NATAL, 2018

Realização: SBC

Organização: UFRN

anais 2018

XXXVIII CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO

3º ETC – ENCONTRO DE TEORIA DA COMPUTAÇÃO

CENTRO DE CONVENÇÕES | NATAL•RN | 22 A 26 DE JULHO DE 2018

#COMPUTAÇÃOESUSTENTABILIDADE

CSBC 2018

anais 2018

XXXVIII CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO
CENTRO DE CONVENÇÕES | NATAL•RN | 22 A 26 DE JULHO DE 2018
#COMPUTAÇÃOESUSTENTABILIDADE

Coordenador Geral

Francisco Dantas de Medeiros Neto (UERN)

Comissão Organizadora

Bartira Paraguaçu Falcão Dantas Rocha (UERN)
Camila Araújo Sena (UERN)
Everton Ranielly de Sousa Cavalcante (UFRN)
Felipe Torres Leite (UFERSA)
Ilana Albuquerque (UERN)
Isaac de Lima Oliveira Filho (UERN)
Priscila Nogueira Krüger (UERN)

Realização

Sociedade Brasileira de Computação

Organização

Universidade do Estado do Rio Grande do Norte

CSBC 2018
XXXVIII Congresso da
Sociedade Brasileira de Computação

Apresentação

Estes anais registram os trabalhos apresentados durante o XXXVIII Congresso da Sociedade Brasileira de Computação (CSBC 2018), realizado em Natal-RN, de 22 a 26 de julho 2018. O evento teve como tema central a Computação e Sustentabilidade, pois se compreende que o avanço da computação e as questões ambientais devem caminhar lado-a-lado, tendo em vista que as técnicas computacionais necessitam ser usadas para possibilitar o desenvolvimento sustentável, e, desse modo, equilibrar as necessidades ambientais, econômicas e sociais.

Organizar o maior evento acadêmico de Computação da América Latina foi um privilégio e um desafio. Foi enriquecedor promover e incentivar a troca de experiências entre estudantes, professores, profissionais, pesquisadores e entusiastas da área de Computação e Informática de todo o Brasil. Ao mesmo foi desafiador termos que lidar, principalmente, com às dificuldades impostas pelo momento de crise que o nosso Brasil vem enfrentando. Uma crise que afeta diretamente nossas pesquisas e, consequentemente, o desenvolvimento e inovação do nosso amado Brasil.

Por meio de seus 25 eventos, o CSBC 2018 apresentou mais de 300 trabalhos, várias palestras e mesas-redondas. O Congresso ainda abrigou diversas reuniões, que incluem a reunião do Fórum de Pós-Graduação, a reunião do CNPq/CAPES, a reunião dos Secretários Regionais SBC, a reunião das Comissões Especiais e a reunião do Fórum IFIP/SBC.

O sucesso do CSBC 2018 só foi possível devido à dedicação e entusiasmo de muitas pessoas. Gostaríamos de agradecer aos coordenadores dos 25 eventos e aos autores pelo envio de seus trabalhos. Além disso, gostaríamos de expressar nossa gratidão ao Comitê Organizador, por sua grande ajuda em dar forma ao evento; e, em especial, à equipe da Sociedade Brasileira de Computação (SBC), por todo apoio.

Por fim, reconhecemos a importância do apoio financeiro da CAPES, do CNPq, do CGI.br, do Governo do Estado do Rio Grande do Norte, da Prefeitura Municipal do Natal, da Prefeitura Municipal de Parnamirim, da CABO Telecom, da ESIG Software e Consultoria, da DynaVideo e do SENAI.

Natal (RN), 26 de julho de 2018.

Chico Dantas (UERN)
Coordenador Geral do CSBC 2018

Anais do CSBC 2018

3º ETC – ENCONTRO DE TEORIA DA COMPUTAÇÃO

III Encontro de Teoria da Computação (ETC)

Apresentação

É com muita satisfação que anunciamos a terceira edição do Encontro de Teoria da Computação. Chegamos a esta edição com a convicção de que os objetivos iniciais que motivaram a realização dos Encontros de Teoria da Computação como evento satélite dos congressos da SBC foram atendidos. De fato, os ETC's, incluindo este, tem continuamente integrado e aproximado estudantes e pesquisadores da área, e nos aproximado da Sociedade Brasileira de Computação, divulgando a área de teoria da computação entre a comunidade de pesquisadores, profissionais e estudantes que participam dos congressos da SBC. Essa série de encontros vem sendo proposta e organizada pela Comissão Especial em Algoritmos, Combinatória e Otimização (CE-ACO) da SBC. Para esta edição foram submetidos 45 trabalhos, e repetindo as edições anteriores, de excelente qualidade. Foram aceitos 40 trabalhos, que serão apresentados ao longo dos dias 23 e 24 de julho de 2018. A apresentação dos trabalhos será intercalada por palestras de pesquisadores renomados, cobrindo amplo espectro da área de teoria, a saber, lógica, algoritmos, grafos e otimização. A preparação do evento mobilizou a nossa comunidade: os artigos aceitos têm 116 autores e os 45 artigos submetidos foram revisados por 86 pesquisadores que emitiram 148 pareceres. Os autores dos trabalhos aceitos estão em 16 instituições distintas, no Brasil e no exterior.

Gostaríamos de expressar nosso imenso agradecimento ao Prof. Jayme Szwarcfiter, que generosamente tem apoiado todas as edições do ETC, não nos negando orientação e experiência, ao Prof. Flávio Miyazawa e Prof. Vinicius Santos, que como membros da Coordenação da CEACO, participaram da formulação da proposta do evento, aos palestrantes Prof. Eduardo Laber, Prof. Fábio Protti, Prof. João Marcos e Prof. Manoel Campêlo, que aceitaram prontamente o nosso convite de abrilhantar o III ETC fazendo as palestras plenárias. Somos imensamente gratas à diretoria da SBC, à organização do CSBC 2018, em particular ao Prof. Francisco Dantas, aos membros do nosso maravilhoso Comitê Científico e aos pareceristas externos pelo seu comprometimento e excelente trabalho.

Cláudia Linhares Sales (UFC)
Rosiane de Freitas (UFAM)

Comitês de Organização

Coordenação Geral

Cláudia Linhares Sales (UFC)
Rosiane de Freitas (UFAM)

Comitê de Honra

Jayme Szwarcfiter (UFRJ e UERJ)
Cláudia Linhares Sales (UFC)
Rosiane de Freitas (UFAM)

Coordenação Local

Francisco Chagas Lima Jr (UERN)

Comitê Científico

Ana Teresa Martins (UFC)
Carlos Eduardo Ferreira (USP)
Cláudia Linhares Sales (UFC)
Claudson Bornstein (UFRJ)
Cristina Fernandes (USP)
Edson Cáceres (UFMS)
Erika Coelho (UFGO)
Fábio Protti (UFF)
Flavio Keidi Miyazawa (UNICAMP)
Jayme Szwarcfiter (UFRJ e UERJ)
Luerbio Faria (UERJ)
Luis Carlos Lamb (UFRGS)
Luiz Satoru Ochi (UFF)
Manoel Campelo (UFC)
Marco Goldbarg (UFRN)
Marcus Ritt (UFRGS)
Mario Benevides (UFRJ)
Rosiane de Freitas (UFAM)
Sulamita Klein (UFRJ)
Vinicius Santos (UFMG)

Revisores Ad-hoc III ETC

Alan Carneiro
Alex Vieira
Alexandre Freire
Aline Silva
Alvaro Franco
Amit Bhaya
Ana da Silva
Ana Karolinna Maia
André Guedes
André Grahl Pereira
Carla Lintzmayer
Carlos Fisch de Brito
Carlos Vinícius Costa Lima
Cândida Silva
Claudia Linhares Sales
Claudson Bornstein
Críston Souza
Cristiana Bentes
Cristiane M. Sato
Cristina Fernandes
Daniel Figueiredo
Daniel Menasché
Diana Sasaki
Diane Castonguay
Diego Nicodemos
Edson Caceres
Elizabeth Goldbarg
Emerson Monte Carmelo
Erika Coelho
Fabiano Oliveira
Fabio Dias
Fabio Protti
Fabricio Benevides
Fernando Carvalho
Flavia Bonomo
Flavio Miyazawa
Guilherme de Castro Mendes Gomes
Guilherme Oliveira Mota
Hebert da Silva Coelho
Jayme Szwarcfiter
Júlio César Araújo
Jose Pinto
Josefran Bastos
Julio Araujo
Lehilton Lelis Chaves Pedrosa
Leonardo Rocha
Les Foulds
Loana Nogueira
Luerbio Faria
Luis Lamb
Luiz Satoru Ochi
Marcio Santos
Marco Goldbarg
Marcus Ritt
Mario Salvatierra
Maristela Oliveira Santos
Matheus Menezes
Mauricio Collares
Márcia Cappelle
Mário Benevides
Mitre Dourado
Murilo da Silva
Nicolas Martins
Paloma Lima
Paulo Asconavieta
Paulo Oliveira
Paulo Pinto
Phablo Moura
Rafael Santiago
Rainer Amorim
Raquel Bravo
Ricardo Dahab
Rodolfo Oliveira
Ronan Soares
Rosiane de Freitas
Rubens Sucupira
Rudini Sampaio
Santiago Ravelo
Sebastian Alberto Urrutia
Sheila Almeida
Tamara Baldo
Thatiana Souza
Thiago Marcilon

Palestras

Uma Estrutura Geral para Convexidades de Caminhos

Fábio Proti (Universidade Federal Fluminense)

No estudo de convexidade em grafos, dedica-se interesse especial às chamadas "convexidades de caminhos", definidas a partir de coleções especiais de caminhos em grafos. Por exemplo, a coleção de caminhos mínimos de um grafo está associada com a conhecida convexidade geodésica, enquanto que a coleção de caminhos induzidos está associada com a convexidade monofônica; e existem muitos outros exemplos na literatura. Neste trabalho propomos uma estrutura geral para convexidades de caminhos, da qual muitas convexidades conhecidas podem ser vistas como casos particulares. Alguns benefícios da estrutura proposta são: sistematização dos estudos algorítmicos sobre convexidade e possibilidade de definir novas convexidades ainda não investigadas.

Este trabalho foi realizado em colaboração com J. V. C. Thompson, L. T. Nogueira, R. S. F. Bravo, M. C. Dourado e U. S. Souza.

What we can DO with and what we can LEARN from Mechanized Reasoning

João Marcos (Universidade Federal do Rio Grande do Norte)

Our civilization is the proud developer of artifacts that excel in playing trivia quizzes and board games, drive cars, and that can participate at distance of conversations in which they show proficiency in emulating well the most vicious teenagers out there. While Artificial Intelligence promises four hundred comparatively honest ways of relieving the people of their burden to think, it should be noted that in some of the recent achievements just mentioned the human spectators have not been left with the faintest idea of what goes in the mind of their new computer overlords.

From the viewpoint of teaching and doing research in Logic, this talk will discuss less esoteric advances that can be made by letting computerized tools act as our collaborators in exploring mathematical phenomena, searching for relevant information in databases of mathematical facts, verifying correctness of proofs, assisting the production of formally verified math, and discovering new theorems. I will also discuss, in particular, how mechanized reasoning may actually teach us a few very useful lessons, and how an understanding of the details of the implementation of automated reasoners can help us in the task of delivering better lessons.

Problemas sobre conjuntos independentes em grafos: uma abordagem poliédrica

Manoel Campelo (Universidade Federal do Ceará)

Apresentamos resultados sobre poliedros associados a alguns problemas de otimização em grafos, entre eles conjunto independente, subgrafo induzido k-partido, coloração, multicoloração e coloração fracionária de vértices. Tais problemas procuram um conjunto de conjuntos independentes do grafo que satisfaçam certa(s) propriedade(s) (por exemplo, serem disjuntos e/ou cobrirem todos os vértices) e que maximizem ou minimizem algum critério. Estudamos a estrutura facial desses poliedros, procurando relacioná-los. Apresentamos

desigualdades válidas, em geral indutoras de facetas, bem como procedimentos para geração automática de tais desigualdades. Alguns resultados computacionais que exploram o uso dessas desigualdades na resolução dos problemas também são reportados.

On partitions with minimum weighted entropy
Eduardo Laber (Pontifícia Universidade Católica-Rio)

The Shannon entropy for a k-dimensional vector v=(v_1,...,v_k), with non-negatives coordinates, is given by - \sum p_i \log p_i, where p_i = v_1/ ||v||_1.

Given a collection S of k-dimensional vectors and a positve integer L>=2, we consider the problem of finding the partition of S into at most L groups so that the weighted sum of the entropies of the groups in minimized, where the entropy of a group G is the entropy of the vector u=\sum_{v \in G} v, and the weight of G is the L_1-norm of de u. This problem has a direct application on the construction of decision trees and random forests and, in the last years, it has been investigated by the Information theory community due to other applications as the design of polar codes. In our talk we present approximation algorithms and results concerning the computational complexity of the problem. A manuscript describing some of our results has been recently accepted for presentation on the International Conference on Machine Learning (ICML 2018).

SUMÁRIO

Practical aspects of ℓ_0-sampling algorithms

Juan P. A. Lopes[1], **Fabiano S. Oliveira**[2], **Valmir C. Barbosa**[1]

[1]PESC/COPPE – Universidade Federal do Rio de Janeiro – Rio de Janeiro – Brasil

[2]IME – Universidade do Estado do Rio de Janeiro – Rio de Janeiro – Brasil

jlopes@cos.ufrj.br, fabiano.oliveira@ime.uerj.br, valmir@cos.ufrj.br

Abstract. *The ℓ_0-sampling problem plays an important role in streaming graph algorithms. In this paper, we revisit a near-optimal ℓ_0-sampling algorithm, proposing a variant that allows proving a tighter upper bound for the probability of failure. We compare experimental results of both variants, providing empirical evidence of their applicability in real-case scenarios.*

1. Introduction

The ℓ_0-sampling problem consists in sampling a nonzero coordinate from a dynamic vector $\boldsymbol{a} = (a_1, \ldots, a_n)$ with uniform probability. This vector is defined in a turnstile model, which consists of a stream of updates $S = \langle s_1, s_2, \ldots, s_t \rangle$ on $\boldsymbol{a}$ (initially $\mathbf{0}$), where $s_i = (u_i, \Delta_i) \in \{1, \ldots, n\} \times \mathbb{R}$ for all $1 \leq i \leq t$, meaning an increment of Δ_i units to a_{u_i}. It is desirable that such sample be produced in a single pass through the stream with sublinear space complexity. The challenge arises from the fact that, since Δ_i can be negative and hence some updates in the stream may cancel others, directly sampling the stream may lead to incorrect results.

The research on ℓ_0-sampling algorithms has recently gained some traction, in part due to results showing that these algorithms can be used as building blocks in many other algorithms [Cormode and Firmani 2014]. For example, the sampling of nonzero coordinates from rows of the incidence matrix of a graph can be used to compute connected components, k-connectivity, bipartiteness, and approximate minimum spanning trees in dynamic graphs using $O(n \log^c n)$ bits, for some constant c [Ahn et al. 2012, McGregor 2014].

In order to achieve sublinear space complexity in a single pass, an ℓ_0-sampling algorithm must represent $\boldsymbol{a}$ through a lower-dimensional projection. This representation is known as a *sketch*. Sketch-based algorithms are common in streaming scenarios, by virtue of allowing compact representations of the original data, whilst retaining some useful information about them.

In [Cormode et al. 2005], a seminal sketch-based algorithm for the ℓ_0-sampling problem is introduced. The algorithm uses a universal family of hash functions to partition the vector $\boldsymbol{a}$ into $O(\log n)$ subvectors with exponentially decreasing probabilities of representing each element of $\boldsymbol{a}$. It is proven that there is a constant lower bound on the probability that at least one of those subvectors has exactly one nonzero coordinate. Through a procedure called 1-*sparse recovery* (Section 2), which requires $O(\log n)$ bits for each subvector, it is possible to recover such coordinate. Considering that the probability of failure has a constant upper bound, running $O(\log(1/\delta))$ independent instances of the algorithm

can ensure a success probability of at least $1 - \delta$. The total space complexity of this algorithm is $O(\log^2 n \log(1/\delta))$. Further studies show stronger results by relaxing assumptions on the hash functions used [Monemizadeh and Woodruff 2010, Jowhari et al. 2011]. Nevertheless, they keep the same worst-case space complexity. In fact, any algorithm that performs ℓ_0-sampling in a single pass should require $\Omega(\log^2 n)$ bits in the worst case [Jowhari et al. 2011]. This holds even if the algorithm allows a relative error of ϵ and a failure probability of δ, for constants ϵ and δ.

2. 1-sparse recovery procedure

A vector is 1-*sparse* when it has a single nonzero coordinate. A 1-sparse recovery procedure allows deciding whether a vector $\boldsymbol{a}$ is 1-sparse, and possibly recover the only nonzero coordinate from it. Note that while $\boldsymbol{a}$ is expected to be 1-sparse at the time of a successful recovery, it may have any number of nonzero coordinates before that. This procedure is a building block for many ℓ_0-sampling algorithms. Here we present a false-biased randomized variant that handles cases where $\boldsymbol{a}$ has negative values [Cormode and Firmani 2014]. It begins by choosing a sufficiently large prime $p \in \Theta(n^c)$, with $c > 1$, and random integer $z \in \mathbb{Z}_p$. Then, iterating through all $s_i = (u_i, \Delta_i) \in S$, three sums are computed:

$$b_0 = \sum_{i=1}^{t} \Delta_i, \qquad b_1 = \sum_{i=1}^{t} \Delta_i u_i, \qquad b_2 = \sum_{i=1}^{t} \Delta_i z^{u_i} \mod p.$$

If $\boldsymbol{a}$ is 1-sparse, it is easy to see that the nonzero coordinate can be recovered as $i = b_1/b_0$, with $a_i = b_0$. However, verifying that $\boldsymbol{a}$ is 1-sparse requires more effort.

Theorem 1. *If $\boldsymbol{a}$ is 1-sparse, then $b_2 \equiv b_0 z^{b_1/b_0} \mod p$. Otherwise, $b_2 \not\equiv b_0 z^{b_1/b_0} \mod p$ with probability at least $1 - n/p$.*

Proof (sketch). If $\boldsymbol{a}$ is 1-sparse, with a nonzero coordinate i, it is trivial to see that $b_2 \equiv a_i z^i \mod p$. Otherwise, $b_2 \equiv b_0 z^{b_1/b_0} \mod p$ may still hold if z is a root in $\mathbb{Z}_p$ of the polynomial $p(z) = b_0 z^{b_1/b_0} - \sum \Delta_i z^{u_i}$. As $p(z)$ is an degree-n polynomial, it has at most n roots in $\mathbb{Z}_p$. Therefore, given that z is chosen at random, the probability of a false recovery is at most n/p. ■

This 1-sparse recovery procedure stores z, and the sums b_0, b_1, and b_2. Assuming that every a_i is limited by a polynomial in n, the total space required is $O(\log n)$ bits.

3. ℓ_0-sampling algorithm

In this work, two variants of the same ℓ_0-sampling algorithm are presented. Both variants define $\boldsymbol{a^{(1)}}, \boldsymbol{a^{(2)}}, \ldots, \boldsymbol{a^{(m)}}$ subvectors of $\boldsymbol{a}$. For all $1 \leq j \leq m$, each $a_i \neq 0$ has a $1/2^j$ probability of being *present* at $\boldsymbol{a^{(j)}}$, that is, $a_i^{(j)} = a_i$ with probability $1/2^j$, otherwise $a_i^{(j)} = 0$. To decide whether $a_i^{(j)}$ is present, we draw a hash function $h_j : \{1, \ldots, n\} \to \{0, \ldots, 2^m - 1\}$ from a universal family, and observe whether $m - \lfloor \log_2 h_j(i) \rfloor = j$, which happens with probability $1/2^j$. An independent 1-sparse recovery is then computed for each $\boldsymbol{a^{(j)}}$. The variants differ only in the number of functions used. Variant (a) uses a single hash function for every $\boldsymbol{a^{(j)}}$ (Algorithm 1), while Variant (b) uses a different function for each subvector (Algorithm 2). While (a) is more useful in practice, the error analysis in (b) is more straightforward. We provide empirical evidence in Section 4 that the error in either variant converges quickly as a function of n.

Algorithm 1 Variant (a)

$M[1..m]$: 1-sparse recoveries
for each $(u_i, \Delta_i) \in S$ **do**
  $k \leftarrow m - \lfloor \log_2 h(u_i) \rfloor$
  $M[k].b_0 \mathrel{+}= \Delta_i$
  $M[k].b_1 \mathrel{+}= \Delta_i u_i$
  $M[k].b_2 \mathrel{+}= \Delta_i M[k].z^{u_i} \mod p$
for $j \in [1..m]$ **do**
  $v \leftarrow M[j].b_0 M[j].z^{M[j].b_1/M[j].b_0} \mod p$
  if $M[j].b_2 = v$ **then**
    return $M[j].b_1/M[j].b_0$
report FAILURE

Algorithm 2 Variant (b)

$M[1..m]$: 1-sparse recoveries
for each $(u_i, \Delta_i) \in S$ **do**
  for $j \in [1..m]$ **do**
    $k \leftarrow m - \lfloor \log_2 h_j(u_i) \rfloor$
    if $k = j$ **then**
      $M[k].b_0 \mathrel{+}= \Delta_i$
      $M[k].b_1 \mathrel{+}= \Delta_i u_i$
      $M[k].b_2 \mathrel{+}= \Delta_i M[k].z^{u_i} \mod p$
for $j \in [1..m]$ **do**
  $v \leftarrow M[j].b_0 M[j].z^{M[j].b_1/M[j].b_0} \mod p$
  if $M[j].b_2 = v$ **then**
    return $M[j].b_1/M[j].b_0$
report FAILURE

Every variant either succeeds in returning a single nonzero coordinate of $\boldsymbol{a}$, or reports a failure. The probability of failure is given by the joint probability of failure of all m 1-sparse recoveries. In Variant (b), those are independent events. Moreover, the probability that a single recovery $M[j]$ fails is the complement of the probability that $\boldsymbol{a}^{(j)}$ is 1-sparse, that is, assuming $\boldsymbol{a}$ has $r \gg 1$ nonzero coordinates:

$$\Pr[\text{FAILURE}] = \prod_{j=1}^{m} \left(1 - r2^{-j}(1-2^{-j})^{r-1}\right) \approx \prod_{j=1}^{m} \left(1 - r2^{-j}e^{-r2^{-j}}\right).$$

Theorem 2. *If* $5 \leq \log_2 r \leq m - 5$*, then* $\Pr[\text{FAILURE}] \leq 0.31$*, for Variant (b).*

Proof (sketch). It is easy to see that the lowest probabilities of failure concentrate around j such that $2^j \leq r < 2^{j+1}$. Letting $q = r/2^{\lfloor \log_2 r \rfloor}$, it holds that

$$\Pr[\text{FAILURE}] \leq \prod_{k=-5}^{5} \left(1 - q2^k e^{-q2^k}\right).$$

Note that $1 \leq q < 2$. In this interval, all factors $1 - q2^k e^{-q2^k}$ are either monotonically increasing or decreasing. Analyzing their global maxima, we arrive at a maximum product of approximately 0.3071, therefore $\Pr[\text{FAILURE}] \leq 0.31$. ■

This result shows that, as n grows, choosing $m = 5 + \lceil \log_2 n \rceil$ is enough to ensure a constant upper bound on the probability of failure. Furthermore, to ensure a success probability of at least $1 - \delta$, it is sufficient to run $\lceil \log_{0.31} \delta \rceil$ instances of the algorithm.

4. Empirical evaluation and outlook

In order to assess the algorithms behavior in a real implementation, an experiment was set up. Both variants were implemented and tested with a vector of size $n = 4\,096$ and increasing values of r. We tested both a correctly sized (i.e., for $m = 17$) and an undersized instance of the ℓ_0-sampling algorithm. The empirical cumulative distribution was also recorded. The experiment was run $100\,000$ times and the mean value for each data point is reported in Figure 1.

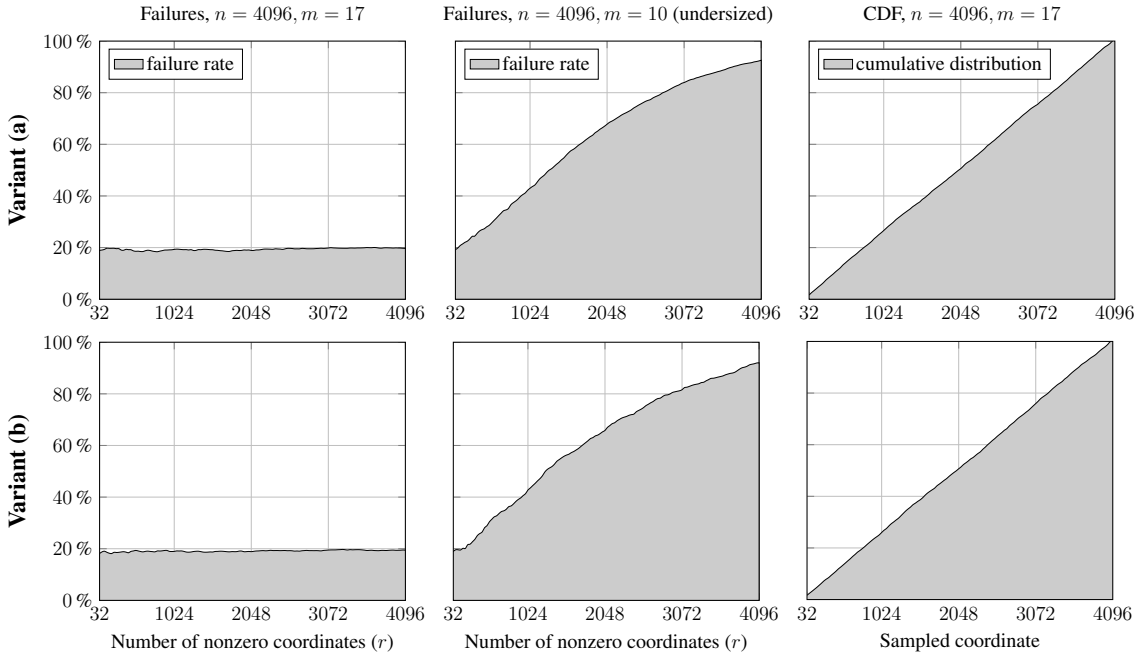


Figure 1. Failure rate and cumulative distribution of successes.

This experiment suggests that in a correctly sized ℓ_0-sampling, the failure probability stays almost constant under 20%. There is little difference between Variants (a) and (b). Furthermore, in an undersized setup, the failure rate rapidly reaches critical levels.

In conclusion, we have introduced a variant of the ℓ_0-sampling algorithm and proved its failure probability to be bounded by a constant value, provided a certain structure-size condition is met. Research is ongoing on the proof of exact probabilities of failure for both algorithm variants. Future research may also include novel graph algorithms that use ℓ_0-sampling as a primitive.

References

Ahn, K. J., Guha, S., and McGregor, A. (2012). Analyzing graph structure via linear measurements. In *Proceedings of SODA'12*, pages 459–467.

Cormode, G. and Firmani, D. (2014). A unifying framework for ℓ_0-sampling algorithms. *Distributed and Parallel Databases*, 32(3):315–335.

Cormode, G., Muthukrishnan, S., and Rozenbaum, I. (2005). Summarizing and mining inverse distributions on data streams via dynamic inverse sampling. In *Proceedings of VLDB'05*, pages 25–36.

Jowhari, H., Sağlam, M., and Tardos, G. (2011). Tight bounds for l_p samplers, finding duplicates in streams, and related problems. In *Proceedings of PODS'11*, pages 49–58.

McGregor, A. (2014). Graph stream algorithms: a survey. *ACM SIGMOD Record*, 43(1):9–20.

Monemizadeh, M. and Woodruff, D. P. (2010). 1-pass relative-error l_p-sampling with applications. In *Proceedings of SODA'10*, pages 1143–1160.

Ramsey-type problems in orientations of graphs *

Bruno Pasqualotto Cavalar[1]

[1]Instituto de Matemática e Estatística, Universidade de São Paulo
Rua do Matão 1010, 05508–090 São Paulo, SP

brunopc@ime.usp.br

Abstract. *The Ramsey number $R(H)$ of a graph H is the minimum number n such that there exists a graph G on n vertices with the property that every two-coloring of its edges contains a monochromatic copy of H. In this work we study a variant of this notion, called the oriented Ramsey problem, for an acyclic oriented graph $\vec{H}$, in which we require that every orientation $\vec{G}$ of the graph G contains a copy of $\vec{H}$. We also study the threshold function for this problem in random graphs. Finally, we consider the isometric case, in which we require the copy to be isometric, by which we mean that, for every two vertices $x, y \in V(\vec{H})$ and their respective copies x', y' in $\vec{G}$, the distance between x and y is equal to the distance between x' and y'.*

1. Introduction

An *ordered graph* G is a pair $G = (G', <_G)$ where G' is a graph and $<_G$ is a total ordering of the vertices of G'. For convenience we write $V(G) := V(G')$ and $E(G) := E(G')$. When a graph G is equipped with a total ordering of its vertices, we will simply refer to G as an ordered graph without further qualifications. Finally, G' is called the *underlying unordered graph* of G.

An ordered graph G is said to *contain* an ordered graph H if there exists a function $\phi : V(H) \to V(G)$ such that, for every $x, y \in V(H)$, we have $\phi(x) <_G \phi(y)$ if and only if $x <_H y$, and $\{i, j\}$ is an edge of H only if $\{\phi(i), \phi(j)\}$ is an edge of G. In this case, we call ϕ a *monotone embedding*.

A *directed graph* or *digraph* $\vec{G}$ is a pair $\vec{G} = (V, E)$ where V is a set of vertices and E is a set such that $E \subseteq (V \times V) \setminus \{(v, v) : v \in V\}$. Just as in the case of undirected graphs, an element of E is called an *edge*; however, it may also be called an *arc* to differ from the undirected case. An *oriented graph* $\vec{G} = (V, E)$ is a digraph where $(u, v) \in E$ implies $(v, u) \notin E$ for every $u, v \in V$. Moreover, an oriented graph $\vec{G} = (V_1, E_1)$ is said to be an *orientation* of a graph $G = (V_2, E_2)$ if $V_1 = V_2$ and, for every $u, v \in V_1 = V_2$, we have $\{u, v\} \in E_2$ if and only if $(u, v) \in E_1$ or $(v, u) \in E_1$. In this case, we say that G is the *underlying undirected graph* of $\vec{G}$. Furthermore, when $\vec{G}$ is an oriented graph, we write G to denote the underlying undirected graph of $\vec{G}$. To avoid confusion, we will always denote a digraph by a capital letter with $\to$.

Given graphs H and G and an integer $r \geqslant 2$, we write $G \to (H)_r$ if every coloring of the edges of G with r colors contains a monochromatic copy of H. When $r = 2$, we may simply write $G \to H$.

*Work supported by FAPESP, Proc. 2015/26678-9.

If the graphs H and G are ordered graphs, we write $G \xrightarrow{\text{ord}} H$ to denote that the monochromatic copy is *ordered*. The *Ramsey number* $R(H)$ of a graph H is defined as

$$R(H) := \inf \{n \in \mathbb{N} : K_n \to H\}.$$

When the graph H is an ordered graph, the *ordered Ramsey number* $R_<(H)$ can be defined analogously. Moreover, for convenience we let $R(H)$ denote the Ramsey number of its underlying unordered graph.

Given an oriented graph $\vec{H}$ and a graph G, we write $G \to \vec{H}$ if every orientation of G has an oriented copy of $\vec{H}$. The *oriented Ramsey number* $\vec{R}(\vec{H})$ is defined as

$$\vec{R}(\vec{H}) := \inf \left\{ n \in \mathbb{N} : K_n \to \vec{H} \right\}.$$

2. A bound for the oriented Ramsey number

To our knowledge, the following is the first bound to appear of the oriented Ramsey number of an oriented graph. Not much has been published since then. Contrary to the traditional Ramsey number, the oriented Ramsey number has been scarcely studied.

Theorem 1 ([Erdős and Moser 1964])**.** Let $\vec{K}_k$ be the acyclic orientation of K_k for some positive integer k. We have $2^{(k-1)/2} \leqslant \vec{R}(\vec{K}_k) \leqslant 2^{k-1}$.

We now give a bound for the oriented Ramsey number of $\vec{H}$ depending on the Ramsey number of H. First, we need a bound for the Ramsey number of ordered graphs.

Theorem 2 ([Conlon et al. 2017])**.** There exists a constant c such that, for every ordered graph H on h vertices, we have $R_<(H) \leqslant R(H)^{c \log^2 h}$.

More precise bounds for $R_<(H)$ for specific classes of ordered graphs can be found in [Conlon et al. 2017] and [Balko et al. 2015]. Our first result is as follows. A slightly stronger result is proven in the full paper.

Theorem 3. There exists a constant c such that the following holds. Let $\vec{H}$ be an acyclic oriented graph with h vertices and H its underlying undirected graph. Then

$$\vec{R}(\vec{H}) \leqslant 2R(H)^{c \log^2 h}.$$

The proof goes roughly as follows. Let $\vec{F}$ be the oriented graph formed by two disjoint copies of $\vec{H}$, in which one has reversed edges. Let F be the (ordered) underlying undirected graph of $\vec{F}$ equipped with a topological ordering of the vertices. We prove that, if $K_n \xrightarrow{\text{ord}} F$, then $K_n \to \vec{H}$, which implies the result.

3. An Oriented Ramsey Theorem for Random Graphs

For a graph H, we denote by $m_2(H)$ its 2-*density*, defined as

$$m_2(H) := \max_{F \subseteq H, |V(F)| \geqslant 3} \frac{|E(F)| - 1}{|V(F)| - 2},$$

where $F \subseteq H$ means that F is a subgraph of H. Let us also denote by $G(n,p)$ the binomial random graph in which each edge appears with probability p, independently of each other edge.

The following is a famous result of Rödl and Ruciński [Rödl and Ruciński 1995], which determines, for an undirected graph H, the threshold function for $G(n,p) \to (H)_r$. Here we state only the 1-statement.

Theorem 4 ([Rödl and Ruciński 1995])**.** Let $r \geqslant 2$ and H be a graph. There exists a constant $C = C(H, r)$ such that, if $p \geqslant Cn^{-1/m_2(H)}$, then $\lim_{n\to\infty} \mathbb{P}[G(n,p) \to (H)_r] = 1$.

We obtained a result for oriented graphs analogous to Theorem 4, described as follows.

Theorem 5. Let $\vec{H}$ be an acyclic oriented graph. There exists a constant $C = C(\vec{H})$ such that, if $p \geqslant Cn^{-1/m_2(\vec{H})}$, then

$$\lim_{n\to\infty} \mathbb{P}\left[G(n,p) \to \vec{H}\right] = 1.$$

Our proof of Theorem 5 adapts the arguments of [Nenadov and Steger 2016], who gave a short proof of Theorem 4 using the method of containers, developed independently by [Balogh et al. 2015] and [Saxton and Thomason 2015]. The technique of using hypergraph containers in random graphs for Ramsey problems was further developed by [Hàn et al. 2016], [Rödl et al. 2016] and [Conlon et al. 2016]. Our approach is also inspired by theirs.

A very rough explanation of the proof idea would be as follows. Applying the hypergraph container lemma of [Balogh et al. 2015] and [Saxton and Thomason 2015], we are able to prove that, if a graph G on n vertices does not satisfy $G \to \vec{H}$, then there exists a s-tuple $\mathcal{T} = (\mathcal{T}_1, \ldots, \mathcal{T}_s)$ of subsets of $E(G)$ and a set $\mathcal{C} = \mathcal{C}(\mathcal{T}) \subseteq E(K_n)$ depending only on $\mathcal{T}$ such that

(i) The number s depends only on $\vec{H}$,
(ii) $\bigcup_{i\in[s]} \mathcal{T}_i \subseteq E(G) \subseteq \mathcal{C}$,
(iii) $|\mathcal{T}_i|$ is "small" (in a precise technical sense) for every $i \in [s]$,
(iv) $|\mathcal{C}|$ is "small" in a precise technical sense.

Observe that, since $E(G) \subseteq \mathcal{C}(\mathcal{T})$, then $D(\mathcal{T}) := E(K_n) \setminus \mathcal{C}(\mathcal{T})$ satisfies $E(G) \cap D(\mathcal{T}) = \emptyset$. Moreover, since $|\mathcal{C}(\mathcal{T})|$ is "small", we have that $|D(\mathcal{T})|$ is "large". Because of this, we can bound the probability of the event $\{G(n,p) \not\to \vec{H}\}$ by the probability that there exists a s-tuple $\mathcal{T}$ satisfying (i)-(iii) and such that G avoids $D(\mathcal{T})$. Since the $\mathcal{T}_i$ are "small" and $D(\mathcal{T})$ is "large", this probability must also be "small" – that is, we must have $\mathbb{P}[G(n,p) \not\to \vec{H}] = o(1)$.

4. The Isometric Oriented Ramsey Number

For an undirected graph G, we denote by $d_G(u,v)$ the distance between two vertices $u, v \in V(G)$. Given two oriented graphs $\vec{H}$ and $\vec{F}$, we say that a copy $f : V(\vec{H}) \to V(\vec{F})$ of $\vec{H}$ in $\vec{F}$ is an *isometric copy* if $d_H(x,y) = d_F(f(x), f(y))$ for every $x, y \in V(\vec{H})$. Note that the distance is taken with respect to the underlying undirected graphs.

Given an oriented graph $\vec{H}$ and a graph G, we write $G \xrightarrow{\text{iso}} \vec{H}$ if every orientation of G has an isometric oriented copy of $\vec{H}$. The *isometric oriented Ramsey number* $\vec{R}_{\text{iso}}(\vec{H})$ is defined as

$$\vec{R}_{\text{iso}}(\vec{H}) := \inf\left\{n \in \mathbb{N} : \text{there exists a graph } G \text{ of order } n \text{ such that } G \xrightarrow{\text{iso}} \vec{H}\right\}.$$

The following result states that the isometric oriented Ramsey number of acyclic oriented graphs is always finite.

Theorem 6 ([Banakh et al. 2017], Theorem 2.1)**.** For every acyclic oriented graph $\vec{H}$, the isometric oriented Ramsey number $\vec{R}_{\text{iso}}(\vec{H})$ is finite.

The problem of estimating $\vec{R}_{\text{iso}}(\vec{H})$ for acyclic oriented graphs $\vec{H}$ first appeared in Banakh, Idzik, Pikhurko, Protasov and Pszczoła [Banakh et al. 2017], where an upper bound for the isometric Ramsey number of oriented trees is given. Our work gives an upper bound on $\vec{R}_{\text{iso}}(\vec{H})$ when $\vec{H}$ is an acyclic orientation of the cycle on k vertices C_k. In particular, we prove the following theorem.

Theorem 7. There exists a positive constant c such that the following holds. Let $\vec{H}$ be an acyclic orientation of C_k and set $R := \vec{R}(\vec{H})$. Then

$$\vec{R}_{\text{iso}}(\vec{H}) \leqslant ck^{12k^3} R^{8k^2}. \tag{1}$$

Our approach to prove Theorem 7 employs the same techniques we used to prove Theorem 5. It goes as follows. We consider the random graph $G(n,p)$ and, imitating the proof of Theorem 5, we prove that, with positive probability, we have $G(n,p) \xrightarrow{\text{iso}} \vec{H}$ for a number n that satisfies (1) and a suitable choice of p. In order to show that $G(n,p) \xrightarrow{\text{iso}} \vec{H}$, we prove that the graph $G(n,p)$ has girth at least k and satisfies $G(n,p) \to \vec{H}$, which implies $G(n,p) \xrightarrow{\text{iso}} \vec{H}$. Once again, the proof makes use of the hypergraph container lemma.

References

Balko, M., Cibulka, J., Král, K., and Kynčl, J. (2015). Ramsey numbers of ordered graphs. *Electronic Notes in Discrete Mathematics*, 49:419 – 424. The Eight European Conference on Combinatorics, Graph Theory and Applications, EuroComb 2015.

Balogh, J., Morris, R., and Samotij, W. (2015). Independent sets in hypergraphs. *J. Amer. Math. Soc.*, 28(3):669–709.

Banakh, T., Idzik, A., Pikhurko, O., Protasov, I., and Pszczoła, K. (2017). Isometric copies of directed trees in orientations of graphs.

Conlon, D., Dellamonica, Jr., D., La Fleur, S., Rödl, V., and Schacht, M. (2016). A note on induced Ramsey numbers. *arXiv e-prints*.

Conlon, D., Fox, J., Lee, C., and Sudakov, B. (2017). Ordered Ramsey numbers. *J. Combin. Theory Ser. B*, 122:353–383.

Erdős, P. and Moser, L. (1964). On the representation of directed graphs as unions of orderings. *Magyar Tud. Akad. Mat. Kutató Int. Közl.*, 9:125–132.

Hàn, H., Retter, T., Rödl, V., and Schacht, M. (2016). Ramsey-type numbers involving graphs and hypergraphs with large girth.

Nenadov, R. and Steger, A. (2016). A short proof of the random Ramsey theorem. *Combin. Probab. Comput.*, 25(1):130–144.

Rödl, V. and Ruciński, A. (1995). Threshold functions for Ramsey properties. *J. Amer. Math. Soc.*, 8(4):917–942.

Rödl, V., Ruciński, A., and Schacht, M. (2016). An exponential-type upper bound for folkman numbers. *Combinatorica*.

Saxton, D. and Thomason, A. (2015). Hypergraph containers. *Invent. Math.*, 201(3):925–992.

Calculando o número de envoltória nas convexidades P_3 e P_3^* [†]

J. Araujo[1,3], **M. Campêlo**[2,3], **G. H. de Sousa**[2,3]

[1]Dep. de Matemática - Universidade Federal do Ceará (UFC)

[2]Dep. de Estatística e Matemática Aplicada - Universidade Federal do Ceará (UFC)

[3]Grupo de Pesquisa ParGO - Paralelismo, Grafos e Otimização

julio@mat.ufc.br, mcampelo@lia.ufc.br, gabriel.hellen@hotmail.com

Abstract. *A subset of vertices S in a graph $G = (V, E)$ is* convex *in the P_3 (resp. P_3^*) convexity if every vertex in $V(G) \setminus S$ does not have two neighbors (resp. that are not adjacent to each other) in S. The* convex hull *of S is the minimum convex set containing it. A* hull set *is a set whose convex hull is $V(G)$. The* hull number *is the cardinality of a minimum hull set. In this work, we propose and study two integer-linear programming formulations to determine the hull number of a graph in the P_3 and P_3^* convexities, which we believe to be the first ones presented in the literature. We carry out some computational experiments to evaluate their performances.*

Resumo. *Um subconjunto de vértices S em um grafo $G = (V, E)$ é* convexo *na convexidade P_3 (resp. P_3^*) se todo vértice $v \in V(G) \setminus S$ não possuir dois vizinhos (resp. que não sejam adjacentes entre si) em S. A* envoltória convexa *de S é o menor conjunto convexo que o contém. Um* conjunto de envoltória *é um conjunto cuja envoltória convexa é $V(G)$. O* número de envoltória *é a cardinalidade de um conjunto de envoltória mínimo. Neste trabalho, propomos e estudamos duas formulações de programação linear-inteira para determinar o número de envoltória de um grafo nas convexidades P_3 e P_3^*, que acreditamos serem as primeiras na literatura. Realizamos experimentos computacionais para avaliar seus desempenhos.*

Introdução

Um *espaço de convexidade* é um par ordenado $(V, \mathcal{C})$, onde V é um conjunto finito não vazio e $\mathcal{C}$ é uma família de subconjuntos de V, chamados de *conjuntos convexos*, satisfazendo:

(C1) $\emptyset, V \in \mathcal{C}$ e **(C2)** $C \cap C' \in \mathcal{C}$, para todos $C, C' \in \mathcal{C}$.

Dado um subconjunto $C \subseteq V$, a *envoltória convexa de C* (com respeito a $(V, \mathcal{C})$) é o único conjunto minimal (com respeito a inclusão) $C' \in \mathcal{C}$ que contém C e é denotado por $\mathrm{H}_{(V,\mathcal{C})}(C)$. Se $\mathrm{H}_{(V,\mathcal{C})}(C) = V$, então C é chamado um *conjunto de envoltória* de $(V, \mathcal{C})$. O *número de envoltória de V com respeito a $\mathcal{C}$* é a cardinalidade de um conjunto de envoltória mínimo. Essas noções remontam aos trabalhos de [Farber and Jamison 1986, Duchet 1988].

[†]Esta pesquisa foi financiada pelo CNPq sob projetos 459466/2014-3, 310234/2015-8 e 401519/2016-3.

Quando se trata de convexidade em grafos, a maioria das convexidades definidas na literatura toma V como o conjunto de vértices e a família $\mathcal{C}$ como subconjuntos de vértices que são pontos fixos de uma função de intervalo. Dado um grafo $G = (V, E)$, uma *função de intervalo* é uma função $I : 2^{V(G)} \to 2^{V(G)}$ tal que $I(S) \supseteq S$, para todo $S \subseteq V(G)$. Os vértices em $I(S) \setminus S$ são ditos *infectados* ou *gerados* por S. Note que, como um função de intervalo I é monótona com respeito a inclusão, seus pontos fixos S (ou seja, $I(S) = S$) definem uma convexidade em $V(G)$, dita convexidade de intervalo.

Há várias convexidades de intervalo estudadas na literatura. Aqui estudamos as convexidades P_3 e P_3^*. Na primeira, $I(S)$ retorna o conjunto de vértices que possuem dois vizinhos em S (ou seja, que pertencem a um P_3 entre vértices de S), enquanto que na segunda $I(S)$ retorna apenas os vértices que possuem dois vizinhos u, v em S tais que $uv \notin E(G)$ (consequentemente, neste caso o P_3 deve ser induzido). Além disso, o nosso interesse é em calcular o número de envoltória nessas convexidades.

Apresentamos e estudamos duas formulações de programação linear-inteira para determinar esses parâmetros, que acreditamos serem as primeiras na literatura. Em seguida, executamos testes em instâncias geradas aleatoriamente para analisar o desempenho de tais formulações quando as instâncias são grafos arbitrários, ou mesmo bipartidos. Vale enfatizar que a determinação de tais parâmetros é um problema *NP*-díficil para grafos bipartidos [Araújo et al. 2013] e para grafos planares com grau máximo limitado [Draque Penso et al. 2014].

Formulações

Modelo 1: Passo de Contaminação.

O primeiro modelo baseia-se no tempo para contaminação, ou seja, no número de aplicações da função de intervalo necessárias para infectar todo o grafo. Seja P um limite superior para esse tempo (por exemplo, $P = |V| - 2$). Para $k \in V$, considere H_k como o conjunto de pares de vértices ambos adjacentes a k (e não adjacentes entre si, para P_3^*). Usamos as variáveis binárias x_i^p e y_{ij}^p, $i, j \in V$ e $p = 0, \ldots, P$, para indicar, respectivamente, se o vértice i e o par $\{i, j\}$ estão contaminados ou não no passo p. Com isso, obtemos o modelo:

$$\min \sum_{i \in V} x_i^0 \tag{1}$$

$$\text{s.a: } x_i^P = 1, \qquad \forall i \in V, \tag{2}$$

$$y_{ij}^p \leq x_i^p, y_{ij}^p \leq x_j^p, \qquad \forall i, j \in V, p = 0, ..., P, \tag{3}$$

$$x_k^{p+1} \leq \sum_{\{i,j\} \in H_k} y_{ij}^p + x_k^0, \qquad \forall k \in V, p = 0, ..., P-1, \tag{4}$$

$$x_i^p \in \{0, 1\}, y_{ij}^p \in \{0, 1\}, \qquad \forall i, j \in V, p = 0, ..., P. \tag{5}$$

As restrições (2) garantem que todos os vértices serão contaminados, enquanto (4) asseguram que i está infectado no passo $p + 1$ se foi infectado no passo 0 ou dois de seus vizinhos estão infectados no passo p. As restrições (3) estabelecem a relação necessária entre as variáveis. Note que esse modelo apresenta um número cúbico, em termos de $|V|$, de variáveis e de restrições, e pode ser submetido diretamente a um *solver*.

Modelo 2: Co-convexo.

Este modelo é inspirado no método implementado em [Sag] para o cálculo do número de envoltória na convexidade geodésica. Para descrevê-lo, precisamos introduzir uma nova definição.

Dado um grafo G, um conjunto $S \subseteq V(G)$ é *co-convexo* se $V(G) \setminus S$ é convexo. Consequentemente, observe que, se nenhum vértice de S for infectado inicialmente, como a envoltória de $V(G) \setminus S$ é o próprio conjunto, nenhum dos vértices de S se tornará contaminado. Desta forma, pode-se deduzir que S' é um conjunto de envoltória se, e somente se, para todo conjunto co-convexo S, temos que $S' \cap S \neq \emptyset$. Essa propriedade leva ao segundo modelo, onde a variável binária x_i indica se o vértice i faz parte do conjunto de envoltória:

$$\begin{aligned} \min \quad & \sum_{i \in V} x_i & \\ \text{s.a:} \quad & \sum_{i \in S} x_i \geq 1 & \forall S \in CC(G) \\ & x_i \in \{0,1\} & \forall i \in V(G) \end{aligned}$$

onde $CC(G)$ é a família de conjuntos co-convexos de G (na respectiva convexidade).

Pode-se diferenciar esse modelo do anterior em alguns aspectos. Nesse, o número de variáveis é linear, o que configura uma vantagem em relação ao anterior, onde tal número é cúbico. Por outro lado, o modelo co-convexo possui um número exponencial de restrições, tornando-se inviável sua submissão direta a um *solver*. Para resolvê-lo, podemos aplicar o método de planos-de-corte. A separação das restrições pode ser feita em tempo polinomial.

Experimentos Computacionais

Avaliamos o desempenho das duas formulações em instâncias geradas aleatoriamente. Cada grafo é gerado a partir de dois parâmetros: n (quantidade de vértices) e $p \in (0, 1]$ (probabilidade de existência de cada aresta). Para obter grafos bipartidos, separamos os vértices em dois conjuntos A e B, com $\lceil n/2 \rceil$ e $\lfloor n/2 \rfloor$ vértices; criamos uma árvore geradora T (mantendo a bipartição) e, para todo par $(i, j) \in (A \times B) \setminus E(T)$ acrescentamos a aresta ij com probabilidade p. Para gerar os grafos aleatórios, segue-se o mesmo procedimento, porém a árvore inicial é qualquer.

Um resumo dos resultados dos experimentos pode ser visto nas Tabelas 1 (grafos bipartidos) e 2 (grafos arbitrários). Em cada tabela, apresentamos, para cada instância, o tempo requerido (em segundos) pelos dois modelos (Pas = modelo 1, CC = modelo 2), considerando as duas convexidades (P_3 e P_3^*), além da solução ótima (Sol = número de envoltória) em cada convexidade. O sinal "-" indica que a formulação não conseguiu encontrar o ótimo no tempo limite de 600s, usando o solver CPLEX.

Podemos observar que o modelo 1 demanda tempos bastante similares para as duas convexidades em bipartidos, mas o mesmo já não ocorre para grafos arbitrários. Ele não conseguiu lidar com a maioria das instâncias com 40 vértices ou mais no tempo limite. Já o modelo 2 mostrou-se bastante eficiente para ambas, com tempos de execução similares. Como trabalho futuro, desejamos estudar uma maior quantidade de instâncias e uma

Tabela 1. Grafos bipartidos

	$\|V\|$	p	P3-Pas	P3*-Pas	P3-CC	P3*-CC	Sol-P3	Sol-P3*
1	10	0,9	0,5598	0,5835	0,0017	0,0019	2	2
2	10	0,5	0,5867	0,5750	0,0016	0,0017	2	2
3	10	0,2	0,1502	0,1739	0,6089	0,6532	5	5
4	20	0,9	18,6940	18,5222	0,0031	0,0033	2	2
5	20	0,5	55,4023	54,0191	0,0024	0,0029	2	2
6	20	0,2	8,8439	8,8338	2,0854	2,0584	4	4
7	40	0,9	566,0112	563,8231	0,0088	0,0104	2	2
8	40	0,5	-	-	0,0067	0,0076	2	2
9	40	0,2	-	-	0,0523	0,0601	2	2
10	80	0,9	–	-	0,0432	0,0535	2	2
11	80	0,5	–	-	0,0229	0,0275	2	2
12	80	0,2	-	-	0,0141	0,0118	2	2

Tabela 2. Grafos arbitrários

	$\|V\|$	p	P3-Pas	P3*-Pas	P3-CC	P3*-CC	Sol-P3	Sol-P3*
1	10	0,9	0,9740	0,1496	0,0025	0,1074	2	2
2	10	0,5	0,9958	0,4127	0,0019	0,0232	2	3
3	10	0,2	0,5135	0,2033	0,5231	0,3269	3	4
4	20	0,9	27,2655	11,7637	0,0051	0,3394	2	2
5	20	0,5	23,2847	22,5997	0,0034	0,0049	2	2
6	20	0,2	-	-	0,0283	0,0359	2	2
7	40	0,9	-	-	0,0219	0,0464	2	2
8	40	0,5	-	-	0,0114	0,0104	2	2
9	40	0,2	-	-	0,0057	0,0063	2	2
10	80	0,9	-	-	0,1303	0,1042	2	2
11	80	0,5	-	-	0,0606	0,0483	2	2
12	80	0,2	-	-	0,0244	0,0236	2	2

maior variação da densidade de cada. Também pretendemos aprimorar o modelo 1 com o uso de propriedades específicas das convexidades para gerar restrições que fortaleçam a formulação.

Referências

Convexity properties of graphs on sagemath. `http://doc.sagemath.org/html/en/reference/graphs/sage/graphs/convexity_properties.html`. Accessed: 2018-03-27.

Araújo, R., Sampaio, R., and Szwarcfiter, J. (2013). The convexity of induced paths of order three. *Electronic Notes in Discrete Mathematics*, 44:109 – 114.

Draque Penso, L., Protti, F., Rautenbach, D., and Souza, U. S. (2014). On p3-convexity of graphs with bounded degree. In Gu, Q., Hell, P., and Yang, B., editors, *Algorithmic Aspects in Information and Management*, pages 263–274, Cham. Springer International Publishing.

Duchet, P. (1988). Convex sets in graphs, ii. minimal path convexity. *Journal of Combinatorial Theory, Series B*, 44(3):307 – 316.

Farber, M. and Jamison, R. E. (1986). Convexity in graphs and hypergraphs. *SIAM J. Algebraic Discrete Methods*, 7:433–444.

α-diperfect digraphs *

M. Sambinelli[1], C. N. da Silva[2], O. Lee[3]

[1]Institute of Mathematics and Statistics – University of São Paulo – Brazil

[2]Department of Computing – Federal University of São Carlos – Brazil

[3]Institute of Computing – University of Campinas – Brazil

Abstract. *Let D be a digraph. A* path partition $\mathcal{P}$ *of D is a collection of paths such that $\{V(P)\colon P \in \mathcal{P}\}$ is a partition of $V(D)$. We say D is* α-diperfect *if for every maximum stable set S of D there exists a path partition $\mathcal{P}$ of D such that $|S \cap V(P)| = 1$ for all $P \in \mathcal{P}$ and this property holds for every induced subdigraph of D. A digraph C is an* anti-directed odd cycle *if (i) the underlying graph of C is a cycle $x_1x_2\cdots x_{2k+1}x_1$, where $k \geq 2$, (ii) the longest path in C has length 2, and (iii) each of the vertices $x_1, x_2, x_3, x_4, x_6, x_8, \ldots, x_{2k}$ is either a source or a sink. Berge (1982) conjectured that a digraph D is α-diperfect if, and only if, D contains no induced anti-directed odd cycle. In this work, we verify this conjecture for digraphs whose underlying graph is series-parallel and for in-semicomplete digraphs.*

1. Introduction

All digraphs considered in this text are finite and contain neither loops nor parallel arcs (but they may contain cycles of length 2). For terminology not defined here, we refer the reader to [Bondy and Murty 2008]. Given a digraph D, we denote its vertex set by $V(D)$ and its arc set by $A(D)$. A pair of vertices $u, v \in V(D)$ is *adjacent* in D if $\{uv, vu\} \cap A(D) \neq \emptyset$. A *stable set* of a digraph D is a set $S \subseteq V(D)$ such that no pair of distinct vertices $u, v \in S$ is adjacent in D. The *stability number* of D, denoted by $\alpha(D)$, is the size of the largest stable set in D. A *path* P in D is a sequence $v_0v_1\cdots v_\ell$ of distinct vertices of D such that $v_iv_{i+1} \in A(D)$ for $i = 0, \ldots, \ell - 1$. A *path partition* $\mathcal{P}$ of D is a collection of paths such that $\{V(P)\colon P \in \mathcal{P}\}$ is a partition of $V(D)$.

In 1960, Gallai and Milgram [Gallai and Milgram 1960] showed that the size of a minimum path partition of a digraph D is at most the stability number of D. Although various proofs of this result were known at the early 80's, no proof implied the existence of a maximum stable set S and a path partition $\mathcal{P}$ such that $|S \cap V(P)| = 1$ for every $P \in \mathcal{P}$ (later, Meyniel [Meyniel 1989] showed that there exist digraphs where such stable set and partition does not exist). Thinking about this matter, Berge [Berge 1982] proposed the class of α-diperfect digraphs. Given a digraph D and a stable set S of D, an *S-path partition* of D is a path partition $\mathcal{P}$ such that $|S \cap V(P)| = 1$ for all $P \in \mathcal{P}$. We say that D satisfies the *α-property* if, for every maximum stable set S of D, there exists an S-path partition of D, and we say that D is *α-diperfect* if every induced subdigraph of D satisfies the α-property.

*M. Sambinelli was partially supported by CNPq (Proc. 141216/2016-6) and FAPESP (Proc. 2017/23623-4). O. Lee was partially supported by CNPq (Proc. 311373/2015-1 and 425340/2016-3) and FAPESP (Proc. 2015/11937-9). E-mails: msambinelli@ic.unicamp.br (M. Sambinelli), candida@ufscar.br (C. N. da Silva), lee@ic.unicamp.br (O. Lee).

Given a digraph D, we denote its underlying graph by $U(D)$ (in this text we always consider that the underlying graph is simple). A digraph C is an *anti-directed odd cycle* if (i) $U(C) = x_1x_2 \cdots x_{2k+1}x_1$ is a cycle, where $k \geq 2$, (ii) the longest path in C has length 2, and (iii) each of the vertices $x_1, x_2, x_3, x_4, x_6, x_8, \ldots, x_{2k}$ is either a source or a sink. Berge [Berge 1982] showed that anti-directed odd cycles do not satisfy the α-property, and hence are not α-diperfect, which led him to conjecture the following characterization for α-diperfect digraphs. Note that it is strikingly similar to Berge's conjecture on perfect graphs – nowadays known as the Strong Perfect Graph Theorem (see Theorem 1 [Chudnovsky et al. 2006]).

Conjecture 1 (Berge, 1982) *A digraph D is α-diperfect if, and only if, D contains no induced anti-directed odd cycle.*

Theorem 1 (Chudnovsky, Robertson, Seymour, and Thomas, 2006) *A graph G is perfect if, and only if, neither G nor its complement contain an induced odd cycle of order at least 5.*

By showing that anti-directed odd cycles do not satisfy the α-property, Berge ended up showing the necessity of Conjecture 1. So the open problem in this conjecture is to verify its sufficiency. Still in his seminal paper, Berge [Berge 1982] showed that digraphs whose underlying graph is perfect and symmetric digraphs are both α-diperfect. To the best of our knowledge these are the only particular cases verified for Conjecture 1. The lack of results for this conjecture and the complexity of the proof of Theorem 1 as well as the time it took to prove it, seem to indicate that this is a very challenging problem.

A graph G is *series-parallel* if it can be obtained from the null graph by applying the following operations repeatedly: (i) adding a vertex v with degree at most one; (ii) adding a loop; (iii) adding a parallel edge; (iv) subdividing an edge. A *clique* of a digraph D is a set $S \subseteq V(D)$ such that every pair of vertices in S are adjacent in D. A digraph D is *semicomplete* if $V(D)$ is a clique, and D is *in-semicomplete* if, for every vertex $v \in V(D)$, the set $\{u\colon uv \in A(D)\}$ is a clique. Note that out-trees, cycles, and semicomplete digraphs are all in-semicomplete digraphs.

Series-parallel graphs are a common start point towards verifying graph theoretical conjectures [Chen et al. 2017, Juvan et al. 1999, Merker 2015] and in-semicomplete digraphs have been well studied in literature [Guo and Volkmann 1994, Bang-Jensen et al. 1997] and have been used to confirm open conjectures on digraphs [Bang-Jensen et al. 2006, Galeana-Sánchez and Gómez 2008]. The contributions of this work are the following theorems.

Theorem 2 *Let D be a digraph containing no induced anti-directed odd cycle. If $U(D)$ is series-parallel, then D is α-diperfect.*

Theorem 3 *If D is an in-semicomplete digraph, then D is α-diperfect.*

2. Outline of the proofs

The proofs of Theorems 2 and 3 use the following auxiliary results.

Lemma 4 *If a digraph D can be partitioned into k induced subdigraphs, say H_1, $H_2, \ldots, H_k$, such that $k \geq 2$, every H_i satisfies the α-property, and $\alpha(D) = \sum_{i=1}^{k} \alpha(H_i)$, then D satisfies the α-property.*

Proof: Let S be a maximum stable set of D and let $S_i = S \cap V(H_i)$ for $i = 1, 2, \ldots, k$. Thus,

$$\alpha(D) = |S| = \sum_{i=1}^{k} |S_i| \leq \sum_{i=1}^{k} \alpha(H_i) = \alpha(D).$$

Hence, S_i is a maximum stable set of H_i, and since the latter satisfies the α-property, there exists an (S_i)-path partition $\mathcal{P}_i$ of H_i, for $i = 1, \ldots, k$. Therefore, $\mathcal{P} = \bigcup_{i=1}^{k} \mathcal{P}_i$ is an S-path partition of D. Since S is an arbitrary maximum stable set of D, the result follows. □

Lemma 5 *If B is a clique cut of a digraph D, then D can be partitioned into two proper induced subdigraphs D_1 and D_2 such that $\alpha(D) = \alpha(D_1) + \alpha(D_2)$. Moreover, if uv is an arc of D such that $u \in V(D_1)$ and $v \in V(D_2)$, then $\{u, v\} \cap B \neq \emptyset$.*

Lemma 6 *Let D be a digraph. If $U(D)$ contains a proper induced cycle containing at most two vertices with degree greater than two, then D can be partitioned into two proper induced subdigraphs D_1 and D_2 such that $\alpha(D) = \alpha(D_1) + \alpha(D_2)$.*

2.1. Outline of the proof of Theorem 2

Towards a contradiction, suppose that the result does not hold, and let D be a counterexample with the smallest number of vertices. It is not hard to check that D has order at least 3. Moreover, since every subgraph of a series-parallel graph is also a series-parallel graph, we have, by the minimality of D, that every proper induced subdigraph of D satisfies the α-property, which means that D does not, since it is a counterexample. Then we show that D can be partitioned into two proper induced subdigraphs D_1 and D_2 such that $\alpha(D) = \alpha(D_1) + \alpha(D_2)$. We prove this result as follows. If D has a cut vertex, then the result follows by Lemma 5. Otherwise, D has no cut vertex, and hence $U(D)$ is 2-connected. We show that $U(D)$ contains an induced cycle C containing at most two vertices with degree greater than 2. If $U(D) = C$, then we show that D satisfies the α-property, a contradiction. Otherwise, C is a proper induced subgraph, and hence the result follows by Lemma 6. Therefore, there exists such partition of D and, by Lemma 4, D satisfies the α-property, a contradiction.

2.2. Outline of the proof of Theorem 3

Towards a contradiction, suppose that the result does not hold and let D be a counterexample with the smallest number of vertices. It is not hard to check that D has order at least 3. Moreover, since every induced subgraph of an in-semicomplete digraph is also an in-semicomplete digraph, we have, by the minimality of D, that every proper induced subdigraph of D satisfies the α-property, which means that D does not, since it is a counterexample. If D is disconnected, then $\alpha(D) = \alpha(C) + \alpha(D - V(C))$, where C is a component of D, and hence, by Lemma 4, D satisfies the α-property, a contradiction. Therefore, we may assume that D is connected. If D is strong, then we use the following result provided in [Bang-Jensen et al. 1993] to show that D satisfies the α-property, a contradiction.

Theorem 7 (Bang-Jensen, Huang, and Prisner, 1993) *An in-semicomplete digraph D of order at least 2 is hamiltonian if, and only if, D is strong.*

Since D is a connected non-strong digraph, there exists a strong component X such that no arc in D is leaving X. Let $Y = \{v \in V(D) \setminus V(X) \colon vu \in A(D) \text{ and } u \in V(X)\}$. By a result of Bang-Jensen and Gutin [Bang-Jensen and Gutin 1998], we have $yx \in A(D)$ for every $y \in Y$ and $x \in V(X)$. Thus, since D is in-semicomplete, Y is a clique. If Y is a cut of D, then, by Lemma 5, D can be partitioned into two proper induced subdigraphs D_1 and D_2 such that $\alpha(D) = \alpha(D_1) + \alpha(D_2)$, and hence, by Lemma 4, D satisfies the α-property, a contradiction. Thus, we may assume that Y is not a vertex cut, and hence $V(D) = Y \cup V(X)$. We note that $u \in Y$ is a vertex adjacent to every vertex of D and prove that if $D - u$ satisfies the α-property, then D also satisfies the α-property, a contradiction.

References

Bang-Jensen, J. r., Guo, Y., Gutin, G., and Volkmann, L. (1997). A classification of locally semicomplete digraphs. *Discrete Math.*, 167/168:101–114. 15th British Combinatorial Conference (Stirling, 1995).

Bang-Jensen, J. r. and Gutin, G. (1998). Generalizations of tournaments: a survey. *J. Graph Theory*, 28(4):171–202.

Bang-Jensen, J. r., Huang, J., and Prisner, E. (1993). In-tournament digraphs. *J. Combin. Theory Ser. B*, 59(2):267–287.

Bang-Jensen, J. r., Nielsen, M. H., and Yeo, A. (2006). Longest path partitions in generalizations of tournaments. *Discrete Math.*, 306(16):1830–1839.

Berge, C. (1982). Diperfect graphs. *Combinatorica*, 2(3):213–222.

Bondy, J. A. and Murty, U. S. R. (2008). *Graph theory*, volume 244 of *Graduate Texts in Mathematics*. Springer, New York.

Chen, G., Ehrenmüller, J., Fernandes, C. G., Heise, C. G., Shan, S., Yang, P., and Yates, A. N. (2017). Nonempty intersection of longest paths in series-parallel graphs. *Discrete Math.*, 340(3):287–304.

Chudnovsky, M., Robertson, N., Seymour, P., and Thomas, R. (2006). The strong perfect graph theorem. *Ann. of Math. (2)*, 164(1):51–229.

Galeana-Sánchez, H. and Gómez, R. (2008). Independent sets and non-augmentable paths in generalizations of tournaments. *Discrete Math.*, 308(12):2460–2472.

Gallai, T. and Milgram, A. N. (1960). Verallgemeinerung eines graphentheoretischen Satzes von Rédei. *Acta Sci. Math. (Szeged)*, 21:181–186.

Guo, Y. and Volkmann, L. (1994). Connectivity properties of locally semicomplete digraphs. *J. Graph Theory*, 18(3):269–280.

Juvan, M., Mohar, B., and Thomas, R. (1999). List edge-colorings of series-parallel graphs. *Electron. J. Combin.*, 6:Research Paper 42, 6 pp. (electronic).

Merker, M. (2015). Decomposing series-parallel graphs into paths of length 3 and triangles. *Electronic Notes in Discrete Mathematics*, 49:367 – 370.

Meyniel, H. (1989). About colorings, stability and paths in directed graphs. *Discrete Math.*, 74(1-2):149–150. Graph colouring and variations.

Sobre Finura Própria de Grafos

Moysés S. Sampaio Jr.[1]*, **Fabiano S. Oliveira**[2]†, **Jayme L. Szwarcfiter**[1,2]‡

[1]COPPE/PESC – Universidade Federal do Rio de Janeiro (UFRJ), Brasil

[2]IME – Universidade do Estado do Rio de Janeiro (UERJ), Brasil

moysessj@cos.ufrj.br, fabiano.oliveira@ime.uerj.br, jayme@nce.ufrj.br

Abstract. *Both graph classes of k-thin and proper k-thin graphs have recently been introduced generalizing interval and unit interval graphs, respectively. The complexity of the recognition of k-thin and proper k-thin are open, even for fixed $k \geq 2$. In this work, we introduce a subclass of the proper 2-thin graphs, called proper 2-thin of precedence. For this class, we present a characterization and an efficient recognition algorithm.*

1. Introdução

A classe dos grafos k-finos foi introduzida em [Mannino et al. 2007] como uma generalização da classe dos grafos de intervalo. Motivados por isto, [Bonomo, Estrada 2017] definiram a classe dos grafos k-finos próprios que, de forma similar, generalizam os grafos de intervalo próprio. As complexidades dos problemas de reconhecer os grafos k-finos e k-finos próprios, mesmo para $k \geq 2$ fixo, encontram-se em aberto. Neste trabalho, estudamos uma subclasse dos grafos k-finos próprios, que chamamos de k-finos próprios de precedência, para a qual apresentamos uma caracterização estrutural e um algoritmo eficiente de reconhecimento.

Um *grafo de intervalo* G é um grafo tal que $V(G)$ é uma família de intervalos fechados da reta real, chamado de *modelo*, tal que $(I, J) \in E(G)$ se, e somente se, $I \cap J \neq \emptyset$. Em [Olariu 1991], mostra-se que um grafo G é de intervalo se, e somente se, existir uma ordenação $<$ de $V(G)$ tal que, para qualquer tripla ordenada (p, q, r) de vértices de G, se $(p, r) \in E(G)$, então $(q, r) \in E(G)$. Tal ordenação $<$ é chamada *canônica* de $V(G)$. A Figura 1(a) ilustra um grafo de intervalo e uma de suas ordens canônicas. Um grafo de intervalo G é dito *próprio* se admite um modelo tal que $I \nsubseteq J$, para todo $I, J \in V(G)$ distintos. Similarmente, G é um grafo de intervalo próprio se, e somente se, admite uma ordenação $<$ de $V(G)$ tal que, para qualquer tripla ordenada (p, q, r) de vértices, se $(p, r) \in E(G)$ então $(p, q), (q, r) \in E(G)$ [Roberts 1969]. Tal ordenação é denominada de *canônica própria*, como ilustrado na Figura 1(b).

Um grafo G é denominado *k-fino* se existir um k-particionamento de $V(G)$ e uma ordenação associada (chamada de *consistente*) tal que, para qualquer tripla ordenada (p, q, r) de $V(G)$, se p e q pertencerem a uma mesma parte e $(p, r) \in E(G)$, então $(q, r) \in E(G)$. Um grafo G é dito *k-fino próprio* se existir uma k-partição de $V(G)$ e uma ordenação associada (chamada de *fortemente consistente*) que é consistente e tal que, para qualquer tripla ordenada (p, q, r) de $V(G)$ se q e r pertencerem a uma mesma

*financiado por CAPES.

†parcialmente financiado por FAPERJ.

‡parcialmente financiado por CNPq.

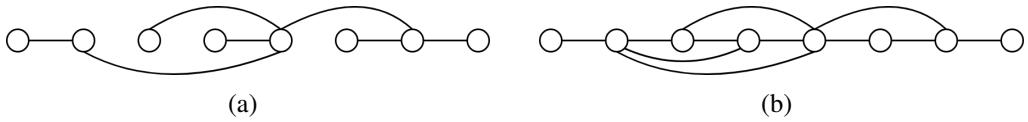


Figura 1. Exemplos de sequências (a) canônicas e (b) canônicas próprias.

parte e $(p, r) \in E(G)$, então $(p, q) \in E(G)$. O valor mínimo de k para o qual G é k-fino (resp. k-fino próprio) é chamado de *finura* (resp. *finura própria*) de G, denotado por $thin(G)$ (resp. $pthin(G)$). Note que, os grafos k-finos são uma generalização dos grafos de intervalo, e que os grafos k-finos próprios generalizam os grafos de intervalo próprios.

Em [Bonomo, Estrada 2017], é apresentado um algoritmo eficiente para determinar o particionamento mínimo de $V(G)$ para o qual uma dada ordenação é consistente, ou fortemente consistente. Além disso, prova-se que o problema de determinar se existe uma ordenação consistente, ou fortemente consistente, dado um particionamento é NP-completo. De uma maneira geral, mesmo para $k \geq 2$ fixo, a complexidade de decidir se um grafo é k-fino ou k-fino próprio encontra-se em aberto. Neste trabalho investigamos o problema para uma classe de grafo mais restrita, definida por admitir uma ordenação fortemente consistente que goza de uma propriedade especial. Mais especificamente, chamaremos um grafo G de 2-*fino próprio de precedência* (2-FPP) se G é um grafo 2-fino próprio que admite bipartição (X, Y) de $V(G)$ e uma ordenação fortemente consistente $<$ na qual todos os vértices de X precedem aqueles de Y em $<$. A Figura 2(a) ilustra um grafo que é 2-FPP e, na Figura 2(c), é ilustrado um grafo que não é 2-FPP com respeito a bipartição escolhida. A convenção da representação destas figuras é a de que os vértices das partes da bipartição (X, Y) associada são dispostos horizontalmente, cada parte em uma horizontal, e a ordem da esquerda para direita é aquela encontrada em uma ordenação fortemente consistente. Além disso, por convenção, os vértices que estão embaixo precedem os que estão em cima na ordenação fortemente consistente.

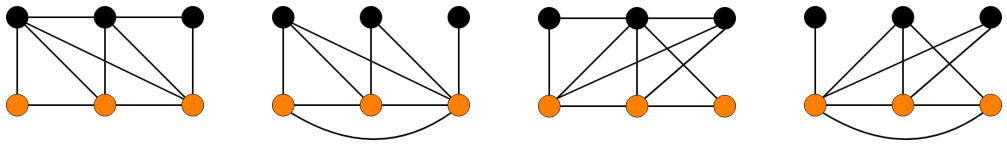


Figura 2. (a)(c) Grafos e bipartição (X, Y) dados; (b)(d) Grafos $S(X, Y)$.

2. Reconhecimento dos Grafos 2-FPP com Dada Bipartição

Nesta seção, apresentaremos um algoritmo eficiente para o seguinte problema:

PROBLEMA:	Reconhecimento dos grafos 2-FPP com bipartição dada
ENTRADA:	Um grafo G e uma bipartição (X, Y) de $V(G)$ com $G[X]$ e $G[Y]$ conexos
QUESTÃO:	Existe uma ordenação $<$ de $V(G)$ consistente com (X, Y) tal que cada vértice de X precede cada vértice de Y em $<$?

Naturalmente, $G[X]$ e $G[Y]$ devem ser grafos de intervalo próprio. Em caso negativo, o algoritmo pode responder NÃO. Em caso positivo, sabe-se que $G[X]$ admite uma única ordenação canônica a menos de reversão e permutação entre os vértices *gêmeos* (vértices

com mesma vizinhança fechada em $G[X]$). O mesmo pode ser dito para $G[Y]$. Sejam s_X e s_Y ordenações canônicas arbitrárias de $G[X]$ e $G[Y]$, respectivamente. Denote por s^{-1} a reversão de uma ordenação s. O algoritmo prossegue para cada $(s_1, s_2) \in (\{s_X, s_X^{-1}\} \times \{s_Y, s_Y^{-1}\})$ verificando se existe uma ordenação s, obtida da concatenação de s_1 com s_2 e de eventual permutação de vértices que são gêmeos em $G[X]$ ou em $G[Y]$, que seja fortemente consistente com (X, Y). Tal verificação é feita da seguinte forma.

O algoritmo utilizará um digrafo D_G associado a G no qual as arestas direcionadas representam ordens forçadas entre os vértices de s. Caso hajam relações de ordem mutuamente contraditórias, o que consiste na aparição de um ciclo em D_G, o par (s_1, s_2) vigente é ignorado e o próximo é considerado. Caso contrário, qualquer ordenação que respeite as relações definidas pelas arestas de D_G será consistente com (X, Y), situação que o algoritmo responderá SIM e retornará uma ordenação topológica de D_G como um certificado dessa resposta. Se um ciclo for encontrado em cada par (s_1, s_2) investigado, o algoritmo responderá NÃO e retornará como certificado da resposta negativa os ciclos encontrados. As regras que forçam ordens relativas entre vértices serão as seguintes:

(i) Seja X' (resp. Y') o conjunto formado pelo último vértice de s_1 e seus gêmeos em $G[X]$ (resp. o conjunto formado pelo primeiro vértice de s_2 e seus gêmeos em $G[Y]$). Para todo $u \in X'$, $v \in Y'$, (u, v) é uma aresta forçada em D_G;

(ii) para todo $u, v \in X$, se u e v não são gêmeos em $G[X]$ e $u < v$ em s_1, então (u, v) é uma aresta forçada em D_G. Analogamente, para todo $u, v \in Y$, se u e v não são gêmeos em $G[Y]$ e $u < v$ em s_2, então (u, v) é uma aresta forçada em D_G;

(iii) para todo $u, v \in X$ e $w \in Y$, se $(u, w) \notin E(G)$ e $(v, w) \in E(G)$, então (u, v) é uma aresta forçada em D_G. Analogamente, para todo $u, v \in Y$ e $w \in X$, se $(u, w) \in E(G)$ e $(v, w) \notin E(G)$, então (u, v) é uma aresta forçada em D_G.

O algoritmo pode ser implementado em tempo $O(n^2)$.

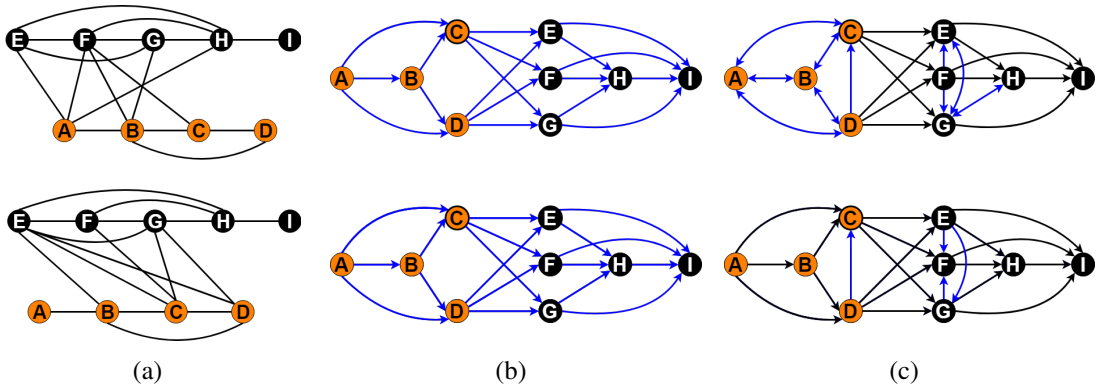


Figura 3. Dois exemplos de execução do algoritmo. (a) grafo e bipartição de entrada; (b) D_G considerando arestas forçadas nos passos (i) e (ii); (c) D_G considerando arestas forçadas no passo (iii).

3. Caracterização dos Grafos 2-FPP com Dada Bipartição

Nesta seção apresentamos uma caracterização dos grafos 2-FPP com dada bipartição. Um grafo G é um grafo *de divisão* se existir uma bipartição (X, Y) de $V(G)$ tal que X seja uma clique e Y um conjunto independente. Um grafo é dito *de limiar* se G é um grafo de

divisão que admite uma ordenação dos vértices de X (resp. Y), chamada de *ordenação de limiar*, de modo que suas vizinhanças estejam ordenadas por inclusão, isto é, se u precede v na ordenação, então $N[u] \subseteq N[v]$ (resp. $N(u) \subseteq N(v)$). Para a caracterização a seguir, definiremos o grafo de divisão $S(X, Y)$ associado à bipartição de entrada (X, Y), obtido de G pela adição e remoção de arestas de modo que X se torne uma clique e Y se torne um conjunto independente. As Figuras 2(b) e 2(d) ilustram os grafos de divisão associados aos grafos das Figuras 2(a) e 2(c), respectivamente.

Teorema 1. *Se G é um grafo e (X, Y) é uma bipartição de $V(G)$, então G é 2-FPP com respeito a (X, Y) se, e somente se, existir ordenações canônicas s_X, s_Y de respectivamente $G[X]$ e $G[Y]$ tais que $S(X, Y)$ é um grafo de limiar para o qual s_X corresponda a uma ordenação de limiar de X e s_Y^{-1} a uma ordenação de limiar de Y.*

Demonstração. Suponha que G é 2-FPP com respeito a bipartição (X, Y) de $V(G)$ e s é uma ordenação consistente própria associada. Seja s_X (resp. s_Y) o prefixo (resp. sufixo) de s correspondente aos vértices de X (resp. Y). Naturalmente, s_X e s_Y são ordens canônicas de $G[X]$ e $G[Y]$, respectivamente. Suponha que s_X não é uma ordenação de limiar de X. Então, existem $u, v \in X$ e $w \in Y$ com $u < v$ em s_X tais que $w \in N[u]$ e $w \notin N[v]$. Mas então $(u, w) \in E(G)$, $(v, w) \notin E(G)$ e $u < v < w$ em s, contrariando s ser uma ordenação fortemente consistente. Suponha agora que s_Y^{-1} não é uma ordenação de limiar para Y, e logo existem $u, v \in Y$ e $w \in X$ com $u < v$ em s_Y tais que $w \in N(v)$ e $w \notin N(u)$. Mas então $(v, w) \in E(G)$, $(u, w) \notin E(G)$ e $w < u < v$ em s, o que contraria s ser uma ordenação fortemente consistente. Por outro lado, considere que s_X e s_Y^{-1} são ordenações de limiar de X e Y e canônicas de $G[X]$ e $G[Y]$. Mostraremos que $s = s_X s_Y$ é uma ordenação fortemente consistente com respeito a bipartição dada (X, Y), seguindo o resultado. Na suposição do contrário, existiria (i) $u, v \in X$, $w \in Y$ com $u < v$ em s tais que $(u, w) \in E(G)$, $(v, w) \notin E(G)$, ou (ii) $u, v \in Y$, $w \in X$ com $u < v$ em s tais que $(v, w) \in E(G)$, $(u, w) \notin E(G)$. No caso (i) (resp. (ii)), há contradição com o fato de s_X (resp. s_Y^{-1}) ser uma ordenação de limiar para X (resp. Y). □

4. Conclusão

Este trabalho foi motivado pelos problemas de determinar a finura, e a finura própria em grafos. A introdução de ambos os parâmetros é um conceito relativamente recente e foi motivada por uma generalização do bem-conhecido problema de reconhecimento de grafos de intervalo e de intervalo unitários. A complexidade de reconhecer se um grafo é k-fino ou k-fino próprio encontra-se em aberto mesmo para $k \geq 2$ fixo. Neste trabalho, introduzimos a classe dos grafos 2-finos próprios de precedência (2-FPP) e apresentamos um algoritmo de reconhecimento polinomial e uma caracterização estrutural da classe.

Referências

Bonomo, F., Estrada, D. (2017). On the thinness and proper thinness of a graph. *CoRR*, abs/1704.00379.

Mannino, C., Oriolo, G., Ricci, F.,, Chandran, S. (2007). The stable set problem and the thinness of a graph. *Operations Research Letters*, 35:1–9.

Olariu, S. (1991). An optimal greedy heuristic to color interval graphs. *Information Processing Letters*, 37:21–25.

Roberts, F. (1969). Indifference graphs. Em Harary, F., editor, *Proof Techniques in Graph Theory*, p. 139–146. Academic Press, New York.

Online Circle and Sphere Packing*

Carla Negri Lintzmayer[1], Flávio Keidi Miyazawa[2], Eduardo Candido Xavier[2]

[1]Center for Mathematics, Computation and Cognition – Federal University of ABC
Av. dos Estados 5001, Santo André, SP, Brazil

[2]Institute of Computing – University of Campinas
Av. Albert Einstein 1251, Campinas, SP, Brazil

carla.negri@ufabc.edu.br,{fkm,eduardo}@ic.unicamp.br

Abstract. *In the Online Circle Packing in Squares, circles arrive one at a time and we need to pack them into the minimum number of unit square bins. We improve the previous best-known competitive ratio for the bounded space version from* 2.439 *to* 2.3536 *and we also give an unbounded space algorithm. Our algorithms also apply to the Online Circle Packing in Isosceles Right Triangles and Online Sphere Packing in Cubes, for which no previous results were known.*

1. Introduction

We consider the Online Circle Packing in Squares, Online Circle Packing in Isosceles Right Triangles, and Online Sphere Packing in Cubes problems. They receive an online sequence of circles/spheres of different radii and the goal is to pack them into the minimum number of unit squares, isosceles right triangles of leg length one, and unit cubes, respectively. By packing we mean that two circles/spheres do not overlap and each one is totally contained in a bin. As applications we can mention origami design, crystallography, error-correcting codes, coverage of a geographical area with cell transmitters, storage of cylindrical barrels, and packaging bottles or cans [Szabó et al. 2007].

In online packing problems, one item arrives at a time and must be promptly packed, without knowledge of further items. Also, after packing an item, it cannot be moved to another bin. At any moment a bin is open or closed: we can only pack items into open bins and, once a bin is closed, it cannot be opened again. The algorithms can have *bounded space*, when the number of open bins is bounded by a constant, or *unbounded space*, when there is no guarantee on this number. Usually, items cannot be reorganized, but our unbounded space algorithms may do this on a constant number of items.

Several works consider items to be squares or rectangles [Christensen et al. 2017], but the Online Circle Packing in Squares was considered only in [Hokama et al. 2016], who showed a lower bound of 2.292 on the competitive ratio of any bounded space algorithm and gave one with asymptotic competitive ratio 2.439. Offline Circle Packing in Squares was considered in [Miyazawa et al. 2016] and there are several results for packing circles or spheres of same radii [Szabó et al. 2007, Tatarevic 2015].

We show bounded and unbounded space algorithms which work for our three problems. The bounded space ones are based on the one given in [Hokama et al. 2016],

*This work was supported by São Paulo Research Foundation (grants 2016/14132-4, 2015/11937-9, 2016/01860-1, 2016/23552-7) and National Counsel of Technological and Scientific Development (grants 306358/2014-0, 311499/2014-7, and 425340/2016-3).

but they have a simpler analysis, we improved the occupation ratio for the class of small circles, we subdivide the bins in a simpler way, and we used another method to find the numerical results. The unbounded space algorithms are a modification of the bounded space ones using an idea presented in [Epstein 2010]. We also give lower bounds on the competitive ratio of any bounded space algorithm for Online Circle Packing in Isosceles Right Triangles and Online Sphere Packing in Cubes. Due to space constraints, in the next section we only describe our algorithms for Online Circle Packing in Squares. The same algorithms, with relatively minor changes work for the other two problems[1].

2. Online Circle Packing in Squares

For an integer M given as a parameter, a circle of radius r is *large* if $r > 2/M$ and it is *small* if $r \leq 2/M$. Large and small circles are packed separately, into *Lbins* and *Sbins*, respectively, which are, nonetheless, unit squares. The only difference between the bounded and unbounded space algorithm is how they pack some of the large circles.

For an integer $i \geq 1$, let ρ_i* be the largest value such that i circles of radius ρ_i* can be packed into a unit square. Let ρ_i be ρ_i*, if $i \leq 30$, or the best-known lower bound for ρ_i*, if $30 < i \leq 9996$ [Specht]. Let $K \in \mathbb{Z}$ be such that $\rho_{K+1} < 2/M \leq \rho_K$. A large circle of radius r has type i, for $1 \leq i < K$, if $\rho_{i+1} < r \leq \rho_i$, or it has type K, if $2/M < r \leq \rho_K$. We denote a large circle of type i as I_i. Let $C \in \mathbb{Z}$ be a parameter. Given a small circle of radius r, we find the largest integer p such that $C^p r \leq 2/M$. We then classify such circle as type (i, p) if $2/(i+1) < C^p r \leq 2/i$, where $M \leq i < CM$.

The hexagonal packing is the densest one for circles of equal radii and we will use it to pack small circles. For each i, with $M \leq i < CM$, an Sbin is of type i if it is used to pack small circles of type (i, p), for any $p \geq 0$. We keep at most one Sbin of type i open at a time. A *q-bin*(i, p) is a square of side length $1/C^{p+1}$. When an Sbin of type i is opened, it is divided into C^2 q-bins$(i, 0)$. A q-bin(i, p) is either subdivided into C^2 q-bins$(i, p+1)$ or it is tiled with hexagons of side $4/(C^p i\sqrt{3})$ to pack small items of type (i, p) (because a circle of radius at most r fits into a hexagon of side length $2r/\sqrt{3}$). Note that we must have $1/C^{p+1} \geq 2/(iC^p)$ for this to work, but since $i < CM$, it suffices to choose $M > 2$.

When a small item of type (i, p) arrives, the algorithms will simply try to pack it into a hexagon of a q-bin(i, p). If there is no such hexagon, they will find an empty q-bin(i, p') (not tiled yet) with the largest p' such that $p' < p$ and subdivide it, until a q-bin(i, p) is found (at which point it will be tiled). Next theorem shows the occupation ratio of a closed Sbin, which is the minimum total area occupied by the circles packed in such bin. Parameter C is chosen to be 5 because it maximizes this ratio.

Theorem 1 *The occupation ratio of a closed Sbin of type i, for $M \leq i < 5M$, is at least* $\frac{551}{600}\left(1 - \frac{43.1}{M} + \frac{18.48}{M^2}\right)\frac{\pi}{\sqrt{12}}\frac{M^2}{(M+1)^2}$.

Now for packing large circles, since $\rho_{9996} < 0.005076143$ and $\rho_i < \rho_{9996}$ for $i < 9996$, we chose $M = 360$, which makes $2/M \leq 0.005555556$. Thus, we can find K such that $\rho_{K+1} < 2/M \leq \rho_K$, and so we can classify all large items. For each i, with $1 \leq i \leq K$, an Lbin is of type i if it packs large circles of type i. At most one Lbin of type i is kept open at a time. When an Lbin of type i is opened, it is divided into i *c-bins*, which are circles of radii ρ_i. When an I_i arrives, it is either packed into an empty c-bin

[1]A full version is available at `https://arxiv.org/abs/1708.08906`.

or the current Lbin of type i (if any) is closed and a new one is opened. Note that at most K Lbins are kept open by the algorithm at any given time. Together with the $(C-1)M$ Sbins, we can see that this algorithm has bounded space. Furthermore, note that each closed Lbin of type i has occupation ratio of at least $i\pi\rho_{i+1}^2$.

The analysis of the competitive ratio uses the weighting method [Epstein 2010], in which we show a weighting function w over the items such that the sum of weights of items in any bin is at least 1, except for a constant number of bins. Afterwards, we find every feasible configuration, i.e., sets of items that can be packed into a bin, calculate their sum of weights, and find the supremum β of such sums. As a result, the asymptotic competitive ratio of the algorithm is bounded from above by β. However, it is not reasonable to list every possible feasible configuration, so we use the following fact. Consider a bin of maximum weight with some known-circles $\mathcal{I}' \subseteq \mathcal{I}$ whose sum of weights is W and sum of areas is A, and let OR be a lower bound on the occupation ratio of the unknown circles of the bin (in $\mathcal{I}\backslash\mathcal{I}'$). If the weighting function is such that the weight of a circle $i \notin \mathcal{I}'$ of area $a(i)$ is at most $a(i)/\mathrm{OR}$, then the total sum of weights in such bin is at most $W + (1-A)/\mathrm{OR}$. With this, we can find only a few configurations and the lower bound on the occupation ratio of circles that are not on them, in order to find an upper bound on the asymptotic competitive ratio of our algorithms. For that, we use a two-phase program. The first phase uses constraint programming to give a set $\mathcal{F}$ of feasible configurations that contain only large circles. Each configuration $\mathcal{I} \in \mathcal{F}$ is associated with an integer $f(\mathcal{I})$ which indicates that all large circles from type 1 to $f(\mathcal{I})$ were tested to be part of $\mathcal{I}$. In the second phase the idea is to add circles of type greater than $f(\mathcal{I})$ in the remaining space of the bin by using a criterion of space: if the circle's area is at most the remaining area of the bin, such circle will be considered. For that we use an integer programming which simulates a knapsack problem. We may create an infeasible configuration, but this is not an obstacle since our goal is to find a configuration with maximum total weight. This two-phase program is used in the proofs of Theorems 2 and 3.

At last, we define $w(I_i) = \frac{1}{i}$, so a closed Lbin of type i has total weight $\sum_{j=1}^{i} \frac{1}{i} = 1$. Let $\mathrm{OR} = \frac{551}{600}\left(1 - \frac{43.1}{M} + \frac{18.48}{M^1}\right)\frac{\pi}{\sqrt{12}}\frac{M^2}{(M+1)^2}$. For a small circle c of type (i,p) and area $a(c)$, we define $w(c) = \frac{a(c)}{\mathrm{OR}}$, so a closed Sbin B of type i has total weight $\sum_{c\in B}\frac{a(c)}{\mathrm{OR}} = \frac{1}{\mathrm{OR}}a(B) \geq 1$, where $a(B)$ is the sum of areas of items in B and, according to Theorem 1, $a(B) \geq \mathrm{OR}$. Theorem 2 concludes with the asymptotic competitive ratio of the algorithm.

Theorem 2 *The algorithm for packing circles in squares with bounded space has asymptotic competitive ratio strictly below* 2.3536.

Packing large circles in the unbounded space algorithm follows an idea of [Epstein 2010]. A *waiting bin* packs either one I_1 with one I_2 (the I_1 at the left bottom corner and the I_2 at the right top corner) or one I_1 with two I_4 (the I_1 centered to the left border with one I_4 at the right bottom corner and the other at the right top corner), if their radius are related with a value D. Other circles are packed as before. The reorganization of packed items is allowed only for I_1s that are inside open waiting bins. The algorithm has unbounded space because we cannot guarantee how many waiting bins are open.

If an I_1 of radius $r > D$ arrives, then we pack it as in the bounded space algorithm (one per bin). Otherwise, $r \leq D$ and we pack it in an already opened waiting bin containing either one I_2 or at least one I_4, if one exists, or we open a new one to pack

the I_1 and let it open waiting for an I_2 or two I_4. This last case is why we allow the reorganization of circles I_1 inside waiting bins. Let $\gamma = \sqrt{2}/(\sqrt{2}+1) - D$. If an I_2 of radius $r > \gamma$ arrives, then we pack it as in the bounded space algorithm (two per bin). When $r \leq \gamma$, the circles are labeled according to their arrival so that the following steps can be repeated at every 72 of them. If the I_2 is among the first 70, then it is packed as in the bounded space algorithm (two per bin); if it is one of the last 2, then it is packed in a waiting bin. Let $\lambda = 3/2 - \sqrt{2D+1}$. If an I_4 of radius $r > \lambda$ arrives, then we pack it as in the bounded space algorithm (four per bin). When $r \leq \lambda$, the circles are labeled according to their arrival so that the following steps are repeated at every 34 of them. If the I_4 is among the first 32, then we pack four per bin; if it is one of the last 2, then it is packed in a waiting bin. Values of γ and λ were chosen through simple algebraic expressions written considering the desired configurations of waiting bins. Since we need $\rho_2 < D \leq \rho_1$, $\rho_3 < \gamma \leq \rho_2$, and $\rho_5 < \lambda \leq \rho_4$ for them to be possible, and this is true for $\rho_2 < D < 0.331553$, we fixed $D = 0.325309$.

To analyse the competitive ratio of this algorithm we use a generalized weighting method [Epstein 2010]. We need to show weighting functions w_1 and w_2 whose sum of weights of items in any bin is at least 1 on average for at least one of the functions, except for a constant number of bins. The supremum β of the sums of weights of feasible configurations, for both functions, is an upper bound on the asymptotic competitive ratio.

When the algorithm ends, either there are open waiting bins with I_1, in which case we apply w_1 over all circles, or there are not, in which case we apply w_2. They differ from w only regarding I_1, I_2, and I_4 if their radii are at most D, γ, and λ, respectively. If I_1 has radius $r \leq D$, then $w_1(I_1) = 1$ and $w_2(I_1) = 0$. If I_2 has radius $r \leq \gamma$, then $w_1(I_2) = \frac{35}{72}$ and $w_2(I_2) = \frac{37}{72}$. If I_4 has radius $r \leq \lambda$, then $w_1(I_4) = \frac{8}{34}$ and $w_2(I_4) = \frac{9}{34}$. For both functions, the sum of weights in any bin is at least 1 if we do not consider I_1, I_2, and I_4, because this is true for w. Now consider w_1 was applied over such items. All waiting bins have total weight at least 1 because they contain an I_1. For every set of 72 I_2, we have 70 of them packed into 35 bins and the last 2 packed into 2 waiting bins with one I_1 each. Thus, the average weight of these bins is $\left(35\left(2\frac{35}{72}\right) + 2\left(1 + \frac{35}{72}\right)\right)/37 = 1$. This is similar for I_4 and w_2. Theorem 3 concludes our result.

Theorem 3 *The algorithm for packing circles in squares with unbounded space has asymptotic competitive ratio strictly below* 2.3105.

References

Christensen, H. I., Khan, A., Pokutta, S., and Tetali, P. (2017). Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79.

Epstein, L. (2010). Two-dimensional Online Bin Packing with Rotation. *Theoretical Computer Science*, 411(31):2899–2911.

Hokama, P., Miyazawa, F. K., and Schouery, R. C. S. (2016). A Bounded Space Algorithm for Online Circle Packing. *Information Processing Letters*, 116(5):337–342.

Miyazawa, F. K., Pedrosa, L. L. C., Schouery, R. C. S., Sviridenko, M., and Wakabayashi, Y. (2016). Polynomial-Time Approximation Schemes for Circle and Other Packing Problems. *Algorithmica*, 76(2):536–568.

Specht, E. Packomania. `http://www.packomania.com/`. Accessed: 2018-03-27.

Szabó, P. G., Markót, M. C., Csendes, T., Specht, E., Casado, L. G., and García, I. (2007). *New Approaches to Circle Packing in a Square*. Springer Optimization and Its Applications. Springer US, New York, USA.

Tatarevic, M. (2015). On Limits of Dense Packing of Equal Spheres in a Cube. *The Electronic Journal of Combinatorics*, 22(1):35.

Sobre as funções racionais multi-sequenciais

Rodrigo de Souza[1*]

[1]Departamento de Computação – Universidade Federal Rural de Pernambuco (UFRPE)

rodrigo.npmsouza@ufrpe.br

Abstract. *Multi-sequential functions were introduced by Choffrut and Schützenberger as the family of rational functions whose graph is a finite union of sequential functions. Recently, Jeckert and Filiot have shown that it is decidable in polynomial time if a rational function is multi-sequential. In this communication we study the rational functions that are a union of k sequential functions, for a fixed integer k. We present three characterizations of these functions, which are related to classical results of the theory of rational relations.*

Resumo. *Funções multi-sequenciais foram introduzidas por Choffrut e Schützenberger como a família das funções racionais cujo gráfico é uma união finita de funções sequenciais. Recentemente, Jeckert e Filiot mostraram que é decidível em tempo polinomial se uma função racional é multi-sequencial. Nesta comunicação estudamos as funções que são uma união de k funções sequenciais, para um inteiro fixo k. Apresentamos três caracterizações dessas funções, que recordam resultados clássicos sobre as relações racionais.*

1. Introdução

A elucidação da estrutura de famílias de objetos realizados por autômatos – linguagens, séries, relações – decompondo-os em objetos mais elementares é uma das linhas de estudo mais tradicionais na área da Teoria dos Autômatos. Por exemplo, no tratado clássico de Samuel Eilenberg [Eilenberg 1974], um capítulo é dedicado à *estrutura dos conjuntos reconhecíveis*, e um algoritmo de decomposição dessas linguagens é apresentado.

Nossa contribuição filia-se a essa linha ao considerar a estrutura das funções entre palavras realizadas por autômatos com entrada e saída – os transdutores. Um *transdutor*[1] é uma extensão do modelo de autômato, no qual as transições são rotuladas por um par de palavras, uma dita *de entrada* e outra *de saída*. Ao invés de reconhecer linguagens, os transdutores realizam relações entre monóides livres A^* e B^*; essas relações são precisamente as *relações racionais* de A^* a B^* (= subconjuntos do produto cartesiano $A^* \times B^*$ que podem ser obtidos com aplicações das operações de união, produto e estrela a partir de unitários). Uma *função racional* é uma relação racional que é uma função parcial.[2]

Em diversos aspectos, relações racionais são objetos muito mais intrincados do que as linguagens reconhecíveis.[3] Por outro lado, as funções racionais, que estudamos aqui, exibem propriedades notáveis e por isso ganharam muita atenção na literatura.

*Este trabalho é apoiado pelo projeto *Problemas estruturais em modelos formais de Computação*, Edital MCTI/CNPQ/Universal 14/2014 (Processo 459957/2014-7).

[1]Por limitação de espaço, não apresentamos neste resumo definições e notações básicas sobre autômatos e transdutores. Esse material pode ser consultado no livro de Jacques Sakarovitch [Sakarovitch 2009].

[2]Ao longo do texto, A e B aparecerão livremente e representam dois alfabetos.

[3]Por exemplo, a equivalência não é decidível para relações racionais.

O problema abordado nesta comunicação refere-se ao conceito de determinismo em transdutores. Contrariamente aos autômatos sobre um monóide livre, nem todo transdutor funcional (= realiza uma função) pode ser determinizado. A caracterização das funções racionais que podem ser realizadas por um *transdutor sequencial*, ou seja, determinístico na entrada – são as chamadas *funções sequenciais* – é um resultado profundo, obtido por Choffrut em 1979 [Choffrut 1979, Lombardy and Sakarovitch 2006].

Em 1986, Choffrut e Schützenberger consideraram um relaxamento da sequencialidade: mesmo que uma função racional não seja sequencial, pode-se escrevê-la como uma união finita de funções sequenciais? Chamamos de *multi-sequencial* uma tal função.[4] O principal resultado apresentado em [Choffrut and Schützenberger 1986] é uma caracterização estrutural dos transdutores que realizam uma função multi-sequencial. Os problemas de decidir se uma função racional é multi-sequencial e de decompor uma tal função em uma união de funções sequenciais foram considerados por Jecker e Filiot [Jecker and Filiot 2015], com um algoritmo polinomial para o primeiro problema.

Nesta comunicação, consideramos uma versão parametrizada, por um inteiro $k > 0$, da questão introduzida por Choffrut e Schützenberger: que funções racionais podem ser escritas como uma união de k funções sequenciais? Chamamos tais funções de k-*sequenciais*. A semelhança com os resultados mencionados é apenas aparente: os trabalhos de Choffrut, Schützenberger, Jeckert e Filiot tratam de investigar a existência de uma decomposição; aqui, deseja-se uma resposta mais refinada, ao fixar-se um tamanho para a decomposição. Essa diferença pode surpreendentemente ser posta em paralelo com dois problemas de decisão clássicos para transdutores: *decidir se um transdutor é k-valorado, para um dado inteiro k* (ou seja, se a imagem de toda palavra no domínio é um conjunto de no máximo k palavras); *decidir se é finitamente valorado* (se existe um tal k). Ambos os problemas podem ser decididos em complexidade polinomial, o que demonstramos com J. Sakarovitch através de construções estruturais com transdutores [Sakarovitch and de Souza 2008]. Mas as construções empregadas são muito diferentes.

Nossa contribuição consiste de três caracterizações das funções k-sequenciais, que referem-se a facetas distintas desses objetos. Cada uma relaciona-se com um resultado clássico sobre transdutores: a primeira, Teorema 3.1, recorda a supracitada caracterização, de sabor topológico, de Choffrut das funções sequenciais (Teorema 2.1); a segunda, Teorema 3.2, também faz referência ao trabalho de Choffrut (Teorema 2.2), mas é mais estrutural, ao lidar com propriedades de passeios em transdutores; finalmente, a terceira, Teorema 3.3, evoca uma propriedade de passeios de transdutores k-valorados implícita em [Weber 1996] e retrabalhada em [Sakarovitch and de Souza 2010] (Teorema 2.3). Por restrições de espaço, limitamo-nos a expor, com um mínimo de notação, os enunciados dos três resultados clássicos referidos, e, em seguida, enunciar nossos resultados. Concluímos com um comentário sobre consequências dos mesmos a serem investigadas.

2. Funções sequenciais e relações k-valoradas: propriedades clássicas

A principal notação envolvida nos resultados que vamos expor é uma métrica sobre um monóide livre; essa função associa a todo par de palavras $u, v \in A^*$ a soma dos comprimentos dos sufixos respectivos após retirar-se o maior prefixo comum entre u e v.

[4] A função que associa toda palavra da forma $u_1x_1\#u_2x_2\#\dots u_kx_k\#$ a $x_1u_1\#x_2u_2\#\dots x_ku_k\#$, onde, para todo i, $u_i \in \{a,b\}^*$ e $x_i \in \{a,b\}$, é um exemplo de função racional que não é multi-sequencial.

Podemos definir essa métrica formalmente com o *grupo livre* sobre A: $||u,\, v|| = |u^{-1}v|$.

Definição 2.1 (Função lipschitziana) *Uma função $f : A^* \to B^*$ é* lipschitziana *se existe um inteiro $L \geq 0$ tal que, para todo u e v no domínio de f, $||uf,\, vf|| \leq L||u,\, v||$.*

Não é difícil demonstrar que toda função sequencial é lipschitziana. A substância da caracterização de Choffrut é a recíproca, cuja demonstração é intrincada:

Teorema 2.1 (Choffrut 1978) *Uma função racional é sequencial sse é lipschitziana.*

Uma caracterização mais estrutural e efetiva das funções sequenciais, também presente no trabalho de Choffrut, pode ser expressa através de propriedades do produto cartesiano de um transdutor $\mathcal{T}$ por ele mesmo: em $\mathcal{T}^2$, um passeio representa simultaneamente dois passeios de $\mathcal{T}$ com a mesma entrada, em $\mathcal{T}^3$, três passeios, etc. Veja detalhes desse produto em [Sakarovitch and de Souza 2010]. Precisamos da seguinte definição:

Definição 2.2 (Passeio (i, j)-conjugado) *Seja c um passeio de $\mathcal{T}^{k+1}$ de um estado inicial $(p_1, p_2, \ldots, p_{k+1})$ ao estado $(q_1, q_2, \ldots, q_{k+1})$; sejam $p_i \xrightarrow{u|u_1} q_i$ e $p_j \xrightarrow{u|u_2} q_j$ as projeções de c nos índices i e j, respectivamente. Dizemos que c é* (i, j)-conjugado *se, para todo circuito d que começa e termina em $(q_1, q_2, \ldots, q_{k+1})$, as projeções de d nos índices i e j, $q_i \xrightarrow{v|v_1} q_i$ e $q_j \xrightarrow{v|v_2} q_j$, satisfazem $u_1^{-1}u_2 = v_1^{-1}u_1^{-1}u_2v_2$.*

A caracterização estrutural de Choffrut é a base de algoritmos para decidir se uma função racional é sequencial (detalhes podem ser vistos em [Béal et al. 2003]):

Teorema 2.2 (Choffrut 1978) *Um transdutor funcional $\mathcal{T}$ realiza uma função sequencial sse todo passeio de $\mathcal{T}^2$ que começa em um estado inicial é $(1, 2)$-conjugado.*

Relativamente às relações racionais k-valoradas, a propriedade que nos interessa envolve o conceito de deslocamento entre passeios c e d com a mesma entrada e que começam em um estado inicial: considerando-se todos os prefixos de c e d de mesmo comprimento, o deslocamento é o máximo entre as diferenças das saídas desses prefixos.

Definição 2.3 (Deslocamento) *Em um transdutor $\mathcal{T}$, o* deslocamento *entre passeios c e d com a mesma entrada, e que começam em um estado inicial, é o inteiro $\langle c,\, d\rangle = \max\left\{||x,\, y|| \;\middle|\; p \xrightarrow{u|x} q \text{ prefixo de } c,\ r \xrightarrow{u|y} s \text{ prefixo de } d\right\}$.*

Se um transdutor $\mathcal{T}$ é k-valorado, não somente toda tupla de $k + 1$ passeios bem-sucedidos com a mesma entrada tem pelo menos um par com a mesma saída (isso é uma paráfrase da definição), mas necessariamente existe um par cujo deslocamento é limitado:

Teorema 2.3 (Weber 1996) *Um transdutor $\mathcal{T}$ é k-valorado se, e somente se, existe um inteiro L tal que, para todo passeio bem-sucedido c de $\mathcal{T}^{k+1}$, existem duas coordenadas i e j tais que as projeções c_i et c_j tem a mesma saída e satisfazem $\langle c_i,\, c_j\rangle < L$.*

3. Contribuições

Teorema 3.1 *Uma função racional $f : A^* \to B^*$ é k-sequencial se, e somente se, existe um inteiro L tal que todo conjunto $u_1, u_2, \ldots, u_{k+1}$ de palavras no domínio de f admite índices i e j tais que $||u_if,\, u_jf|| \leq L||u_i,\, u_j||$.*

Teorema 3.2 *Um transdutor funcional $\mathcal{T}$ realiza uma função k-sequencial se, e somente se, todo passeio c de $\mathcal{T}^{k+1}$, que começa em um estado inicial, satisfaz: para todo prefixo c' de c, existe pelo menos um par i, j de índices tais que c' é (i, j)-conjugado.*

Teorema 3.3 *Um transdutor funcional $\mathcal{T}$ realiza uma função k-sequencial se, e somente se, existe um inteiro L tal que, para todo passeio c de $\mathcal{T}^{k+1}$ que começa em um estado inicial, existem coordenadas i e j tais que as projeções c_i e c_j satisfazem $\langle c_i,\ c_j \rangle < L$.*

4. Comentários

As demonstrações dos teoremas 3.1, 3.2 e 3.3 são relacionadas. Por exemplo, no Teorema 3.2, a existência de um passeio que não satisfaz as condições enunciadas permite a construção de um conjunto de palavras que não satisfaz as condições do Teorema 3.1; a existência de um passeio que não satisfaz as condições do Teorema 3.3 permite construir um passeio que não satisfaz as condições do Teorema 3.2.

A sequência natural de nossos resultados é a investigação de um algoritmo para decidir se uma função é k-sequencial com base no Teorema 3.2 e de uma decomposição de uma função k-sequencial em k transdutores sequenciais partindo do Teorema 3.3.

Por fim, agradecemos aos revisores anônimos pela leitura cuidadosa da versão preliminar deste texto, que nos permitiu melhorá-lo substancialmente.

Referências

Béal, M. P., Carton, O., Prieur, C., and Sakarovitch, J. (2003). Squaring transducers: an efficient procedure for deciding functionality and sequentiality. *Theoretical Computer Science*, 292(1):45–63.

Choffrut, C. (1979). A generalization of Ginsburg and Rose's characterization of gsm mappings. In *ICALP'79*, volume 71 of *Lecture Notes in Computer Science*, pages 88–103. Springer-Verlag.

Choffrut, C. and Schützenberger, M. P. (1986). Décomposition de fonctions rationnelles. In *STACS 86*, volume 210 of *Lecture Notes in Computer Science*, pages 213–226. Springer.

Eilenberg, S. (1974). *Automata, Languages, and Machines*. Academic Press, Inc.

Jecker, I. and Filiot, E. (2015). Multi-sequential word relations. In *DLT 2015*, volume 9168 of *Lecture Notes in Computer Science*, pages 288–299. Springer.

Lombardy, S. and Sakarovitch, J. (2006). Sequential? *Theoretical Computer Science*, 356(1-2):224–244.

Sakarovitch, J. (2009). *Elements of Automata Theory*. Cambridge University Press.

Sakarovitch, J. and de Souza, R. (2008). On the decidability of bounded valuedness for transducers. In *MFCS 2008*, volume 5162 of *Lecture Notes in Computer Science*, pages 588–600. Springer.

Sakarovitch, J. and de Souza, R. (2010). Lexicographic decomposition of k-valued transducers. *Theory of Computing Systems*, 47(3):758–785.

Weber, A. (1996). Decomposing A *k*-valued transducer into *k* unambiguous ones. *ITA*, 30(5):379–413.

Expected Emergent Algorithmic Creativity and Integration in Dynamic Complex Networks

Felipe S. Abrahão, Klaus Wehmuth, Artur Ziviani

[1]National Laboratory for Scientific Computing (LNCC)
Av. Getúlio Vargas, 333, Quitandinha – CEP 25651-075 – Petrópolis, RJ – Brazil

fsa@lncc.br, klaus@lncc.br, ziviani@lncc.br

Abstract. *We present a theoretical investigation of the emergence of complexity or irreducible information in networked computable systems when the network topology may change over time. For this purpose, we build a network model in which nodes are randomly generated Turing machines that obey a communication protocol of imitation of the fittest neighbor. Then, we show that there are topological conditions that trigger a phase transition in which eventually these networked computable systems begin to produce an unlimited amount of bits of expected emergent algorithmic complexity, creativity and integration as the network size goes to infinity.*

1. Introduction

We present a theoretical investigation of systemic properties in networked complex systems, focusing on the emergence of complexity or irreducible information when computable systems are connected by communication channels. For this purpose, we define a general mathematical model based on computability theory, information theory, and graph theory. This model enables one to represent and build definitions as well as theorems on networked computable systems where the set of nodes (or vertices) is defined as a population of Turing machines (or programs) and edges or arrows (i.e., connections or communication channels) are defined by a MultiAspect Graph (MAG) [Wehmuth et al. 2016]. We call this model as *algorithmic networks* [Abrahão 2016, Abrahão et al. 2017].

As pointed and exemplified in [Abrahão et al. 2017], the motivations for our definitions and problems as well as the range of our intended applications of the present study belongs to an intersection of complex networks, distributed computing, evolutionary game theory, algorithmic and statistical information theory, theory of computation, complex systems theory. Thus, the present work is developed as an interdisciplinary theoretical investigation. In order to start tackling this broad subject, we narrow our scope to study a toy model and the general problem of how much more algorithmic complexity, creativity or integration a complex network of randomly generated computable systems produces when playing a game of optimizing the average fitness (or payoff) of the population through diffusion. Given certain topologies and information-sharing (or communication) protocols we aim at mathematically investigating the expected amount of emergent irreducible information. Comparing the algorithmic complexity of networked nodes with the algorithmic complexity of isolated nodes gives a direct formal way to measure this amount of emergent irreducible information. In this work, we will show that the results in [Abrahão et al. 2017] can be directly extended in a way such that there is also a lower

bound for the amount of expected emergent algorithmic creativity or (behavior) integration.

Therefore, in order to be able to build definitions and prove theorems, we study these properties of complex systems by simplifying or making an abstraction of the intricate and vast set of different behaviors of networked complex systems. As discussed in [Abrahão et al. 2017], the present work achieves results that may be related to questions ranging, for instance, from the problem of symbiosis, cooperation, and integration to biological, economic, and social networks. Moreover, it follows the pursuit of a universal framework for the problem of complexity in complex networks as supported by [Barabási et al. 1999, Barabasi 2009]. While it is conjectured in network science that different types of real-world networks are related by graph properties (e.g. a small diameter compared to the network size or the small-world phenomenon), the theoretical approach we are developing suggests that these relations may be sound. Our results go in the direction of answering the problem of why a network topology and an information-sharing (or communication) protocol become relevant from an emergent open-ended evolutionary point of view that takes into account synergy, complexity, irreducible information, and computational power in solving problems.

2. Model

We defined in [Abrahão et al. 2017] a particular model of algorithmic networks with a randomly generated population of nodes (i.e., Turing machines) that plays the Busy Beaver Imitation Game (BBIG). This is a class of synchronous dynamic algorithmic networks $\mathfrak{N}_{BB}$ that is based on an information-sharing protocol of the simple imitation of the fittest neighbor. A network Busy Beaver Game is a general game in which each player attempts to calculate the largest integer it can using the information shared by its neighbors. In this sense, the BBIG is a special case of the network Busy Beaver Game in which every node only propagates or diffuses the largest integer, taking into account the one produced by itself and the ones from their neighbors.

Thus, as presented in [Abrahão et al. 2017], these algorithmic networks $\mathfrak{N}_{BB}$ can be seen as playing an optimization procedure where the whole pursues the synergistic increase of the average fitness through diffusing over the network the best randomly generated solution to a problem. This is so, assuming that this problem is Turing equivalent to the halting problem and assuming the Busy Beaver function as a measure of fitness. Moreover, the network topology may vary over time, which comes from the formalism of Time-Varying Graphs (TVG) as in [Costa et al. 2015]. While optimization through selection in a random sampling may refer to evolutionary computation or genetic algorithms, for instance, in which the best solution eventually appears and remains sufficiently stable over time, in our model optimization is obtained in a manner that the best solution also eventually appears, but it is diffused over time in order to make every individual as averagely closer to the best solution as they can.

3. Results

We have proved in [Abrahão et al. 2017] that there is a lower bound for the expected emergent algorithmic complexity in algorithmic networks $\mathfrak{N}_{BB}$. It depends on how much larger is the average diffusion density in a given time interval compared with the cycle-bounded conditional halting probability. This lower bound also depends on the parameter

for which one is calculating the number of cycles. In fact, we have proved a corollary showing that this parameter can be calculated, for example, from the cover time, as defined in [Costa et al. 2015]. Thus, our results hold even in the case of spending a computably larger number of cycles compared to the cover time. Furthermore, we have proved that there are asymptotic conditions on the increasing power of the cover time as a function of the population size such that they ensure that there is a central time to trigger *expected emergent open-endedness* [Abrahão et al. 2017]. That is, there is a central time to allow communication between nodes with the purpose of generating unlimited expected emergent algorithmic complexity with the least amount of cycles (i.e., communication rounds).

We have also made a small modification on the family of Time-Varying Graphs of $\mathfrak{N}_{BB}$ with the purpose of investigating what would happen if the networks had a relative small-diameter. We have replaced the cover time with the temporal diffusion diameter (i.e., the number of time intervals to reach every node from any node)—or the classical diameter (i.e., the maximum shortest path length) in the static case—in the definition of this family of graphs. Indeed, we have proved in [Abrahão et al. 2017] that, in this case, a small diameter (i.e., dominated by $\mathbf{O}(\lg(N))$, where N is the network size) is sufficient for existing a central time to trigger expected emergent open-endedness.

These proofs stems from the main idea of combining an estimation of a lower bound for the average algorithmic complexity/information of a networked node and an estimation of the expected algorithmic complexity/information of an isolated node. While the estimation of the former comes from the very BBIG dynamics through diffusion of the biggest sent partial outputs, the estimation of the latter comes from the strong law of large numbers, Gibb's inequality, and algorithmic information theory applied to the randomly generated population of Turing machines.

As indicated in [Abrahão 2016] for static algorithmic networks only, if one defines the emergent creativity of a node as

$$I_A(\mathbf{U}(p^b_{net}(o_i, c))|\mathbf{U}(p_{iso}(o_i, c)),$$

where $I_A(y|x)$ is the conditional (prefix) algorithmic complexity/information (program-size complexity, Kolmogorov complexity, or Solomonoff-Kolmogorov-Chaitin complexity) necessary to return the string y as output given the string x as input, then our results also hold for replacing the expected emergent algorithmic complexity/information with expected emergent algorithmic creativity. Thus, there would also be an emergent open-endedness on creativity from the studied conditions, and not only on algorithmic complexity. To this end, since we were estimating lower bounds for the expected emergent algorithmic complexity/information in [Abrahão et al. 2017], note that from algorithmic information theory we have that

$$I_A(\mathbf{U}(p^b_{net}(o_i, c))|\mathbf{U}(p_{iso}(o_i, c)) \geq I_A(\mathbf{U}(p^b_{net}(o_i, c)) - I_A(\mathbf{U}(p_{iso}(o_i, c)) + \mathbf{O}\big(1\big).$$

Therefore, $I_A(\mathbf{U}(p^b_{net}(o_i, c))|\mathbf{U}(p_{iso}(o_i, c))$ defines the amount of algorithmic information necessary to produce the same output that an isolated node would do if networked. In other words, it gives a formal measure on how much more algorithmically creative a node becomes when networked compared with when isolated.

Additionally, if one defines the measure of how much integrated with the network the behavior of node is as $Int(o_i, c)$, where $Int(o_i, c)$ is the information that one would need to actually inform as partial inputs in order to make an isolated node o_i behaves like it was networked during c cycles, then it directly follows from algorithmic information theory that

$$Int(o_i, c) + \mathbf{O}\big(1\big) \geq I_A(\mathbf{U}(p^b_{net}(o_i, c))|\mathbf{U}(p_{iso}(o_i, c)).$$

Thus, our results in [Abrahão et al. 2017] also hold for replacing the expected emergent algorithmic complexity/information with expected (behavior) integration of nodes.

4. Conclusions

We have shown in [Abrahão et al. 2017] that there are topological conditions that trigger a phase transition in which, eventually, algorithmic networks $\mathfrak{N}_{BB}$ begin to produce an unlimited amount of bits of expected emergent algorithmic complexity as the population size goes to infinity. In this work, we have also shown that the same statement holds for expected emergent algorithmic creativity and for expected emergent (behavior) integration. These topological conditions come from a positive trade-off between the average diffusion density and the number of cycles. Therefore, the diffusion power of a dynamic network has proven to be paramount with the purpose of optimizing the average fitness of an algorithmic network that plays the Busy Beaver Imitation Game (BBIG). Further, this diffusion power may come either from the cover time, as in [Costa et al. 2015], or from a small diameter compared to the network size.

Acknowledgements

This work is partially funded by CAPES, CNPq, FAPESP, and FAPERJ.

References

Abrahão, F. S. (2016). Emergent algorithmic creativity on networked Turing machines. In *The 8th International Workshop on Guided Self-Organization at the Fifteenth International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*, Cancún.

Abrahão, F. S., Wehmuth, K., and Ziviani, A. (2017). Algorithmic Networks: central time to trigger expected emergent open-endedness.

Barabasi, A.-L. (2009). Scale-Free Networks: A Decade and Beyond. *Science*, 325(5939):412–413.

Barabási, A.-L., Albert, R., and Jeong, H. (1999). Mean-field theory for scale-free random networks. *Physica A: Statistical Mechanics and its Applications*, 272(1-2):173–187.

Costa, E. C., Vieira, A. B., Wehmuth, K., Ziviani, A., and da Silva, A. P. C. (2015). Time Centrality in Dynamic Complex Networks. *Advances in Complex Systems*, 18(07n08).

Wehmuth, K., Fleury, É., and Ziviani, A. (2016). On MultiAspect graphs. *Theoretical Computer Science*, 651:50–61.

Jogos de Transporte Sequenciais*

Francisco J. M. Silva[1], **Flávio K. Miyazawa**[1], **Rafael C. S. Schouery**[1]

[1]Instituto de Computação - Universidade Estadual de Campinas (UNICAMP)
Av. Albert Einstein, 1251 - 13083-852 - Campinas - SP - Brasil

francisco.silva@students.ic.unicamp.br, {fkm,rafael}@ic.unicamp.br

Abstract. *In this paper we analyze a transportation game where all players want to be transported to a common destination as quickly as possible, and in order to achieve this goal, they have to choose one of the available buses. We provide bounds for the Sequential Price of Anarchy considering two social functions in both metric and non-metric instances.*

Resumo. *Neste artigo, consideramos um jogo de transporte onde todos os jogadores querem ser transportados a um destino em comum o mais rápido possível, e para isso eles devem escolher um dentre os ônibus disponíveis. Apresentamos limitantes para o Preço da Anarquia Sequencial considerando duas funções sociais, para ambas instâncias métricas e não-métricas.*

Introdução

Problemas relacionados com meios de transporte são comumente encontrados na área de Otimização Combinatória, como, por exemplo, o problema do caixeiro viajante e o de roteamento de veículos. Recentemente, [Fotakis et al. 2017] propuseram e analisaram um problema de transporte sob a perspectiva da teoria de jogos algorítmica no qual exibem resultados quanto à existência e à ineficiência de equilíbrios puros de Nash. A seguir, apresentamos o jogo proposto por eles, o qual também será objeto dos nossos resultados.

Um jogo de transporte Γ é uma tupla (N, M, G), onde N é um conjunto de n jogadores; M é um conjunto de $m \geq 2$ ônibus; $G = (V, E)$ é um grafo completo não-direcionado com um vértice de partida s e um vértice de destino t onde cada aresta $e \in E$ possui uma distância $d_e \in \mathbb{R}_+$, e $V = N \cup \{s, t\}$. Se todas as distâncias d_e obedecerem a desigualdade triangular, dizemos que G é métrico. Cada jogador está localizado em um vértice de G e eles possuem o objetivo de serem transportados de suas localizações ao vértice t com o menor custo possível. Todos os ônibus estão inicialmente em s e devem seguir um caminho até o vértice t e consideramos que, assim como [Fotakis et al. 2017], o percurso de cada ônibus $j \in M$, é determinado por uma permutação $\pi_j : \{1, \ldots, n\} \to N$.

Um perfil de estratégia em Γ é uma atribuição $\sigma : N \to M$ na qual um jogador i escolhe um ônibus j que irá buscá-lo, e chamamos de $\mathcal{S}$ o conjunto de todos estes perfis. Considerando um perfil de estratégia $\sigma \in \mathcal{S}$, o custo do jogador sob este perfil $c_i(\sigma)$ é definido com a distância percorrida por σ_i, o ônibus escolhido por i no perfil σ, entre a localização de i e o destino t. Observe que as permutações π são independentes de

*Pesquisa financiada pela FAPESP Proc. 2015/11937-9, 2016/01860-1, 2016/23552-7, 2017/05223-9, CNPQ Proc. 311499/2014-7, 425340/2016-3, 148027/2016-4, 308689/2017-8.

qualquer perfil $\sigma \in \mathcal{S}$. Também consideramos que um ônibus j visita apenas os jogadores que o escolheram para realizar o trajeto, ou seja, j realizará $shortcuts$ (atalhos) em seu trajeto sempre que possível.

Um **Equilíbrio (Puro) de Nash** (PNE) em Γ é um perfil de estratégia σ onde nenhum jogador pode de forma unilateral reduzir seu custo escolhendo um ônibus diferente. Uma forma de avaliar a ineficiência de equilíbrios é o Preço da Anarquia. Considere uma função f representando a função social de um jogo $\mathcal{G}$, e seja $\mathrm{PNE}(\mathcal{G})$ o conjunto de todos os PNEs de $\mathcal{G}$. O **Preço da Anarquia** (PoA) é definido como $\mathrm{PoA}(f, \mathcal{G}) = \frac{\max_{\sigma \in PNE(\mathcal{G})} f(\sigma)}{\min_{\sigma^* \in \mathcal{S}} f(\sigma^*)}$, onde σ^* representa um perfil ótimo.

A seguir, apresentamos as duas funções sociais definidas por [Fotakis et al. 2017]. A primeira é relacionada com a distância total percorrida pelos ônibus, o que reflete o impacto ambiental do resultado do jogo. Considere um perfil de estratégia σ e para $j \in M$, seja $n_j = |\{i : \sigma_i = j\}|$ o número de jogadores que escolheram o ônibus j em σ e $(j_1, \ldots, j_{n_j})$ a ordenação em que os jogadores serão buscados pelo ônibus j. Então, $D(\sigma) = \sum_{j=1}^{m} \sum_{i=1}^{n_j} d(j_i, j_{i+1})$, onde $j_{n_j+1} = t$ para todo ônibus j. A segunda função social é chamada de custo igualitário $E(\sigma)$ e representa a maior distância percorrida por um único ônibus, definida como $E(\sigma) = \max_{i \in N} c_i(\sigma)$. Note que as funções não consideram a distância percorrida entre s e o primeiro passageiro.

Jogos de Transporte Sequenciais

Motivado pelo arcabouço dado por [Leme et al. 2012], nós analisamos o jogo de transporte em sua versão sequencial. Neste cenário os jogadores tomam suas decisões sequencialmente, por exemplo, do jogador 1 ao n, e essa decisão é final. Portanto, quando estiver decidindo em que ônibus viajar, um jogador i terá conhecimento somente das decisões de seus predecessores. Para isso, consideramos o jogo em sua forma extensiva onde a estrutura do jogo é representada por uma árvore m-ária T, onde em cada nível i ($1 \leq i \leq n$) o jogador i é responsável por tomar as decisões de todos os nós pertencentes ao seu nível, ou seja, escolher um dos m ônibus tendo como informação somente todas as escolhas possíveis dos seus $i - 1$ predecessores. Nas folhas estão as informações dos custos de cada jogador.

Um subjogo é um jogo induzido pela restrição de T a um nó i e a seus descendentes, e um perfil de estratégia $s = (s_1, \ldots, s_n)$ é um **Equilíbrio Perfeito de Subjogo** (SPE) se e somente se ele é simultaneamente um equilíbrio de Nash para todos os subjogos (subárvores) do jogo quando restrito à s. Temos que um SPE sempre existe, ele é um equilíbrio puro de Nash do jogo em sua versão simultânea e também pode ser encontrado através de indução retrógrada. Referimos o leitor a [Nisan et al. 2007] para mais sobre jogos na forma extensiva.

[Leme et al. 2012] também introduziram a noção de Preço da Anarquia Sequencial o qual corresponde à pior relação entre o custo de um SPE e o custo de um resultado social ótimo relacionado à alguma função social. Dado uma função c representando o custo social de um jogo $\mathcal{G}$, seja $\mathrm{SPE}(\mathcal{G})$ o conjunto de todos os SPE de $\mathcal{G}$. O **Preço da Anarquia Sequencial** (SPoA) é definido como $\mathrm{SPoA}(c, \mathcal{G}) = \max_{A \in \mathrm{SPE}(\mathcal{G})} \frac{c(A)}{c^*}$, onde c^* representa o custo de um resultado social ótimo. Na próxima seção apresentamos nossos resultados sobre os valores do SPoA relacionados a ambas as funções sociais D e E para instâncias métricas e não-métricas do jogo de transporte sequencial.

Resultados

Instâncias não-métricas

Seguindo os resultados de [Fotakis et al. 2017] que mostram que o valor do Preço da Anarquia (PoA) é ilimitado para instâncias não-métricas do jogo de transporte na sua versão simultânea, mostramos que o valor do SPoA também é ilimitado considerando ambas as funções sociais D e E quando lidando com instâncias não-métricas, mesmo que todas as permutações dos ônibus sejam iguais.

Proposição 1 *O* SPoA *é ilimitado para* D *e* E *se as distâncias não forem métricas, mesmo que todas as permutações dos ônibus sejam iguais.*

Prova. (*esboço*) Considere a instância exibida na Fig. 1 com $n = m = 2$ e π sendo igual à permutação identidade, ou seja, $\pi_{b_1} = \pi_{b_2} = (1, 2)$. Na Fig. 2 podemos observar o jogo em sua forma extensiva, onde as folhas representam o custo dos jogadores 1 e 2 respectivamente. Observe que o jogador 2 sempre terá custo 1 não importando em qual ônibus ele decida viajar, então um possível SPE é aquele em que o jogador 2 sempre escolhe o mesmo ônibus escolhido pelo jogador 1 e portanto o jogador 1 terá custo $M + 1$. Temos que para D, o ótimo social vale 2 e o SPE vale $M+2$, enquanto que para E o ótimo social vale 1 e o SPE vale $M + 1$. Analisando o valor do SPoA para ambas as funções observa-se que para um M suficientemente grande ($M \to \infty$), temos que $\mathrm{SPoA} = \infty$. □

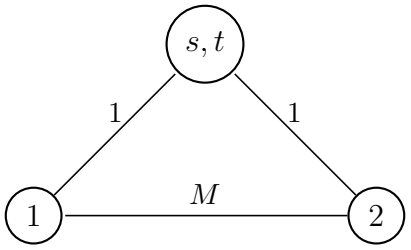


Figura 1. Instância não-métrica com $n = 2$ jogadores.

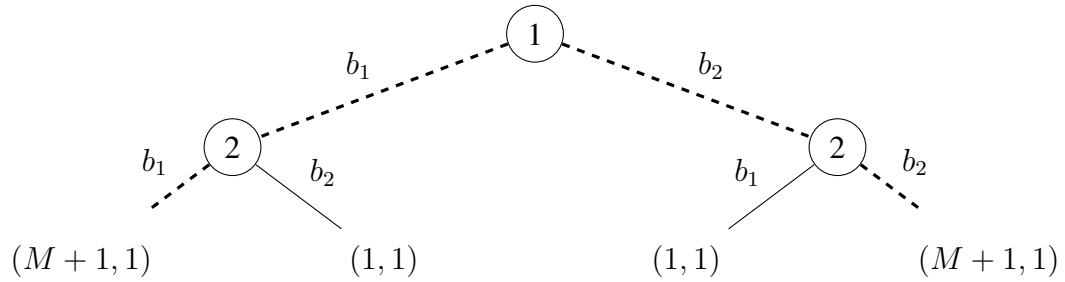


Figura 2. Jogo em sua forma extensiva.

Função D com instâncias métricas

A seguir mostramos que o valor do SPoA para instâncias métricas em relação à função social D é n. Esse valor é igual ao valor do PoA [Fotakis et al. 2017].

Teorema 3.1 *Em jogos de transporte métricos e sequenciais com* n *jogadores, com função social* D*,* $\mathrm{SPoA} = n$*.*

Prova. O limitante superior vem de [Fotakis et al. 2017] onde temos que $D(\sigma) \leq nD(\sigma^*)$ para qualquer perfil σ e um perfil ótimo σ^*. Para o limitante inferior, considere uma instância (N, M, G) onde $|N| = n$, $M = \{b_1, \ldots, b_n\}$, e um grafo G tal

que $d(s,u) = d(u,t) = 1$ para todo $u \in N$ e $d(u,v) = 0$ para todo $u, v \in N$. Considere que cada π_j, para $j \in M$, é uma permutação qualquer. É possível verificar que o único perfil ótimo σ^* é aquela em que todos os jogadores estão viajando juntos em um único ônibus, e portanto o $D(\sigma^*)$ é 1.

Agora basta mostrarmos que estas instâncias possuem um SPE com valor n, como por exemplo, uma solução onde cada ônibus está sendo usado por um único jogador. Observe que um jogador i sempre terá um custo de 1 não importando o que os outros jogadores escolherem. Consequentemente, considere o perfil σ onde o jogador i escolhe o ônibus b_i. Em σ o custo de cada jogador i é 1 e em qualquer outro ônibus o custo seria 1 também. Portanto, σ é um SPE com valor n. □

Função E com instâncias métricas

Em contraste com os resultados sobre os valores do PoA relacionados com a função E apresentados em [Fotakis et al. 2017] ($\text{PoA} = 2\lceil \frac{n}{m} \rceil - 1$ para $n > m$ e $\text{PoA} = 1$ se $n \leq m$), o próximo teorema mostra que o valor do SPoA é pior do que o encontrado no jogo em sua versão simultânea mesmo quando $n = m$.

Teorema 3.2 *Em jogos de transporte métricos e sequenciais com n jogadores, com função social E,* $\text{SPoA} = 2n - 1$.

Prova. (*esboço*) Seja σ^* um perfil ótimo, o maior valor que uma solução pode alcançar é quando todos os jogadores escolhem viajar em um único ônibus, e argumentamos que este é um limitante superior para o SPoA de $(2n-1)E(\sigma^*)$. Esse valor é obtido através de um resultado auxiliar de [Fotakis et al. 2017] o qual afirma que $d(u,v) \leq 2E(\sigma^*)$ para todos os pares $(u,v) \in N$ e $d(u,t) \leq E(\sigma^*)$ para todos $u \in N$. Então, uma vez que todos os jogadores estão em um único ônibus, temos que o caminho que será percorrido pelo ônibus contém somente uma aresta conectada a t e as $(n-1)$ arestas restantes são utilizadas para buscar todos os jogadores. Portanto, esse caminho vale no máximo $(n-1)2E(\sigma^*) + E(\sigma^*) = (2n-1)E(\sigma^*)$.

Para o limitante inferior, mostramos uma família de instâncias com um SPE com valor igual ao limitante superior. Estas instâncias, dadas por (N, M, G) onde $|N| = |M| = n$, e um grafo G onde $d(s,u) = d(u,t) = 1$ para todos $u \in N$ e $d(u,v) = 2$ para todos $u, v \in N$. Considere que π_j, para $j \in M$, seja a permutação identidade, ou seja, $\pi_j = (1, \ldots, n)$. Observe que o perfil ótimo σ^* é aquele onde cada ônibus está sendo usado por um único jogador, e portanto temos que $E(\sigma^*) = 1$. Pode-se mostrar, através de indução retrógrada, que existe pelo menos um SPE nestas instâncias onde todos os jogadores escolhem o mesmo ônibus, e com isso teremos que $\text{SPoA} \geq 2n - 1$. □

Referências

Fotakis, D., Gourvès, L., and Monnot, J. (2017). Selfish transportation games. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 176–187. Springer.

Leme, R. P., Syrgkanis, V., and Tardos, É. (2012). The curse of simultaneity. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 60–67. ACM.

Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V. (2007). *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA.

Um Limite Superior para a Complexidade do ShellSort

Raquel M. Souza[1]*, **Fabiano S. Oliveira**[1]†, **Paulo E. D. Pinto**[1]†

[1]Universidade do Estado do Rio de Janeiro, Brasil

raquelmarcolino25@gmail.com, {fabiano.oliveira,pauloedp}@ime.uerj.br

Abstract. *The worst-case time complexity of the ShellSort algorithm is known only for some specific sequences (a sequence is a parameter of the algorithm). Relating the algorithm to the Frobenius number concept, we present an algorithm for determining the maximum number of comparisons for any sequence and array to be ordered. We apply this method together with the empirical determination of complexity to analyze several sequences whose worst case complexity are known. We show that the empirical approach succeeded in determining the same complexities which are analytically known and presented its results for sequences with unknown worst-case time complexity.*

1. Introdução

O algoritmo ShellSort é um algoritmo de ordenação por comparação criado por [Shell 1959] com o objetivo de ter complexidade de pior caso de tempo ótima de $\Theta(n \log n)$, como aquela do MergeSort, e de fazer uso de espaço auxiliar constante. O algoritmo utiliza uma sequência de inteiros, chamados de *passos*, que influencia na complexidade do algoritmo. A complexidade de pior caso para a sequência de Shell não atingiu tal objetivo ([Frank 1960] mostraram posteriormente que o pior caso resultante da sequência de Shell é de $\Theta(n^2)$), mas abriu caminho para que diversos trabalhos estudassem outras sequências de passos, que resultaram em complexidades de pior caso de tempo tais como $\Theta(n^{3/2})$, $\Theta(n \log^2 n)$ e $O(n^{4/3})$ [Frank 1960, Pratt 1972, Sedgewick 1986].

Embora a relação entre o algoritmo ShellSort e o número de Frobenius já tenha sido abordada [Incerpi 1983], este trabalho apresenta uma nova relação. Tal relação conduz a um algoritmo para determinar um limite superior para o número de comparações numa execução do ShellSort dados uma sequência de passos e um vetor. Com tal limite, é realizado um estudo empírico a fim de encontrar complexidades de pior caso para sequências clássicas, algumas cuja complexidade de pior caso são desconhecidas.

2. ShellSort

ShellSort é um algoritmo de ordenação por comparação [Shell 1959] de um vetor V de n elementos que executa o algoritmo InsertionSort em diferentes subsequências de V. Tais subsequências são definidas por uma *sequência de passos* $p_1, p_2, \ldots, p_k$ com $p_1 = 1$, $p_i < p_{i+1}$ para todo $1 \leq i < k$ e $p_k < n$. O ShellSort é descrito pelo Algoritmo 1, no qual InsertionSort(V, S) representa a ordenação por inserção da subsequência de V definida pela sequência de índices S. Diversos trabalhos estudaram a complexidade de diferentes sequências, sumarizadas na Tabela 1.

O menor limite superior conhecido que independa de sequência é dado a seguir.

*Projeto financiado por CAPES.

†Projeto parcialmente financiado por FAPERJ.

Entrada: $V[1..n]$ de inteiros, sequência de passos $p_1, p_2, \ldots, p_k$
Resultado: V ordenado
para $j \leftarrow k$ **até** 1 **incremento** -1 **faça**
  para $i \leftarrow 1$ **até** p_j **faça**
    InsertionSort(V, $S_{i,j}$), com $S_{i,j} = i, i + p_j, i + 2p_j, \ldots, i + \left\lfloor \frac{n-i}{p_j} \right\rfloor p_j$

Algoritmo 1: Algoritmo do ShellSort.

Tabela 1: Complexidade analítica para algumas sequências de passos.

[Autor Ano]	**Passos** $\{p_1, \ldots, p_k\}$	**Pior caso**
[Shell 1959]	$\{\lfloor n/2^i \rfloor : 0 < i \leq \lfloor \log_2 n \rfloor\}$	$\Theta(n^2)$
[Frank 1960]	$\{2 \cdot \lfloor n/2^i \rfloor + 1 : 0 < i \leq \lfloor \log_2 n \rfloor\}$	$\Theta(n^{3/2})$
[Hibbard 1963]	$\{2^i - 1 : 0 < i \leq \lfloor \log_2 n \rfloor\}$	$\Theta(n^{3/2})$
[Papernov 1965]	$\{1\} \cup \{2^{i-1} + 1 : 1 < i \leq \lfloor \log_2 n \rfloor\}$	$\Theta(n^{3/2})$
[Pratt 1972]	$\{q = 2^r 3^s : r, s \in \mathbb{N} \mid q < n\}$	$\Theta(n \log^2 n)$
[Knuth 1973]	$\{q = (3^i - 1)/2 : i \in \mathbb{N} \mid q \leq \lceil n/3 \rceil\}$	$\Theta(n^{3/2})$
[Sedgewick 1986]	$\{q = 4^i + 3 \cdot 2^{i-1} + 1 : 1 < i \in \mathbb{N} \mid q < n\}$	$O(n^{4/3})$
[Gonnet 1991]	$\{\lfloor (0,45454)^i \cdot n \rfloor : 0 < i \leq \lfloor \log_{2,2} n \rfloor\}$	em aberto
[Tokuda 1992]	$\{q = \lceil (9^i - 4^i)/(5 \cdot 4^{i-1}) \rceil : 1 < i \in \mathbb{N} \mid q < n\}$	em aberto

Teorema 1. *A complexidade de pior caso do algoritmo ShellSort é* $O(n^2 \log \log n)$ *para qualquer constante* c *e sequência com* $O(\log^c n)$ *passos.*

3. Relação com Número de Frobenius

Considere o conjunto $A = \{a_1, a_2, \ldots, a_n\} \subset \mathbb{N}$, $|A| > 1$, tal que $a_1, a_2, \ldots, a_n > 1$ e MDC$(a_1, a_2, \ldots, a_n) = 1$. Seja $F(A) \subset \mathbb{N}$ tal que, para cada elemento $f \in F(A)$, não existem $k_1, k_2, \ldots, k_n \in \mathbb{N}$ de modo que $k_1 a_1 + k_2 a_2 + \ldots + k_n a_n = f$, isto é, f não pode ser representado como uma combinação linear dos elementos do conjunto A. O *número de Frobenius*, denotado por $g(A)$, é o maior número de F. O problema de determinar $g(A)$ possui solução algébrica para $|A| = 2$, a saber, $g(\{a_1, a_2\}) = a_1 a_2 - a_1 - a_2$, mas sua solução algébrica é um problema em aberto quando $n \geq 3$. Contudo, elaboramos um algoritmo pseudo-polinomial como uma variação do algoritmo da mochila que é eficiente na prática. Tal algoritmo determina o número de Frobenius dado um limite superior $t \in \mathbb{N}$ para $g(A)$. Mais especificamente, dados $A = \{a_1, a_2, \ldots, a_n\}$ e $t \in \mathbb{N}$, o algoritmo determina $g(A, t) = \max\{f \in F(A) : f \leq t\}$, que é igual a $g(A)$ quando $t \geq g(A)$. No contexto do ShellSort, este algoritmo pode ser estendido para determinar também $g_m(A, t) = \max\{f \in F(A) : f \leq t, f \bmod m = 0\}$. Durante o processamento do passo p_s, pode ser mostrado que o número máximo de comparações para cada elemento do vetor neste passo é de $g_{p_s}(\{p_k, p_{k-1}, \ldots, p_{s+1}\}, n)$. Tal demonstração se baseia em mostrar que não ocorrem comparações de elementos cuja diferença de posições seja acima deste valor, pois os elementos já estariam ordenados entre si no processamento de algum dos passos anteriores. Baseado nestas ideias, o Teorema 2 permite estabelecer um limite superior de comparações de um elemento arbitrário para qualquer sequência utilizada. Este resultado é utilizado na elaboração do Algoritmo 2.

Teorema 2. *Durante a ordenação do vetor* V *pelo ShellSort, o número de comparações realizadas na iteração relativa ao passo* p_s *para cada elemento de* V *é, no máximo,* $\lfloor g_{p_s}(\{p_k, p_{k-1}, \ldots, p_{s+1}\}, n)/p_s \rfloor + 1$.

Demonstração (Esboço). Mostra-se que o limite superior do intervalo percorrido por um elemento do vetor na ordenação com a iteração relativa ao passo p_s é $g_{p_s}(\{p_k, p_{k-1}, \ldots, p_{s+1}\}, n)$. Para percorrer tal intervalo são necessárias $\lfloor g_{p_s}(\{p_k, p_{k-1}, \ldots, p_{s+1}\}, n)/p_s \rfloor + 1$ comparações. □

4. Algoritmo de Limite Superior para o Número de Comparações

Com base no Teorema 2, é possível enunciar o Algoritmo 2, que determina o número máximo de comparações de qualquer sequência do ShellSort.

Entrada: Sequência de passos $p_1, p_2, \ldots, p_k$ e inteiro n
Resultado: Número máximo de comparações realizadas pelo ShellSort com n elementos e com a sequência de passos $p_1, p_2, \ldots, p_k$
ncomp $\leftarrow 0$
para $s \leftarrow k$ **até** 1 **incremento** -1 **faça**
  limite $\leftarrow \lfloor g_{p_s}(\{p_k, p_{k-1}, \ldots, p_{s+1}\})/p_s \rfloor + 1$
  para $i \leftarrow 1$ **até** n **faça**
    limite$_i \leftarrow \lfloor (n-i)/p_s \rfloor$
    ncomp $\leftarrow$ ncomp + $\min\{$limite, limite$_i\}$
retorna *ncomp*

Algoritmo 2: Determinação do número máximo de comparações do ShellSort.

O algoritmo determina tal limite com o suporte do Teorema 2, observando-se que o limite dado pelo teorema é comparado com o tamanho dos subvetores a serem ordenados por InsertionSort. Em alguns casos, o limite dado pelo teorema pode ser maior que o número de elementos, e contabiliza-se o menor entre os dois. Como resultado, o algoritmo computa um limite superior para a complexidade de pior caso, em número de comparações, dados uma sequência de passos e o número n de elementos do vetor a ser ordenado. Este algoritmo é utilizado com o método empírico de se escolher diversos valores de n e de se determinar o limite superior para tais valores pelo Algoritmo 2. O resultado é analisado então com a ferramenta EMA [Oliveira 2016] para estimar a complexidade assintótica correspondente a tal conjunto de pontos. Apesar de não haver uma garantia de que o pior caso assim obtido coincida com os resultados da literatura, este foi o observado nos experimentos (ver Seção 5).

5. Resultados e Conclusão

A Tabela 2 mostra que os limites superiores para as complexidades retornados pelo algoritmo são justos para as sequências cujas complexidades de pior caso são conhecidas. Além disso, para a sequência [Sedgewick 1986], para a qual a complexidade de pior caso exata é desconhecida, o limite encontrado pelo método proposto neste trabalho coincide com aquele analítico. Portanto, os limites conhecidos analíticos foram todos obtidos através do método empírico apresentado. Para as sequências de complexidade analítica desconhecidas, apresentamos os resultados desta abordagem. Finalmente, observamos que o melhor limite superior analítico para sequências gerais, dado pelo Teorema 1, não é justo para nenhuma sequência específica, enquanto que a abordagem empírica obteve limites justos para todas elas.

Tabela 2: Tabela de complexidades retornadas pelo EMA para cada sequência.

[Autor Ano]	**Literatura**	**Empírico**
[Shell 1959]	$\Theta(n^2)$	$O(n^2)$
[Frank 1960]	$\Theta(n^{1,5})$	$O(n^{1,5})$
[Hibbard 1963]	$\Theta(n^{1,5})$	$O(n^{1,5})$
[Papernov 1965]	$\Theta(n^{1,5})$	$O(n^{1,5})$
[Pratt 1972]	$\Theta(n \log^2 n)$	$O(n \log^2 n)$
[Knuth 1973]	$\Theta(n^{1,5})$	$O(n^{1,5})$
[Sedgewick 1986]	$O(n^{1,333...})$	$O(n^{1,329})$
[Gonnet 1991]	em aberto	$O(n^{1,145})$, $O(n \log^3 n)$
[Tokuda 1992]	em aberto	$O(n^{1,255})$, $O(n \log^{5,5} n)$

Referências

Frank, R. M. Lazarus, R. B. (1960). A high-speed sorting procedure. *Communications of the ACM*, 3(1):20–22.

Gonnet, G. H. Baeza-Yates, R. A. (1991). *Handbook of Algorithms and Data Structures: in Pascal and C*. Addison-Wesley.

Hibbard, T. N. (1963). An empirical study of minimal storage sorting. *Communications of the ACM*, 6(5):206–213.

Incerpi, J. Sedgewick, R. (1983). Improved upper bounds on shellsort. *Anais do 24º Annual Symposium on Foundations of Computer Science*, p. 48–55.

Knuth, D. E. (1973). *The Art of Computer Programming 3: Sorting and Searching*. Addison-Wesley.

Oliveira, F. S. (2016). EMA - WebPage. `http://fabianooliveira.ime.uerj.br/ema`. [Último acesso: 10 de Março de 2018].

Papernov, A. A. Stasevich, G. V. (1965). A method of information sorting in computer memories. *Problemy Peredachi Informatsii*, 1(3):81–98.

Pratt, V. R. (1972). Shellsort and sorting networks. Relatório técnico, Documento DTIC.

Sedgewick, R. (1986). A new upper bound for shellsort. *Journal of Algorithms*, 7(2):159–173.

Shell, D. L. (1959). A high-speed sorting procedure. *Communications of the ACM*, 2(7):30–32.

Tokuda, N. (1992). An improved shellsort. *Anais do 12º IFIP World Computer Congress on Algorithms, Software, Architecture-Information Processing*, 1:449–457.

Uma estratégia para selecionar vértices como candidatos a roteadores em uma árvore de Steiner

João Guilherme Martinez[1], Rosiane de Freitas[1], Altigran da Silva[1], Fábio Protti[2]

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Manaus – AM – Brasil

[2]Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói – RJ – Brasil

{joaogam,rosiane,alti}@icomp.ufam.edu.br, fabio@ic.uff.br

Abstract. *The Steiner tree optimization problem in graphs is NP-hard, however, algorithms that use heuristics obtain results close to optimal solution in polynomial time. In this paper we present a new algorithm based on an exact enumerative algorithm from literature. The proposed heuristic selects vertices as candidates to be routers in the Steiner tree.*

Resumo. *O problema da árvore de Steiner em grafos é NP-difícil, no entanto, algoritmos que fazem uso de heurísticas conseguem obter resultados próximos da solução ótima em tempo polinomial. Neste trabalho apresentamos um novo algoritmo baseado em um algoritmo exato enumerativo da literatura. A heurística proposta seleciona vértices como candidatos a serem roteadores na árvore de Steiner.*

1. Introdução

Considere um grafo $G = (V(G), E(G))$ onde $V(G)$ representa o conjunto de vértices do grafo e $E(G)$ o conjunto de suas arestas. Dado um subconjunto W de vértices de $V(G)$, uma árvore de Steiner consiste num subgrafo conexo T de G que contêm W e cuja soma dos pesos das arestas de $E(T)$ seja mínimo. Os vértices do conjunto W são chamados de terminais e os demais vértices de $V(G)$ utilizados para construir T são chamados de vértices de Steiner. A versão de decisão do problema da árvore de Steiner em grafos é NP-Completo [Karp 1972], portanto não possui algoritmos polinomiais que encontrem a solução ótima, entretanto existem algoritmos na literatura que fazem uso de heurísticas e alcançam soluções próximas da ótima. O problema da árvore de Steiner possui aplicações práticas como o desenho de layouts de circuitos e o projeto de redes, além de outras variações do problema que surgem no campo prático.

Considere n como o número de vértices em $V(G)$, m como o número de arestas em $E(G)$ e k como o número de vértices terminais do conjunto W. A Tabela 1 apresenta alguns trabalhos relacionados publicados na literatura com a complexidade do algoritmo no pior caso e a razão de aproximação das soluções obtidas em relação à solução ótima, sendo alguns destes aproximativos ([Takahashi and Matsuyama 1980],[Wu et al. 1986],[Kou et al. 1981],[Zelikovsky 1993], [Robins and Zelikovsky 2000]) e outros exatos ([Dreyfus and Wagner 1971], [Vygen 2011],[Dourado et al. 2014],[Byrka et al. 2013],[Pajor et al. 2014]).

Título	Autores	Ano	Complexidade	Razão de Aproximação
The Steiner problem in graphs	Dreyfus e Wagner	1971	$O(n^3 + n^2 2^{k-1} + n3^{k-1})$	1
An approximate solution for the steiner problem in graphs	Takahashi e Matsuyama	1980	$O(kn^2)$	2
A fast algorithm for Steiner trees	Kou, Markowsky e Berman	1981	$O(kn^2)$	$2(1 - 1/l)$[a]
A faster approximation algorithm for the Steiner problem in graphs	Wu, Widmayer e Wong	1986	$O(m \log n)$	$2(1 - 1/l)$[a]
An 11/6-approximation algorithm for the network steiner problem	Zelikovsky	1993	$O(nm + k^4)$	1.833
Improved Steiner Tree Approximation in Graphs	Robins e Zelikovsky	2000	$O(kn^2)$	1.55
Faster algorithm for optimum Steiner trees	Vygen	2011	$O(nk2^{k+\log_2 k \log_2 n})$	1
Steiner Tree Approximation via Iterative Randomized Rounding	Byrka, Grandoni, Rothvoss e Sanità	2013	indefinido	1.38
Algorithmic aspects of Steiner convexity and enumeration of Steiner trees	Dourado, Oliveira e Protti	2014	$O(n^2(n+m) + n^{k-2} + n\alpha)$[b]	1
A Robust and Scalable Algorithm for the Steiner Problem in Graphs	Pajor, Uchoa e Werneck	2018	$O(m(min\{nk, m\}))$	indefinido

Tabela 1. Algoritmos conhecidos na literatura para árvore de Steiner.

Na seção 2 são mostrados os conceitos preliminares para o funcionamento do algoritmo, o pseudo-código e a complexidade no pior caso. Na seção 3 são apresentados os resultados dos testes computacionais e na seção 4 são feitas as considerações finais.

2. Descrição do Algoritmo

O trabalho de Dourado et al. (2014) propõe um algoritmo exato enumerativo para obter as árvores de Steiner de um dado grafo em tempo exponencial para o número de terminais k. Nos conceitos preliminares, os autores explicam que os vértices de Steiner podem ser de dois tipos: *linkers* e roteadores. Os *linkers* são vértices de grau igual a 2 e os roteadores devem possuir sempre grau maior que 2. A partir disso, provam que em uma árvore de Steiner, sempre existe no máximo somente $k-2$ roteadores e que todo roteador ou terminal possui um caminho que é mínimo em G até outro roteador ou terminal. Os vértices que formam esses caminhos são os *linkers*, por isso sempre possuem grau igual a 2.

No algoritmo proposto pelos autores, são explorados todos os possíveis subconjuntos de roteadores que fazem parte da solução ótima, portanto consideram todos os subconjuntos com 0 a $k-2$ vértices como roteadores. Neste ponto o algoritmo torna-se exponencial na ordem de $O(n^{k-2})$.

A heurística proposta neste trabalho consiste em analisar todos os vértices não-terminais e usá-los como roteadores, um por vez, e analisar aquele que melhora a qualidade da última melhor solução obtida. No entanto esse procedimento só é executado até no máximo $k-2$ vezes, pois esta é a quantidade máxima de roteadores em uma árvore de Steiner mínima ótima. O Algoritmo 1 mostra o pseudo-código do algoritmo heurístico proposto.

[a] l representa o número de folhas em uma árvore de Steiner ótima.

[b] α representa o número de árvores de Steiner ótimas enumeradas.

Assim como no algoritmo exato [Dourado et al. 2014], a ideia geral consiste em utilizar uma matriz d_G com as distâncias dos caminhos mínimos entre todos os vértices do grafo G e construir templates, que são árvores geradoras mínimas de grafos completos G_{RW} formados com os terminais e os roteadores, utilizando as distâncias em d_G para os pesos das arestas.

A complexidade geral do algoritmo fica definida a seguir: para encontrar as distâncias d_G é $O(n^3)$. A iteração principal do algoritmo é da ordem de $O(nk^3)$ devido ao algoritmo para gerar uma árvore geradora mínima no grafo completo G_{RW}. Portanto a complexidade total do algoritmo é $O(n^3 + nk^3)$.

Algoritmo 1: Heurística para árvore de Steiner

Entrada: Grafo G, conjunto W
Saída: Árvore de Steiner T
início
  $melhor \leftarrow \infty$;
  $d_G \leftarrow$ todas as menores distâncias dos caminhos mínimos entre todos os vértices;
  $G_{RW} \leftarrow$ grafo completo somente com os vértices de W usando as distâncias de d_G;
  se $|AGM(G_{RW})| < melhor$ **então**
    $melhor \leftarrow |\text{AGM}(G_{RW})|$;
    $T \leftarrow \text{AGM}(G_{RW})$;
  fim
  $r \leftarrow 0$;
  $W' \leftarrow W$;
  $changed \leftarrow true$;
  enquanto $changed = true$ *E* $r < k - 2$ **faça**
    $changed \leftarrow$ false;
    para *cada vértice* v *de* G *que não esteja em* W' **faça**
      $G_{RW} \leftarrow$ grafo completo somente com os vértices de $W' \cup v$ usando as distâncias de d_G;
      se $|AGM(G_{RW})| < melhor$ **então**
        $melhor \leftarrow |\text{AGM}(G_{RW})|$;
        $T \leftarrow \text{AGM}(G_{RW})$;
        $changed \leftarrow true$;
        $v' \leftarrow v$;
      fim
    fim
    se $changed = true$ **então**
      $r = r + 1$;
      $W' \leftarrow W' \cup v'$;
    fim
  fim
  expande os caminhos mínimos em T;
  retorna T;
fim

3. Experimentos Computacionais

A Tabela 2 apresenta os resultados dos testes computacionais realizados com as bases de teste de um *benchmark* conhecido[c]. As árvores encontradas pelo algoritmo foram comparadas com as soluções ótimas esperadas das instâncias em relação ao grau de

[c]Bases de testes disponíveis em: http://steinlib.zib.de/testset.php

aproximação. Para os experimentos foram utilizadas somente instâncias em que a solução ótima é conhecida.

	Esparsos com pesos aleatórios	**Completos com pesos aleatórios**	**Esparsos com pesos euclidianos**	**Completos com pesos euclidianos**	**Esparsos com pesos incidentes**	**TOTAL**
Instâncias testadas	60	27	14	14	395	514
Soluções ótimas encontradas	31	23	14	12	130	212
Porcentagem de ótimas encontradas	51,67%	85,19%	100%	85,71%	32,91%	41,25%
Média de aproximação	1,01	1,00	1,00	1,00	1,01	1,01
Pior aproximação encontrada	1,09	1,04	1,00	1,00	1,12	1,12

Tabela 2. Qualidade das soluções encontradas utilizando a heurística proposta.

4. Considerações Finais

A heurística proposta apresenta bons resultados em termos de qualidade da solução obtida em tempo $O(n^3 + nk^3)$ e é de fácil compreensão e implementação. Melhorar os critérios de escolha dos vértices roteadores, obter a razão de aproximação do algoritmo e fazer um estudo comparativo com outras heurísticas são atividades para trabalhos futuros.

Referências

Byrka, J., Grandoni, F., Rothvoss, T., and Sanità, L. (2013). Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6:1–6:33.

Dourado, M., Oliveira, R., and Protti, F. (2014). Algorithmic aspects of steiner convexity and enumeration of steiner trees. *Annals of Operations Research*, 223(1):155–171.

Dreyfus, S. E. and Wagner, R. A. (1971). The steiner problem in graphs. *Networks*, 1(3):195–207.

Karp, R. M. (1972). *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA.

Kou, L., Markowsky, G., and Berman, L. (1981). A fast algorithm for steiner trees. *Acta Informatica*, 15(2):141–145.

Pajor, T., Uchoa, E., and Werneck, R. F. (2014). A robust and scalable algorithm for the steiner problem in graphs. *CoRR*, abs/1412.2787.

Robins, G. and Zelikovsky, A. (2000). Improved steiner tree approximation in graphs. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, pages 770–779, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Takahashi, H. and Matsuyama, A. (1980). An approximate solution for the steiner problem in graphs. In *Math. Jap*, pages 24:573–577.

Vygen, J. (2011). Faster algorithm for optimum steiner trees. *Information Processing Letters*, 111(21):1075 – 1079.

Wu, Y. F., Widmayer, P., and Wong, C. K. (1986). A faster approximation algorithm for the steiner problem in graphs. *Acta Inf.*, 23(2):223–229.

Zelikovsky, A. Z. (1993). An 11/6-approximation algorithm for the network steiner problem. *Algorithmica*, 9(5):463–470.

The Hidden Binary Search Tree

Saulo Queiroz[1], **Edimar Bauer**[1]

[1]Academic Department of Informatics
Federal University of Technology (UTFPR)
Ponta Grossa, PR – Brazil

sauloqueiroz@utfpr.edu.br, edimarbauer@alunos.utfpr.edu.br

Abstract. *In this paper we review and enhance the Hidden Binary Search Tree (HBST) presented in [Queiroz 2017]. The HBST idea builds on the assumption an n-node self-balanced tree (e.g. AVL) requires to assure $O(\log_2 n)$ worst-case search, namely, comparison between keys takes constant time. Therefore the size of each key in bits is fixed to $B = O(\log_2 n)$ once n is determined, otherwise the $O(1)$-time comparison assumption does not hold. HBST generalizes the search-tree property such that the position of a node in the tree results from comparing its key against 'ideal' reference values associated to its ancestors. The first ideal values comes from the mid-point of the interval $0..2^B$. The strategy follows recursively such that the HBST height is bounded by $O(B)$ regardless the input sequence of keys nor self-balancing procedures. In this paper we enhance the HBST to enable keys with arbitrary number of bits.*

1. Introduction

In a Binary Search Tree (BST) and variants thereof the 'search-tree' property relies on a field of the nodes' abstract data type to determine the position of a given key in the tree. That definition causes BST's height $h(n)$ to vary depending on the values of the given n-size input sequence of integer keys s=<k_i> ($0 \leq k_i < n$ and $1 \leq i \leq n$) such that $h(n) = O(n)$ in the worst case. Self-balanced BSTs (e.g. AVL, Red-black) decouple the $h(n)$ performance from s by means of self-balancing procedures. These procedures may walk the path back from an inserted (deleted) node to the root in order to update height-related informations and, if needed, trigger rotation(s) to ensure $h(n) = O(\log_2 n)$. Therefore, self-balanced BSTs are preferred than BSTs when the asymptotic worst-case scenario is the sole criterion. However, if n is not large enough or s 'naturally' causes BST's height to be logarithmic – as is the case of random uniform inputs – BSTs may be preferred because of its "*substantially less overhead and simpler programming*" [Knuth 1998].

[Queiroz 2017] presents the Hidden Binary Search Tree (HBST), a data structure characterized by a logarithmic worst-case height and the 'simpler programming' of BSTs (i.e. no self-balancing procedures). The first remark of the HBST design is that the $O(\log_2 n)$ search of self-balanced BSTs results from assuming that a comparison between keys takes constant time. This implies the number of bits B per key must be a constant bounded to $O(\log_2 n)$ after n is determined. From this, the author generalizes the search-tree property such that the position of a given key in the tree can result from comparisons against *reference* values other than the key of prior inserted nodes. Relying on the assumption that B is constant, HBST matches its search reference values to the sequence of keys that 'naturally' causes a BST to be balanced. In this work we review the HBST procedures (Section 2) and enhance its design to enable unbounded B (Section 3).

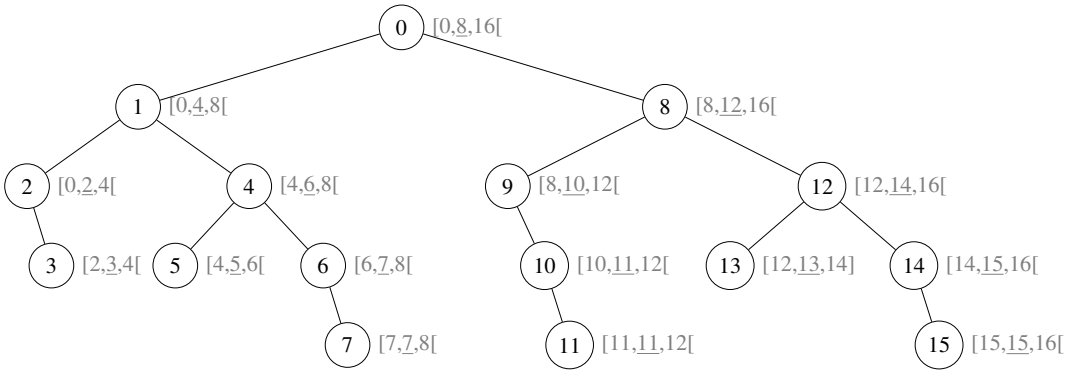


Figura 1. HBST built from the sequence $0, \ldots, 15$, $B = 4$. The upper and lower bounds intervals are computed across iterations. Search-property holds if a key is compared against the ideal hidden reference values of its ancestors (underlined values).

2. Elementary Procedures of HBST

The elementary HBST procedures exploits a widely known lesson from the design of algorithms, namely, an algorithm whose iteration (or recursive call) takes $O(1)$ time to halve an n-size problem has $O(\log_2 n)$ complexity. In several practical scenarios, the largest size n of a data structure is implicitly established when the programmer declares a B-bit key field, i.e. $n = 2^B$. From these points, HBST considers the lower and upper bounds of the interval $0..2^B$ to guide the insertion, search and deletion of a given input key $0 \leq k_i < 2^B$, as we describe next.

Insertion and Search. If the given (sub-)tree is empty, k_i is inserted as root. Otherwise the search reference value "idealRef" is set to $\lfloor (lower + upper)/2 \rfloor$. The variables $lower$ and $upper$ are respectively set to 0 and 2^B in the first iteration. If $k_i < idealRef$ the insertion follows recursively to the left subtree setting $upper$ to "idealRef". Otherwise it follows to the right subtree updating $lower$ to "idealRef". The interval $[lower, upper]$ halves at a cost $O(1)$ per iteration, leading to $O(B)$ iterations in the worst-case. An HBST built from the insertion sequence $0, 1, 2, \ldots, 15$ with $B = 4$ is illustrated in Fig. 1. Note that the search-tree property holds only if the keys at the left and right subtrees of an arbitrary node r are compared against the hidden (underlined) reference value associated to r. The same idea applies to find a key in the tree as illustrated in the recursive Algorithm 1.

Deletion. The first part of deleting a node r works just as in BST following the hidden search strategy to find it (Algorithm 1). If r is leaf, it is removed (of course the pointer to it is updated accordingly in the corresponding parent node). If r is non-leaf, it can be overwritten by one of its descendants leaf node[1].

3. Practical Concerns and HBST Enhanced Design

The careful reader may observe that HBST is nothing but a typical BST that exploits the constant number of bits of the key field to present an alternative search-tree property ensuring $O(B)$ height. For any quantity $n' \leq n$, HBST *worst-case* has $B + 1$ levels (i.e. $0, 1, ..., B$) while the height of AVL and Red-Black

[1] this fixes the statement "*the substitute can be any ... less than 2 children*" of [Queiroz 2017].

Algorithm 1: HBST Search.

```
1  Function Search (root, key, lower, upper)
2  # Assumptions: B-bit keys. Possible key values: 0, 1, ..2^B − 1.
3  # First call: lower=0, upper=n=2^B.
4  if root = nill OR root → key = key then
5  |  return root;
6  end
7  idealRef := ⌊(lower + upper)/2⌋;
8  if key < idealRef then
9  |  return Search (root → left, key, lower, idealRef);
10 else
11 |  return Search (root → right, key, idealRef, upper);
12 end
```

are $1.4405 \log_2(n' + 2) - 0.3277$ [Knuth 1998, Adelson-Velsky and Landis 1962] and $2 \log_2(n' + 1)$ [Guibas and Sedgewick 1978], respectively. For relatively small n', the *height* of self-balanced BSTs can outperform HBST's. However, the worst-case of insertion and deletion procedures might be added by rotations of those trees. Moreover, in the worst-case the path from the root to the node being inserted/removed is traversed twice (irrespective of rotations) to assure that balance factors (colors) of the ancestors are updated. Thus, the worst-case of insertion/deletion in self-balanced BSTs is at least twice their respective maximum heights plus rotations. Since HBST does not need self-balancing procedures nor rotations its actual performance compares to self-balanced BSTs. At the same time, its programming 'simplicity' and maintainability compare to the classical BST. We believe both these characteristics turn HBST very appealing for practical scenarios. Next we describe an enhanced HSBT design to handle keys with arbitrary number of bits.

The enhanced HBST data structure consists of a sequence of HSBTs as illustrated in Fig. 2. Given a non-negative integer key k_i for insertion, deletion or search, we firstly count the *minimum* number of bits $B(k_i)$ needed to represent k_i in binary. This can be done at a cost $O(B(k_i))$ by successively dividing k_i by two while the result is not zero. In practice, these divisions can be efficiently implemented by successive right shifts. We insert the number $B(k_i)$ in a sorted linked list. Each node of the list represents (i.e., has a pointer to) a specific HBST where all keys can be represented with the same minimum number of bits $B(k_i)$. For instance, keys 2 and 3 are inserted in the same HBST because $B(2) = B(3) = 2$ bits. Thus, $B(k_i)$ indicates which HBST k_i must be inserted into. If $B(k_i) = 1$ the hidden interval associated to the root node is $[0, 1]$. If $B(k_i) > 1$ the corresponding HBST interval is $[2^{B(k_i)-1}, 2^{B(k_i)}[$. Hence, the maximum number of nodes in the HBST of k_i is $n_{B(k_i)} = 2^{B(k_i)} - 2^{B(k_i)-1} = 2^{B(k_i)-1}$ (please, remark the base case $n_1 = 2$ keys) which yields a height of $O(\log_2(2^{B(k_i)-1})) = O(B(k_i))$.

The worst-case time for insertions in the sorted linked list of Fig. 2 is linear on the number of bits $B(k_i)$ of the key k_i. To make the HBST linked list to grow linearly on the number of nodes, one needs an input sequence in which each key is twice higher than its preceding key e.g. $s=<2^1, 2^2, 2^3, \ldots, 2^n>$. In this extreme case, there are n unitary

HBSTs and the largest key needs n bits. Thus an insertion takes $O(n) = O(B_{max})$ where B_{max} is the largest key in number of bits. In the opposite case where all n keys are in a single HBST, the size of the linked list is one (keys have the same size in bits) and the height of the tree is $O(B_{max})$. Accounting for the fact that a single key comparison in an arbitrary level of the tree is $O(B_{max})$ rather than $O(1)$, the overall search complexity is $O(B_{max}^2)$.

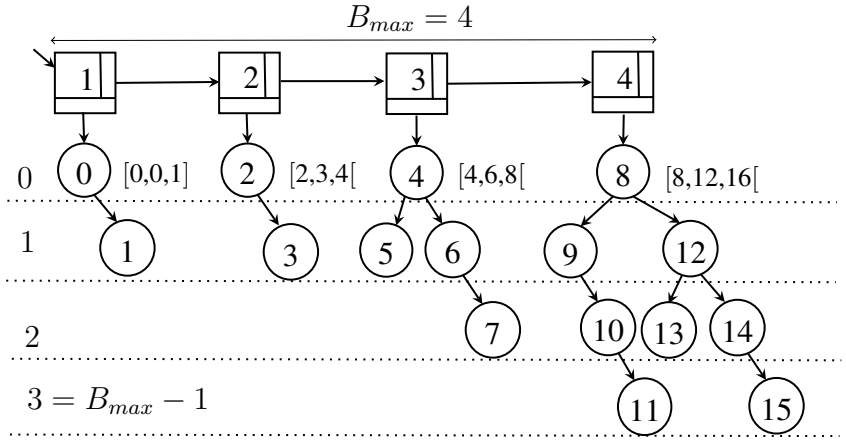


Figura 2. HSBT enhanced for keys with arbitrarily number of bits B. built from the sequence $0, \ldots, 15$. The first hidden interval for HSBTs with 1-bit and $B(k_i)$-bit keys are $[0, 1]$ and $[2^{B(k_i)-1}, 2^{B(k_i)}[$, respectively.

4. Summary

We reviewed and enhanced a variation of the Binary Search Tree (BST), named Hidden BST (HBST). Under the same assumptions of self-balanced BSTs (e.g. AVL, red-black), HBST's height is $O(B)$ where B is the size of keys in bits and n is a given number of keys. For keys with arbitrary number of bits, we show elementary $O(B_{max}^2)$ procedures where B_{max} is the currently known largest key in bits. Possible related topics for future work are linear-time in-order traversal, priority queues and external memory.

Referências

Adelson-Velsky, G. M. and Landis, E. M. (1962). An algorithm for the organization of information. In *Proceedings of the USSR Academy of Sciences*, volume 146, pages 263–266.

Guibas, L. J. and Sedgewick, R. (1978). A dichromatic framework for balanced trees. In *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, pages 8–21.

Knuth, D. E. (1998). *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

Queiroz, S. (2017). The hidden binary search tree: A balanced rotation-free search tree in the AVL RAM model. *CoRR*, abs/1711.07746.

Half Cuts em Grafos Bipartidos Completos

Rubens A. Sucupira[1], **Sulamita Klein**[2], **Luerbio Faria**[1]

[1]Instituto de Matemática e Estatística – Universidade do Estado do Rio de Janeiro (UERJ)

[2]COPPE Sistemas – Universidade Federal do Rio de Janeiro (UFRJ)

Abstract. *A graph is Half Cut if it admits an edge cut with exactly $\lceil \frac{m}{2} \rceil$ edges. It is known that graceful graphs are Half Cut and that complete bipartite graphs are graceful. In this paper we give an alternative proof that complete bipartite graphs are Half Cut, exhibiting an edge cut with exactly $\lceil \frac{m}{2} \rceil$ edges.*

Resumo. *Um grafo é Half Cut se admite um corte de arestas de cardinalidade $\lceil \frac{m}{2} \rceil$. É sabido que grafos graciosos são Half Cut e os grafos bipartidos completos são graciosos. Neste artigo damos uma prova alternativa de que os grafos bipartidos completos são Half Cut, exibindo um corte de arestas de cardinalidade $\lceil \frac{m}{2} \rceil$.*

1. Introdução

Dados um grafo $G = (V, E)$ com n vértices e m arestas, e um subconjunto não vazio próprio $S \subset V$, definimos o *corte de arestas* ∂S como o conjunto formado por todas as arestas com exatamente um extremo em S. Dizemos que um corte de arestas ∂S tal que $|\partial S| = \lceil \frac{m}{2} \rceil$ é um *half cut*. Um grafo $G = (V, E)$ é do tipo *Half Cut* se admite um *half cut*. Um modo de caracterizar um corte de arestas é através da partição associada a esse corte $V = S \cup (V \setminus S)$ que também representamos por $V = (S, V \setminus S)$.

Como exemplo de grafo Half Cut podemos considerar o grafo ciclo C_4 no qual encontramos cortes de arestas com cardinalidade $2 = \lceil \frac{4}{2} \rceil$. Em [Sucupira et al. 2017], exibimos partições Half Cut para quaisquer caminhos P_n e mostramos que só existem tais partições para os ciclos C_n com $n \equiv 0 (\text{mod } 4)$ ou $n \equiv 3 (\text{mod } 4)$. Um grafo $G = (V, E)$ é *bipartido completo* se seu conjunto de vértices V pode ser particionado em dois conjuntos independentes V_1 e V_2, de modo que cada vértice de V_1 é adjacente a todo vértice de V_2. Se $|V_1| = p$ e $|V_2| = q$, representamos o grafo bipartido completo correspondente por $K_{p,q}$. No contexto do artigo atual, o grafo C_4 pode ser visto como $K_{2,2}$ e apresenta uma outra maneira de construção da partição Half Cut. A Figura 1 a seguir apresenta essas duas partições.

Outros exemplos de grafos Half Cut são as rodas W_n que têm $m = 2n$ arestas e podem ser decompostas em uma estrela central S_n e um ciclo C_n, de modo que a partição Half Cut fica trivialmente determinada pela decomposição $V = (\{c\}, V \setminus \{c\})$ na qual c é o centro da estrela S_n.

Neste artigo mostramos que todo grafo bipartido completo $K_{p,q}$ é do tipo Half Cut, exibindo explicitamente um corte de arestas de cardinalidade $\lceil \frac{m}{2} \rceil$.

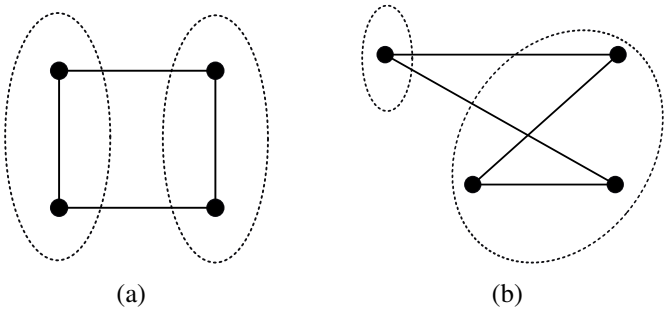


Figura 1. (a) Half cut de C_4 e (b) Half cut de $K_{2,2}$.

2. Grafos Graciosos

Uma rotulação graciosa de um grafo $G = (V, E)$ é composta por uma rotulação dos vértices $f : V \to \{0, 1, \cdots, m\}$ e uma rotulação das arestas $f_\gamma : E \to \{1, 2, \cdots, m\}$ definida por $f_\gamma(\{u, v\}) = |f(u) - f(v)|$ ambas injetivas. Se G admite uma rotulação graciosa, então G é dito um *grafo gracioso*. A Figura 2 apresenta as rotulações graciosas de C_4 e de $K_{2,3}$.

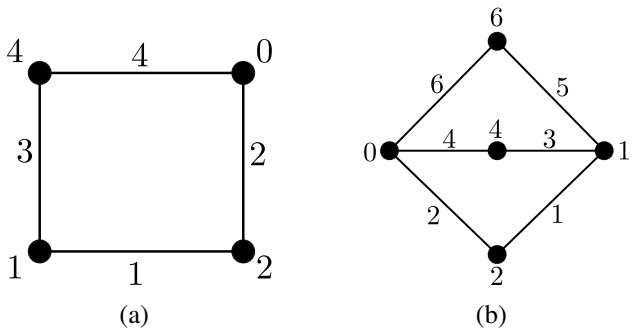


Figura 2. Rotulações graciosas de (a)C_4 e de (b) $K_{2,3}$.

[Golomb 1972] provou que todo grafo gracioso é do tipo Half Cut. A demonstração de Golomb pode ser encontrada em [Sucupira et al. 2017]. [Rosa 1966] e [Golomb 1972] provaram que todo grafo bipartido completo é gracioso, ou seja, já era conhecido o fato de que os grafos bipartidos completos são Half Cut.

3. Grafos Bipartidos Completos

A seguir demonstramos que os grafos bipartidos completos são Half Cut fornecendo um processo que define explicitamente uma partição Half Cut para tais grafos.

Teorema 3.1. *Todo grafo bipartido completo é do tipo Half Cut.*

Demonstração. Seja $K_{p,q}$ um grafo bipartido completo. Temos dois casos a considerar: No primeiro caso, pelo menos uma das cardinalidades p ou q é par; e no segundo caso, ambas as cardinalidades p e q são ímpares.

Primeiro caso: Sem perda de generalidade, consideremos p par, já que o caso q par se trata de maneira análoga. Então $p = 2k$ para algum inteiro positivo k. Vamos

considerar $V_1 = \{u_1, u_2, \cdots, u_k, u_{k+1}, \cdots, u_p\}$. Como todos os vértices de V_1 são adjacentes a todos os de V_2, todos os vértices $u_i \in V_1$ têm o mesmo grau q, ou seja, $K_{p,q}$ tem exatamente pq arestas, mais especificamente, $m = 2kq$ e assim $\lceil \frac{m}{2} \rceil = kq$. Uma maneira simples de obter um corte de arestas com tal cardinalidade é formar a partição $V = (A, B)$ com $A = \{u_1, u_2, \cdots, u_k\}$ e $B = V \setminus A$, ou seja, transferir metade dos vértices de V_1 para a outra parte da partição, eliminando metade das arestas de corte. A Figura 3 abaixo ilustra essa situação com exemplos de partições Half Cut para os grafos bipartidos completos $K_{4,3}$ e $K_{6,4}$.

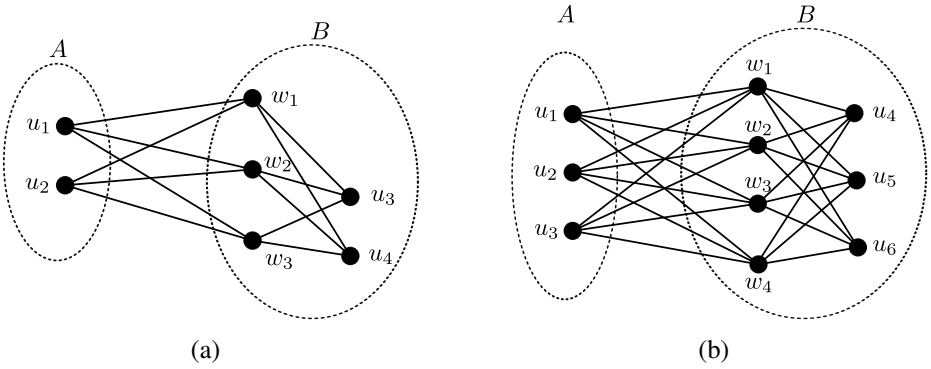


Figura 3. (a) Half cut de $K_{4,3}$ e (b) Half cut de $K_{6,4}$.

Segundo caso: Se p e q são ambos ímpares, existem inteiros positivos i e j tais que $p = 2i + 1$ e $q = 2j + 1$. Desse modo, o grafo bipartido completo $K_{p,q}$ tem $m = (2i + 1)(2j + 1)$ arestas e $\lceil \frac{m}{2} \rceil = 2ij + i + j + 1$. Nesse caso consideremos $V_1 = \{u_1, u_2, \cdots, u_{i+1}, \cdots, u_p\}$ e $V_2 = \{w_1, w_2, \cdots, w_{j+1}, \cdots, w_q\}$ e a partição $V = (A, B)$ na qual $A = \{w_1, w_2, \cdots, w_{j+1}, u_{i+2}, \cdots, u_p\}$ e $B = \{u_1, u_2, \cdots, u_{i+1}, w_{j+2}, \cdots, w_q\}$. Essa partição consiste em transferir $i + 1$ vértices de V_1 para a segunda parte da partição, bem como transferir $j + 1$ vértices de V_2 para a primeira parte, o que dará origem a um corte de arestas com cardinalidade $(i+1)(j+1)+ij$, já que a inversão de posições elimina as $(i + 1)j + i(j + 1)$ arestas do corte original $V = (V_1, V_2)$. A Figura 4 a seguir ilustra essa situação, exibindo as partições Half Cut para os grafos $K_{5,5}$ e $K_{5,7}$.

□

4. Trabalhos Futuros

[Acharya and Gill 1981] provaram que as grades planares completas $P_r \times P_s$ são grafos graciosos. Consequentemente tais grafos admitem uma partição Half Cut. Pretendemos em artigos futuros exibir partições Half Cut para as grades planares completas $P_r \times P_s$. Já foi verificado em [Sucupira et al. 2017] que os grafos graciosos formam uma subclasse própria dos grafos Half Cut, pois há grafos do tipo Half Cut que não são graciosos, como por exemplo os grafos completos K_n com $n = q^2 > 4$. Desejamos também estudar em que condições um grafo bipartido admite uma partição Half Cut, já que nem todo grafo bipartido é Half Cut. Um exemplo de grafo bipartido que não é Half Cut são os ciclos C_n com $n \equiv 2 (\mathrm{mod}\ 4)$. Verificamos também em [Sucupira et al. 2017] que toda árvore é do tipo Half Cut, embora ainda continue em aberto a Conjectura de Ringel-Kotzig [Huang et al. 1982] a qual afirma que todas as árvores são graciosas. Essa dificuldade em provar a conjectura para as árvores já nos dá indícios de que não é uma tarefa

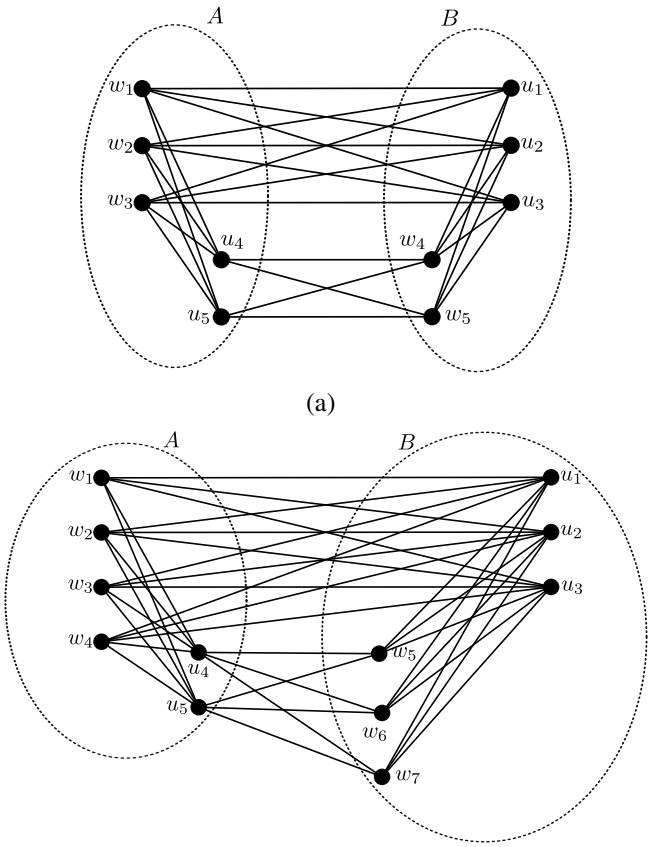


Figura 4. (a) Half cut de $K_{5,5}$ e (b) Half cut de $K_{5,7}$.

simples determinar sob que condições um grafo bipartido é gracioso. Desejamos também observar se no caso dos grafos graciosos há alguma relação entre as partições Half Cut que obtivemos e possíveis rotulações graciosas, de modo a utilizar uma partição Half Cut para determinar uma rotulação graciosa e vice-versa.

Referências

Acharya, B. and Gill, M. (1981). On the index of gracefulness of a graph and the gracefulness of two-dimensional square lattice graphs. *Indian J. Math*, 23:81–94.

Golomb, S. W. (1972). How to number a graph. *Graph theory and computing*, pages 23–37.

Huang, C., Kotzig, A., and Rosa, A. (1982). Further results on tree labellings. *Util. math., 21c*, pages 31–48.

Rosa, A. (1966). On certain valuations of the vertices of a graph. In *Theory of Graphs (Internat. Symposium, Rome*, pages 349–355.

Sucupira, R., Faria, L., and Klein, S. (2017). Grafos half cut. In de Computação, S. B., editor, *Anais do XXXVII Congresso da SBC*, pages 115–118. XXXVII Congresso da Sociedade Brasileira de Computação.

Problema de partição em conjunto independente e árvore quando restrito à classe dos grafos-P_4-tidy

Fábio Silva[1], Raquel Bravo[1], Rodolfo Oliveira[2], Uéverton Souza[1]

[1]Instituto de Computação
Universidade Federal Fluminense (UFF)

[2]Instituto do Noroeste Fluminense de Educação Superior
Universidade Federal Fluminense (UFF)

fabiojunior@id.uff.br, {raquel,ueverton}@ic.uff.br,

rodolfo.oliveira@infes.uff.br

Abstract. *We consider in this work the problem of partitioning the vertex set of a graph into an independent set (a graph without edges between its vertices) and into a tree (a connected graph that has no cycles) called $(\mathcal{S}, \mathcal{T})$-partition. We will present a characterization by minimal forbidden subgraphs of $(\mathcal{S}, \mathcal{T})$-partition of the P_4-tidy graphs. As an additional result, we have provided an algorithm of recognizing this partition in linear time for P_4-tidy graphs, using the proof of the characterization by forbidden subgraphs.*

Resumo. *Consideramos neste trabalho o problema de particionar o conjunto de vértices de um grafo em um conjunto independente (sem arestas entre os vértices) e em uma árvore (grafo livre de ciclos e conexo) denominada $(\mathcal{S}, \mathcal{T})$-partição. Apresentaremos uma caracterização por subgrafos proibidos minimais da $(\mathcal{S}, \mathcal{T})$-partição dos grafos restritos à classe dos grafos P_4-tidy. Como resultado adicional, disponibilizamos um algoritmo de reconhecimento desta partição em tempo linear para os grafos P_4-tidy, utilizando a prova de caracterização dos subgrafos proibidos.*

1. Introdução

Podemos descrever formalmente os problemas de partição de vértices como aqueles que tem por objetivo particionar conjunto de vértices de um grafo em subconjuntos $V_1, V_2, \ldots, V_k$, onde $V = V_1 \cup V_2 \cup \ldots \cup V_k$ e $V_i \cap V_j = \emptyset$, para $1 \leq i < j \leq k$, exigindo-se algumas propriedades internas ou externas sobre este conjunto de vértices. Esta classe de problemas tem despertado amplo interesse devido às pesquisas em grafos perfeitos [Golumbic 2004]. Pode-se encontrar aplicação para problemas de partição em diversas áreas como a biologia, engenharias, informática e mais recentemente na ciência de dados. Estudar subconjuntos de usuários em redes sociais, design de chips VLSI, escalonamento de recursos são algumas aplicações práticas que podem ser abordadas através de problemas de partição.

Outro problema de grande importância para a teoria dos grafos fortemente relacionado aos problemas de partição é o problema da k-coloração [Saaty 1977], que tem por objetivo colorir o conjunto de vértices de um grafo com k cores distintas de modo que dois vértices adjacentes não recebam a mesma cor. Este problema pode ser abordado

através do problema em partição de dividir o conjunto de vértices do grafo em k conjuntos independentes.

Chamamos a classe dos grafos que podem ser particionados em conjunto independente e uma árvore de grafos-$(\mathcal{S}, \mathcal{T})$. A classe dos grafos-$(\mathcal{S}, \mathcal{T})$ está estritamente contida em outra classe também conhecida como classe dos grafos *quase-bipartidos*. A classe dos grafos quase-bipartidos é definida como classe dos grafos que podem ser particionados em um conjunto independente e uma floresta. Foi introduzida por [Yang and Yuan 2006], onde os autores demonstraram que o problema da quase-bipartição é NP-completo para grafos gerais mesmo quando restrito a grafos de grau máximo 3 ou diâmetro 4, podendo ser resolvido em tempo polinomial para grafos de grau máximo 2 ou diâmetro 2, deixando a questão do diâmetro 3 em aberto. Recentemente, [Bonamy et al. 2018] provaram que a NP-completude se estende para grafos de diâmetro 3 também. Entretanto, o resultado não se aplica a todas as subclasses de grafos e isto tem despertado o interesse de diversos autores ([Dross et al. 2016], [Bonamy et al. 2017]) em pesquisar algoritmos eficientes para solucionar o problema da quase-bipartição quando restrito a certas subclasses de grafos.

[Brandstädt et al. 2013] provaram que este problema continua NP-completo para os grafos perfeitos, todavia, apresentaram um algoritmo polinomial para a classe dos cografos (contida estritamente nos grafos perfeitos) e iniciaram o estudo deste problema na classe dos grafos com poucos P_4's. Seguindo a mesma linha de investigação, estudamos o problema da $(\mathcal{S}, \mathcal{T})$-partição na classe dos grafos P_4-tidy, também pertencente à classe dos grafos com poucos P_4's, que será apresentada na próxima seção.

Apresentamos na Seção 3 uma caracterização para subgrafos desconexos $(\mathcal{S}, \mathcal{T})$-particionáveis e utilizamos este resultado para fornecer uma caracterização por subgrafos proibidos da partição $(\mathcal{S}, \mathcal{T})$ em grafos P_4-tidy em geral, apresentada na Seção 4. Este resultado pode ser aplicado no desenvolvimento de um algoritmo linear para o reconhecimento desta classe de grafos como demonstrado ao final desta mesma seção.

2. Preliminares

Seja $G = (V(G), E(G))$ um grafo tal que $V(G)$ denota seu conjunto de vértices e $E(G)$ seu conjunto de arestas. Dois vértices $u, v \in V(G)$ são ditos adjacentes se a aresta $(u, v) \in E(G)$. Denotamos como $N(X)$, para $X \subseteq V(G)$, a vizinhança do conjunto de vértices X, sendo $N(X)$ o conjunto dos vértices de $V(G)$ cujas arestas possuem apenas um dos extremos em X. Abreviamos $N(\{x\})$ como $N(x)$ por comodidade. Um subgrafo H de G é um grafo tal que $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$. Um subgrafo H de G é chamado de subgrafo induzido se, para qualquer par de vértices $u, v \in V(H)$, a aresta $(u, v) \in E(H)$ se, e somente se, $(u, v) \in E(G)$.

Um grafo G é dito da classe P_4-tidy se para qualquer P_4 (caminho formado por 4 vértices) induzido H de G, existe no máximo um vértice fora de H que juntamente com três vértices de H induzem no máximo um P_4. Esta classe foi apresentada por [Roussel et al. 1999] para generalizar a já conhecida classe dos grafos com poucos P_4's. Uma caracterização estrutural para os grafos P_4-tidy foi apresentada em [Giakoumakis et al. 1997].

Teorema 1. *[Giakoumakis et al. 1997] Um grafo G é P_4-tidy se e somente se para todo subgrafo induzido H de G, exatamente uma das seguintes condições é satisfeita: (a) H*

é desconexo; (b) $\overline{H}$ é desconexo; (c) H é uma aranha, tal que a cabeça induz um grafo P_4-tidy; (d) H é uma quase-aranha, tal que a cabeça induz um grafo P_4-tidy; (e) H é isomorfo a C_5 (ciclo induzido com 5 vértices), P_5 (caminho induzido com 5 vértices) ou $\overline{P_5}$ (complemento de um caminho induzido com 5 vértices); (f) H possui exatamente um vértice ou $V(G) = \emptyset$. □

As demais definições necessárias para este trabalho podem ser encontradas em [Giakoumakis et al. 1997]. Por questão de espaço as provas das seções seguintes foram omitidas.

3. Caracterização da $(\mathcal{S}, \mathcal{T})$-partição em grafos desconexos

Mostraremos agora uma caracterização para a $(\mathcal{S}, \mathcal{T})$-partição nos grafos desconexos. Seja G um grafo desconexo.

Lema 1. *G possui uma partição-$(\mathcal{S}, \mathcal{T})$ se e somente se existe um componente conexo $G_i \subseteq G$ tal que G_i possui uma partição $(\mathcal{S}, \mathcal{T})$ e $G \backslash G_i$ é conjunto independente.* □

Corolário 1. *Se G possui uma partição-$(\mathcal{S}, \mathcal{T})$, então existe no máximo um componente conexo não trivial (com aresta) em G.* □

Lema 2. *Seja G' um subgrafo induzido de G, isomorfo a $2K_2$. Se G é $(\mathcal{S}, \mathcal{T})$-particionável, então todos os vértices de G' pertencem ao mesmo componente conexo de G.* □

Pelos resultados do Corolário 1 e Lema 2 garantimos que se G é $(\mathcal{S}, \mathcal{T})$-particionável, então G possui no máximo um componente conexo contendo aresta, sendo assim livre de qualquer $2K_2$ induzido por dois componentes conexos distintos. Assim podemos afirmar que uma vez verificada a existência de apenas um componente conexo não trivial, podemos enunciar o seguinte teorema.

Teorema 2. *Seja G um grafo com exatamente um componente conexo não trivial G'. G possui uma $(\mathcal{S}, \mathcal{T})$-partição se, e somente se, G' possui uma $(\mathcal{S}', \mathcal{T}')$-partição.* □

Concluímos então que é possível identificar a $(\mathcal{S}, \mathcal{T})$-partição de um grafo G desconexo pela caracterização de uma partição $(\mathcal{S}', \mathcal{T}')$ do seu único componente conexo não trivial G', caso exista. Assim, garantir que o componente conexo não trivial único G' de G seja $(\mathcal{S}', \mathcal{T}')$-particionável é uma condição necessária e suficiente para garantir a $(\mathcal{S}, \mathcal{T})$-partição de qualquer grafo G.

4. Caracterização da $(\mathcal{S}, \mathcal{T})$-partição em grafos P_4-tidy

Utilizando os resultados da seção anterior, ofereceremos uma caracterização para os grafos P_4-tidy-$(\mathcal{S}, \mathcal{T})$ através de uma caracterização por subgrafos proibidos para grafos P_4-tidy conexos.

Teorema 3. *Seja G um grafo P_4-tidy. G possui uma $(\mathcal{S}, \mathcal{T})$-partição se, e somente se, existe no máximo um componente conexo não trivial G' em G, e G' é livre dos grafos da Figura 1 como subgrafos induzidos.* □

Concluímos então a análise do problema do reconhecimento da partição $(\mathcal{S}, \mathcal{T})$ em grafos P_4-tidy apresentando um algoritmo que reconhece esta partição em tempo linear.

Teorema 4. *Podemos construir um algoritmo linear para o reconhecimento de grafos P_4-tidy $(\mathcal{S}, \mathcal{T})$-particionáveis.* □

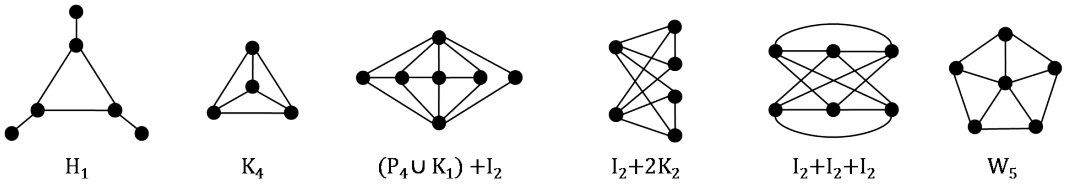


Figura 1. Subgrafos proibidos para grafo P_4-tidy-$(\mathcal{S}, \mathcal{T})$

5. Conclusão

Como principal resultado trouxemos uma caracterização da $(\mathcal{S}, \mathcal{T})$-partição nos grafos desconexos, reduzindo o problema a encontrar um partição $(\mathcal{S}', \mathcal{T}')$ em um subgrafo não trivial e conexo único, caso exista. Estendemos o resultado anterior para caracterizar os grafos P_4-tidy através de uma família de subgrafos proibidos minimais, utilizando as características da decomposição modular da classe P_4-tidy. Este resultado incrementa o estado da arte no estudo do problema da $(\mathcal{S}, \mathcal{T})$-partição em grafos com poucos P_4's, mostrando que a classe dos grafos P_4-tidy, assim como as demais classes que estão contidas propriamente nesta, podem ser reconhecidas em tempo linear. Em trabalhos futuros pretendemos estender estes resultados para a classe dos grafos P_4-laden e P_4-laden estendido, completando assim a análise do problema para a classe dos grafos com poucos P_4's.

Referências

Bonamy, M., Dabrowski, K. K., Feghali, C., Johnson, M., and Paulusma, D. (2017). Independent feedback vertex set for p_5-free graphs. *arXiv preprint arXiv:1707.09402*.

Bonamy, M., Dabrowski, K. K., Feghali, C., Johnson, M., and Paulusma, D. (2018). Independent feedback vertex sets for graphs of bounded diameter. *Information Processing Letters*, 131:26–32.

Brandstädt, A., Brito, S., Klein, S., Nogueira, L. T., and Protti, F. (2013). Cycle transversals in perfect graphs and cographs. *Theoretical Computer Science*, 469:15–23.

Dross, F., Montassier, M., and Pinlou, A. (2016). Partitioning sparse graphs into an independent set and a forest of bounded degree. *arXiv preprint arXiv:1606.04394*.

Giakoumakis, V., Roussel, F., and Thuillier, H. (1997). On p4-tidy graphs. *Discrete Mathematics and Theoretical Computer Science*, 1.

Golumbic, M. C. (2004). *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier.

Roussel, F., Rusu, I., and Thuillier, H. (1999). On graphs with limited number of p4-partners. *International Journal of Foundations of Computer Science*, 10(01):103–121.

Saaty, T. (1977). *The four-color problem : assaults and conquest*. McGraw-Hill International Book Co, New York.

Yang, A. and Yuan, J. (2006). Partition the vertices of a graph into one independent set and one acyclic set. *Discrete mathematics*, 306(12):1207–1216.

A b-continuidade de grafos com cintura alta*

Allen Ibiapina , Ana Silva

[1]Departamento de Matemática – Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brazil
ParGO - Paralelismo, Grafos e Otimização

allenr.roossim@gmail.com, anasilva@mat.ufc.br

Abstract. *A b-coloring of a graph is a proper coloring such that each color class has at least one vertex which is adjacent to each other color class. The b-spectrum of G is the set $S_b(G)$ of integers k such G has a b-coloring with k colors and $b(G) = \max S_b(G)$ is the b-chromatic number of G. A graph is b-continous if $S_b(G) = [\chi(G), b(G)] \cap \mathbb{Z}$. An infinite number of graphs that are not b-continuous is known. Also it is known that graphs with girth at least 10 are b-continuous. In this article, we prove that graphs with girth at least 8 are b-continuous and that the b-spectrum of graphs with girth at least 7 contains the integers between $2\chi(G)$ and $b(G)$.*

Resumo. *Uma b-coloração de um grafo é uma coloração própria, tal que cada classe de cor possui um vértice que é vizinho de pelo menos um vértice das outras classes de cores. O b-espectro de G é o conjunto $S_b(G)$ dos inteiros k tais que G tem uma b-coloraçao com k cores e $b(G) = \max S_b(G)$ é o número b-cromático de G. Um grafo é b-contínuo se $S_b(G) = [\chi(G), b(G)] \cap \mathbb{Z}$. É conhecida uma infinidade de grafos que não são b-contínuos. Também é sabido que grafos com cintura maior ou igual a 10 são b-contínuos. Neste artigo, mostramos que grafos com cintura pelo menos 8 são b-contínuos e que o b-espectro de grafos com cintura pelo menos 7 contém os inteiros entre $2\chi(G)$ e $b(G)$.*

Introdução

Seja G um grafo simples[1]. Uma *k-coloração própria* de G, doravante chamada apenas de k-coloração, é uma função $\psi\colon V(G) \to \mathbb{N}$ tal que $|\psi(V(G))| = k$ e $\psi(u) \neq \psi(v)$ sempre que $uv \in E(G)$. Dizemos que $u \in V(G)$ é um *b-vértice em ψ (de cor $\psi(u)$)* se para toda cor $c \in \psi(V(G)) \setminus \{\psi(u)\}$, existe v vizinho de u tal que $\psi(v) = c$. Se para uma classe de cor $c \in \psi(V(G))$, não existem b-vértices da cor c, podemos obter uma $(k-1)$-coloração recolorindo cada vértice da cor c com uma cor de $\psi(V(G)) \setminus \{c\}$ que não aparece em sua vizinhança; dizemos que essa nova coloração é obtida de ψ pela *limpeza da cor c*. Para uma coloração que não podemos aplicar esse algorítmo para diminuir a quantidade de cores, toda classe de cor c possui um b-vértice. Uma coloração assim é chamada de *b-coloração*. Como o problema de coloração é NP-completo, nem sempre uma b-coloração usa apenas $\chi(G)$ cores. Note que toda coloração de G usando $\chi(G)$ cores é uma b-coloração, caso contrário poderíamos diminuir o número de cores usadas. Por isso, a

*Este trabalho foi parcialmente financiado pelo Projeto CNPq/Universal 401519/2016-3

[1]Aqui usamos a terminologia de [Bondy and Murty 2008].

menor quantidade de cores em uma b-coloração coincide com o número cromático, e portanto só trabalhamos com o parâmetro de maximização. Em [Irving and Manlove 1999] é definido o *número b-cromático de G*, que é denotado por *b(G)*, como o maior natural k tal que G tem uma b-coloração que usa k cores. No mesmo artigo os autores mostraram que o problema de achar $b(G)$ é NP-completo.

Considerando uma b-coloração com $b(G)$ cores, cada b-vértice claramente tem grau maior ou igual $b(G) - 1$. Assim, definindo $m(G)$ como o maior inteiro k tal que G possui k vértices de grau maior ou igual $k-1$, segue que $m(G) \geq b(G)$. Note que pode-se calcular $m(G)$ em tempo polinomial. Irving e Manlove provaram que calcular o número b-cromático de árvores é polinomial. Eles demonstraram primeiro que $b(G) \geq m(G) - 1$ e depois que pode-se decidir se $b(G) = m(G)$ em tempo polinomial. Estes resultados foram generalizados em [Campos et al. 2015] para grafos de cintura pelo menos 7. Tais estudos sugerem que grafos de cintura alta são mais fáceis de lidar no que diz respeito à b-coloração.

Irving and Manlove observaram que o cubo tem b-coloração com 2 cores e com 4 cores, mas não possui b-coloração com 3 cores. Inspirado nisso, em [Kratochvíl et al. 2002] é mostrado que, para $n \geq 1$ inteiro, o grafo obtido de $K_{n,n}$ pela deleção de arestas de um emparelhamento perfeito possui b-colorações usando 2 e n cores, mas não com uma quantidade de cores estritamente entre esses dois números. Daí surge a definição do *b-espectro de G*, que é o conjunto dos inteiros k tais que G tem uma b-coloração com k cores; tal conjunto é denotado por $S_b(G)$. Naturalmente surge também a definição de b-continuidade: dizemos que um grafo é *b-contínuo* se $S_b(G) = [\chi(G), b(G)] \cap \mathbb{Z}$. Em [Barth et al. 2007] é provado que para todo subconjunto finito $S \subset \mathbb{N} \setminus \{1\}$, existe G tal que $S_b(G) = S$, e também que o problema de decidir se um dado grafo G é b-contínuo é NP-completo ainda que sejam dadas b-colorações com $\chi(G)$ e $b(G)$ cores. Existem muitos resultados positivos no que diz respeito à b-continuidade de grafos de alguma família. É demonstrado que grafos regulares com cintura maior ou igual a 6, sem ciclos de tamanho 7 são b-contínuos em [Balakarishnan and Kavaskar 2012], e, mais recentemente, que grafos de cintura pelo menos 10 são b-contínuos [Sales and Silva 2017]. Lá, os autores, propõem a seguinte questão: Qual é o menor inteiro $\hat{g}$ tal que G é b-contínuo sempre que tem cintura pelo menos $\hat{g}$. Usando o resultado lá demonstrado, tem-se $\hat{g} \leq 10$. Por outro lado, como $K_{n,n}$ menos um emparelhamento perfeito não é b-contínuo temos que $5 \leq \hat{g}$. Até onde sabemos, nenhum grafo de cintura pelo menos 5 que não é b-continuo é conhecido. O principal resultado desse artigo que melhora o resultado provado em [Sales and Silva 2017] é.

Teorema 1. *Se G é um grafo de cintura pelo menos 8, então G é b-contínuo.*

E, ainda na direção de mostrar que b-colorações de grafos com cintura alta são fáceis, demonstramos o seguinte teorema:

Teorema 2. *Se G é um grafo de cintura pelo menos 7, então* $[2\chi(G), b(G)] \cap \mathbb{Z} \subseteq S_b(G)$

De maneira geral, a prova consiste em, dada uma b-coloração com k cores, obter uma b-coloração com $k - 1$ cores usando recolorações. Mencionamos que as provas possuem um passo não construtivo, o que é de esperar já que calcular o número cromático de grafos com cintura ao menos k é NP-completo, para todo $k \geq 3$ fixo [Lozin and Kaminski 2007].

As seguintes definições serão essenciais para a prova. A *cintura* de um grafo é o tamanho do seu menor ciclo. Para z inteiro, e $u \in V(G)$, $N_{\leq z}(u)$ denota o conjunto de

vértices à distância no máximo z de u. Em [Sales and Silva 2017], um vértice $u \in V(G)$ é definido para ser uma *k-íris* se existe $S \subseteq N(u)$ tal que $|S| \geq k-1$ e $d(v) \geq k-1$ para todo $v \in S$. Dada ψ uma k-coloração de G, cada $i \in \{1,\ldots,k\}$ é chamado de *cor*, enquanto que $\psi^{-1}(i)$ é chamado de *classe de cor*. Dizemos que um vértice u *realiza* a cor i, se $u \in \psi^{-1}(i)$ e u é b-vértice; dizemos também que i é *realizada por* u. Para $x \in V(G)$ e $i \in \{1,\ldots,k\}$, $N^i(x) = \{v \in N(x) \mid \psi(v) = i\}$, ou seja, os vizinhos de x que tem cor i. Para $X \subseteq V(G)$, $N^i(X) = \cup_{x \in X} N^i(x) \setminus X$. Denotamos por $B(\psi)$ o conjunto de b-vértices de ψ e, para $i \in \{1,\ldots,k\}$, B_i é o conjunto de b-vértices da cor i, isto é, $B_i = B(\psi) \cap \psi^{-1}(i)$. Para cada $x \in V(G) \setminus B(\psi)$, seja $U(x) = \{i \in \{1,\ldots,k\} \mid N^{\psi(x)}(B_i) = \{x\}\}$, ou seja, o conjunto de cores que dependem de x para possuirem b-vértices. Para finalizar, para uma cor $j \in \{1,\ldots,k\}$, dizemos que $x \in V(G)$ é *j-mutável* se existe cor c tal que, ao mudarmos a cor de x para c, não criamos b-vértices da cor j; caso contrário x é dito *j-imutável*. Em [Balakarishnan and Kavaskar 2012], é provado o seguinte lema:

Lema 1. *Seja G um grafo de cintura pelo menos 6 e sem ciclos de tamanho 7. Se G tem uma k-íris com $k \geq \chi(G)$, então G tem uma b-coloração com k cores.*

Dado $x \in V(G) \setminus B(\psi)$, quando $U(x) = \emptyset$, podemos mudar a cor de x para uma cor livre sem perder a b-coloração com k cores. Em nossa prova, nos preocupamos especialmente com vértices $x \in V(G) \setminus B(\psi)$ tais que $|U(x)| \geq 2$; a esses chamamos de *úteis*. Para um conjunto $K \subseteq V(G) \setminus B(\psi)$ monocromático de cor i, diremos que uma cor j é *dependente* de K se $N^i(B_j) \subseteq K$. Note que, ao mudarmos as cores de K para quaisquer cores diferente da inicial, perderemos b-vértices de todas as cores dependentes de K e somente destas. Note que se j depende de K, então $B_j \subseteq N^j(K)$.

O teorema principal do artigo e de onde os Teoremas 1 e 2 são corolários é o seguinte.

Teorema 3. *Seja $G = (V, E)$ um grafo com cintura maior ou igual a 7. Se G possui uma b-coloração com k cores, onde $k \geq \chi(G)+1$, então G possui uma b-coloração com $k-1$ cores ou uma $(k-1)$-íris.*

Acreditamos que as técnicas de recoloração alcançam seu limite quando a cintura chega em 7, tanto que o resultado que calcula o número cromático de grafos com cintura 7 apresentado em [Campos et al. 2015] até hoje ainda não foi melhorado. Desta maneira, uma tentativa de encontrar grafos com cintura 6 que não sejam b-contínuos é válida, mesmo se restringimos para grafos bipartidos. Em particular, uma resposta positiva para ambas as perguntas abaixo definiria o valor de $\hat{g}$ em 7.

Pergunta 1. *O Lema 1 pode ser generalizado de maneira a permitir ciclos de tamanho 7?*

Pergunta 2. *Existe grafo bipartido com cintura 6 que não seja b-contínuo?*

Idéia da Prova

Nossa prova segue semelhante à feita em [Sales and Silva 2017], porém damos atenção a uma cor que queremos retirar. Isso faz com que os b-vértices das outras cores se aproximem dos b-vértices dessa cor específica. Seja ψ uma coloração com k cores de G. Suponha que ψ minimiza em primeiro lugar a quantidade de b-vértices da cor 1 e, em segundo lugar, que minimiza a quantidade de vértices da cor 1. Seja $u \in B_1$. Tentamos recolorir $N(u)$ de forma que u deixe de ser b-vértice, tomando cuidado de não criar mais

b-vértices da cor 1 e de que todas as outras cores ainda sejam realizadas. Daí, pela minimalidade de $|B_1|$, essa nova coloração não é b-coloração e, como todas as cores que não 1 são realizadas, a cor 1 é a única não realizada. Pela limpeza de 1, conseguimos uma b-coloração com $k-1$ cores. Argumentamos também que, quando não conseguimos isso, tem-se que u é k-íris e, em particular, $(k-1)$-íris; daí a b-coloração desejada é obtida pelo Lema 1. Um fato importante e que já nos direciona para um entendimento do que está acima, é o seguinte.

Fato 1. *Para cada cor $j \in \{2, \ldots, k\}$, tem-se que todo $x \in N^j(u) \setminus B(\psi)$ é 1-mutável. Além disso, vale um dos seguintes itens:*

1. *Existe $v \in N(u) \cap B_j$;*
2. *Existe cor $d \in \{2, \ldots, k\} \setminus \{j\}$ tal que d depende de $N^j(u)$.*

Após a prova do fato, argumentamos que se a quantidade de cores que segue o item 2 é não nula, passando por algumas colorações parciais conseguimos uma b-coloração com $k-1$ cores. Sobrando então o caso onde o item 1 vale para todas as cores. É trivial ver que temos uma $(k-1)$-íris.

Referências

Balakarishnan, R. and Kavaskar, T. (2012). b-coloring of kneser graphs. *Discrete Applied Mathematics*, 160:9–14.

Barth, D., Cohen, J., and Faik, T. (2007). On the b-continuity property of graphs. *Discrete Applied Mathematics*, 155:1761–1768.

Bondy, J. and Murty, U. (2008). *Graph Theory*. Springer.

Campos, V., Lima, C., and Silva, A. (2015). Graphs of girth at least 7 have high b-chromatic number. *European Journal of Combinatorics*, 48:154–164.

Irving, R. W. and Manlove, D. (1999). The b-chromatic number of a graph. *Discrete Applied Mathematics*, 91:127–141.

Kratochvíl, J., Tuza, Z., and Voigt, M. (2002). On the b-chromatic number of graphs. In Goos, G., Hartmanis, J., van Leeuwen, J., and Kučera, L., editors, *Graph-Theoretic Concepts in Computer Science*, pages 310–320. Springer Berlin Heidelberg.

Lozin, V. and Kaminski, M. (2007). Coloring edges and vertices of graphs without short or long cycles. *Contributions do Discrete Mathematics*, 2.

Sales, C. L. and Silva, A. (2017). The b-continuity of graphs with large girth. *Graphs and Combinatorics*, 33:1138–1146.

On Total and Edge-colouring of Proper Circular-arc Graphs

João Pedro W. Bernardi[1,2*], Sheila M. de Almeida[3*], Leandro M. Zatesko[1,2*†]

[1]Federal University of Fronteira Sul, Chapecó, Brazil

[2]Federal University of Paraná, Curitiba, Brazil

[3]Federal University of Technology — Paraná, Ponta Grossa, Brazil

{winckler,leandro.zatesko}@ufpr.br, sheilaalmeida@utfpr.edu.br

Abstract. *Deciding if a graph is Δ-edge-colourable (resp. $(\Delta+1)$-total colourable), although it is an* NP*-complete problem for graphs in general, is polynomially solvable for interval graphs of odd (resp. even) maximum degree Δ. An interesting superclass of the proper interval graphs are the proper circular-arc graphs, for which we suspect that Δ-edge-colourability is linear-time decidable. This work presents sufficient conditions for Δ-edge-colourability, $(\Delta+1)$-total colourability, and $(\Delta+2)$-total colourability of proper circular-arc graphs. Our proofs are constructive and yield polynomial-time algorithms.*

1. Introduction

The chromatic index and the total chromatic number of a graph G with maximum degree Δ clearly satisfy $\chi'(G) \geq \Delta$ and $\chi''(G) \geq \Delta + 1$ (see definitions in the sequel). Also, $\chi'(G) \leq \Delta + 1$ [Vizing 1964], and the *Total Colouring Conjecture* states that $\chi''(G) \leq \Delta + 2$ [Behzad 1965, Vizing 1968]. A graph G is *Class 1* if $\chi'(G) = \Delta$, or *Class 2* otherwise. Since no graph with $\chi''(G) \geq \Delta+3$ is known, graphs with $\chi''(G) = \Delta+1$ have been called *Type 1*, and those with $\chi''(G) = \Delta + 2$ *Type 2*. Deciding if G is *Class 1* and deciding if G is *Type 1* are NP-complete problems [Holyer 1981, Sánchez-Arroyo 1989].

The classes of the unit and the proper interval graphs are the same [Roberts 1969], but the classes of the unit and the proper circular-arc graphs are not (see Figure 1). The

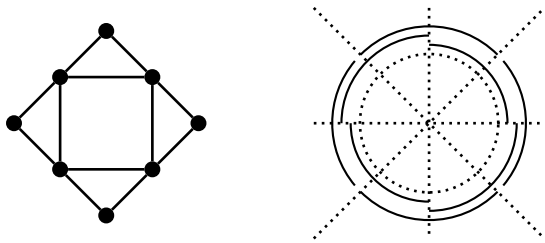


Figure 1. A proper non-unit circular-arc graph with a corresponding arc model

Total Colouring Conjecture holds for proper interval graphs, often referred to as *indifference graphs*, which are *Class 1* when they have odd Δ, and *Type 1* when Δ is even [Figueiredo et al. 1997]. For edge-colouring of indifference graphs with even Δ or total colouring of these graphs with odd Δ, partial results are known [Figueiredo et al. 2003,

*Partially supported by CNPq, Proc. 428941/2016-8.

†Partially supported by UFFS, Proc. 23205.001243/2016-30.

Campos et al. 2012]. Recall that interval graphs are perfect graphs and, thus, admit polynomial-time vertex-colouring algorithms [Grötschel et al. 1981], in contrast to vertex-colouring of circular-arc graphs, which is NP-hard [Garey et al. 1980]. To the best of our knowledge, there is no published work on total or edge-colouring of circular-arc graphs.

Let G be an n-vertex proper circular-arc graph. We show that if $n \equiv 0 \pmod{(\Delta+1)}$, or if G has a maximal clique of size 2 and $n \not\equiv k \pmod{(\Delta+1)}$ for all $k \in \{1, \Delta\}$, then: $\chi'(G) = \Delta$ and $\chi''(G) \leq \Delta + 2$ if Δ is odd; $\chi''(G) = \Delta + 1$ if Δ is even. This implies that the Total Colouring Conjecture holds for the class of all such graphs.

This paper is organised as follows: the remaining of this section provides further definitions and discusses other related results in the literature; Section 2 presents our results; at last, Section 3 makes remarks on edge-colouring proper circular-arc graphs.

Preliminary definitions and other related results

This work deals only with *simple graphs*, referred to simply as *graphs*. Usual terms concerning graph-theoretical concepts follow their definitions and notation in the literature. In particular, the *degree* of a vertex u in a graph G, the *set of neighbours* of u in G, and the *set of the edges incident* to u in G are denoted by $d_G(u)$, $N_G(u)$, and $\partial_G(u)$, respectively.

Let $G = (V, E)$ be a graph and $\mathscr{C}$ a set of t colours. A function with $\mathscr{C}$ as its codomain is: a *t-edge-colouring* if its domain is E and it is injective in $\partial_G(u)$ for all $u \in V$; a *t-total colouring* if its domain is $V \cup E$ and it is injective in $\partial_G(u) \cup \{u\}$ for all $u \in V$ and injective in $\{u, v\}$ for all $uv \in E$. In a total or edge-colouring, we say that a colour is *missing* at a vertex u if it is not assigned to u or to any edge incident to u. The chromatic index (denoted by $\chi'(G)$) and the total chromatic number (denoted by $\chi''(G)$) of G are the least t for which G is t-edge-colourable and t-total colourable, respectively.

An n-vertex graph with more than $\Delta\lfloor n/2 \rfloor$ edges (thus *Class 2*, since at most $\lfloor n/2 \rfloor$ edges can be coloured the same) is said to be *overfull*. It is conjectured that every graph G with $\Delta > n/3$ is *Class 2* if and only if it is *subgraph-overfull* (shortly, *SO*), i.e. if G has an overfull subgraph with the same maximum degree [Hilton and Johnson 1987].

The complete graph K_n is: *Class 1* and *Type 2* if n is even; *Class 2* and *Type 1* if n is odd [Behzad et al. 1967]. Let $V(K_n) \coloneqq \{0, \ldots, n-1\}$ and let $\mathrm{even}(n)$ be 1 if n is even or 0 otherwise. We call the *canonical* total and edge-colourings of the K_n the functions $\varphi^{\text{even}}_{\text{edge}}$, $\varphi^{\text{odd}}_{\text{edge}}$, and φ_{total} given by: $\varphi^{\text{even}}_{\text{edge}}(uv) \coloneqq (u+v) \bmod (n-1)$, if neither u nor v is $n-1$; $\varphi^{\text{even}}_{\text{edge}}(uv) \coloneqq (2u) \bmod (n-1)$, if $v = n-1$; $\varphi^{\text{odd}}_{\text{edge}}(uv) = \varphi_{\text{total}}(uv) \coloneqq (u+v) \bmod (n + \mathrm{even}(n))$; $\varphi_{\text{total}}(u) \coloneqq (2u) \bmod (n + \mathrm{even}(n))$.

A *circular-arc graph* G is the intersection graph of a finite set S of arcs of a circle, in which case S is an *arc model* corresponding to G. Furthermore, G is said to be: *proper*, if there is a corresponding arc model wherein no arc properly contains another; a *unit* circular-arc graph, if there is a model wherein all the arcs have equal length. The vertices of a proper circular-arc graph admit a *proper circular-arc order*, i.e. a circular order in which vertices belonging to the same maximal clique appear consecutively. Homonymous terms are defined for *interval graphs* analogously, but being S a set of intervals on the real line and the *proper interval* (or *indifference*) *order* a linear order. Interval and circular-arc graphs can be recognised in linear time [Booth and Lueker 1976, McConnell 2003].

A *pullback* from $G_1 = (V_1, E_1)$ to $G_2 = (V_2, E_2)$ is a *homeomorphism* $\pi \colon V_1 \to$

V_2 (i.e. $\pi(u)\pi(v) \in E_2$ for all $uv \in E_1$) injective in $N_{G_1}(u) \cup \{u\}$ for all $u \in V_1$. If such a pullback exists and G_2 has a: t-edge-colouring φ, then a t-edge-colouring for G_1 can be given by $\psi(uv) \coloneqq \varphi(\pi(u)\pi(v))$; t-total colouring φ, then a t-total colouring for G_1 can be given by $\psi(uv) \coloneqq \varphi(\pi(u)\pi(v))$ and $\psi(u) \coloneqq \varphi(\pi(u))$ [Figueiredo et al. 1997].

2. Results

Throughout this section, let G be an n-vertex proper circular-arc graph. Remark that when we say that G is $\Delta + 2$-total colourable, it does not mean that G is *Type 2*.

Theorem 1. *If* $n \equiv 0 \pmod{(\Delta + 1)}$*, then* G *is:* Class 1 *and* $(\Delta + 2)$*-total colourable if* Δ *is odd;* Type 1 *if* Δ *is even.*

Proof. It suffices to show that if $n \equiv 0 \pmod{(\Delta + 1)}$, then there is a pullback from G to the $K_{\Delta+1}$. Let $\sigma \coloneqq u_0, \ldots, u_{n-1}$ be a proper circular-arc order of G and $0, \ldots, \Delta$ be the vertices of the $K_{\Delta+1}$. Assume, by the sake of contradiction, that the function $\pi\colon V(G) \to V(K_{\Delta+1})$ defined by $\pi(u_i) \coloneqq i \bmod (\Delta + 1)$ is *not* a pullback from G to the $K_{\Delta+1}$. As π is clearly a homeomorphism, there must be two distinct vertices v_1 and v_2 in $V(G)$ which have a neighbour w in common and satisfy $\pi(v_1) = \pi(v_2)$. However, since σ is a proper circular-arc order of G, all vertices between v_1 and v_2 in σ are thus neighbours of w, which straightforwardly implies $d_G(w) > \Delta$. □

Theorem 2. *If* $n \not\equiv k \pmod{(\Delta+1)}$*, for all* $k \in \{1, \Delta\}$*, and* G *has a maximal clique of size 2, then* G *is:* Class 1 *and* $(\Delta + 2)$*-total colourable if* Δ *is odd;* Type 1 *if* Δ *is even.*

Proof. If $r \coloneqq n \bmod (\Delta + 1) = 0$, we are done by Theorem 1. If $\Delta \leq 2$, then G is a cycle or a disjoint union of paths and the theorem clearly holds. Hence, we assume that $\Delta \geq 3$ and $r \neq 0$. Let $\sigma \coloneqq u_0, \ldots, u_{n-1}$ be a proper circular-arc order of G, being $\{u_0, u_{n-1}\}$ a maximal clique. Because σ is a proper circular-arc order, we have $u_\Delta \notin N_G(u_0)$ and $u_{n-1-\Delta} \notin N_G(u_{n-1})$, otherwise $d_G(u_0) > \Delta$ or $d_G(u_{n-1}) > \Delta$.

Let $V(K_{\Delta+1}) \coloneqq \{0, \ldots, \Delta\}$ and let $\varphi \in \{\varphi^{\text{even}}_{\text{edge}}, \varphi_{\text{total}}\}$ be the canonical total or edge-colouring of the $K_{\Delta+1}$. The function $\pi\colon V(G') \to V(K_{\Delta+1})$ defined by $\pi(u_i) \coloneqq i \bmod (\Delta + 1)$ is clearly a pullback from $G' \coloneqq G - u_{n-1}u_0$ to the $K_{\Delta+1}$ and brings a total or an edge-colouring ψ of G' using the same set of colours as φ. Ergo, we have only to colour $u_{n-1}u_0$ in order to complete the proof.

Observe that $\pi(u_{n-1}) = r - 1$, $\pi(u_{n-1-\Delta}) = r$, and, since neither r nor $r-1$ is Δ, $\varphi(r, r-1) = (2r-1) \bmod d =: q$, with $d \coloneqq \Delta$ if $\varphi = \varphi^{\text{even}}_{\text{edge}}$, or $d \coloneqq \Delta+1+\text{even}(\Delta+1)$ if $\varphi = \varphi_{\text{total}}$. Therefore, as $\pi(v) \neq \Delta$ and $\pi(w) \neq r$ for all $v \in N_{G'}(u_0)$ and all $w \in N_{G'}(u_{n-1})$, the colour $\varphi(0, \Delta)$ is missing at u_0 and the colour q at u_{n-1}. If $q = \varphi(0, \Delta)$, then we assign the colour q to $u_{n-1}u_0$ and we are done. Otherwise, since $q \in \{0, \ldots, \Delta\}$, we exchange Δ and q in the codomain of π, that is, we redefine π so that every vertex which has been mapped by π to Δ is now mapped to q and vice versa. Notice that the images of u_0, $u_{n-1-\Delta}$, and u_{n-1} by π remain the same, but $\pi(u_\Delta)$ becomes q, which now is also a colour missing at u_0. Then, we colour $u_{n-1}u_0$ with q. □

3. Final remarks

Let $\mathcal{A}$ be the class of the proper circular-arc graphs with odd Δ and a maximal clique of size 2. Overfull graphs in $\mathcal{A}$ can be constructed for $n \equiv 1$ and for $n \equiv \Delta \pmod{(\Delta+1)}$ (see Figures 2(a) and 2(b), respectively). Since Theorem 2 can be interestingly used to

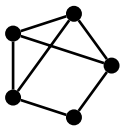
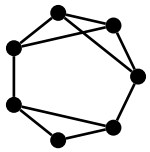
a)  b) 

Figure 2. Two overfull graphs in $\mathcal{A}$

show a graph in $\mathcal{A}$ is *SO* if and only if it is overfull, we conclude proposing the following:

Conjecture. *A graph in $\mathcal{A}$ is* Class 2 *if and only if it is overfull.*

References

Behzad, M. (1965). *Graphs and their chromatic numbers*. PhD thesis, Michigan State University.

Behzad, M., Chartrand, G., and Cooper Jr., J. K. (1967). The color numbers of complete graphs. *J. London Math. Soc.*, 42:226–228.

Booth, K. S. and Lueker, G. S. (1976). Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *J. Comput. Syst. Sci.*, 13:335–379.

Campos, C. N., Figueiredo, C. M. H., Machado, R., and Mello, C. P. (2012). The total chromatic number of split-indifference graphs. *Discrete Math.*, 312:2690–2693.

Figueiredo, C. M. H., Meidanis, J., and Mello, C. P. (1997). On edge-colouring indifference graphs. *Theor. Comput. Sci.*, 181:91–106.

Figueiredo, C. M. H., Meidanis, J., Mello, C. P., and Ortiz, C. (2003). Decompositions for the edge colouring of reduced indifference graphs. *Theor. Comput. Sci.*, 297:145–155.

Garey, M. R., Johnson, D. S., Miller, G. L., and Papadimitriou, C. H. (1980). The complexity of coloring circular arcs and chords. *SIAM J. Algebraic Discrete Methods*, 1(2):216–227.

Grötschel, M., Lovász, L., and Schrijver, A. (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197.

Hilton, A. J. W. and Johnson, P. D. (1987). Graphs which are vertex-critical with respect to the edge-chromatic number. *Math. Proc. Cambridge Philos. Soc.*, 102:103–112.

Holyer, I. (1981). The NP-completeness of edge-colouring. *SIAM J. Comput.*, 10(4):718–720.

McConnell, R. M. (2003). Linear-time recognition of circular-arc graphs. 37:93–147.

Roberts, F. S. (1969). Indifference graphs. In *Proc. 2nd Ann Arbor Graph Theory Conference*, pages 139–146, Ann Arbor, USA.

Sánchez-Arroyo, A. (1989). Determining the total colouring number is NP-hard. *Discrete Math.*, 78:315–319.

Vizing, V. G. (1964). On an estimate of the chromatic class of a p-graph (in Russian). *Diskret. Analiz.*, 3:25–30.

Vizing, V. G. (1968). Some unsolved problems in graph theory. *Russian Math. Surveys*, 23:125–141.

Número de envoltória em classes de grafos orientados*

J. Araujo[1] **, P. Arraes**[1]

[1]ParGO – Departamento de Matemática – Universidade Federal do Ceará (UFC)

julio@mat.ufc.br, pedro.arraes@alu.ufc.br

Abstract. *In this paper we study the hull number for some classes of oriented graphs. First we present an upper bound for the hull number of tournaments, and a tournament for which that bound is attained. Next we prove that the hull number problem is NP-complete for bipartite oriented graphs. For that we use the result of [Araujo et al. 2013], which asserts that the hull number problem is NP-complete for bipartite non-oriented graphs. After that we show a charecterization for the smallest hull set of a tree. Moreover, we generalize this result by showing a polynomial-time algorithm that computes the hull number of any oriented cactus graph.*

Resumo. *Neste trabalho estudamos o número de envoltória para algumas classes de grafos orientados. Primeiramente apresentamos um limitante superior para o número de envoltória restrito a torneios, além de um torneio para o qual atingimos esse limite. Em seguida provamos que esse problema é NP-completo para grafos bipartidos orientados. Para tanto utilizamos o resultado de [Araujo et al. 2013], o qual afirma que tal problema é NP-completo para grafos bipartidos não-orientados. Depois mostramos uma caraterização para o menor conjunto de envoltória de uma árvore orientada. Além disso, generalizamos esse resultado ao mostrar um algoritmo de tempo polinomial para calcular o número de envoltória de qualquer grafo cacto orientado.*

1. Introdução

Nessa seção apresentaremos algumas definições básicas de grafos orientados e o suficiente para se compreender o problema do número de envoltória. Para outras definições em grafos, veja [West 2000]; para definições sobre algoritmos e complexidade computacional, sugere-se [Garey and Johnson 1990].

Um *grafo orientado* é um grafo direcionado cujo grafo subjacente é simples. Dados um grafo orientado D e $u, v \in V(D)$, um (u,v)*-caminho direcionado* P é um subgrafo orientado de D tal que $V(P) = \{u = v_1, v_2, \ldots, v_k = v\}$ e $A(P) = \{(v_i, v_{i+1}) \mid 1 \leq i \leq k-1\}$; esse, por sua vez, é denotado por $(v_1, v_2, \ldots, v_k)$. Um *ciclo direcionado* C é um ciclo orientado tal que $V(C) = \{v_1, v_2, \ldots, v_k\}$ e $A(C) = \{(v_i, v_{i+1}) \mid 1 \leq i \leq k-1\} \cup \{(v_k, v_1)\}$.

Dados um grafo orientado D e vértices $u, v \in V(D)$, um (u,v)*-geodésico* é um caminho direcionado de u para v com menor comprimento possível, enquanto $d_D(u,v)$ denota a quantidade de arcos num (u,v)-geodésico. Perceba que um (u,v)-geodésico não é um (v,u)-geodésico (e vice-versa) e pode acontecer de $d_D(u,v) \neq d_D(v,u)$.

*Esta pesquisa foi financiada pelo CNPq sob projetos 459466/2014-3, 310234/2015-8 e 401519/2016-3.

A *função de intervalo* $I : \mathcal{P}(V(D)) \to \mathcal{P}(V(D))$ aplicada em $S \subseteq V(D)$ retorna o conjunto $I[S]$, composto pelos vértices de todos os (u, v)-geodésicos e de todos os (v, u)-geodésicos com $u, v \in S$. Para todo inteiro $n \geq 2$ definimos $I^n[S] := I[I^{n-1}[S]]$ e $I^0[S] = S$. Um conjunto S é *convexo* se $I[S] = S$, ou seja, para todo vértice w em algum (u, v)-geodésico com $u, v \in S$ temos $w \in S$. Em contrapartida, S é dito *co-convexo* se $V(D) \setminus S$ é convexo. A *envoltória* de $S \subseteq V(D)$, denotada por $[S]$, é o menor conjunto convexo que contém S. Esse pode ser obtido aplicando a função de intervalo iterativamente em S até obtermos um conjunto S' tal que $I[S'] = S'$. Tal conjunto sempre é atingido pois, como os grafos tratados aqui são finitos, em alguma iteração deixaremos de adicionar vértices. Isso quer dizer que existe um natural mínimo n tal que $I^{n+1}[S] = I^n[S] = [S]$.

Quando a envoltória de um $S \subseteq V(D)$ é todo o conjunto de vértices do grafo orientado, dizemos que S é um *conjunto de envoltória* de D. Além disso, se S for um conjunto de envoltória de D mínimo, definimos o *número de envoltória* de D por $\overrightarrow{hn}(D) := |S|$. De forma similar, S é dito um *conjunto geodético* de D quando $I[S] = V(D)$ e, no caso em que esse é mínimo, definimos o *número geodético* de D por $\overrightarrow{gn}(D) := |S|$.

Em grafos orientados, existem vértices que exercem um papel importante no problema do número de envoltória: os *extremais*. Esses podem ser de um dos três tipos a seguir: *transmissor*: $d_D^-(v) = 0$ e $d_D^+(v) > 0$; *receptor*: $d_D^+(v) = 0$ e $d_D^-(v) > 0$; e *transitivo*: $d_D^-(v) > 0$, $d_D^+(v) > 0$ e, para qualquer par de arcos (u, v), $(v, w) \in A(D)$, temos $(u, w) \in A(D)$.

Lema 1 ([Chartrand et al. 2003])**.** Todo conjunto de envoltória (geodético) de um grafo orientado conexo D deve conter os vértices extremais de D. Em particular, se o conjunto de vértices extremais for um conjunto de envoltória (geodético), então esse é o único conjunto de envoltória (geodético) mínimo.

A maioria dos trabalhos na literatura sobre Convexidade em Grafos lida com o caso não-orientado. Ademais, os que lidam com o caso orientado focam nas orientações de grafos cujo número de envoltória é ou o maior ou o menor possível.

Nossa primeira contribuição é um limitante superior para $\overrightarrow{hn}(D)$ quando D é um torneio, além de um exemplo no qual esse é atingido. Em seguida mostramos que, restrito à classe de bipartidos orientados, o problema do número de envoltória é ***NP***-completo.

Depois provamos que o conjunto de vértices extremais é um conjunto de envoltória mínimo para grafos árvore orientados. Para a classe de cactos orientados, mostramos como obter um conjunto de envoltória mínimo. Finalmente, discutimos os resultados obtidos e apresentamos problemas em aberto na última seção.

2. Torneios

Um *torneio* é um grafo completo orientado. Nossa abordagem inicial foi tomar a envoltória do conjunto composto pelos extremais e analisar os vértices que não fazem parte dela. Observamos que, para um subconjunto $\mathcal{S} \subset V(D)$ qualquer, os vértices extremais de $\mathcal{S}$ não influenciam na adição de outros vértices em $I[\mathcal{S}]$. Veja que, aplicando esse mesmo argumento mais $n - 1$ vezes, podemos afirmar o mesmo para $I^n[\mathcal{S}]$.

Portanto, sendo S o conjunto composto pelos vértices extremais do torneio D, $\mathcal{S}$ é um conjunto de envoltória de D se $S \subseteq \mathcal{S}$ e $[\mathcal{S} \setminus S]_D = V(D) \setminus S$. Ou seja, nosso problema se resume a encontrar um conjunto de vértices (não extremais) cuja envoltória seja composta por todos os vértices não extremais. Abaixo mostramos que, para obter todos os vértices de $V(D)$, além dos extremais precisamos de no máximo dois terços dos outros vértices.

Proposição 1. Seja D um torneio e S o conjunto de vértices extremais de D. Então $\overrightarrow{hn}(D) \leq |S| + \frac{2}{3}(n(D) - |S|)$ e esse limite é apertado.

Para obter um torneio no qual o limite acima é atingido basta fazer o produto lexicográfico de $K_3 \equiv C_3$, orientado de forma a ser um ciclo direcionado, com um K_5, orientado de forma transitiva, de modo que cada vértice do segundo é substituido pelo C_3 direcionado.

3. Bipartidos

O objetivo dessa seção é mostrar que, dados um grafo orientado D e um inteiro positivo k, decidir se $\overrightarrow{hn}(G) \leq k$ é um problema NP-completo, mesmo que G seja bipartido. Para tanto, fazemos uso do resultado abaixo, o qual mostra a dificuldade desse problema para grafos bipartidos.

Teorema 1 ([Araujo et al. 2013])**.** Dados um grafo G e um inteiro positivo k, decidir se $\mathrm{hn}(G) \leq k$ é um problema ***NP***-completo, mesmo se G for um grafo bipartido.

O método que usaremos para provar o resultado final é o seguinte. Dado um grafo bipartido G podemos construir um bipartido orientado $G_{\overrightarrow{C4}}$, pela substituição de cada aresta por um ciclo direcionado com 4 vértices, cujo número de envoltória é o mesmo.

Teorema 2. Se G for um grafo bipartido, então $hn(G) = \overrightarrow{hn}(G_{\overrightarrow{C4}})$.

Corolário 1. Dados um grafo bipartido orientado G e um inteiro positivo k, decidir se $\overrightarrow{hn}(G) \leq k$ é um problema ***NP***-completo.

4. Árvores e Cactos

Antes de mostrarmos um algoritmo polinomial para determinar o número de envoltória de um cacto orientado, vamos mostrar um resultado sobre uma subclasse de cactos: as árvores. Para essa classe, tentamos a mesma abordagem dos torneios: tomar a envoltória dos extremais.

Proposição 2. Seja D uma árvore orientada e $S \subseteq V(D)$ o conjunto formado pelos vértices extremais de D. S é um conjunto de envoltória e geodético de D.

Um grafo G é dito um *cacto* se cada um de seus blocos ou é um ciclo induzido ou é uma aresta. Uma maneira mais informal de definir essa classe é que quaisquer dois subgrafos ciclos do grafo possuem no máximo um vértice em comum, o qual seria uma articulação. Descobrimos que além dos extremais existem ciclos no cacto tais que todo conjunto de envoltória do grafo deve ter pelo menos um de seus vértices. A esses ciclos damos o nome de *insatisfeitos*. Além disso, vimos que esses vértices não são apenas necessários num conjunto de envoltória, eles são suficientes.

Teorema 3. Seja D um cacto orientado. Então, existe um conjunto de envoltória mínimo S de D composto pelos vértices extremais e exatamente um vértice (não-extremal) de

cada ciclo insatisfeito tal que $I^2[S] = V(D)$ e todos os vértices que podem não estar em $I[S]$ pertencem a ciclos satisfeitos. Consequentemente, todo conjunto de envoltória mínimo S de D é composto pelos vértices extremais e exatamente um vértice (não-extremal) de cada ciclo insatisfeito.

5. Conclusões

Nas Seções 2 e 4, abordamos o problema do número de envoltória da mesma forma: analisando a envoltória dos extremais e formulando uma maneira de obter os demais. Um caminho natural é tentar utilizar esse método para encontrar mais resultados desse parâmetro em outras classes de grafos orientados.

Seguindo essa linha de raciocínio, será possível aplicar um método similar utilizado na Seção 3 em subclasses de bipartidos para provar a *NP*-completude? Por exemplo, foi demonstrado em [Albenque and Knauer 2016] que encontrar o número de envoltória também é *NP*-difícil para cubos parciais. *Cubos parciais* são subgrafos isométricos de hipercubos, os quais por sua vez são bipartidos.

Além disso, ainda do ponto de vista de Complexidade Computacional, há inúmeras classes de grafos orientados a serem abordadas, por exemplo: cordais, planares, grafos com largura em árvore limitada, etc. Também não conhecemos resultados do ponto de vista de Complexidade Parametrizada para problemas de convexidade em grafos orientados. Finalmente, só encontramos na literatura resultados sobre os parâmetros número de envoltória e número geodético, ou seja, todos os demais parâmetros de convexidade ainda devem ser estudados no caso orientado [Chartrand et al. 2002, Duchet 1988].

Referências

Albenque, M. and Knauer, K. (2016). Convexity in partial cubes: The hull number. *Discrete Mathematics*, 339(2):866 – 876.

Araujo, J., Campos, V., Giroire, F., Nisse, N., Sampaio, L., and Soares, R. (2013). On the hull number of some graph classes. *Theoretical Computer Science*, 475(Supplement C):1 – 12.

Chartrand, G., Fink, J. F., and Zhang, P. (2003). The hull number of an oriented graph. *International Journal of Mathematics and Mathematical Science*, 2003(36):2265–2275.

Chartrand, G., Wall, C., and Zhang, P. (2002). The convexity number of a graph. *Graphs and Combinatorics*, 18(2):209–217.

Duchet, P. (1988). Convex sets in graphs, ii. minimal path convexity. *Journal of Combinatorial Theory, Series B*, 44(3):307–316.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

West, D. B. (2000). *Introduction to Graph Theory*. Prentice Hall, 2 edition.

Fluxos Ramificados Arco-disjuntos em Redes de Capacidade Restrita*

A. Karolinna Maia[1], **Jonas Costa**[1,2], **Raul Lopes**[1,2]

[1]Departamento de Computação - Universidade Federal do Ceará (UFC)
Grupo de Pesquisa ParGO
Fortaleza – CE – Brazil

karolmaia@lia.ufc.br, jonascosta@lia.ufc.br, raul.wayne@gmail.com

Abstract. *The problem of determining if a given network has a feasible flow is largely studied and known to be polynomial-time solvable. In this work, we consider an specific type of flow, called branching flow, which can be useful for finding branchings in digraphs. We focus on the problem of finding multiple arc-disjoint branching flows on a given network and we study its complexity on networks with different capacities on the arcs. A previous result shows that, under a common theoretical assumption (ETH), in networks with n vertices where all the arcs have capacity equal to $n - f(n)$, for a bounded function f, this problem is hard. We extended this result showing that, under the same assumption, the problem is also hard when the capacities are equal to $f(n)$.*

Resumo. *Determinar se uma rede possui um fluxo viável é um problema amplamente estudado e que pode ser resolvido em tempo polinomial. Investigamos um tipo específico de fluxo, o fluxo ramificado, que pode ser aplicado na busca por ramificações em digrafos. Consideramos o problema de encontrar múltiplos fluxos ramificados arco-disjuntos em uma rede e estudamos a complexidade deste problema com diferentes capacidades nos arcos. Um resultado anterior prova que, sob uma suposição teórica conhecida (ETH), em redes com n vértices tais que todos os arcos possuem capacidade $n - f(n)$, para uma função limitada f, o problema é difícil. Estendemos este resultado mostrando que, sob a mesma hipótese, o problema também é difícil quando as capacidades são iguais a $f(n)$.*

Introdução

Neste trabalho, seguimos a terminologia básica contida em [Bang-Jensen and Gutin 2008]. Uma *rede* $\mathcal{N} = (V, A, u)$ é definida por um digrafo $D = (V, A)$ e uma função de capacidade $u : A \to \mathbb{Z}_+$. Usamos a notação u_{vw} para nos referir a $u(vw)$, para todo arco $vw \in A$. Podemos escrever $\mathcal{N} = (V, A, u \equiv l)$ quando a capacidade de todos os arcos da rede é l.

Um *fluxo* em uma rede $\mathcal{N}$ é uma função $x : A \to \mathbb{Z}_+$, onde x_{vw} denota o valor de x no arco vw. O *balanço de um fluxo* x é uma função que atribui a cada vértice $v \in V$ o valor

$$b_x(v) = \sum_{vw \in A} x_{vw} - \sum_{zv \in A} x_{zv}.$$

[2]Financiado pela CAPES.

*Este trabalho foi parcialmente apoiado pelo CNPq - projeto Universal 425297/2016-0 e FUNCAP - PRONEM PNE-0112-00061.01.00/16.

Um fluxo x em $\mathcal{N}$ é dito *viável* se $x_{vw} \leq u_{vw}$, para todo arco $vw \in A$. Frenquentemente, no estudo do problema de fluxo é considerado a sua conservação, isto é, $b_x(v) = 0$ para todo vértice $v \in V$.

Fluxos em redes são amplamente estudados, pois além de modelarem problemas práticos de grande importância, também despertam grande interesse teórico sendo várias vezes uma ferramenta útil para a resolução de problemas em grafos e digrafos. Uma vasta coleção de resultados de fluxo é apresentada em [Ahuja 1993].

As variações mais conhecidas de problemas de fluxo podem ser resolvidas em tempo polinomial. No entanto, existem problemas de otimização que lidam com caminhos e ciclos em digrafos e se assemelham tipicamente a fluxos, mas que não podem ser modelados em sua forma tradicional. Um exemplo clássico é o problema de encontrar k caminhos arco-disjuntos em um digrafo D. Neste, recebemos como entrada $2k$ vértices $s_1, s_2, \ldots, s_k, t_1, t_2, \ldots, t_k \in V(D)$ e queremos decidir se existem k caminhos arco-disjuntos $P_1, P_2, \ldots, P_k$ em D tal que P_i é um (s_i, t_i)-caminho para $i \in \{1, 2, \ldots, k\}$. Este problema foi mostrado ser $\mathcal{NP}$-completo em [Fortune et al. 1980] para $k \geq 2$. Com o objetivo de obter uma estrutura mais geral e abranger alguns desses casos, [Bang-Jensen and Bessy 2014] começaram a investigar a existência múltiplos fluxos viáveis arco-disjuntos em uma rede pré-determinada. Dois fluxos x e y em $\mathcal{N}$ são ditos arco-disjuntos se $x_{vw} \cdot y_{vw} = 0$ para todo arco $vw \in A$.

Uma *ramificação* em um digrafo D é uma árvore geradora de D na qual todos os vértices, com exceção de um chamado de *raiz*, possuem grau de entrada um. Uma *s-ramificação* é uma ramificação cuja a raiz é o vértice s. O estudo de ramificações em digrafos também possui motivações práticas. Por exemplo, em um contexto em que se deseje enviar informações de uma fonte para todos os demais vértices do digrafo, os arcos de uma ramificação podem ser utilizados para fazer o roteamento das mensagens. Porém, essas estruturas podem não ser muito seguras. Por esse motivo se torna interessante o estudo de medidas de qualidade de uma ramificação. Alguns exemplos de parâmetros considerados podem ser vistos em [Bang-Jensen and Yeo 2015, Bang-Jensen et al. 2016].

Um *fluxo s-ramificado* em uma rede $\mathcal{N} = (V, A, u)$ é um fluxo x no qual $b_x(s) = n - 1$ e $b_x(v) = -1$ para todo $v \in V \setminus \{s\}$, onde n é o número de vértices de $\mathcal{N}$. Neste tipo de fluxo cada vértice da rede recebe pelo menos uma unidade de fluxo. Observe que o subgrafo de $\mathcal{N}$ definido por V e pelos arcos em que o valor de $x \neq 0$ não necessariamente é uma s-ramificação.

Em [Bang-Jensen and Bessy 2014], os autores introduzem o problema de decidir se existem 2 fluxos s-ramificados arco-disjuntos em uma rede $\mathcal{N}$, e [Bang-Jensen et al. 2016] se aprofundaram nessa questão, considerando diferentes capacidades para $\mathcal{N}$. No presente trabalho, resolvemos um importante caso deixado em aberto relativo ao mesmo problema, além de termos obtido uma prova alternativa do caso complementar apresentada por [Bang-Jensen et al. 2016]. Devido a restrições de espaço, todas as provas serão omitidas.

Fluxos ramificados em redes com capacidades específicas

Através de uma prova algorítmica dada por [Lovász 1976] para um teorema originalmente descrito por [Edmonds 1973], é possível obter um algoritmo polinomial para

decidir se uma rede $\mathcal{N} = (V, A, u \equiv n - 1)$ admite k fluxos s-ramificados $x^1, x^2, \ldots, x^k$ arco-disjuntos. Reduzir a capacidade dos arcos por um fator constante não é suficiente para alterar a dificuldade do problema quando consideramos apenas dois fluxos:

Teorema 1. *[Bang-Jensen et al. 2016] Para todo inteiro positivo k fixo, existe um algoritmo que recebe como entrada uma rede $\mathcal{N} = (V, A, u \equiv n - k)$ e um vértice especial s e decide em tempo polinomial se $\mathcal{N}$ admite dois fluxos s-ramificados arco-disjuntos.*

Um resultado de [Bang-Jensen and Bessy 2014] mostra que é polinomial decidir se uma rede com capacidade unitárias $(u \equiv 1)$ possui dois fluxos arco-disjuntos com o mesmo balanço. Note que, em particular, o resultado vale para fluxos ramificados. Por outro lado, o seguinte teorema foi provado:

Teorema 2. *[Bang-Jensen and Bessy 2014] É $\mathcal{NP}$-completo decidir se uma rede $\mathcal{N} = (V, A, u)$, com $u_{vw} \in \{1, 2\}$ $\forall vw \in A$, possui dois fluxos s-ramificados arco-disjuntos.*

O Teorema 2 é utilizado para provar que se aumentarmos a capacidade em no máximo uma constante o problema continua $\mathcal{NP}$-completo.

Teorema 3. *[Bang-Jensen et al. 2016] Para todo inteiro $k \geq 2$ fixo, é $\mathcal{NP}$-completo decidir se uma rede $\mathcal{N} = (V, A, u \equiv k)$ possui dois fluxos s-ramificados arco-disjuntos.*

Pelos resultados anteriores, observamos que, aparentemente, ao aumentarmos a capacidade das arestas de $\mathcal{N}$, o problema de encontrar fluxos arco-disjuntos pode apenas se tornar mais fácil. Dessa forma, conjecturamos que deve existir algum valor $\ell \in \{2, ..., n - 1\}$ onde é difícil decidir se uma rede com arcos de capacidade $u \leq \ell$ admite dois fluxos s-ramificados e arco-disjuntos, porém o problema é fácil para capacidades maiores (Teorema 1).

Estendemos resultados obtidos em [Bang-Jensen et al. 2016] para mostrar que, como consequência da *Hipótese de Tempo Exponencial* (ETH) [Impagliazzo et al. 2001], o problema considerado é difícil para quase todos os valores possíveis de ℓ. A ETH é uma conjectura que afirma não existir algoritmo para resolver o problema 3-SAT que seja sub-exponencial no número de variáveis.

Mostramos que a existência de um algoritmo polinomial no número de vértices para o problema, quando consideramos as capacidades por uma função limitada de n, implicaria na existência de um algoritmos sub-exponencial para 3-SAT.

Teorema 4. *Assumindo válida a ETH, seja $\epsilon > 0$ arbitrário e $f(n)$ uma função inteira tal que $\log(n)^{1+\epsilon} \leq f(n) \leq n/2$ para todo $n > 0$. Além disso, suponha que existe uma constante C^* no qual para todo $c \geq C^*$ existe um n tal que $f(n) = c$. Dadas essas condições, seja $\mathcal{N} = (V, A, u \equiv f(n))$ uma rede com n vértices e um vértice especial $s \in V$. Não existe algoritmo polinomial (em n) para decidir se $\mathcal{N}$ possui dois fluxos s-ramificados arco-disjuntos.*

O Teorema 4, principal contribuição deste trabalho, considera o caso de redes com capacidades $f(n)$. A demonstração segue uma adaptação da demonstração para o caso de redes com capacidades $n - f(n)$ contida em [Bang-Jensen et al. 2016]. Ressaltamos que, com algumas alterações, a prova oferecida para o Teorema 4 se mostra

como uma prova alternativa para o resultado já conhecido para redes com capacidades $n - f(n)$. A Figura 1 ilustra para quais capacidades a complexidade do problema considerado é conhecida. Segmentos contínuos marcam resultados previamente conhecidos ([Bang-Jensen et al. 2016, Bang-Jensen and Bessy 2014]) e o segmento curvo marca nossa contribuição.

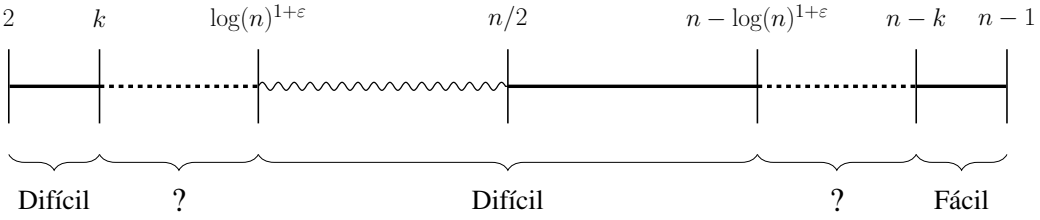


Figura 1. Intervalos de capacidades e complexidades.

Referências

Ahuja, R. (1993). *Network flows : theory, algorithms, and applications*. Prentice Hall, Englewood Cliffs, N.J.

Bang-Jensen, J. and Bessy, S. (2014). (Arc-) disjoint flows in networks. *Theoretical Computer Science*, 526:28–40.

Bang-Jensen, J. and Gutin, G. Z. (2008). *Digraphs: theory, algorithms and applications*. Springer Science & Business Media.

Bang-Jensen, J., Havet, F., and Yeo, A. (2016). The complexity of finding arc-disjoint branching flows. *Discrete Applied Mathematics*, 209:16–26.

Bang-Jensen, J. and Yeo, A. (2015). Balanced branchings in digraphs. *Theoretical Computer Science*, 595:107 – 119.

Edmonds, J. (1973). Edge-disjoint branchings. *Combinatorial Algorithms*, pages 91–96.

Fortune, S., Hopcroft, J., and Wyllie, J. (1980). The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111 – 121.

Impagliazzo, R., Paturi, R., and Zane, F. (2001). Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512 – 530.

Lovász, L. (1976). On two minimax theorems in graph. *Journal of Combinatorial Theory, Series B*, 21(2):96 – 103.

Politopo baseado em distâncias para o problema clássico de coloração de vértices em grafos

Bruno Dias[1], **Rosiane de Freitas**[1], **Javier Marenco**[2], **Nelson Maculan**[3]

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Av. Rodrigo Otávio, 3000 – 69000-000 – Manaus – AM – Brasil

[2]Departamento de Computación - Universidad de Buenos Aires (UBA)
Buenos Aires – Argentina

[3]PESC/COPPE – Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 68530 – 21945-970 – Rio de Janeiro – RJ – Brasil

{bruno.dias, rosiane}@icomp.ufam.edu.br, jmarenco@dc.uba.ar,
maculan@cos.ufrj.br

Abstract. *In this work, we present a distance-based model for the classic vertex coloring problem in graphs (VCP). The integer linear programming formulation uses decision variables representing the distance between the colors assigned to every pair of distinct vertices instead of explicitly providing the colors assigned to each vertex. We show close relations between this formulation and the so-called orientation model for VCP, also proposed by the authors of this work. In particular, we prove that we can translate many facet-inducing inequalities for the orientation model polytope into facet-inducing inequalities for the distance model polytope, and vice versa.*

Resumo. *Neste trabalho, apresenta-se o modelo baseado em distâncias para o problema clássico de coloração de vértices em grafos (VCP). A formulação de programação linear inteira utiliza variáveis de decisão que representam a distância entre cores atribuídas para cada par de vértices distintos, no lugar de fornecer explicitamente tais cores. Mostra-se que há uma relação próxima entre esta formulação e o modelo baseado em orientação para o VCP, proposto também pelos autores deste trabalho. Em particular, prova-se que desigualdades indutoras de facetas para o modelo baseado em orientação podem ser traduzidas em desigualdades indutoras de facetas para o modelo baseado em distâncias e vice-versa.*

1. Introdução

O problema clássico de coloração de vértices em grafos (em inglês, *vertex coloring problem* - VCP) consiste em, para um grafo simples não direcionado $G = (V, E)$, encontrar um mapeamento $c : V \to \mathbb{Z}_{\geq 0}$ tal que $c(i) \neq c(j)$ para toda aresta $(i, j) \in E$ (ou, equivalentemente, $|c(i) - c(j)| \geq 1$) onde a quantidade de cores usadas seja a menor possível. Esse problema é um dos mais conhecidos em otimização combinatória e teoria dos grafos, com diversas aplicações, como alocação de canais em redes sem fio [Dias 2014, Koster 1999], escalonamento de tarefas [de Freitas et al. 2010], ensalamento escolar [Burke et al. 2010] e outros. Porém, o VCP é NP-difícil [Karp 1972], exigindo

a utilização de técnicas avançadas para a obtenção de soluções ótimas ou próximas do ótimo em tempo computacional aceitável.

Diversos modelos de programação inteira (PI) foram propostos para o VCP, tais como: clássico baseado em restrições de atribuição, usado em método *branch-and-cut* [Méndez-Díaz and Zabala 2006]; baseado em conjuntos independentes maximais, utilizado em métodos de geração de colunas [Mehrotra and Trick 1996]; e formulação de representantes assimétricos, onde vértices são eleitos como representantes das classes de cores usadas [Campêlo et al. 2008].

Um outro modelo proposto é o baseado em orientação, proposto originalmente para a coloração em largura de banda (*bandwidth coloring problem* - BCP) [Dias et al. 2017], na qual as soluções induzem a uma orientação do grafo de entrada. Com base nesta formulação, deriva-se um modelo baseado em distâncias, onde, ao invés de determinar-se as cores explicitamente, as distâncias entre cores de vértices adjacentes devem ser retornadas, o que explora a correlação entre coloração de vértices e geometria de distâncias [Dias et al. 2013]. Sendo assim, neste trabalho, apresenta-se um estudo poliedral inicial dessa formulação baseada em distâncias, para a qual fornece-se duas famílias de desigualdades válidas indutoras de facetas do politopo associado.

2. Formulação baseada em distância

Dado um grafo $G = (V, E)$ e um conjunto de cores $C = \{1, 2, \ldots, |V|\}$, o modelo baseado em distâncias utiliza variáveis inteiras x_{ij} para todo $i, j \in V$ tal que $i < j$ que denotam a diferença entre as cores de i e j (de modo que $x_{ij} = c(i) - c(j)$). Além disso, também são usadas as variáveis de orientação y_{ij}, com valor 1 se $x_{ij} < 0$ e 0 caso contrário. Uma solução factível para a coloração de vértices é dada pelas restrições a seguir:

$$x_{ik} = x_{ij} + x_{jk} \qquad \forall i, j, k \in V,\ i < j < k \tag{1}$$
$$x_{ij} \geq 1 - |C| y_{ij} \qquad \forall (i, j) \in E,\ i < j \tag{2}$$
$$x_{ij} \leq -1 + |C|(1 - y_{ij}) \qquad \forall (i, j) \in E,\ i < j \tag{3}$$
$$x_{ij} \in \{-|C| + 1, \ldots, |C| - 1\} \qquad \forall i, j \in V,\ i < j \tag{4}$$
$$y_{ij} \in \{0, 1\} \qquad \forall (i, j) \in E,\ i < j \tag{5}$$

As restrições (1) refletem a separação de cores entre os vértices, inclusive entre os não adjacentes entre si, de acordo com a interpretação das variáveis x_{ij}. Os conjuntos de restrições (2) e (3) induzem à orientação do grafo de acordo com as cores atribuídas. Por fim, (4) e (5) referem-se a integralidade e limites das variáveis.

O conjunto de restrições é composto de $O(|V|^3)$ equações, porém, o mesmo pode ser substituído por um conjunto de $O(|V|^2)$ equações, como visto a seguir.

Teorema 1 *Se $V = \{1, \ldots, n\}$, então as restrições (1) equivalem a:*

$$x_{i,i+1} + x_{i+1,i+2} = x_{i,i+2} \qquad \forall i \in V, i \leq n - 2 \tag{6}$$
$$x_{ij} + x_{i+1,j-1} = x_{i,j-1} + x_{i+1,j} \qquad \forall i, j \in V, i \leq n - 3, i + 3 \leq j \tag{7}$$

Seja $PD(G, C)$ o fecho convexo dos vetores (x, y) que satisfazem as restrições (1)-(5). Para tal politopo, algumas propriedades do modelo são enunciadas, sendo a primeira relacionada à dimensão de $PD(G, C)$, como visto a seguir.

Teorema 2 *Se* $|C| > \chi(G) + 1$*, então* $\dim(PD(G, C)) = |V| + |E| - 1$.

Como dito anteriormente, o modelo baseado em orientação serve de base para esta formulação. Nele, existem variáveis inteiras $z_i \in \{1, \ldots, |C|\}$ para cada $i \in V$, além das mesmas variáveis y. Pode-se então transportar desigualdades válidas indutoras de facetas do politopo $PO(G, C)$ do modelo com orientação para a formulação baseada em distâncias, como definido a seguir.

Teorema 3 *Seja* $\alpha z_i + \pi y \leq \alpha z_j + \pi_0$ *uma desigualdade válida (indutora de faceta) para* $PO(G, C)$*, onde* $(i, j) \in E$*. Então,* $\alpha x_{ij} + \pi y \leq \pi_0$ *é válida (indutora de faceta se* $|C| \geq \chi(G) + 2$*) para* $PD(G, C)$ *e vice-versa.*

3. Facetas para modelo baseado em distâncias

Uma importante implicação do Teorema 3 é que pode-se transformar desigualdades válidas indutoras de facetas do modelo baseado em orientação ($PO(G, C)$) para o modelo baseado em distâncias ($PD(G, C)$). A seguir, são mostradas duas famílias de desigualdades para $PO(G, C)$ e suas transformações, por meio do Teorema 3, para $PO(G, C)$. Ressalta-se que, nas expressões abaixo, o termo z_i é a variável de $PO(G, C)$ que indica a cor do vértice $i \in V$ (com $z_i \in \mathbb{Z}_{\geq 0}$). Também considere que $y_{ji} = 1 - y_{ij}$ para $i < j$.

3.1. Desigualdade clique

Seja $i \in V$ e $K \subseteq N(i)$ e uma clique. Define-se então a desigualdade clique associada a i e K para o $PO(G, C)$ do seguinte modo:

$$z_i \ \geq \ \sum_{f \in K} y_{fi}$$

Para utilizar a desigualdade no modelo baseado em distâncias pelo Teorema 3, deve-se adicionar um termo z_j (onde j é algum outro vértice diferente de i) para o lado direito da desigualdade, de modo a obter-se uma expressão $z_i - z_j$ a ser substituída por x_{ij}. Porém, deve-se adicionar um termo suficientemente grande também ao lado esquerdo para compensar tal adição. Desse modo, adiciona-se um termo $|C|$ do lado esquerdo, já que $z_j \in \{1, 2, \ldots, |C|\}$. Assim, a desigualdade para $PD(G, C)$ é:

$$x_{ij} + |C| \ \geq \ \sum_{f \in K} y_{fi}$$

3.2. Desigualdade clique dupla

Seja $(i, j) \in E$ e considere uma clique $K \subseteq N(i) \cap N(j)$. Define-se então a desigualdade clique dupla associada ao vértice i e à clique K para $PO(G, C)$ como se segue:

$$z_i + \sum_{k \in K} (y_{ik} - y_{jk}) \ \leq \ z_j + (s - |K|) y_{ij}$$

Pela aplicação do Teorema 3 e pela observação que a constante s do modelo baseado em orientação, que deve ser um limite superior para a maior cor usada, é equivalente a $|C|$ no modelo baseado em distâncias, obtém-se a desigualdade a seguir para $PD(G, C)$:

$$x_{ij} + \sum_{k \in K} (y_{ik} - y_{jk}) \ \leq \ (|C| - |K|) y_{ij}$$

Tabela 1. Sumário das formulações de programação inteira para a coloração de vértices em grafos.

Formulação	Variáveis	Restrições	Autor(es)
Clássica	$O(\|V\|^2)$	$O(\|V\|^2 E)$	-
Conjuntos independentes	$O(3^{\|V\|/3})$	$O(\|V\|)$	[Mehrotra and Trick 1996]
Representantes assimétricos	$O(\|V\|^2 - \|E\|)$	$O(\|V\|^3 - \|V\|\|E\|)$	[Campêlo et al. 2008]
Clique cover	$O(\|V\|\|H\|)$	$O(\|H\| + \|V\|\|E'\|)$	[Burke et al. 2010]
Baseada em orientação	$\boldsymbol{O(\|V\| + \|E\|)}$	$\boldsymbol{O(\|V\| + \|E\|)}$	**[Dias et al. 2017]**
Baseada em distância	$\boldsymbol{O(\|V\|^2)}$	$\boldsymbol{O(\|V\|^2)}$	**Este trabalho**

4. Considerações finais

Neste trabalho, apresentou-se uma formulação de programação inteira para o problema clássico de coloração de vértices em grafos (VCP), onde as variáveis de decisão indicam as distâncias entre cores dos vértices, desenvolvida com base em outra formulação de programação linear inteira proposta pelos autores, o modelo de orientação. Mostrou-se que as desigualdades válidas indutoras de facetas do modelo baseado em orientação podem ser usadas na formulação com distâncias e vice-versa.

Resultados em andamento incluem a determinação de funções objetivo para o modelo baseado em distâncias, bem como a implementação de um método *cut-and-branch* utilizando a formulação e suas desigualdades válidas. Este trabalho também é a base para elaboração de um modelo de distâncias para a generalização do VCP conhecida como coloração em largura de banda (*bandwidth coloring probem* - BCP), que está em desenvolvimento.

Referências

Burke, E., Mareček, J., Parkes, A., and Rudová, H. (2010). A supernodal formulation of vertex colouring with applications in course timetabling. *Annals of Operat. Research*, 179:105–130.

Campêlo, M., Campos, V., and Corrêa, R. (2008). On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics*, 156:1097–1111.

de Freitas, R., Dourado, M., and Szwarcfiter, J. (2010). Graph coloring and scheduling problems. 4th Latin American Workshop on Cliques in Graphs.

Dias, B. (2014). Modelos teóricos e algoritmos para a otimização da alocação de canais em redes móveis sem fio. Master's thesis, Instituto de Computação – Universidade Federal do Amazonas, in portuguese.

Dias, B., de Freitas, R., Maculan, N., and Marenco, J. (2017). Facet-inducing inequalities and a cut-and-branch for the bandwidth coloring polytope based on the orientation model. *Electronic Notes in Discrete Mathematics*, 62:141 – 146. LAGOS'17 – IX Latin and American Algorithms, Graphs and Optimization.

Dias, B. R. C., Rodrigues, R. F., and Szwarcfiter, J. (2013). On graph coloring problems with distance constraints. In *Proc. of I Workshop on Distance Geometry and Appl. (DGA 2013)*.

Karp, R. (1972). Reducibility Among Combinatorial Problems. In Miller, R. E. and Thatcher, J. W., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

Koster, A. (1999). *Frequency assignment: models and algorithms*. PhD thesis, Univ. Maastricht.

Mehrotra, A. and Trick, M. (1996). A column generation approach for graph coloring. *INFORMS Journal on Computing*, 8:344–354.

Méndez-Díaz, I. and Zabala, P. (2006). A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics*, 154(5):826–847.

Exact and Heuristic Approaches to the Maximum Capacity Representatives Problem

Italos Estilon da Silva de Souza[1], **Mauro Roberto Costa da Silva**[1], **Welverton Rodrigues da Silva**[1], **Rafael C. S. Schoeury**[1]

[1]Institute of Computing – University of Campinas (UNICAMP)
Caixa Postal 6176 – 13083-852 – Campinas – SP – Brazil

italosestilon@gmail.com, maurorcsc@gmail.com,
welvertonrodrigues@gmail.com, rafael@ic.unicamp.br

Abstract. *This paper approaches the problem of finding the system of representatives of a family of disjoint sets. To solve this problem, three methods were used: integer programming, branch-and-bound, and the BRKGA metaheuristic. We observed that, in randomly generated instances, the branch-and-bound algorithm was the best exact method but it was surpassed by BRKGA for large instances.*

1. Introduction

In this work, we address the problem of maximizing the sum of capacities of representatives of disjoint sets introduced by [Bellare 1993], who showed that this problem is NP-complete and gave some inapproximability results. Formally, the maximum capacity representatives problem (MCR) can be defined as follows: let $S_1, S_2, \ldots, S_k$ be disjoint sets. For any $i \neq j$, $x \in S_i$ and $y \in S_j$, let $c(x, y)$ be the nonnegative capacity between x and y. Let R be a set such that $\forall_{1 \leq i \leq k} |R \cap S_i| = 1$, we say that R is a system of representatives. The capacity C_R of a system of representatives R is given by $C_R = \sum_{x,y \in R} c(x, y)$. The problem is finding R that maximizes C_R.

One can think of an instance of this problem as an undirected graph in which the vertex set V is $\bigcup_{i=1}^{k} S_i$ and edge set E is $\{(u, v) \,|\, u \in S_i,\, v \in S_j, \text{ and } i \neq j\}$. The edge weights represent the capacity between the vertices and each S_i is an independent set. Note that this graph has the property of being k-partite complete.

Thus the problem of finding the system of representatives of maximum capacity is equivalent to the problem of finding a maximum edge-weighted clique in a k-partite complete graph. This is interesting because the maximum edge-weighted clique problem is widely studied. For simplicity, we will think about instances of the MCR problem as graphs and then, throughout this text, we will refer to sets S_i as parts, to capacities between elements as edge weights, and to elements of S_i as vertices.

We have not found the MCR problem to be studied experimentally in the literature, but we link to a variant problem, the so-called minimum distance representative (MinDR) [Blanco et al. 2014].

We approached the MCR problem with integer programming, branch-and-bound, and BRKGA, comparing these techniques.

2. Integer Programming

[Gouveia and Martins 2015] consider a version of the problem of maximum edge-weighted clique where there are edges with negative weight. They gave the formulation

below to this problem based on [Park et al. 1996]. Let $G = (V, E)$ be an undirected graph. For all $i \in V$, variable x_i equals 1 if i is in the clique, otherwise x_i equals 0. For all $i, j \in V$, variable y_{ij} equals 1 if edge (i, j) is in the clique, otherwise y_{ij} equals 0.

$$\text{maximize} \sum_{i,j \in E} c_{ij} y_{ij} \tag{1}$$

$$\text{subject to } y_{ij} \leq x_i,\ y_{ij} \leq x_j \qquad (i,j) \in E \tag{2}$$

$$x_i + x_j \leq y_{ij} + 1 \qquad (i,j) \in E \tag{3}$$

$$x_i + x_j \leq 1 \qquad (i,j) \notin E \tag{4}$$

$$x_i \in 0, 1 \qquad i \in V \tag{5}$$

$$y_{ij} \in 0, 1 \qquad (i,j) \in E \tag{6}$$

We propose to exchange constraint (4) by $\sum_{x_j \in S_i} x_j = 1$ for all S_i, provided that we already know the graph is k-partite complete and we know each part S_i. The formulation with constraint (4) may lead to infeasible solutions to our problem because it does not force each solution to have exactly one vertex from each part, although an optimal solution of the former formulation is also an optimal solution to our problem.

3. BRKGA

The biased random-key genetic algorithm (BRKGA) is an evolutionary metaheuristic for optimization problems. This metaheuristic works with a fixed-size population composed of p vectors (chromosomes) of randomly generated numbers in the interval [0, 1). The population is divided into groups, the so-called elite, non-elite and mutants. For each iteration, the best ones are kept (elite), new ones are generated by crossover (non-elite) and a small number of mutants are introduced into the population.

The BRKGA uses a decoder to calculate the value of a chromosome [Resende 2011]. We create a deterministic algorithm for decoding that takes as input a chromosome and associates with it a feasible solution to the MCR problem and outputs this solution value. Each chromosome is a real vector $\mathbf{x} \in [0, 1)^k$, where each locus is sequentially associated to a disjoint set. Algorithm 1 presents the decoder.

Algorithm 1 Decoder algorithm

```
1: function DECODER(x)
2:     R ← ∅
3:     for i ← 1, ..., k do
4:         j ← 1 + ⌊x_i · |S_i|⌋
5:         R ← R ∪ {x}, where x is j-th element of S_i
6:     return Σ_{x,y∈R} c(x, y)
```

In addition, we create a greedy constructive heuristic to introduce a non-random key chromosome in the initial population. The heuristic works as follows: for all $i = 1, 2, \ldots, k$, the locus i is equal to $(j - 1)/|S_i|$, where x is the j-th element of S_i such that $\sum_{y \in V \setminus S_i} c(x, y)$ is the greatest sum of capacities.

4. Branch-and-Bound

We create an initial feasible solution by choosing for each S_i with $i = 1, 2, \ldots, k$ an $x \in S_i$ that maximizes $\sum_{y \in V \setminus S_i} c(x, y)$. That is, for each set S_i we choose the vertex that have the greatest sum of edge weights.

At level l of the search tree, we choose which vertex of part S_l will be in the clique. Note that the search tree has depth k, the number of sets. Let σ be a solution under construction, that is, the algorithm is on level $l < k$, and let σ_i be the vertex of part i in σ. Let $R(\sigma) = \sum_{\sigma_i,\sigma_j \in \sigma} c(\sigma_i, \sigma_j)$ be the value of solution under construction σ. Let $\mathrm{opt}(\sigma)$ be the greatest value σ can have after choose vertices for the remaining levels $l' > l$. Let $\rho(\sigma)$ be the sum of weights of heaviest edges between part S_i and S_j for all $i, j > l'$ plus the weight of the edge between σ_i and S_j for all $i \leq l$ and $j > l$. Note that $\mathrm{opt}(\sigma) \leq \rho(\sigma)$, that is, the best solution that can be made is less or equal to the sum of the weight of the heaviest edge that can be in a solution between each pair of parts plus the value of the solution under construction. Algorithm 2 presents the branch-and-bound algorithm.

Algorithm 2 Branch-and-Bound algorithm

```
1: σ_max ← ∅
2: BB(∅, 1)
3: function BB(σ, l)
4:     if l = k then
5:         if R(σ) > R(σ_max) then σ_max ← σ
6:         return
7:     if ρ(σ) ≤ R(σ_max) then return
8:     for all u ∈ S_l in non-increasing order by total capacity do
9:         BB(σ ∪ {u}, l + 1)
```

5. Computational Experiments

We generated the instances used in the experiments randomly. There are three types of instances, the ones with part sizes and weights were generated using a uniform probability distribution ("uniform"), the ones with parts of equal sizes ("similar"), and the ones with weights generated using the normal probability distribution ("normal"). Part sizes were chosen from the range of 1 to a maximum size using a uniform probability distribution. Edge weights were chosen from the range of 1 to 100, for instances of type "uniform", and from the range of 1 to 50, for type "similar", using a uniform probability distribution. For instances of type "normal", edge weights were chosen using a normal probability distribution with mean 100 and standard deviation of 2. Instances may have 5, 10, 30 or 50 parts and we used three values for the maximum size of a part: 10, 30 and 50. We made all combinations of instance types, the number of parts and maximum size of a part.

We implement the branch-and-bound algorithm (BB) in C++ and the ILP model in Java using Gurobi 7.5.2. The BRKGA was implemented in C++, using the API described and proposed in [Toso and Resende 2015]. We set parameters of BRKGA as follows: size of population $p = 1000$, elite set fraction $p_e = 0.16$, fraction of population to be replaced by mutants $p_m = 0.08$, probability that offspring inherit an allele from elite parent $rhoe = 0.65$, number of independent populations $p_n = 3$ and the number of generations $gen = 300$. Then we execute all instances for each algorithm with 1 hour of timeout in a machine with Intel (R) Xeon (R) Silver 4114 CPU @ 2.20GHz, 32GB of memory and Linux 64bits.

6. Results and Conclusions

The approach with integer programming did not lead to good results, provided that the solver only solved six instances. In cases where the solver found solutions close to optimal the gap was still large indicating that the solver was having difficulty to show that the solutions found were optimal. Instances of type "similar" were harder to solve by the ILP solver and by the branch-and-bound algorithm by the fact instances of type "similar" have more edges and vertices than the others instances. Instances of type "normal" and "uniform" had no significant difference in the number of vertices and edges. The branch-and-bound algorithm performed in a similar way for instances of those types, provided that, the number of vertices showed to be more determinative than edge weights for this method. For the ILP solver, edge weights distribution was not a significant factor, although it solved more instances of type "uniform" where the variance of weights is greater. The branch-and-bound approach obtained better results than the ILP since the BB algorithm solved sixteen instances.

The BRKGA was executed 100 times for each instance. The approach with BRKGA found, in average, solutions very close to optimal for instances solved by the branch-and-bound algorithm with optimality proof. For larger instances, BRKGA found solutions better than the ones found by the branch-and-bound algorithm within very competitive time, taking, in average, less than 15 seconds per instance. BRKGA found solutions equal to or better than those found by the exact methods for 31 instances. We tried to use BRKGA solutions as an initial lower bound to the branch-and-bound algorithm, but it was not able to prove optimality for those solutions, although it did not find better solutions before reaching the time limit. We conclude that BRKGA is a good approach for this problem given that it was better than exact methods for large instances. Provided that the exact methods did not lead to good results, we plan to analyze the use of other heuristics approaches as future work.

This project was supported by the São Paulo Research Foundation (FAPESP) grants #2014/25892-4, #2015/11937-9, #2016/23552-7, #2016/01860-1 and #2017/23343-1; and the Brazilian National Council For Scientific and Technological Development (CNPq) grants #311499/2014-7, #425340/2016-3 and #304856/2017-7.

References

[Bellare 1993] Bellare, M. (1993). Interactive proofs and approximation: reductions from two provers in one round. In *Theory and Computing Systems, 1993., Proceedings of the 2nd Israel Symposium on the*, pages 266–274. IEEE.

[Blanco et al. 2014] Blanco, R., Boldi, P., and Marino, A. (2014). Entity-linking via graph-distance minimization. *ArXiv e-prints*.

[Gouveia and Martins 2015] Gouveia, L. and Martins, P. (2015). Solving the maximum edge-weight clique problem in sparse graphs with compact formulations. *EURO Journal on Computational Optimization*, 3(1):1–30.

[Park et al. 1996] Park, K., Lee, K., and Park, S. (1996). An extended formulation approach to the edge-weighted maximal clique problem. *European Journal of Operational Research*, 95(3):671–682.

[Resende 2011] Resende, M. G. (2011). Introdução aos algoritmos genéticos de chaves aleatórias viciadas. *Simpósio Brasileiro de Pesquisa Operacional*, pages 3680–3691.

[Toso and Resende 2015] Toso, R. and Resende, M. (2015). A c++application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*, 30(1):81–93.

Uma Aproximação Ótima para o Problema do Caixeiro Alugador*

Lehilton L. C. Pedrosa[1], **Rafael C. S. Schouery**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Av. Albert Einstein, 1251 – 13083-852 – Campinas – SP – Brasil

{lehilton,rafael}@ic.unicamp.br

Abstract. *In the classical Traveling Salesman Problem (TSP), we want to find a route that visits each of n cities, minimizing the total traveled distance. An often considered generalization is the Traveling Car Renter Problem, in which the route is traveled by renting a set of cars. Every car has a distinct distance function, but each rented car incurs the payment of a return fee $g \geq 0$. The objective is to minimize the sum of traveled distances plus return fees. In this work, we present a $O(\log n)$-approximation for the problem. This algorithm is asymptotically optimal, since we show that there is no approximation with factor $o(\log n)$, unless P = NP.*

Resumo. *No clássico Problema do Caixeiro Viajante (TSP), queremos encontrar uma rota que passa por um conjunto de n cidades e que minimize a distância total percorrida. Uma generalização conhecida é o chamada Problema do Caixeiro Alugador, em que a rota pode ser percorrida alugando-se um subconjunto de veículos. Cada veículo tem uma função de distância distinta, mas para cada veículo alugado, paga-se uma taxa de retorno $g \geq 0$. O objetivo é minimizar a soma das distâncias percorridas mais as taxas de retorno. Neste trabalho, apresentamos uma $O(\log n)$-aproximação para o problema. Esse algoritmo é assintoticamente ótimo já que também mostramos que não há aproximação com fator $o(\log n)$, a não ser que P = NP.*

1. Introdução

Problemas de transporte aparecem no contexto de logística ou transporte urbano. Particularmente, serviços de transporte individual são oferecidos nas mais diversas formas, como compartilhamento e aluguel de veículos. Na maior parte das vezes, os problemas de aluguel de veículos são discutidos do ponto de vista de uma companhia, que detém uma frota de veículos e cujo objetivo é maximizar o lucro total ou o número de clientes atendidos. Alguns trabalhos consideram também o problema do ponto de vista de um usuário (Goldbarg et al., 2012; da Silva and Ochi, 2016; Rios et al., 2017), que tem um conjunto de opções de aluguel disponíveis e deseja escolher quais serviços contratar para minimizar o seu custo total.

Um problema que visa minimizar os custos de um usuário aparece no setor de turismo (da Silva and Ochi, 2016). Goldbarg et al. (2012) introduziram o *Problema do*

*Este trabalho foi parcialmente financiado pelo processo nº 2015/11937-9, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) e pelos processos nºs 425340/2016-3, 308689/2017-8 e 313026/2017-3, Conselho Nacional de Desenvolvimento Científico e Tecnológico.

Caixeiro Alugador (CARS), ilustrado a seguir. Um turista deseja visitar um conjunto de cidades e, para isso, tem um conjunto de veículos disponíveis para alugar. O custo de alugar cada veículo é dado por uma função de distância do trajeto percorrido, além do custo de retornar o veículo até o local alugado. Tanto a função de distância quanto a taxa de retorno dependem do veículo escolhido. O objetivo é alugar veículos para visitar todas as cidades e retornar ao local de origem pagando o mínimo possível.

Seguindo Goldbarg et al. (2012), que desenvolveram algoritmos meméticos, outros trabalhos consideraram o problema (da Silva and Ochi, 2016; Rios et al., 2017; Dias et al., 2016). Observamos que, se exigirmos que a solução seja um ciclo sem repetição de cidades, ou permitirmos funções de custo arbitrárias, então o problema não admite aproximação por qualquer função computável, a não ser que o Problema do Caminho Hamiltoniano possa ser resolvido em tempo polinomial. Assim, consideramos a versão do problema em que as funções são métricas e a taxa de retorno é a mesma para todo veículo e independe do local de devolução. Formalmente, no *Problema do Caixeiro Alugador Uniforme* (UCARS), dado um conjunto V de n cidades, r funções de distância $d_i : V \times V \to \mathbb{R}_{\geq 0}$ e uma taxa de retorno $g \geq 0$, o objetivo é encontrar um conjunto de passeios $P_1, \ldots, P_s$ associados a índices $e_1, e_2, \ldots, e_s$, tais que $(P_1 \ldots P_s)$ é um passeio fechado que passa por todas cidades e que minimiza a soma

$$\sum_{i=0}^{s} \left(g + \sum_{(u,v) \in E(P_i)} d_{e_i}(u, v) \right).$$

Contribuições Neste trabalho, mostramos que UCARS é tão difícil quanto o Problema da Cobertura de Conjuntos (SC), o que implica que UCARS não tem aproximação com razão $o(\log n)$, a não ser que P = NP (Guha and Khuller (1999)). Em seguida, observamos que UCARS pode ser reduzido para o Group-TSP (versão de TSP em que se deseja percorrer um elemento de cada grupo dado), derivando diretamente em uma aproximação de razão $O(\log^2 n \log \log n \log r)$. Finalmente, obtemos uma aproximação melhor, apresentando um algoritmo de arredondamento probabilístico com razão de aproximação $O(\log n)$. Esse algoritmo é assintoticamente ótimo por causa do limite inferior.

2. Inaproximabilidade

No Problema da Cobertura de Conjuntos, dados conjuntos $S_1, S_2, \ldots, S_m$ cuja união é U, queremos encontrar $S_{e_1}, S_{e_2}, \ldots, S_{e_s}$ cuja união é U e s é mínimo.

Teorema 2.1. Se UCARS admite α-aproximação, então SC admite α-aproximação.

Corolário 2.2. Se P $\neq$ NP, então UCARS não admite $(c \log n)$-aproximação com $c < 1$.

Por outro lado, reduzimos UCARS para Group-TSP. Uma instância de Group-TSP é formada por um conjunto V', uma função de distância d' e conjuntos $S_1, S_2, \ldots, S_l \subseteq V'$ e uma solução é uma passeio fechado P tal que $S_i \cap V(P) \neq \emptyset$ para $1 \leq i \leq l$. Construímos uma instância de Group-TSP. Primeiro, criamos um grafo $G = (V', E)$ em que, para cada $v \in V$, adicionamos um grupo $\{v_1, v_2, \ldots, v_r\}$ formando uma clique com arestas de peso g. Depois, para cada $u, v \in V$ e $1 \leq i \leq r$, adicionamos arestas (u_i, v_i) com custo $d_i(u, v)$. Observe que, dada uma solução para a instância de Group-TSP com custo SOL, podemos criar uma solução para a instância de UCARS de custo O(SOL) e vice-versa. Portanto, obtemos uma aproximação para UCARS com fator $O(\log^2 n \log \log n \log r)$ (Garg et al. (2000)). Na seção seguinte, mostramos que há uma $O(\log n)$-aproximação para UCARS.

3. Algoritmo de arredondamento de PL

A ideia principal do algoritmo é resolver o problema em duas fases. Primeiro, obtemos uma cobertura de todos os vértices de V utilizando subárvores. Assim, ao invés de procurar um conjunto de segmentos que forme um passeio fechado, nos preocupamos apenas em encontrar um conjunto de componentes conexas que cubram o conjunto de vértices. Na segunda fase, transformamos a cobertura em um passeio fechado.

3.1. Uma formulação de cobertura

No seguinte programa linear inteiro, denotamos por $\mathcal{S}$ o conjunto de todos os pares (i, S), com $S \subseteq V$ e $1 \leq i \leq r$. Para cada $(i, S) \in \mathcal{S}$, $\text{MST}_i(S)$ denota o custo de uma árvore geradora mínima de S com respeito a d_i e $x_{(i,S)}$ indica se (i, S) faz parte da solução.

$$\begin{array}{lll} \text{minimizar} & \displaystyle\sum_{(i,S)\in\mathcal{S}} x_{(i,S)}(\text{MST}_i(S) + g) & \\ \text{sujeito a} & \sum_{(i,S)\in\mathcal{S}: v\in V} x_{(i,S)} \geq 1 & \forall\ v \in V, \\ & x_{(i,S)} \in \{0, 1\} & \forall\ (i, S) \in \mathcal{S}. \end{array} \tag{PLI}$$

Observe que uma solução de uma instância de UCARS induz uma solução para esse programa, então o valor ótimo do (PLI) é um limitante inferior para o valor ótimo dessa instância. Seja (PL) a relaxação linear de (PLI). Observe que (PL) tem um número exponencial de variáveis e, portanto, não sabemos resolver esse problema diretamente. No entanto, podemos obter uma solução aproximada com um número polinomial de variáveis não nulas, conforme o lema a seguir, que é um resultado central para o algoritmo. Para um programa linear (Q), denote por $\text{OPT}(Q)$ o valor ótimo de (Q).

Lema 3.1. É possível, em tempo polinomial, encontrar uma solução viável para (PL) de valor $O(\text{OPT}(\text{PLI}))$.

A prova desse lema pode ser resumida do seguinte modo. Obtemos a formulação dual de (PL), que tem restrições $\sum_{v\in V} y_v - \text{MST}_i(S) \leq g$ para cada $(i, S) \in \mathcal{S}$ e executamos o método dos elipsoides. Para cada solução candidata y dada pelo método, queremos resolver o problema da separação correspondente. Se pudéssemos encontrar (i, S) que maximiza o valor do lado esquerdo, então poderíamos ou encontrar uma restrição violada, ou concluir que y é viável. Infelizmente, esse problema é NP-difícil e sequer tem aproximação constante. Para contornar essa dificuldade, utilizamos uma estratégia mais elaborada. Observe que se $\text{MST}_i(S)$ for pequeno quando comparado com g, então maximizar $\sum_{v\in V} y_v$ é uma boa aproximação para esse valor e esse problema pode ser bem resolvido a menos de um fator constante. Assim, a principal observação para derivar o lema é que o valor de PLI quando restrito a componentes com $\text{MST}_i(S) \leq g$ está a um múltiplo constante de $\text{OPT}(\text{PLI})$.

3.2. Arredondando uma solução fracionária

Dada uma solução x de (PL) de valor $O(\text{OPT}(\text{PLI}))$, o algoritmo obtém uma solução para a instância de UCARS arredondando o valor das variáveis $x_{(i,S)}$ para cada $(i, S) \in \mathcal{S}$. Observe que, como podemos supor que $0 \leq x_{(i,S)} \leq 1$, podemos incluir (i, S) em uma solução integral com probabilidade $x_{(i,S)}$. Fazendo isso para todas as componentes, o custo esperado é igual $O(\text{OPT}(\text{PLI}))$. A ideia do algoritmo é que, se repetirmos o procedimento acima um número suficiente de vezes, então a probabilidade de um elemento continuar descoberto tende a zero. Todos os passos são descritos em Algoritmo 1.

1. Obtenha uma solução x de (PL) usando o Lema 3.1.
2. Faça $\mathcal{C} \leftarrow \emptyset$ e repita $\lceil \log n \rceil$ vezes:
   - Para cada (i, S), inclua (i, S) em $\mathcal{C}$ com probabilidade $x_{(i,S)}$.
3. Para cada $v \in V$ tal que $v \notin S$ para todo $(i, S) \in \mathcal{C}$, adicione $(1, \{v\})$ a $\mathcal{C}$.
4. Faça $F \leftarrow \emptyset$ e para cada $(i, S) \in \mathcal{C}$:
   - Construa uma árvore geradora mínima H de $S \setminus V(F)$ com respeito a d_i.
   - Construa uma rota P duplicando as arestas de H.
   - Rotule cada aresta de H com índice i e adicione H a F.
5. Defina $d_{\min} : V \times V \to \mathbb{R}$ de forma que $d_{\min}(u, v) = \min_{1 \leq i \leq r} d_i(u, v)$.
6. Contraia cada componente conexa de F e obtenha F'.
7. Construa uma árvore geradora mínima T' de F' com respeito a $d_{\min}$.
8. Para cada aresta (u, v) de T', adicione duas arestas paralelas (u, v) a F rotuladas com índice i tal que $d_{\min}(u, v) = d_i(u, v)$.
9. Construa um ciclo euleriano C de F e devolva C.

Algoritmo 1: Aproximação para UCARS.

Nos passos de 1 a 3, obtemos uma cobertura de V por componentes de $\mathcal{C}$. Após o passo 4, F é formado por um conjunto de rotas que cobrem todos os vértices V, mas é possivelmente desconexo. Assim, queremos adicionar arestas para conectar essas rotas. Para cada aresta, escolhemos o veículo de menor custo correspondente e obtemos um conjunto de arestas que conecta F com menor custo. Daí, após o passo 8, F é um grafo conexo em que todo vértice tem grau par e, portanto, tem um ciclo euleriano C. Observe que C pode ser decomposto em uma sequência de passeios maximais de arestas com o mesmo rótulo e, portanto, induz uma solução viável do problema.

O custo da solução devolvida é composto das distâncias percorridas e das taxas de retorno. Observe que ambos custos podem ser limitados pelos custos das componentes em $\mathcal{C}$. Como o custo esperado das componentes sorteadas em cada iteração do passo 2 é de no máximo o valor de x, que é $O(\text{OPT}(\text{PLI}))$, e o passo 2 é executado $O(\log n)$ vezes, obtemos o seguinte teorema.

Teorema 3.2. O Algoritmo 1 é uma $O(\log n)$-aproximação para UCARS.

Referências

da Silva, A. R. V. and Ochi, L. S. (2016). An efficient hybrid algorithm for the traveling car renter problem. *Expert Systems with Applications*, 64:132 – 140.

Dias, S. S., Ochi, L. S., Machado, V. M. C., Simonetti, L. G., and da Silva, A. R. V. (2016). Uma heurística baseada em ils para o problema do caixeiro alugador. In *Anais do XLVIII SBPO Simpósio Brasileiro de Pesquisa Operacional*.

Garg, N., Konjevod, G., and Ravi, R. (2000). A polylogarithmic approximation algorithm for the group steiner tree problem. *Journal of Algorithms*, 37(1):66 – 84.

Goldbarg, M. C., Asconavieta, P. H., and Goldbarg, E. F. G. (2012). Memetic algorithm for the traveling car renter problem: an experimental investigation. *Memetic Computing*, 4(2):89–108.

Guha, S. and Khuller, S. (1999). Greedy Strikes Back: Improved Facility Location Algorithms. *Journal of Algorithms*, 31(1):228–248.

Rios, B. H. O., Goldbarg, E. F. G., and Quesquén, G. Y. O. (2017). A hybrid metaheuristic using a corrected formulation for the traveling car renter salesman problem. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 2308–2314.

On the Approximability of the Minimum Subgraph Diameter Problem

Arthur Pratti Dadalto[1], **Fábio Luiz Usberti**[1], **Mário César San Felice**[2]

[1]Instituto de Computacão - Universidade Estadual de Campinas (Unicamp)
Av. Albert Einstein, 1251 - Cidade Universitária - 13083-852 - Campinas - SP - Brazil

[2]Departamento de Computação - Universidade Federal de São Carlos (UFSCar)
Rod. Washington Luís, Km 235 - 13565-905 - São Carlos - SP - Brazil

ra166858@students.ic.unicamp.br, fusberti@ic.unicamp.br, felice@ufscar.br

Abstract. *This work addresses the minimum subgraph diameter problem (MSDP) by answering an open question with respect to its approximability. Given a graph with lengths and costs associated to its edges, the MSDP consists in finding a spanning subgraph with total cost limited by a given budget, such that its diameter is minimum. We prove that there is no β-approximation algorithm for the MSDP, for any constant β, unless P = NP. Our proof is grounded on the non-approximability of the minimum spanning tree diameter problem, proven by Bálint in 2013.*

1. Introduction

In a combinatorial optimization problem, there is a set of instances, a finite set of feasible solutions for each instance and an objective function. A feasible solution is one that respects a set of restrictions associated with the problem.

The problem can be of minimization or maximization. In a minimization problem, the goal is to find a feasible solution such that its value, evaluated by the objective function, is minimal. Such a solution is called optimal.

A naive algorithm to solve a combinatorial optimization problem consists in evaluating all feasible solutions in search of an optimal one. This easily becomes impractical, since many problems have an exponential amount of feasible solutions. While there are combinatorial optimization problems for which polynomial time algorithms are known, many of them are NP-hard problems for which there is no polynomial time algorithm, unless $P = NP$ [Papadimitriou and Steiglitz 1998].

For those problems, arises a need for approximation algorithms, which find feasible solutions in polynomial time to NP-hard optimization problems with provable guarantees on the distance between the returned solution and the optimal ones [Williamson and Shmoys 2011]. For a given constant β, we say that an algorithm is a β-approximation if, on every input, the algorithm finds a solution whose cost is at most β times the optimum. However, for some problems, it is possible to prove that even the design of a β-approximate algorithm is impossible for small values of β or even for any constant value of β, unless $P = NP$ [Trevisan 2014].

This paper studies the approximability of the minimum subgraph diameter problem (MSDP). Given a graph with lengths and costs associated to its edges, the MSDP

consists in finding a spanning subgraph with total cost limited by a given budget, such that its diameter is minimum.

The MSDP can appear in real world applications, such as the construction of a telecommunication network with a budget constraint. There are costs related to building connections, e.g. acquisition of cables, and there are transmission delays between nodes inherent to the network topology. A relevant goal could be to minimize the largest delay between two nodes in the resulting network.

Our contribution. We prove that there is no polynomial time β-approximation algorithm for the MSDP for any constant β, unless $P = NP$.

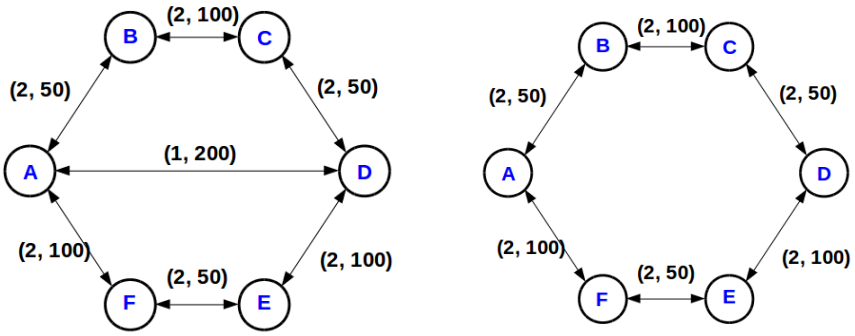
Paper organization. Section 2 formally describes the MSDP. Section 3 explains previous results on the MSDP as well as related work. Section 4 shows the result obtained and Section 5 presents our future research directions.

2. Problem description

In this paper we study the minimum subgraph diameter problem (MSDP). The problem input consists of a graph $G = (V, E)$, a resource budget $R \in \mathbb{R}_+$ and two weight functions for the edges: the length $l : E \to \mathbb{R}_+$ and the cost $c : E \to \mathbb{R}_+$.

Given a spanning subgraph $S = (V, E')$ of G and two vertices $u, v \in V$, we define $d_s(u, v)$ as the length of the shortest path between u and v in S (using the l weight function). If there is no path between vertices u and v, then $d_s(u, v) = \infty$. The diameter of the subgraph is defined as $\mathrm{diam}(S) = \max_{u,v \in V} d_s(u, v)$. The cost of the subgraph is defined as $\mathrm{cost}(S) = \sum_{e \in E'} c(e)$.

A feasible solution to this problem is a spanning subgraph S with $\mathrm{cost}(S)$ at most R and the goal is to minimize $\mathrm{diam}(S)$.



(a) MSDP instance. (b) An optimal solution.

Figure 1. Sample instance for the MSDP and an optimal solution for it.

Figure 1(a) shows a possible instance to the MSDP. Each edge is associated with a pair of values: the left one is its length and the right one is its cost. For feasible solutions a total cost budget of 450 will be considered.

When trying to minimize the diameter one would not remove any edges, but the budget constraint does not allow it. When trying to minimize the total cost one would get to the minimum spanning tree, that does not minimize the diameter. An optimal solution for this instance is shown in Figure 1(b). It has a higher diameter than the original graph, but it is the lowest possible without violating the budget.

Let diam^* be the diameter value for an optimal solution of the MSDP. Given a constant β, a β-approximation for the MSDP consists in finding a spanning subgraph S such that $\mathrm{cost}(S) \leq R$ and $\mathrm{diam}(S) \leq \beta \cdot \mathrm{diam}^*$.

3. Related work

It has been proved that, unless $P = NP$, there is no polynomial time β-approximation algorithm for the MSDP in each of the following four cases [Plesnik 1981]:

1. All the edge lengths (but not costs) of G are equal and $\beta < 5/4$.
2. $l(e) = c(e)$ for every edge e and $\beta < 5/4$.
3. $\beta < 2$.
4. G is a planar graph with maximum degree 3 and $\beta < 3/2$.

It is worth mentioning that our work strengthens the result given in item 3.

A more generic bicriteria network problem, the $(A, B, \lambda, \mathfrak{C})$-problem, has been studied. The problem input consists of a graph $G = (V, E)$ and two weight functions for the edges: the length $l : E \to \mathbb{R}_+$ and the cost $c : E \to \mathbb{R}_+$. A is a criterion on the weight function c and B is a criterion on the weight function l. For instance, A could be cost and B could be diam (as defined in section 2). λ is a real number not less than 1 and $\mathfrak{C}$ is a class of graphs that limits the possible solutions (e.g., trees) [Bálint 2003].

Let $\mathfrak{C}(G)$ denote the class of all spanning subgraphs of G belonging to the class $\mathfrak{C}$, i.e., $\mathfrak{C}(G) := \{S \in \mathfrak{C} | S \text{ is a spanning subgraph of } G\}$. The minimum value of criterion A is defined as $A^* = \min\{A(S) | S \in \mathfrak{C}(G)\}$. The minimum value of criterion B under a budget on A is defined as $B_\lambda = \min\{B(S) | S \in \mathfrak{C}(G), A(S) \leq \lambda A^*\}$. Note that λ indirectly defines a budget [Bálint 2003].

A solution to the $(A, B, \lambda, \mathfrak{C})$-problem is a spanning subgraph $S \in \mathfrak{C}(G)$ such that $A(S) \leq \lambda A^*$ and $B(S) = B_\lambda$. As an example, the MSDP, defined in Section 2, is almost equivalent to the $(cost, diam, \lambda, \text{spanning subgraph})$-problem. The only difference is that here the budget is set indirectly by λ instead of directly by R on the MSDP, i.e., for a budget R we should set $\lambda = R/A^*$.

An (α, β)-approximation to the $(A, B, \lambda, \mathfrak{C})$-problem is a spanning subgraph $S \in \mathfrak{C}(G)$ such that $A(S) \leq \alpha\lambda A^*$ and $B(S) \leq \beta B_\lambda$ [Bálint 2003].

It has been proved that, unless $P = NP$, for any constant β there is no polynomial time algorithm to solve the $(1, \beta)$-approximation to the $(cost, diam, 1, \text{spanning tree})$-problem, even if both criteria are under the same cost function $c : E \to \{1, 2\}$ [Bálint 2003]. In other words, there is no polynomial time algorithm to find a minimum spanning tree (since $\lambda = \alpha = 1$) with at most β times the diameter of any minimum spanning tree. We shall use this result to prove that there is no polynomial time algorithm to solve the β-approximation to the MSDP for any β constant.

4. Our results

Suppose that for some constant β there is a polynomial time β-approximation algorithm for the MSDP. Given an instance I of the $(cost, diam, 1,$ spanning tree$)$-problem, we can create an instance I' of the MSDP by using the same graph, weight functions and setting the budget $R = \mathrm{cost}(\mathrm{MST}(G))$, i.e., the cost of a minimum spanning tree for G (since $\lambda = 1$).

If we solve the MSDP for I', the result might not be a tree, due to the possibility of zero cost edges in the solution. Then, there would be no trivial way to transform this solution back to one for the $(cost, diam, 1,$ spanning tree$)$-problem, since removing any edges forfeits our guarantees on the diameter. In this case, we would not be able to translate the lower bound for spanning trees from Bálint to a lower bound for general spanning subgraphs. However, we can use the stronger lower bound by Bálint which holds even when both criteria are under the same cost function $c : E \to \{1, 2\}$.

Thus, assuming that I only has edges with costs 1 or 2, we create I' as before and then we solve the MSDP for I'. Since there are no zero cost edges, the solution found would be a minimum spanning tree with diameter at most β times that of any other minimum spanning tree.

The minimum spanning tree found would be a $(1, \beta)$-approximation to the $(cost,$ $diam, 1,$ spanning tree$)$-problem, contradicting the result by Bálint. Therefore, there is no polynomial time β-approximation algorithm for the MSDP for any constant β, unless $P = NP$.

5. Future work

Our result cuts short any effort to find a β-approximation for the MSDP. Nonetheless, a promising line of research is to investigate (α,β)-approximation algorithms for the MSDP.

References

Bálint, V. (2003). The non-approximability of bicriteria network design problems. *Journal of Discrete Algorithms*, 1:339–355.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*. Courier Corporation.

Plesnik, J. (1981). The complexity of designing a network with minimum diameter. *Networks*, 11:77–85.

Trevisan, L. (2014). *Inapproximability of Combinatorial Optimization Problems. In: Paradigms of Combinatorial Optimization*, chapter 13, pages 381–434. Wiley-Blackwell.

Williamson, D. P. and Shmoys, D. B. (2011). *The Design of Approximation Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition.

Centralities in High Order Networks

Klaus Wehmuth, Artur Ziviani

[1]National Laboratory for Scientific Computing (LNCC)
Av. Getúlio Vargas, 333, Quitandinha – CEP 25651-075 – Petrópolis, RJ – Brazil

{klaus,ziviani}@lncc.br

Abstract. *We propose a method for computing centralities based on shortest paths in time-varying, multilayer, and time-varying multilayer networks using MultiAspect Graphs (MAG). Thanks to the MAG abstraction, these high order networks are represented in a way that is isomorphic to a directed graph. We then show that well-known centrality algorithms can be adapted to the MAG environment in a straightforward manner. Moreover, we show that, by using this representation, pitfalls usually associated with spurious paths resulting from aggregation in time-varying and multilayer networks can be avoided.*

1. Introduction

In this paper, we discuss notions of centrality in high order networks. A high order network is any network that has additional structures, such as, layers, time representations, or any other structure with similar properties. A high order network is not restricted to have only one of the mentioned structures, as it is possible to create networks that are layered and time-varying, or even layered networks with multiple time structures representing a multi-scale time structure. In this paper, we represent high order networks by means of MultiAspect Graphs (MAGs) [Wehmuth et al. 2016, Wehmuth et al. 2017]. MAGs are an abstraction that can be used to represent time-varying, multilayer, combined time-varying-multilayer, or even higher order graphs. In a MAG, each independent structure of a high order network is represented by an aspect. In a MAG, aspects are used to represent the vertices set, layer set, time instants set, and so on. In [Wehmuth et al. 2016], MAGs are shown to be isomorphic to an object composed of a directed graph and one tuple. Algebraic representations and basic algorithms for MAGs are investigated in [Wehmuth et al. 2017]. These previous works pave the way for the notions of centralities in high order networks introduced here. In particular, we focus on shortest path centralities, since they are widely applied on complex network analysis and relatively straightforward to interpret. Moreover, we show that by using the proposed method, pitfalls usually associated with aggregation in time-varying and multilayer networks can be avoided. These aggregation side-effects are well-known in the literature of multilayer/time-varying networks [Pan and Saramäki 2011, Nicosia et al. 2012, Ribeiro et al. 2013, Kivela et al. 2014, De Domenico et al. 2016] and are manifested as additional paths that do not exist in the original network, or as additional shortest paths that also do not exist in the original networks. The presence of these artifacts on the aggregated network causes the shortest path based centrality algorithms to include these shortest paths into the calculation, leading to spurious results that are not consistent to the original network.

2. MultiAspect Graphs (MAGs)

We define a MAG as $H = (A, E)$, where E is a set of edges and A is a finite list of sets, each of which is called an *aspect*. Each aspect $\sigma \in A$ is a finite set, and the number of aspects $p = |A|$ is called the order of H. Each edge $e \in E$ is a tuple with $2 \times p$ elements. All edges are constructed so that they are of the form $(a_1, \ldots, a_p, b_1, \ldots, b_p)$, where a_1, b_1 are elements of the first aspect of H, a_2, b_2 are elements of the second aspect of H, and so on, until a_p, b_p which are elements of the p-th aspect of H. As a matter of notation, we say that $A(H)$ is the aspect list of MAG H and $A(H)[n]$ is the n-th aspect of MAG H. We also define the set

$$\mathbb{V}(H) = \bigtimes_{n=1}^{p} A(H)[n], \tag{1}$$

which is the Cartesian product of all the p aspects of the MAG H. We call the elements $\mathbf{v} \in \mathbb{V}(H)$ *composite vertices*. Note that each composite vertex $\mathbf{v} \in \mathbb{V}(H)$ has the form $(a_1, \ldots, a_p)$. Therefore, there is a close relation between MAG edges and pairs of composite vertices, since $(a_1, \ldots, a_p, b_1, \ldots, b_p) \sim (\mathbf{v}, \mathbf{u}) = ((a_1, \ldots, a_p), (b_1, \ldots, b_p))$, so that $\mathbf{v} = (a_1, \ldots, a_p)$ and $\mathbf{u} = (b_1, \ldots, b_p)$. From this relation between MAG edges and pairs of composite vertices, it is possible to build a directed graph of composite vertices, which is shown to be isomorphic to the MAG in [Wehmuth et al. 2016].

As a consequence of the isomorphism between MAGs and a directed graph, we then define the function

$$\begin{aligned} g : (A(H), E(H)) &\to (\mathbb{V}(H), \mathbb{V}(H) \bigtimes \mathbb{V}(H)) \\ H &\mapsto (\mathbb{V}(H), \psi(E(H)), \end{aligned} \tag{2}$$

which maps every MAG H to its isomorphic directed graph $g(H)$. Further, we define the set of functions

$$\begin{aligned} \pi_i : \mathbb{V}(H) &\to A(H)[i] \\ (a_1, a_2, \ldots, a_p) &\mapsto a_i, \end{aligned} \tag{3}$$

which extracts the i-th element of a composite vertex tuple.

Note that by the definition of $g(H)$ in Expression 2, the vertices of the directed graph $g(H)$ are tuples with $2 \times p$ elements. It is also possible, if desired, to have a more traditional graph representation where the vertices are simply points of a set with no additional meaning. In this case, the directed graph has to be complemented by a *companion tuple*, which allows the association of each vertex with its original MAG tuple. This companion tuple is a tuple with p elements, where each element $0 < i \leqslant p$ is given as $|A(H)[i]|$, the number of elements of the i-th aspect. Further details regarding the construction and usage of the companion tuple of a MAG can be found in [Wehmuth et al. 2017].

2.1. MAG sub-determination

A MAG sub-determination is a generalization of the aggregation applied to high order networks, in which layers or time instants can be aggregated, resulting in a traditional graph. Since a high order network can be represented by a MAG with 2 or more aspects,

a sub-determination can be performed in more ways than the classic aggregation. The formal definition of MAG sub-determination can be found in [Wehmuth et al. 2016]. As an example, consider a MAG with two aspects, where the first aspect is a set of geographic localities and the second aspect is a set of layers. Such structure could be used, for instance, to represent a multi-modal urban transportation system, where the layers represent the distinct transportation modes (e.g. Bus, Subway, Ferry) and geographic location represent the stations. A possible sub-determination of such MAG, can be the aggregation upon the stations, which is a classic directed graph.

3. Centralities in high order networks

In graph theory and network analysis, a centrality can be understood as an indicator of the relative importance of the vertices or edges on the graph under analysis. Formally, it follows that for a given graph $G = (V, E)$, a vertex centrality C_v can be seen as a function from the set of vertices of a graph to a set of nonnegative real numbers, i.e.

$$C_v : V(G) \rightarrow \mathbb{R}^+ \bigcup \{0\}, \tag{4}$$

where $V(G)$ is the vertex set of the graph G. Note that Function 4 induces a linear order relation upon its domain, which can, in turn, be understood as a centrality notion. The adopted centrality function should then properly reflect the kind of vertex relative importance to be expressed by the centrality notion. Similarly, edge centralities may be defined to reflect the relative importance of an edge.

For instance, the vertex betweenness centrality [Freeman 1977] is defined as

$$C_B(v) = \sum_{s,t \in V(G)} \frac{\sigma(s, t/v)}{\sigma(s, t)}, \tag{5}$$

where $V(G)$ is the the vertex set of the graph G, $s, t \in V(G)$ are vertices of the graph G, $v \in V(G)$ is the vertex for which the centrality is calculated, $\sigma(s, t)$ is the number of shortest paths connecting vertices s and t, and $\sigma(s, t/v)$ is the number of shortest paths from s to t which pass through vertex v. If calculated for each $v \in V(G)$, Equation 5 can be seen as a possible implementation of Function 4.

3.1. Extending the centrality notion to MAGs

The evaluation of edge centralities in a MAG can be done in a straightforward way using the MAG full representation. Since this representation is a traditional directed graph in which the vertices are a subset of the composite vertices set and it carries all the edges of the MAG, it thus preserves the topological properties (i.e., their adjacency properties). It follows that edge centralities can be computed using the same methods applied to traditional directed graphs.

However, the evaluation of vertex centralities in MAGs is more complex than in a traditional graph. The first difference to be considered is that a MAG can be sub-determined and at the limit any aspect element can become a vertex. Nevertheless, the full representation of a MAG, which is shown to be isomorphic to the MAG [Wehmuth et al. 2016], is a directed graph. Hence, vertex centralities in a MAG can be defined in terms of the composite vertices present on the MAG, as well as, in

terms of any sub-determination of the composite vertices. Moreover, it is possible to find the shortest paths necessary for centrality calculation in the MAG full representation and performing the sub-determination in a later step. This assures that only the paths present in the full MAG representation will be considered in the centrality calculation, leading to the conclusion that the additional paths created by the sub-determination process will not be considered in the calculation.

4. Conclusion

In this paper, we discuss the implementation of shortest path based centralities for MAGs. Since MAGs are isomorphic to directed graphs added with a tuple, it follows that well-known centrality algorithms can be adapted for the MAG environment. Further, due to the structure of MAGs, it follows that centralities can also be calculated in sub-determined form. We show that the sub-determined centrality algorithms use only valid paths present on the MAG, making them different from using the traditional (not sub-determined) algorithm upon a sub-determined MAG.

As with the well-known centrality algorithms for directed graphs, MAG centrality algorithms can be adapted in various ways, such as different distance definitions, restricted ranges, weighted graphs, and so on. We also show that these adaptations can be ported to MAG algorithms in a similar way as the traditional algorithms themselves are adapted for the MAG environment. Finally, as an additional contribution, we make a python implementation of the algorithms discussed in this paper publicly available.[1]

Acknowledgements

This work is partially funded by CAPES, CNPq, FAPERJ, and FAPESP.

References

De Domenico, M., Granell, C., Porter, M. A., and Arenas, A. (2016). The physics of multilayer networks. *arXiv*, pages 1–22.

Freeman, L. C. (1977). A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1):35 – 41.

Kivela, M., Arenas, A., Barthelemy, M., Gleeson, J. P., Moreno, Y., and Porter, M. a. (2014). Multilayer networks. *Journal of Complex Networks*, 2(3):203–271.

Nicosia, V., Tang, J., Musolesi, M., Russo, G., Mascolo, C., and Latora, V. (2012). Components in time-varying graphs. *Chaos*, 22(2):023101.

Pan, R. and Saramäki, J. (2011). Path lengths, correlations, and centrality in temporal networks. *Physical Review E*, 84(1):1–10.

Ribeiro, B., Perra, N., and Baronchelli, A. (2013). Quantifying the effect of temporal resolution on time-varying networks. *Scientific reports*, 3:3006.

Wehmuth, K., Fleury, É., and Ziviani, A. (2016). On MultiAspect graphs. *Theoretical Computer Science*, 651:50–61.

Wehmuth, K., Fleury, E., and Ziviani, A. (2017). Multiaspect graphs: Algebraic representation and algorithms. *Algorithms*, 10(1).

[1] `http://github.com/wehmuthklaus/MAG_Algorithms`

Sobre o problema da precificação livre de inveja na indústria de entretenimento esportivo

Marcos Salvatierra[1], **Rosiane de Freitas**[1]

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)

marcosms,rosiane@icomp.ufam.br

Abstract. *In this paper, we discuss the feasibility of applying the envy-free pricing problem as strategy in modeling the sale of tickets for sporting events. A mathematical formulation in integer nonlinear programming of the problem is proposed, where assumes that the products have different price ranges, that the customers' valuations of the products are given by reservation prices, that consumers are grouped into segments, that the supply of each product is limited, that consumers can claim several units of the same product and also that there is no price difference for the same product. The goal is to maximize the seller's revenue, but in a way that there is buyers' satisfaction. Preliminary computational experiments demonstrate the viability of the proposed formulation.*

Resumo. *Neste trabalho, se analisa a viabilidade de aplicação do problema da precificação livre de inveja como estratégia na modelagem da venda de ingressos para eventos esportivos. Uma formulação matemática em programação não-linear inteira do problema é proposta, onde se assume que as valorações dos produtos atribuídas pelos consumidores se dão pelos preços de reserva, que a oferta de cada produto é limitada, que os consumidores podem pleitear várias unidades de um mesmo produto e, também, que não há diferença de preços para um mesmo produto. O objetivo é maximizar o lucro do vendedor, mas de maneira que haja satisfação dos compradores. Experimentos computacionais preliminares demonstram a viabilidade da formulação proposta.*

1 - Introdução

As mudanças repentinas que vêm ocorrendo no mercado apresentam desafios às companhias exigindo destas a elaboração de novas ideias inovadoras e produtivas a fim de manter a competitividade e a rentabilidade. Um segmento que encontra-se no bojo das discussões de mercado na atualidade é a indústria de entretenimento esportivo, que movimenta bilhões de dólares anualmente. No processo de obtenção de uma estimativa do volume de movimentação financeira deste segmento, uma série de fatores são levados em conta, como a venda de ingressos, produtos e artigos esportivos, acordos de transmissão, patrocínios, entre outros. Os resultados apresentados neste artigo têm como foco o aspecto da venda de ingressos, que é a mais tradicional fonte de renda na indústria do esporte.

Motivado pela realização do maior evento esportivo deste ano, a Copa do Mundo da Rússia de 2018, este trabalho visa apresentar uma formulação matemática em programação não-linear inteira mista do problema da precificação livre de inveja aplicada à venda de ingressos de jogos de futebol.

Para se alcançar este objetivo, primeiramente será apresentado o problema da precificação livre de inveja na venda de ingressos esportivos. Em seguida, será feita uma formulação MINLP (*Mixed-Integer Nonlinear Programming*) seguido de uma proposta de linearização para o caso hipotético da venda de ingressos para a Copa do Mundo da Rússia 2018. Por fim, serão apresentados e discutidos resultados de algumas simulações.

2 - O problema da precificação livre de inveja na venda de ingressos esportivos

No problema da precificação livre de inveja [Guruswami et al. 2005], considera-se como dados de entrada um conjunto de consumidores, um conjunto de produtos e uma valoração atribuída a cada produto por cada consumidor, produzindo como saídas uma precificação e uma alocação livres de inveja, ou seja, ao final espera-se que cada comprador esteja tão satisfeito com a alocação atribuída a ele quanto qualquer outra, levando em consideração a utilidade de cada produto para cada comprador. De maneira formal, tem-se como entrada um conjunto $I = \{1, ..., m\}$ de compradores, um conjunto $J = \{1, ...n\}$ de produtos e uma valoração $v \in \mathbb{R}_+^{m \times n}$ em que v_{ij} é a valoração atribuída pelo consumidor i ao produto j. As saídas são uma precificação $\pi \in \mathbb{R}_+^n$ em que π_j é o preço do produto j e uma alocação $X \in \{0, 1\}^{m \times n}$ em que $x_{ij} = 1$ se o consumidor i compra o produto j e $x_{ij} = 0$, caso contrário. Assim, a receita total do vendedor é dada por $\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}\pi_j$ e o problema busca maximizar esta receita, respeitando a satisfação dos compradores (no sentido da compra livre de inveja), dada pela não negatividade (ou excedente) do comprador i para o produto j: $e_{ij} = v_{ij} - \pi_j \geq 0$.

3 - O caso hipotético da venda de ingressos para a Copa do Mundo da Rússia 2018

3.1 - Formulação MINLP

Para fins de motivação e desenvolvimento de um protótipo, suponhamos que a empresa responsável pelo evento quer vender ingressos para uma partida entre duas seleções. Uma proposta de modelagem matemática para este fim pode levar em consideração os seguintes aspectos: faixas de preço (os ingressos são tipificados e valorizados de acordo com a proximidade e visibilidade dos assentos em relação ao campo); determinação de valorações (a empresa recorre a análises de mercado que observam o comportamento de compra de potenciais consumidores, estimando as valorações que os consumidores atribuem aos vários tipos de ingresso como sendo os preços de reserva [Farris et al. 2012]); segmentos de consumidores (consumidores com comportamentos de compra semelhantes são agrupados em segmentos); demanda múltipla (os consumidores estão dispostos a adquirir várias unidades de um mesmo tipo de ingresso); oferta limitada (existe um número máximo de unidades disponíveis para cada tipo de ingresso, de acordo com a lotação máxima do estádio); preço não diferenciado (consumidores que desejam o mesmo tipo de ingresso pagam o mesmo preço por ele); compra livre de inveja (cada consumidor se sente pelo menos tão feliz com a alocação atribuída a ele quanto com qualquer outra, considerando a precificação escolhida).

Feitas estas considerações, um modelo matemático que maximiza a receita da empresa ao mesmo tempo que satisfaz o desejo dos consumidores (no sentido da precificação

livre de inveja) é:

$$
\begin{aligned}
\max & \sum_{i=1}^{m}\sum_{j=1}^{n} N_i x_{ij} c_{ij} \pi_j \\
\text{s. a.} \quad & \sum_{i=1}^{m} x_{ij} c_{ij} \leq u_j,\ \forall j \in J \\
& \sum_{i=1}^{m} N_i x_{ij} c_{ij} \leq T_j,\ \forall j \in J \\
& (r_{ij} - \pi_j) x_{ij} \geq (r_{ik} + \delta_i) x_{ij} - \pi_k,\ \forall i \in I,\ \forall j,k \in J,\ j \neq k \\
& (r_{ij} - \pi_j) x_{ij} \geq 0,\ \forall i \in I,\ \forall j \in J \\
& \pi_j \geq 0,\ \forall j \in J \\
& x_{ij} \in \{0,1\},\ \forall i \in I,\ \forall j \in J
\end{aligned}
$$

em que: $I = \{1, ..., m\}$, é o conjunto de segmentos de consumidores; N_i é o número de consumidores do segmento i; $J = \{1, ..., n\}$ é o conjunto de tipos de ingresso; u_j é o número de unidades disponíveis do tipo j para cada consumidor; T_j é o total de ingressos do tipo j disponíveis; r_{ij} é a valoração (preço de reserva) que os consumidores do segmento i atribuem ao ingresso do tipo j; c_{ij} é o número unidades do tipo j que os consumidores do segmento i desejam; π_j é o preço a ser determinado para o ingresso do tipo j; δ_i é um parâmetro que determina o desempate nos casos em que consumidores possuem mesma utilidade máxima para produtos diferentes [Shioda et al. 2011]; x_{ij} é a variável de decisão que determinará se os consumidores do segmento i adquirirão o ingresso do tipo j.

3 .2 - Proposta de linearização

A fim de linearizar o modelo acima, será introduzida uma variável auxiliar contínua p tal que $p_{ij} = \pi_j$, se $x_{ij} = 1$, e $p_{ij} = 0$ caso contrário, e será considerado também $\bar{r}_j = \max_i\{r_{ij}\}$. Assim, o modelo linear correspondente é:

$$
\begin{aligned}
\max & \sum_{i=1}^{m}\sum_{j=1}^{n} N_i c_{ij} p_{ij} \\
\text{s. a.} \quad & \sum_{i=1}^{m} x_{ij} c_{ij} \leq u_j,\ \forall j \in J \\
& \sum_{i=1}^{m} N_i x_{ij} c_{ij} \leq T_j,\ \forall j \in J \\
& \sum_{j \neq k} (r_{ij} x_{ij} - p_{ij}) \geq (r_{ik} + \delta_i)(\sum_{j \neq k} x_{ij}) - \pi_k,\ \forall i \in I,\ \forall k \in J \\
& r_{ij} x_{ij} - p_{ij} \geq 0,\ \forall i \in I,\ \forall j \in J \\
& p_{ij} \geq \pi_j - \bar{r}_j(1 - x_{ij}),\ \forall i \in I,\ \forall j \in J \\
& p_{ij} \leq \pi_j,\ \forall i \in I,\ \forall j \in J \\
& \pi_j, p_{ij} \geq 0,\ \forall j \in J \\
& x_{ij} \in \{0,1\},\ \forall i \in I,\ \forall j \in J
\end{aligned}
$$

4 - Resultados e discussão

Os resultados preliminares obtidos indicam a viabilidade da formulação, suportados por experimentos computacionais usando o solver GLPK v. 4.57.

Em um cenário hipotético, considerou-se um conjunto de oito segmentos de consumidores e quatro tipos de ingresso, sendo que a disponibilidade de todos os tipos era de duas unidades por consumidor, com $c_{ij} = 1\ \forall i \leq 4$ e $c_{ij} = 2\ \forall i \geq 5$, $N_i = 10.000\ \forall i$, $\delta_i = 40\ \forall i$, $T_j = 10.000\ \forall j \leq 3$ e $T_4 = 20.000$. Ao final da simulação, a receita do vendedor foi de \$38.900.000, o que verifica a maximização da receita observada a condição livre de inveja para os compradores, como mostram as Figuras 1 e 2.

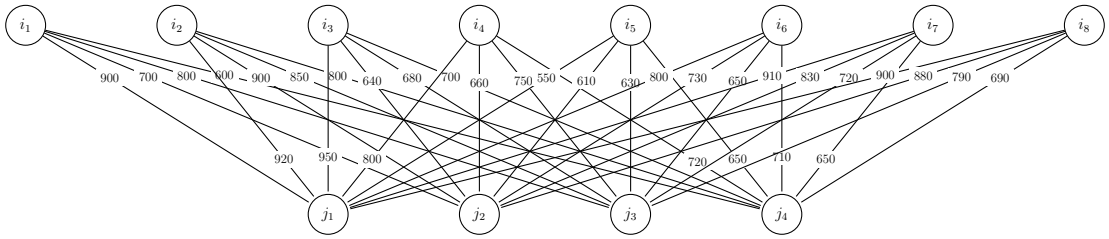


Figura 1. Valorações.

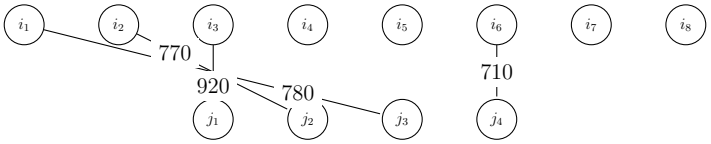


Figura 2. Alocação e precificação.

5 - Conclusões

Neste trabalho foram apresentados os resultados iniciais da aplicação de um modelo de precificação livre de inveja no âmbito da indústria de entretenimento esportivo. Em andamento, está sendo feita uma análise de sensibilidade dos parâmetros envolvidos e definindo-se instâncias maiores para testes computacionais robustos. Com base nos experimentos iniciais realizados, percebe-se que para um consumidor i que deseja n cópias de um item, sendo que a disponibilidade de unidades do item é menor que n, esse consumidor fica sem ser atendido, o que parece ser algo similar ao problema de precificação livre de inveja com objetivo único [Cheung and Swamy 2008]. Dessa forma, pretendemos elaborar formulações em que estes clientes possam ter suas demandas parcialmente aceitas.

Referências

Cheung, M. and Swamy, C. (2008). Approximation algorithms for single-minded envy-free profit- maximization problems with limited supply. In *49th Annual IEEE Symposium on Foundations of Computer Science*, pages 35–44.

Farris, P. W., Bendle, N. T., Pfeifer, P. E., and Reibstein, D. J. (2012). *Métricas de marketing - o guia definitivo de avaliação de desempenho de marketing*. Bookman.

Guruswami, C., Hartline, J. D., Karlin, A. R., Kempe, D., Kenyon, C., and Mcsherry, F. (2005). On profit-maximizing envy-free pricing. In *18th Annual ACM-SIAM Symposium on Discrete Algorithms, 2005, Vancouver*, pages 1164–1173. SIAM.

Shioda, R., Tuncel, L., and Myklebust, T. G. J. (2011). Maximum utility product pricing models and algorithms based on reservation price. *Computational Optimization and Applications*, 48(2):157–198.

Vertex partition problems in digraphs *

M. Sambinelli[1], C. N. Lintzmayer[2], C. N. da Silva[3], O. Lee[4]

[1]Institute of Mathematics and Statistics – University of São Paulo – Brazil

[2]Center for Mathematics, Computation and Cognition – Federal University of ABC – Brazil

[3]Department of Computing – Federal University of São Carlos – Brazil

[4]Institute of Computing – University of Campinas – Brazil

Abstract. *Let D be a digraph and k be a positive integer. Linial (1981) conjectured that the k-norm of a k-minimum path partition of a digraph D is at most $\max\{\sum_{C\in\mathcal{C}} |C| \colon \mathcal{C}$ is a partial k-coloring of $D\}$. Berge (1982) conjectured that every k-minimum path partition contains a partial k-coloring orthogonal to it. It is well known that Berge's Conjecture implies Linial's Conjecture. In this work, we verify Berge's Conjecture, and consequently Linial's Conjecture, for locally in-semicomplete digraphs and k-minimum path partitions containing only two paths. Moreover, we verify a conjecture related to Berge's and Linial's Conjectures for locally in-semicomplete digraphs.*

1. Introduction

Given a digraph D, we denote its vertex set by $V(D)$ and its arc set by $A(D)$. A *path partition* $\mathcal{P}$ of a digraph D is a collection of paths such that $\{V(P) \colon P \in \mathcal{P}\}$ is a partition of $V(D)$. Given a positive integer k, the *k-norm* of $\mathcal{P}$, denoted by $|\mathcal{P}|_k$, is $\sum_{P\in\mathcal{P}} \min\{|V(P)|, k\}$. We say that $\mathcal{P}$ is *k-minimum* if $|\mathcal{P}|_k \leq |\mathcal{P}'|_k$ for every path partition $\mathcal{P}'$ of D, and we denote by $\pi_k(D)$ the k-norm of a k-minimum path partition of D. A *partial k-coloring* $\mathcal{C}$ of D is a collection of stable sets such that $\mathcal{C}$ is a packing of $V(D)$ and $|\mathcal{C}| \leq k$. Let $\alpha_k(D) = \max\{\sum_{C\in\mathcal{C}} |C| \colon \mathcal{C}$ is a partial k-coloring of $D\}$. In 1981, Linial [Linial 1981] proposed Conjecture 1 which extends the classical result of Gallai and Milgram (see [Hartman 2006]) that says that the size of a minimum path partition of a digraph is at most its stability number.

Conjecture 1 (Linial, 1981) *If D is a digraph and $k \in \mathbb{Z}^+$, then $\pi_k(D) \leq \alpha_k(D)$.*

A path partition $\mathcal{P}$ of a digraph D and a partial k-coloring $\mathcal{C}$ of D are *orthogonal* if each $P \in \mathcal{P}$ meets $\min\{|V(P)|, k\}$ stable sets in $\mathcal{C}$. In an attempt to unify the proof of Gallai and Milgram's result with another classical result in graph theory (see [Berge 1997]), Berge [Berge 1982] proposed Conjecture 2. It is known that this conjecture implies Conjecture 1 (see [Hartman 2006]).

Conjecture 2 (Berge, 1982) *Let D be a digraph and $k \in \mathbb{Z}^+$. If $\mathcal{P}$ is a k-minimum path partition of D, then there exists a partial k-coloring of D orthogonal to $\mathcal{P}$.*

*O. Lee was partially supported by CNPq (Proc. 311373/2015-1) and FAPESP (Proc. 2015/11937-9). C. N. Lintzmayer was supported by FAPESP (Proc. 2016/14132-4). M. Sambinelli was partially supported by CNPq (Proc. 141216/2016-6) and FAPESP (Proc. 2017/23623-4). E-mails: candida@ufscar.br (C. N. da Silva), lee@ic.unicamp.br (O. Lee), carla.negri@ufabc.edu.br (C. N. Lintzmayer), msambinelli@ic.unicamp.br (M. Sambinelli).

Conjectures 1 and 2 remain open, but they were verified for some particular cases. Conjecture 2, and hence Conjecture 1, was verified for $k = 1$ [Linial 1978], $k = 2$ [Berger and Hartman 2008], $k \geq \lambda - 3$ (where λ is the order of a longest path) [Herskovics 2016], when all the paths from the k-minimum path partition have order at most k [Berge 1982] or at least k [Aharoni and Hartman 1993], acyclic digraphs [Aharoni et al. 1985], and bipartite digraphs [Berge 1982]. Moreover, Conjecture 1 was verified for a superclass of split digraphs [Sambinelli et al. 2017].

Let D be a digraph and k be a positive integer. A *path k-pack $\mathcal{P}$* of D is a collection of paths such that $\{V(P)\colon P \in \mathcal{P}\}$ is a packing of $V(D)$ and $|\mathcal{P}| \leq k$. The *weight* of $\mathcal{P}$, denoted by $||\mathcal{P}||$, is $\sum_{P\in\mathcal{P}} |V(P)|$, and we say that $\mathcal{P}$ is *maximum* if $||\mathcal{P}|| \geq ||\mathcal{P}'||$ for every path k-pack $\mathcal{P}'$ of D. A *coloring $\mathcal{C}$* of D is a collection of stable sets such that $\mathcal{C}$ is a partition of $V(D)$. A path k-pack $\mathcal{P}$ and a coloring $\mathcal{C}$ are *orthogonal* if each stable set $C \in \mathcal{C}$ meets $\min\{|C|, k\}$ paths in $\mathcal{P}$. As a generalization for a conjecture related to Conjecture 1, Aharoni, Hartman, and Hoffman [Aharoni et al. 1985] proposed Conjecture 3 – to understand the relationship among these conjectures, see [Hartman 2006]. Conjecture 3 was verified for $k = 1$ [Gallai 1968], when the maximum path k-pack has at least one trivial path [Hartman et al. 1994], bipartite digraphs [Hartman et al. 1994], and acyclic digraphs [Aharoni et al. 1985].

Conjecture 3 (Aharoni, Hartman, and Hoffman, 1985) *Let D be a digraph and $k \in \mathbb{Z}^+$. If $\mathcal{P}$ is a maximum path k-pack of D, then there is a coloring of D orthogonal to $\mathcal{P}$.*

A digraph D is *semicomplete* if $V(D)$ is a clique, and it is *in-semicomplete* if, for every vertex $v \in V(D)$, the set $\{u\colon uv \in A(D)\}$ is a clique. Note that out-trees, cycles, and semicomplete digraphs are all in-semicomplete digraphs. In-semicomplete digraphs have been well studied in literature [Guo and Volkmann 1994, Bang-Jensen et al. 1997] and have been used as a particular case to confirm open conjectures on digraphs [Bang-Jensen et al. 2006, Galeana-Sánchez and Gómez 2008]. The contributions of this work are the following theorems.

Theorem 1 *If $\mathcal{P}$ is a k-minimal path partition of an in-semicomplete digraph D, then there exists a partial k-coloring of D orthogonal to $\mathcal{P}$.*

Theorem 2 *If $\mathcal{P}$ is a maximum path k-pack of an in-semicomplete digraph D, then there exists a coloring of D orthogonal to $\mathcal{P}$.*

Theorem 3 *Let D be a digraph and let k be a positive integer. If $\mathcal{P} = \{P_1, P_2\}$ is a k-minimum path partition of D, then there exists a partial k-coloring orthogonal to $\mathcal{P}$.*

Theorems 1 and 3 confirm, respectively, Conjecture 2 for in-semicomplete digraphs and for k-minimum path partitions containing only two paths. Theorem 2 confirms Conjecture 3 for in-semicomplete digraphs.

2. Brief outline of the proofs of Theorems 1 and 2

Our proofs for both theorems use induction and follow similar ideas. One important structure that they rely on is the following characterization of in-semicomplete digraphs [Bang-Jensen and Gutin 2009]. Given a collection of paths $\mathcal{P}$, the set of terminal vertices of paths in $\mathcal{P}$ is denote by $\mathrm{ter}(\mathcal{P})$, i.e., $\mathrm{ter}(\mathcal{P}) = \{v_\ell\colon v_1v_2\cdots v_\ell = P \in \mathcal{P}\}$.

Theorem 4 (Bang-Jensen and Gutin, 2009) *A digraph D is in-semicomplete if, and only if, for every vertex v and every pair of internally vertex-disjoint paths P and Q ending at v, there exists a path R ending at v such that $V(R) = V(P) \cup V(Q)$.*

2.1. Theorem 1

Let D be an in-semicomplete digraph, let k be a positive integer, and let $\mathcal{P}$ be a path partition of D. Our proof consists in showing that (i) there exists a partial k-coloring of D orthogonal to $\mathcal{P}$; or (ii) there exists a path partition $\mathcal{B}$ of D such that $|\mathcal{B}|_k < |\mathcal{P}|_k$ and $\mathrm{ter}(\mathcal{B}) \subseteq \mathrm{ter}(\mathcal{P})$. We start our proof by showing that if (ii) does not hold, then there exists a path partition $\mathcal{Q}$ such that $|\mathcal{Q}|_k = |\mathcal{P}|_k$, $\mathrm{ter}(\mathcal{Q}) \subseteq \mathrm{ter}(\mathcal{P})$, $\mathrm{ter}(\mathcal{Q})$ is stable, and every partial k-coloring orthogonal to $\mathcal{Q}$ is also orthogonal to $\mathcal{P}$. Our proof for such result follows by induction on the number of paths in $\mathcal{P}$ with order smaller than k. This reduces the problem of proving the result for $\mathcal{P}$ to the problem of proving it for $\mathcal{Q}$. The remaining proof follows by induction of k. If $k = 1$, then the stable set $\mathrm{ter}(\mathcal{Q})$ is a partial 1-coloring orthogonal to $\mathcal{Q}$, and to $\mathcal{P}$, and the result follows. Otherwise, $k > 1$ and let $D' = D - \mathrm{ter}(\mathcal{Q})$ and $\mathcal{Q}' = \{Q_1 \colon Q_1 u = Q \in \mathcal{Q}\}$. Note that D' is an in-semicomplete digraph and that $\mathcal{Q}'$ is a path partition of D'. By the induction hypothesis applied to D', $\mathcal{Q}'$, and $k-1$, there exists (a) a partial $(k-1)$-coloring $\mathcal{C}$ of D' orthogonal to $\mathcal{Q}'$, or (b) a path partition $\mathcal{R}'$ of D' such that $|\mathcal{R}'|_{k-1} < |\mathcal{Q}'|_{k-1}$ and $\mathrm{ter}(\mathcal{R}') \subseteq \mathrm{ter}(\mathcal{Q}')$. If (a) holds, then $\mathcal{C} \cup \{\mathrm{ter}(\mathcal{Q})\}$ is a partial k-coloring orthogonal to $\mathcal{Q}$ and (i) holds. So we assume that (b) holds and, with the help of Theorem 4, we show how to build a path partition of D satisfying (ii) from $\mathcal{Q}'$.

2.2. Theorem 2

Let D be an in-semicomplete digraph, let k be a positive integer, and let $\mathcal{P}$ be a path k-pack of D. Our proof consists in showing that (i) there exists a coloring of D orthogonal to $\mathcal{P}$, or (ii) there exists a path k-pack $\mathcal{B}$ of D such that $||\mathcal{B}|| = ||\mathcal{P}|| + 1$ and $\mathrm{ter}(\mathcal{B}) \subseteq \mathrm{ter}(\mathcal{P}) \cup \overline{V}_{\mathcal{P}}$, where $\overline{V}_{\mathcal{P}} = V(D) \setminus \cup_{P \in \mathcal{P}} V(P)$. Our proof follows by induction on $|\overline{V}_{\mathcal{P}}|$. If $\overline{V}_{\mathcal{P}} = \emptyset$, then the coloring $\{\{v\} \colon v \in V(D)\}$ is orthogonal to $\mathcal{P}$ and (i) holds. Thus, we may assume $\overline{V}_{\mathcal{P}} \neq \emptyset$. Let w be a vertex in $\overline{V}_{\mathcal{P}}$ and let $\mathcal{Q} = \mathcal{P} \cup \{w\}$. If $|\mathcal{Q}| \leq k$, then $\mathcal{B} = \mathcal{Q}$ satisfies (ii) and the result follows. Thus we may assume that $|\mathcal{Q}| > k$, and, in this case, $|\mathcal{Q}| = k+1$ and $|\mathcal{P}| = k$, since $|\mathcal{P}| \leq k$. Let $S \subseteq \overline{V}_{\mathcal{P}}$ be a maximum stable set in $D[\overline{V}_{\mathcal{P}}]$ and let $Z = \mathrm{ter}(\mathcal{P}) \cup S$. We can prove that Z is a stable set of D. Let $D' = D - Z$, and let $\mathcal{P}' = \{P' \colon P'u = P \in \mathcal{P}\}$. Note that D' is an in-semicomplete digraph, $\mathcal{P}'$ is a path k-pack of D', $||\mathcal{P}'|| = ||\mathcal{P}|| - k$, and $\overline{V}'_{\mathcal{P}} \subset \overline{V}_{\mathcal{P}}$. By the induction hypothesis applied to D' and $\mathcal{P}'$ we have (a) there exists a coloring $\mathcal{C}$ of D' orthogonal to $\mathcal{P}'$, or (b) there exists a path k-pack $\mathcal{Q}'$ of D' such that $||\mathcal{Q}'|| = ||\mathcal{P}'|| + 1$ and $\mathrm{ter}(\mathcal{Q}') \subseteq \mathrm{ter}(\mathcal{P}') \cup \overline{V}'_{\mathcal{P}}$. If (a) holds, then $\mathcal{C} \cup \{Z\}$ is a coloring of D orthogonal to $\mathcal{P}$ and (i) holds. So we assume that (b) holds and, with the help of Theorem 4 and Hall's theorem, we show how to extend the path k-pack $\mathcal{Q}'$ to a path k-pack of D that satisfies (ii).

3. Brief outline of the proof of Theorem 3

Let D be a digraph, let k be a positive integer, and let $\mathcal{P} = \{P_1, P_2\}$ be a k-minimum path partition of D. Let $P_1 = u_1 u_2 \cdots u_\ell$ and let $P_2 = v_1 v_2 \cdots v_r$. If either $\ell, r \leq k$ or $\ell, r \geq k$, then there exists a partial k-coloring orthogonal to $\mathcal{P}$ and the result follows [Berge 1982, Aharoni and Hartman 1993]. Thus we may assume, without loss of generality, that $\ell > k$ and $r < k$, and hence $|\mathcal{P}|_k = k + r$. We can prove that P_1 is a longest path of D. Let $X = V(P_2)$, $Y = V(P_1)$, and G be the $\{X, Y\}$ bipartite graph defined by $V(G) = X \cup Y$ and $E(G) = \{uv \colon u \in X, v \in Y,$ and u and v are not adjacent in $D\}$. Using Hall's theorem, we can show that there exists a matching M in G covering the vertices of X,

otherwise P_1 would not be a longest path in D. Given a vertex $v \in X$, we write $M(v)$ to denote the vertex in $Y = V(P_1)$ matched to v in M. By the construction of G, v and $M(v)$ are non-adjacent. Let $\mathcal{S}_1 = \{\{u, M(u)\} : u \in X\}$, and let $\mathcal{S}_2$ be a set of $k - r$ vertices of P_1 not matched by M. Thus, $\mathcal{S}_1 \cup \{\{u\} : u \in \mathcal{S}_2\}$ is a partial k-coloring orthogonal to $\mathcal{P}$.

References

Aharoni, R. and Hartman, I. B.-A. (1993). On Greene-Kleitman's theorem for general digraphs. *Discrete Math.*, 120(1-3):13–24.

Aharoni, R., Hartman, I. B.-A., and Hoffman, A. J. (1985). Path partitions and packs of acyclic digraphs. *Pacific J. Math.*, 118(2):249–259.

Bang-Jensen, J. and Gutin, G. (2009). *Digraphs: Theory, algorithms and applications*. Springer Monographs in Mathematics. Springer-Verlag London, Ltd., London, 2nd edition.

Bang-Jensen, J. r., Guo, Y., Gutin, G., and Volkmann, L. (1997). A classification of locally semicomplete digraphs. *Discrete Math.*, 167/168:101–114. 15th British Combinatorial Conference (Stirling, 1995).

Bang-Jensen, J. r., Nielsen, M. H., and Yeo, A. (2006). Longest path partitions in generalizations of tournaments. *Discrete Math.*, 306(16):1830–1839.

Berge, C. (1982). k-optimal partitions of a directed graph. *European J. Combin.*, 3(2):97–101.

Berge, C. (1997). Motivations and history of some of my conjectures. *Discrete Math.*, 165/166:61–70. Graphs and combinatorics (Marseille, 1995).

Berger, E. and Hartman, I. B.-A. (2008). Proof of Berge's strong path partition conjecture for $k = 2$. *European J. Combin.*, 29(1):179–192.

Galeana-Sánchez, H. and Gómez, R. (2008). Independent sets and non-augmentable paths in generalizations of tournaments. *Discrete Math.*, 308(12):2460–2472.

Gallai, T. (1968). On directed paths and circuits. In *Theory of Graphs (Proc. Colloq., Tihany, 1966)*, pages 115–118. Academic Press, New York.

Guo, Y. and Volkmann, L. (1994). Connectivity properties of locally semicomplete digraphs. *J. Graph Theory*, 18(3):269–280.

Hartman, I. B.-A. (2006). Berge's conjecture on directed path partitions—a survey. *Discrete Math.*, 306(19-20):2498–2514.

Hartman, I. B.-A., Saleh, F., and Hershkowitz, D. (1994). On Greene's theorem for digraphs. *J. Graph Theory*, 18(2):169–175.

Herskovics, D. (2016). Proof of Berge's path partition conjecture for $k \geq \lambda - 3$. *Discrete Appl. Math.*, 209:137–143.

Linial, N. (1978). Covering digraphs by paths. *Discrete Math.*, 23(3):257–272.

Linial, N. (1981). Extending the Greene-Kleitman theorem to directed graphs. *J. Combin. Theory Ser. A*, 30(3):331–334.

Sambinelli, M., Nunes da Silva, C., and Lee, O. (2017). On Linial's conjecture for spine digraphs. *Discrete Math.*, 340(5):851–854.

Sobre a Dificuldade de Reconhecimento de Grafos B_1-EPG-Helly

Claudson Bornstein[1], Tanilson Santos[1,2], Uéverton Souza[3], Jayme Szwarcfiter[1,4]

[1]Programa de Engenharia de Sistemas e Computação – UFRJ (Coppe/PESC)
Caixa Postal 68511 – 21941-972 – Rio de Janeiro – RJ – Brasil

[2]Curso de Ciência da Computação – Universidade Federal do Tocantins - UFT
Palmas-TO, Brasil

[3]Instituto de Computação – Universidade Federal do Fluminense - UFF
Niterói, Brasil

[4]Departamento de Informática e Ciência da Computação - IME
Universidade do Estado do Rio de Janeiro - UERJ – Rio de Janeiro, RJ – Brasil

{cbornstein,uevertonssouza}@gmail.com,{jayme,tanilson}@cos.ufrj.br

Abstract. *Golumbic, Lipshteyn e Stern have defined in 2009 the EPG graphs, a intersection graph class based on edge intersection of paths on a grid. An EPG graph G is a graph that admits a representation scheme where its vertices are represented by paths on a grid Q, such that two vertices of G are adjacent if and only if their corresponding paths in Q have a common edge. If the paths in Q have at most k changes of direction (bends), we say that this is a B_k-EPG representation. A collection C of sets satisfies the Helly property when every sub-collection of C that is pairwise intersecting has at least one common element. In this paper we show that the problem of recognizing B_1-EPG graphs that satisfy the Helly property is NP-hard.*

Resumo. *Golumbic, Lipshteyn e Stern definiram em 2009 os grafos EPG, uma classe de grafos de intersecção baseada na intersecção de arestas de caminhos em uma grade. Um grafo EPG G é um grafo que admite um esquema de representação onde seus vértices são representados por caminhos de uma grade Q, de forma que dois vértices em G são adjacentes se e somente se os caminhos correspondentes em Q compartilham arestas. Se tais caminhos em Q tem no máximo k mudanças de direção (dobras), então a representação é dita B_k-EPG. Uma coleção C de conjuntos satisfaz a propriedade Helly quando toda subcoleção de C que é mutuamente intersectante possui pelo menos um elemento comum. Neste trabalho mostramos que o reconhecimento de grafos que admitem representação B_1-EPG satisfazendo a propriedade Helly é NP-difícil.*

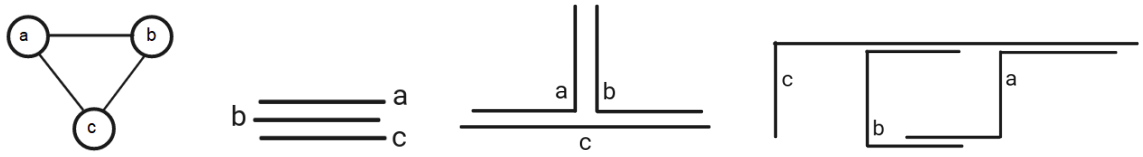
1. Introdução

Um grafo EPG G é um grafo que admite um esquema de representação onde seus vértices são representados por caminhos de uma grade Q, de tal forma que dois vértices em G são adjacentes se e somente os caminhos correspondentes em Q possuem arestas em comum. O estudo de grafos EPG tem motivação prática relacionada ao problema de *layout* de

circuitos digitais, que podem ser representados como caminhos em uma grade ortogonal retangular.

Todo grafo G possui uma representação EPG, onde cada vértice v é representado como um caminho $p(v)$ e dois vértices u, v são adjacentes se e somente se $p(u) \cap p(v) \neq \emptyset$, ver [Golumbic et al. 2009]. Um grafo possui uma representação B_k-EPG se existe uma representação para o grafo na qual cada caminho tem no máximo k dobras.

A classe de grafos EPG tem sido estudada por diversos pesquisadores, atualmente, como [Ries 2009, Alcón et al. 2016, Cohen et al. 2014], entre outros. Essas pesquisas abordam com frequência caracterizações com relação a quantidade de dobras na representação dos grafos. Com relação à complexidade de reconhecimento, somente 3 classes de grafos EPG estão determinadas: B_0-EPG [Golumbic et al. 2009], B_1-EPG [Heldt et al. 2014] e mais recentemente B_2-EPG [Martin Pergel 2017], sendo somente B_0-EPG reconhecido em tempo polinomial, pois corresponde à classe de grafos de intervalo, ver [Booth and Lueker 1976]. A título de exemplo, a Figura 1(a) esboça o grafo C_3, na Figura 1(b) é exibida uma representação EPG onde os caminhos não possuem dobra, na Figura 1(c) uma representação com 1 dobra e na Figura 1(d) uma representação onde todo caminho possui no máximo 2 dobras.



(a) Grafo C_3 (b) Representação B_0-EPG(C_3) (c) Representação B_1-EPG(C_3) (d) Representação B_2-EPG(C_3)

Figura 1. Grafo C_3 e algumas de suas representações sem dobra, com 1 dobra e com 2 dobras

Uma coleção C de conjuntos satisfaz a *propriedade Helly* quando toda subcoleção de C que é mutuamente intersectante possui pelo menos um elemento comum. A *propriedade Helly* tem esse nome em homenagem ao grande matemático austríaco Eduard Helly, que em 1923 propôs seu famoso teorema a respeito do relacionamento de conjuntos intersectantes.

Este trabalho trata de um estudo sobre grafos de intersecção de arestas em grade que possuem a propriedade Helly em sua representação. Nossa contribuição é apresentar uma prova da NP-dificuldade relacionada ao reconhecimento de grafos B_1-EPG-Helly.

2. As classes B_1-EPG e B_1-EPG-Helly

Embora as classes B_1-EPG e B_1-EPG-Helly não coincidam, o mesmo não ocorre entre as classes B_0-EPG e B_0-EPG-Helly que coincidem, como observado no Lema 1.

Lema 1. *Toda representação B_0-EPG satisfaz a propriedade Helly.*

Corolário 2. *A classe de grafos B_0-EPG coincide com a classe B_0-EPG-Helly.*

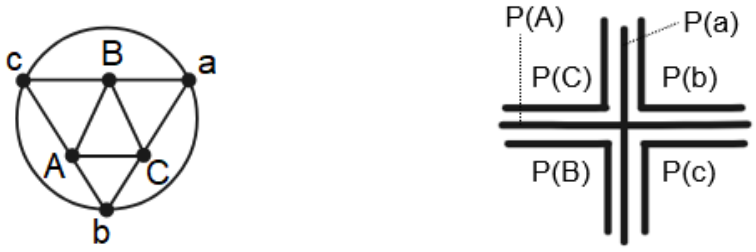
Definição 3. *[Golumbic et al. 2009] Dada uma coleção de caminhos $P = \{p_1, \ldots, p_4\}$ de uma grade Q em uma representação B_1-EPG do grafo G, considere um subgrafo 4-estrela de Q cujo centro está posicionado sobre o ponto b e as arestas $(a_1, b), (a_2, b), (a_3, b), (a_4, b)$, tomadas em sentido horário. Define-se:*

- *Uma* torta verdadeira *é uma* 4*-estrela onde cada segmento* $(a_i, b) \cup (a_{i+1}, b)$*, para* $1 \leq i \leq 4$*, é um membro diferente de* P*, onde adicionalmente é assumido ser módulo 4. Todos os 4 caminhos de* G *fazem dobra em* b*.*
- *Uma* torta falsa *é uma* 4*-estrela onde cada segmento* $(a_1, b) \cup (a_2, b); (a_2, b) \cup (a_4, b); (a_4, b) \cup (a_3, b); (a_3, b) \cup (a_1, b)$*, para* $1 \leq i \leq 4$*, é um membro diferente de* P*. Na torta falsa, somente 2 caminhos, não adjacentes entre si, fazem dobra em* b*.*
- *Considerando um retângulo de qualquer tamanho com 4 cantos* $(x_1, y_1); (x_2, y_1); (x_2, y_2); (x_1, y_2)$*. Um* quadro ou moldura *é um retângulo em que em cada canto está uma dobra diferente para cada membro de* $p_1, \ldots, p_4 \in P$*. Os subcaminhos* $p_1 \cap p_2, p_2 \cap p_3, p_3 \cap p_4, p_4 \cap p_1$*, compartilham pelo menos uma aresta. Enquanto os subcaminhos* $p_2 \cap p_4$ *e* $p_1 \cap p_3$ *não compartilham aresta.*

Figura 2. Representação B_1-EPG de ciclo de tamanho 4: moldura (à esquerda), torta verdadeira (centro) e torta falsa (direita), [Golumbic et al. 2009].

Lema 4. *[Golumbic et al. 2009] Dada uma coleção de caminhos* $P = \{p_1, \ldots, p_4\}$ *de uma grade* Q *em uma representação* B_1-*EPG do grafo* G*. Todo* C_4 *induzido em* G *corresponde, em qualquer representação, a uma torta verdadeira, a uma torta falsa ou a uma moldura.*

Lema 5. *O grafo octaedro* O_3 *possui representação* B_1*-EPG(*O_3*) minimal, a menos de isomorfismos.*



(a) Grafo octaedro O_3 (b) Representação B_1-EPG(O_3)

Figura 3. Grafo octaedro O_3 e sua representação B_1-EPG, [Heldt et al. 2014]

Pelo Lema 5 o grafo octaedro O_3, da Figura 3(a), é isomorfo, em qualquer representação B_1-EPG minimal, à representação da Figura 3(b). Pela Figura 3(b) podemos observar que, por exemplo, os caminhos $p(a), p(b)$ e $p(c)$ formam uma estrutura que não satisfaz a propriedade Helly. Por essa observação e da informação mostrada no Lema 5 segue o seguinte Corolário 6.

Corolário 6. *A classe* B_1*-EPG-Helly é subclasse própria de* B_1*-EPG.*

3. O Reconhecimento de B_1-EPG-Helly é NP-difícil

A prova de NP-dificuldade de reconhecimento de B_1-EPG-Helly envolve uma redução ao problema de (1em3)-3SAT POSITIVO, similar à utilizada por [Heldt et al. 2014] para B_1-EPG. O problema relacionado é o seguinte:

RECONHECIMENTO DE GRAFOS B_1-EPG-HELLY	
Instância:	Um grafo G.
Objetivo:	Determinar se existe um conjunto de caminhos de dobra simples $P = \{p_1, p_2, \ldots, p_n\}$ em uma grade Q representando $V(G)$ tal que: • $u, v \in V(G)$ são adjacentes em G se e somente se p_u, p_v são intersectantes em Q; • P satisfaz a propriedade de Helly.

A redução codifica uma fórmula F pertencente a (1em3)-3SAT POSITIVO em um grafo dispositivo específico G_F. O dispositivo G_F força que a construção de uma representação B_1-EPG-Helly que só existirá caso F seja satisfatível. G_F é subdividido em 3 partes: um dispositivo cláusula, para cada cláusula de F; um dispositivo variável, para cada variável em F; e um dispositivo base para construção da representação.

Uma fórmula F pertencente a (1em3)-3SAT POSITIVO é satisfatível se e somente se existir uma H-B_1-EPG(G_F), onde G_F é um dispositivo construído apropriadamente a partir de F.

Referências

Alcón, L., Bonomo, F., Durán, G., Gutierrez, M., Mazzoleni, M. P., Ries, B., and Valencia-Pabon, M. (2016). On the bend number of circular-arc graphs as edge intersection graphs of paths on a grid. *Discrete Applied Mathematics*, 234:12–21.

Booth, K. and Lueker, G. (1976). Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *Journal of Computer and System Sciences*, 13:335–379.

Cohen, E., Golumbic, M. C., and Ries, B. (2014). Characterizations of cographs as intersection graphs of paths on a grid. *Discrete Applied Mathematics*, 178:46–57.

Golumbic, M., Lipshteyn, M., and Stern, M. (2009). Edge intersection graphs of single bend paths on a grid. *Networks*, 54:130–138.

Heldt, D., Knauer, K., and Ueckerdt, T. (2014). Edge-intersection graphs of grid paths: the bend-number. *Discrete Appl. Math*, 167:144–162.

Martin Pergel, P. R. (2017). On edge intersection graphs of paths with 2 bends. *Discrete Applied Mathematics*, 226:106–116.

Ries, B. (2009). Some properties of edge intersection graphs of single bend paths on a grid. *Electronic Notes in Discrete Mathematics*, 34:29–33.

Grafos Bipartidos Completos em ORTH$[3, 3, t]$

Claudson F. Bornstein[1], **José Wilson C. Pinto**[2,3], **Jayme L. Szwarcfiter**[2,4]

[1]DCC-IM – Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

[2]PESC-COPPE – Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

[3]Faculdade de Educação Tecnológica do Estado do Rio de Janeiro (FAETERJ)
Rio de Janeiro – RJ – Brasil

[4]Universidade do Estado do Rio de Janeiro (UERJ)
Rio de Janeiro – RJ – Brasil

cfb@cos.ufrj.br, jwcoura@cos.ufrj.br, jayme@cos.ufrj.br

Abstract. *In this paper, we investigate under what conditions a graph $K_{m,n}$ belongs to the class* ORTH$[3, 3, t]$ *introduced by [Jamison and Mulder 2000]. We show that $K_{4,4} \notin$ ORTH$[3, 3, 4]$, corroborating a conjecture of Jamison and Mulder in [Jamison and Mulder 2005]. The main result of this work is the proof of the existence of a graph $G \subseteq K_{n,n}$ and $G \in$ ORTH$[3, 3, 2n - 3]$, if n is a power of 2 and $n \geq 4$.*

Resumo. *Neste trabalho, nós investigamos sob quais condições um grafo $K_{m,n}$ pertence à classe* ORTH$[3, 3, t]$ *introduzida por [Jamison and Mulder 2000]. Mostramos que $K_{4,4} \notin$ ORTH$[3, 3, 4]$, corroborando uma conjectura de Jamison e Mulder em [Jamison and Mulder 2005]. O principal resultado deste trabalho é a prova da existência de um grafo $G \subseteq K_{n,n}$ e $G \in$ ORTH$[3, 3, 2n-3]$, se n é uma potência de 2 e $n \geq 4$.*

1. Introdução

Uma (h, s, t)-representação de um grafo G [Jamison and Mulder 2000] consiste em representar G como grafo de interseção de subárvores, de grau máximo s, com $s \leq h$, de uma árvore hospedeira T de grau máximo h, tal que dois vértices são adjacentes em G se e somente se as suas subárvores têm interseção de tamanho pelo menos t, ou seja, compartilham no mínimo t nós em T. Denotamos por $[h, s, t]$ a classe de grafos que admitem uma (h, s, t)-representação. Um grafo G é ORTH$[h, s, t]$ se possui uma (h, s, t)-*representação ortodoxa*. Isto é, uma (h, s, t)-representação de G na qual cada uma das subárvores que representam os vértices de G são tais que suas folhas também são folhas da árvore hospedeira T, e duas subárvore compartilham uma folha se e somente se os vértices correspondentes são adjacentes em G. As classes $[h, s, t]$ e ORTH$[h, s, t]$ já são estudadas há algum tempo. Os grafos da classe $[\infty, \infty, 1]$ são grafos de interseção de subárvores, e em 1974 Gavril [Gavril 1974] mostrou que esta classe corresponde à família dos grafos cordais. Por outro lado, a classe $[\infty, \infty, 2]$ inclui todos os grafo, uma vez que é possível representar um grafo arbitrário G como um grafo de interseção de subestrelas de uma estrela. A estrela consiste de um

vértice central c e um vértice para cada aresta de $E(G)$. Cada vértice $v \in V(G)$ corresponde a uma estrela de centro c e os vértices correspondentes as arestas que incidem em v [Jacobson et al. 1991, Teorema 2.1]. Em [Jamison and Mulder 2000, Jamison and Mulder 2005], os autores atribuem a [McMorris and Scheinerman 1991] o resultado $[\infty, \infty, 1] = \mathrm{ORTH}[3, 3, 1] = \mathrm{ORTH}[3, 3, 2]$. As classes $[\infty, 2, 1]$ e $[\infty, 2, 2]$ que coincidem, respectivamente, com grafos VPT e grafos EPT foram bastante estudas em [Golumbic and Jamison 1985b, Golumbic and Jamison 1985a]. Golumbic, Lipshteyn, e Stern [Golumbic et al. 2008] estudaram em detalhes as classes $[h, 2, t]$ e $\mathrm{ORTH}[h, 2, t]$, e mostraram que $\mathrm{ORTH}[\infty, 2, 1] = \mathrm{ORTH}[3, 2, 1] = \mathrm{ORTH}[3, 2, 2]$, e que $\mathrm{ORTH}[\infty, 2, 1]$ é uma subclasse própria de $\mathrm{ORTH}[\infty, 2, 2]$. Neste trabalho os autores perguntam se $\mathrm{ORTH}[\infty, 2, t]$ e $\mathrm{ORTH}[3, 2, t]$ coincidem. Em [Bornstein et al. 2017], foi mostrado que $\mathrm{ORTH}[\infty, 2, t]$ e $\mathrm{ORTH}[3, 2, t]$ não coincidem e foi exibido um conjunto infinito de grafos que pertencem a $\mathrm{ORTH}[\infty, 2, t]$ e não pertencem a $\mathrm{ORTH}[3, 2, t]$. Em [Jamison and Mulder 2000], os autores estudam detalhadamente as classes $[3, 3, t]$ e $\mathrm{ORTH}[3, 3, t]$. Eles mostram que $[\infty, \infty, 1] = [3, 3, 1] = \mathrm{ORTH}[3, 3, 1] = [3, 3, 2] = \mathrm{ORTH}[3, 3, 2]$. Além disso, a Conjectura 1 é apresentada juntamente com uma questão *Qual o menor valor de t tal que* $K_{n,n} \in [3, 3, t]$ *?* Nosso trabalho exibe na Seção 2 a prova que $K_{4,4} \notin \mathrm{ORTH}[3, 3, 4]$ que corrobora a Conjectura 1. Na Seção 3 provamos que um grafo $G \subseteq K_{n,n} \in \mathrm{ORTH}[3, 3, 2n - 3]$ para n potência de 2 maior ou igual a 4.

Conjectura 1. *Para* $n \geq 3$*, o grafo* $K_{n,n}$ *tem uma representação* $(3, 3, n)$*-fiel, mas não tem uma representação* $(3, 3, n)$*-ortodoxa ou uma representação* $(3, 3, t)$*, com* $t < n$*.*

2. $K_{n,n} \notin \mathbf{ORTH}[3, 3, n]$

Nesta seção mostramos que o grafo $K_{4,4}$ não pertence à classe $\mathrm{ORTH}[3, 3, 4]$, o que reforça a Conjectura 1.

Teorema 2. *O grafo* $K_{4,4} \notin \mathrm{ORTH}[3, 3, 4]$.

Prova. Considere o grafo $G = K_{4,4}$ com partes $\mathcal{L} = \{a, b, c, d\}$ e $\mathcal{N} = \{1, 2, 3, 4\}$. Suponha que $K_{4,4} \in \mathrm{ORTH}[3, 3, 4]$ e que $(T, \mathcal{S})$ seja sua representação. Cada vértice $v \in V(G)$ corresponde a uma subárvore $S_v \in \mathcal{S}$. Considere $\mathcal{S}_\mathcal{L} \subset \mathcal{S}$ o conjunto das subárvores que correspondem a vértices de $\mathcal{L}$ e $\mathcal{S}_\mathcal{N} \subset \mathcal{S}$ o conjunto das subárvores que correspondem a vértices de $\mathcal{N}$. Como a representação é ortodoxa a árvore hospedeira T deve possuir, no mínimo, $|\mathcal{L} \times \mathcal{N}| = 16$ folhas. Para facilitar vamos dividir nossa prova em três casos. Caso 1: em $(T, \mathcal{S})$, a distância entre duas subárvores de $\mathcal{S}_\mathcal{N}$, digamos S_1 e S_2, é maior ou igual a 2 (duas arestas). Neste caso, existe um caminho mais curto P entre S_1 e S_2, com $|P| \geq 3$ (nós). Sejam $x = P \cap S_1$, $z = P \cap S_2$ e y um nó de P tal que $y \notin S_1 \cup S_2$. Obrigatoriamente, as quatro subárvores de $\mathcal{S}_\mathcal{L}$ compartilham todos os nós de P. Assim, $|P| \geq 4$, implica uma interseção proibida entre subárvores de $\mathcal{S}_\mathcal{L}$. Se $|P| = 3$, como duas subárvores quaisquer de $\mathcal{S}_\mathcal{L}$ não podem compartilhar folhas, , e T possui grau máximo 3, existe um nó $x' \notin P$, que é compartilhado por ao menos duas subárvores de $\mathcal{S}_\mathcal{L}$. Então, existem duas subárvores de $\mathcal{S}_\mathcal{L}$ cuja interseção tem tamanho 4, o que não é permitido nesta representação. Caso 2: a distância entre duas subárvores de $\mathcal{S}_\mathcal{N}$, digamos S_1 e S_2, é igual a 1. Considere a aresta xy onde $x \in S_1$ e $y \in S_2$. Obrigatoriamente, as quatro subárvores de $\mathcal{S}_\mathcal{L}$ compartilham xy. Sejam T_x e T_y subárvores de $T - xy$ com raízes x e y, respectivamente. Duas subárvores de $\mathcal{S}_\mathcal{L}$ não podem compartilhar um nó de nível maior que 2 em T_x ou T_y, caso contrário a interseção entre elas será maior que 3, o que produz

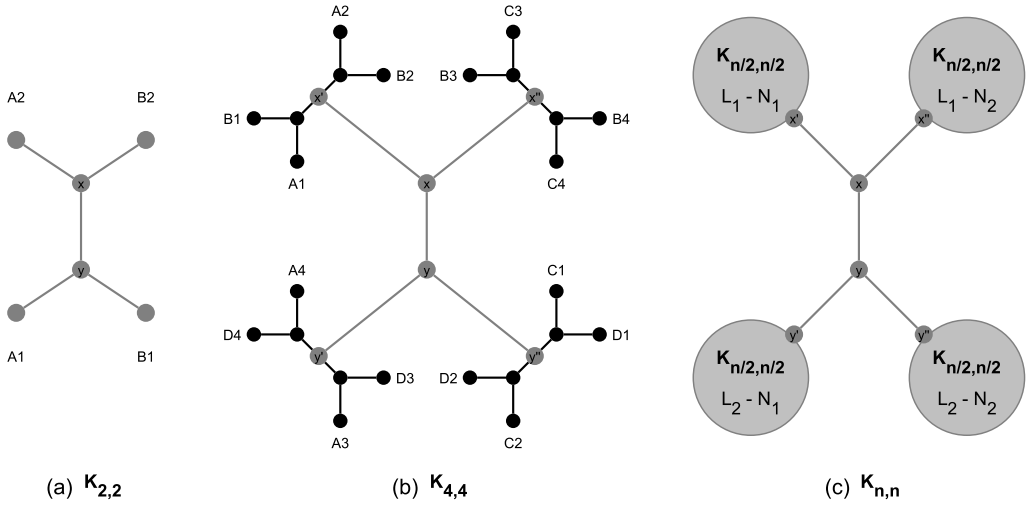


Figure 1. Construção da representação do $\mathrm{K_{n,n}}$ a partir da construção do $\mathrm{K_{n/2,n/2}}$

uma adjacência proibida. Duas subárvores de $\mathcal{S}_{\mathcal{N}}$ não podem compartilhar um nó de nível maior que 3 em T_x ou T_y, caso contrário existirá interseção de tamanho 4. Em uma representação ortodoxa todos os nós internos de T são compartilhados por pelo menos duas subárvores da mesma parte. Então, todos os nós de nível 4 são folhas. Como já vimos anteriormente, T possui no mínimo 16 folhas. Sendo assim, a subárvore T_x possui 4 nós internos de nível 3, os quais devem ser compartilhados por pares de subárvores de $\mathcal{S}_{\mathcal{N}}$. Em T_x, as três subárvores S_1, S_3 e S_4 de $\mathcal{S}_{\mathcal{N}}$, compartilham o nó x, caso contrário a distância até S_2 seria maior que 1. Cada par de subárvores de $\mathcal{S}_{\mathcal{N}}$ compartilha apenas um nó no nível 3. Como em T_x temos quatro nós de nível 3 e apenas os três pares (S_1S_3), (S_1S_4) e (S_3S_4) não é possível construir esta representação. Caso 3: a distância entre duas subárvores quaisquer de $\mathcal{S}$ é zero, ou seja, todas as subárvores de $\mathcal{S}$ compartilham um nó x. O nível de um nó v é a quantidade de nós do caminho entre x e v. Como todo nó interno é compartilhado por pelo menos duas subárvores da mesma parte, não podemos ter nós internos com nível maior que 3. Sendo assim, todos os nós de nível 4 são folhas. Como o grau máximo de T é 3, a quantidade máxima de folhas nesta árvore é 12. Entretanto, são necessárias no mínimo 16 folhas para construir a representação. Logo, não é possível construir uma representação $(3, 3, 4)$-ortodoxa para um $K_{4,4}$. □

3. $K_{n,n} \in$ ORTH$[3, 3, t]$

Nesta seção, apresentamos um valor para t tal que existe um grafo $G \subseteq K_{n,n}$ que pertence à classe $\mathrm{ORTH}[3, 3, t]$. Para simplificar, consideramos que duas subárvores que compartilham uma mesma folha possuem entre si interseção mínima de tamanho t. A Figura 1(a) é uma representação $(3, 3, 3)$-ortodoxa $(T, \mathcal{S})$ de um grafo $G = K_{2,2}$. Considere em G duas partes $\mathcal{L} = \{A, B\}$ e $\mathcal{N} = \{1, 2\}$. Cada vértice $i \in V(G)$ está associado a uma subárvore $S_i \in \mathcal{S}$. Cada subárvore S_i é gerada pelas folhas que possuem i em seu rótulo, ou seja, é formada pela união destas folhas e os caminhos entre elas. A Figura 1(b) é uma representação $(3, 3, 5)$-ortodoxa $(T, \mathcal{S})$ de um grafo $H = K_{4,4}$. Considere em H duas partes $\mathcal{L} = \{A, B, C, D\}$ e $\mathcal{N} = \{1, 2, 3, 4\}$. Observe que a representação $(T, \mathcal{S})$ de H é composta por uma aresta xy, chamada *aresta central*, dois

nós x', x'' adjacentes a x e dois nós y', y'' adjacentes a y. Cada um dos nós x', x'', y' e y'' é raiz da subárvore binária X', X'', Y' e Y'', respectivamente. Repare que a subárvore X' é exatamente a representação de um $K_{2,2}$ com sua aresta central subdividida pelo nó x'. As demais subárvores X'', Y' e Y'' são construídas de forma análoga. Seguindo este esquema, construímos uma representação $(3, 3, 2n-3)$-ortodoxa de um grafo $K_{n,n}$, baseado na representação de $K_{n/2,n/2}$.

Teorema 3. *Seja $n \geq 4$ uma potência de 2. Então existe $G \subseteq K_{n,n}$ e $G \in \mathrm{ORTH}[3, 3, 2n-3]$.*

Prova. Sejam $\mathcal{L} = \{a_1, ..., a_n\}$ e $\mathcal{N} = \{b_1, ..., b_n\}$ as partes da bipartição de $K_{n,n}$. Considere $\mathcal{L}_1$ e $\mathcal{L}_2$ dois subconjuntos disjuntos de $\mathcal{L}$, cada um com $n/2$ vértices, e $\mathcal{N}_1$ e $\mathcal{N}_2$ dois subconjuntos disjuntos de $\mathcal{N}$, cada um com $n/2$ vértices. Considere uma aresta central xy e quatro nós x', x'', y' e y'' conforme a Figura 1(c). Cada um dos nós x', x'', y' e y'' é raiz da subárvore binária X', X'', Y' e Y'', respectivamente. Cada uma das subárvores X', X'', Y' e Y'' é uma representação $(3, 3, 2(n/2) - 3)$-ortodoxa de um $K_{n/2,n/2}$. Como observado na Figura 1(c), em X' as partes do $K_{n/2,n/2}$ são $\mathcal{L}_1$ e $\mathcal{N}_1$, em X'' são $\mathcal{L}_1$ e $\mathcal{N}_2$, em Y' são $\mathcal{L}_2$ e $\mathcal{N}_1$, em Y'' são $\mathcal{L}_2$ e $\mathcal{N}_2$. Na representação $(T, \mathcal{S})$ do $K_{4,4}$, Figura 1(b), o tamanho da maior interseção entre subárvores da mesma parte é 4. Pela construção da representação $(T, \mathcal{S})$ de um $K_{n,n}$, temos que o tamanho da maior interseção entre subárvores da mesma parte é definido pela relação de recorrência $I(n) = 2I(n/2) + 4$, para $n > 4$ e potência de 2. Assim, o tamanho da maior interseção entre subárvores da mesma parte é $t = 2n - 4$. Logo, $G \subseteq K_{n,n}$ e $G \in \mathrm{ORTH}[3, 3, t]$ para $t = 2n - 3$. □

References

Bornstein, C. F., Coura Pinto, J. W., Rautenbach, D., and Szwarcfiter, J. L. (2017). Constant Threshold Intersection Graphs of Orthodox Paths in Trees. *ArXiv e-prints*.

Gavril, F. (1974). The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47 – 56.

Golumbic, M. C. and Jamison, R. E. (1985a). Edge and vertex intersection of paths in a tree. *Discrete Mathematics*, 55(2):151 – 159.

Golumbic, M. C. and Jamison, R. E. (1985b). The edge intersection graphs of paths in a tree. *Journal of Combinatorial Theory, Series B*, 38(1):8 – 22.

Golumbic, M. C., Lipshteyn, M., and Stern, M. (2008). Equivalences and the complete hierarchy of intersection graphs of paths in a tree. *Discrete Applied Mathematics*, 156(17):3203 – 3215.

Jacobson, M., McMorris, F., and Mulder, H. (1991). An introduction to tolerance intersection graphs. *Graph Theory, Combinatorics and Applications*, 2:705–723.

Jamison, R. E. and Mulder, H. M. (2000). Tolerance intersection graphs on binary trees with constant tolerance 3. *Discrete Mathematics*, 215(1 - 3):115 – 131.

Jamison, R. E. and Mulder, H. M. (2005). Constant tolerance intersection graphs of subtrees of a tree. *Discrete Mathematics*, 290(1):27 – 46.

McMorris, F. R. and Scheinerman, E. R. (1991). Connectivity threshold for random chordal graphs. *Graphs and Combinatorics*, 7(2):177–181.

O problema probe particionado split bem-coberto é polinomial

S. R. Alves[1], F. Couto[2], L. Faria[3], S. Klein[4], U. dos S. Souza[5]

[1]FAETEC/RJ – Rio de Janeiro, [2]UFRRJ – Nova Iguaçu, [3]UERJ – Rio de Janeiro
[4]UFRJ – Rio de Janeiro, [5]UFF – Niterói
Brazil

sancrey@gmail.com, nandavdc@gmail.com, luerbio@cos.ufrj.br,

sula@cos.ufrj.br, uevertonssouza@gmail.com

Abstract. *A graph $G = (V, E)$ is* well-covered *if each maximal independent set of G is maximum. A graph $G = (V, E)$ is* split *if there is a partition $V = (S, K)$, where S is an independent set and K is a clique. A* well-covered split *graph is at a same time well-covered and split. Given a class $\mathcal{C}$ of graphs, a graph $G = (V, E)$ is $\mathcal{C}-$probe if there is a partition for $V = (N, P)$ into probes P and non-probes N, where N is an independent set and new edges may be added between non-probes such that the resulting graph is in the graph class $\mathcal{C}$. We say that (N, P) is a $\mathcal{C}$ probe partition for G. The $\mathcal{C}-$*PARTITIONED PROBE *problem consists of an input graph G and a probe partition $V = (N, P)$ and the question: Is (N, P) a $\mathcal{C}-$probe partition? In this paper we consider $\mathcal{C}$ the class of well-covered split graphs, and we prove that $\mathcal{C}-$*PARTITIONED PROBE *problem belong to the polynomial problem class P.*

Resumo. *Um grafo $G = (V, E)$ é* bem-coberto *se cada conjunto independente maximal de G é máximo. Um grafo $G = (V, E)$ é* split *se existe uma partição $V = (S, K)$, onde S é um conjunto independente e K é uma clique. Um grafo* split bem-coberto *é, ao mesmo tempo, split e bem-coberto. Dada uma classe $\mathcal{C}$ de grafos, um grafo $G = (V, E)$ é $\mathcal{C}-$*probe *se existe uma partição para $V = (N, P)$ em probes P e não-probes N, onde N é um conjunto independente e novas arestas podem ser adicionadas entre não-probes de maneira que o grafo resultante permaneça na classe de grafos $\mathcal{C}$. Dizemos que (N, P) é uma $\mathcal{C}$* probe partição *para G. O problema $\mathcal{C}-$*PROBE PARTICIONADO *consiste de um grafo de entrada G e uma partição probe $V = (N, P)$ e a questão: (N, P) é uma $\mathcal{C}-$partição probe? Neste artigo, consideramos $\mathcal{C}$ como a classe dos grafos split bem-cobertos, e provamos que o problema $\mathcal{C}-$*PROBE PARTICIONADO *pertence à classe de problemas polinomiais P.*

Dado um grafo $G = (V, E)$, um vértice u é *adjacente* a v se $uv \in E(G)$. Dado $S \subset V$, o conjunto $N_S(u) = \{\forall v \in V(G) : uv \in E\}$ é chamado de *vizinhança aberta de u em S*. Denotamos $N(v)$ a vizinhança aberta de v em G. H é um *subgrafo* de G se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$. Uma *clique* $K \subset V$ é tal que para cada par de vértice $u, v \in K$, $uv \in E$. Um *conjunto independente* $S \subset G$ é tal que para cada par de vértice $u, v \in S$, $uv \notin E$. Um conjunto $S \subset V$ é um *conjunto independente maximal* de G, se não há conjunto independente de G contendo propriamente S. Um grafo é *split* quando seu conjunto de vértices pode ser particionado em uma clique e um conjunto independente. Um grafo G é dito *bem-coberto* se cada conjunto independente

maximal é máximo. Embora o o problema de reconhecimento BEM COBERTO seja coNP-completo [Chvátal and Slater 1993] para grafos em geral, provamos que o problema de reconhecimento SPLIT BEM-COBERTO [Alves et al. 2016] é um problema polinomial.

Teorema 1 *[Alves et al. 2016] Um grafo G é um grafo split bem-coberto se, e somente se, admite uma partição* split bem coberta $V = (S, K)$ *onde S é um conjunto independente e K uma clique tal que, ou $\forall x \in K$, $|N_S(x)| = 0$, ou $\forall x \in K$, $|N_S(x)| = 1$.*

Os vértices $u \in S$ adjacentes a vértices $v \in K$ são ditos *testemunhas* de v. Assim, o Teorema 1 garante, em outras palavras, que um grafo é split bem-coberto se, e somente se admite uma partição de V em um conjunto independente S e uma clique K, tal que, ou os vértices de K não têm testemunhas, ou cada vértice de K tem exatamente uma testemunha. Golumbic, Kaplan e Shamir [Golumbic et al. 1995] provaram que PROBE PARTICIONADO SPLIT é polinomial. Nesta seção propomos um algoritmo polinomial para o problema de decisão PROBE PARTICIONADO SPLIT BEM-COBERTO.

PROBE PARTICIONADO SPLIT BEM-COBERTO - (PPSPLIT-WC)
INSTÂNCIA: Grafo $G = (V, E)$ e uma partição probe $V = (N, P)$, onde N é um conjunto independente de G.
PERGUNTA: Existe um conjunto $E' \subseteq N \times N$ tal que $H = (V, E \cup E')$ é um grafo split bem-coberto?

Teorema 2 *Se a instância $(G = (V, E), N, P)$ é sim para o problema* PROBE PARTICIONADO SPLIT*, então G é split e uma solução com o grafo split bem-coberto H com partição split bem coberta (S, K) é tal que que a partição split $P = (P_S, P_K) = (P \cap S, P \cap K)$ de P é uma das $O(n^2)$ possíveis partições de P do grafo split H.*

O Algoritmo 1 proposto utiliza o conceito de partição esparso-denso proposto por [Feder et al. 1999], precisamente no fato que existe um número polinomial $O(n^2)$ de partições esparso-densas (S, K) de um grafo split. Mais ainda, uma vez conhecendo uma, as demais podem ser encontradas em tempo polinomial. Nosso algoritmo vai olhar para cada uma dessas partições uma única vez. Assim, dada uma partição split obtida em tempo polinomial [Golumbic 1980] para o grafo de entrada G e, dada $P = (P_S, P_K)$ uma partição split esparso-densa [Feder et al. 1999] obtida a partir da partição inicial split para G, onde P_S é um conjunto independente e P_K é uma clique de G, observamos que N pode ser particionado em dois conjuntos: $N_S = \{u \in N :$ existe $uv \notin E$ tal que $v \in P_K\}$ e $N_K = \{u \in N : uv \in E$ para todo $v \in P_K\}$. Nós consideramos cada um dos possíveis casos: $(N_S, N_K) = (\emptyset, \emptyset)$, $(N_S, N_K) = (\emptyset, \neq \emptyset)$, $(N_S, N_K) = (\neq \emptyset, \emptyset)$, e $(N_S, N_K) = (\neq \emptyset, \neq \emptyset)$. A seguir, explicaremos cada um destes casos bem como as respectivas tomadas de decisão. Em nossa notação $K(S), S \subset V$ significa o grafo obtido de $G[S]$ pela adição de todas as arestas entre os vértices de $N \cap S$.

Para a corretude do Algoritmo 1, vamos analisar os seguintes casos:

1. G é split bem-coberto. Caso trivial que contempla os casos: $(N_S, N_K) = (\emptyset, \emptyset)$ e $(N_S, N_K) = (\neq \emptyset, \emptyset)$ descritos acima. Com partição split bem coberta $(N_S \cup P_S, N_K \cup P_K) = (S, K)$.
2. $N_S = \emptyset$, $N_K \neq \emptyset$ e $P_S = \emptyset$. Observe que, basta adicionar todas as arestas possíveis entre os vértices de N_K que teremos um grafo split bem-coberto com partição split bem coberta $(S, K) = (\emptyset, K)$ onde cada vértice de K tem zero vizinhos em S.

Algoritmo 1: GRAFO SPLIT BEM-COBERTO PROBE PARTICIONADO.

Entrada: Grafo $G = (V, E)$ tal que $V = P \cup N$, N é um conjunto independente.
Saída: Resposta SIM com grafo-split bem-coberto $H = (V, E \cup E')$, tal que $E' \subseteq N \times N$ junto com partição split bem coberta (S, K), ou resposta NÃO caso tal grafo H não exista.

```
1  início
2    se G é split bem-coberto então
3      retorna SIM – (N_S ∪ P_S, N_K ∪ P_K) = (S, K).
4    para CADA PARTIÇÃO SPLIT DE G[P] faça
5      se N_S = ∅ e N_K = ∅ então
6        se G[P] é um grafo split bem-coberto então
7          retorna SIM – (P_S, P_K) = (S, K).
8      se N_S = ∅ e N_K ≠ ∅ então
9        se P_S = ∅ então
10         retorna SIM – (∅, N_K ∪ P_K) = (S, K).
11       se P_S ≠ ∅ então
12         se (N(P_S) ∩ P_K) ≠ ∅ então
13           se (P_S, K(N_K ∪ P_K)) é split bem-coberto então
14             retorna SIM – (P_S, N_K ∪ P_K) = (S, K).
15         senão
16           se ∃u ∈ (N_K\N(P_S)), tal que o grafo H = (V, E ∪ E'),
               E' = {uv : v ∈ ((N_K\N(P_S)) ∪ P_K)}∪
               {xy : x, y ∈ N_K \ {v}, x ≠ y} é split bem-coberto então
17             retorna SIM – Onde H = (V, E ∪ E'),
                 E' = {uv : v ∈ ((N_K\N(P_S)) ∪ P_K)}∪
                 {xy : x, y ∈ N_K \ {u}, x ≠ y},
                 (S, K) = (P_S ∪ {u}, P_K ∪ (N_K\{u})).
18     se (N_S ≠ ∅ e N_K = ∅) então
19       se (P_S ∪ N_S, P_K) é uma partição split bem coberta, então
20         retorna SIM – (P_S ∪ N_S, P_K).
21     se (N_S ≠ ∅ e N_K ≠ ∅) então
22       se (P_S ∪ N_S, K(N_K ∪ P_K)) é uma partição split bem coberta, então
23         retorna SIM – (N_S ∪ P_S, K(N_K ∪ P_K)).
24   retorna NÃO
25 Fim.
```

3. $N_S = \emptyset$, $N_K \neq \emptyset$, $P_S \neq \emptyset$ e $(N(P_S) \cap P_K) \neq \emptyset$. Uma vez que existe pelo menos um vértice em P_S com adjacência em P_K, pelo Teorema 1, é necessário que cada um dos vértices de P_K tenha exatamente um vizinho em P_S. Primeiramente, note que as testemunhas dos vértices de P_K devem pertencer apenas a P_S, caso contrário, pela definição de N_S, um vértice em P_K teria mais de uma testemunha. Assim, após tornarmos N_K uma clique, basta verificar se o grafo resultante satisfaz tal condição necessária. Se o grafo resultante não satisfizer a condição necessária descrita pelo Teorema 1, uma nova partição deve ser considerada.
4. $N_S = \emptyset$, $N_K \neq \emptyset$, $P_S \neq \emptyset$ e $(N(P_S) \cap P_K) = \emptyset$. Como P_S não tem vizinhança em P_K e G não é split bem-coberto, P_S tem vizinhança em N_K. Neste caso,

todos os vértices de K têm que possuir testemunha e, como os vértices de P_K não têm vizinhos em P_S e $N_S = \emptyset$, é necessário (e suficiente) que exista um vértice u em N_K que servirá como testemunha para os vértices de P_K. Note que u é adjacente a todos os vértices de P_K. Além disso, se, eventualmente, existirem outros vértices em N_K que não estão na vizinhança de P_S, u também servirá como testemunha para eles. Portanto, se a existência do vértice u em N_K for confirmada e adicionarmos arestas entre cada par de vértices de $N_K \setminus \{u\}$, basta verificarmos se o grafo resultante é split bem-coberto. Em caso negativo, existe um vértice em K com mais de uma testemunha (não é possível ter vértice em K sem testemunha, por construção). Observe que tais testemunhas estão em P_S, sendo inerentes, portanto, à partição considerada. Logo, neste caso, uma nova partição deve ser considerada.

5. $N_S \neq \emptyset$, $N_K \neq \emptyset$. Observe que, como G não é split bem-coberto, em $N_S \cup P_S$ deve existir pelo menos uma testemunha para $N_K \cup P_K$ e, neste caso, todas as testemunhas pertencem a $N_S \cup P_S$. Sendo assim, após fazermos $G[P_K \cup N_K]$ ser uma clique, resta-nos verificar se o grafo resultante é split bem-coberto. Em caso negativo, como, por construção, o grafo é split, existem vértices em K sem testemunha ou com mais de uma testemunha. Como as arestas adicionadas não acrescentam testemunhas, o problema de ter mais de uma testemunha está na partição considerada. Se, eventualmente, um vértice de N_K ficou sem testemunha, este problema também se deve à partição, uma vez que todas as testemunhas estão restritas ao conjunto $N_S \cup P_S$. Logo, em ambos os casos, uma nova partição deve ser analisada.

Finalmente, checar se um grafo é split bem-coberto, toma $O(m)$ passos, o passo 8 que é o mais custoso, toma $O(nm)$, e o número de partições esparso-densas é $O(n^2)$. O que totaliza a complexidade de tempo $O(n^3m)$.

Referências

Alves, S. R., Dabrowski, K. K., Faria, L., Klein, S., Sau, I., and Souza, U. S. (2016). On the (parameterized) complexity of recognizing well-covered (r, ℓ)-graphs. In *COCOA 2016, Hong Kong, China, 2016*, pages 423 – 437. LNCS.

Chvátal, V. and Slater, P. J. (1993). A note on well-covered graphs. In John Gimbel, J. W. K. and Quintas, L. V., editors, *Quo Vadis, Graph Theory?*, volume 55 of *Annals of Discrete Mathematics*, pages 179 – 181. Elsevier.

Feder, T., Hell, P., Klein, S., and Motwani, R. (1999). Complexity of graph partition problems. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, pages 464 – 472. ACM, New York, NY, USA.

Golumbic, M., Kaplan, H., and Shamir, R. (1995). Graph sandwich problems. volume 19, pages 449 – 473.

Golumbic, M. C. (1980). Algorithmic graph theory and perfect graphs. In *Academic Press*, volume 1, pages 149 – 156. Elsevier Science.

O Problema da Deposição Gamma é NP-Completo

Marcelo Fonseca Faraj[1], **Sebastián Urrutia**[1], **João Fernando Machry Sarubbi**[2]

[1]DCC – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

[2]DECOM – Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)
Belo Horizonte – MG – Brasil

marcelo@gmail.com, surrutia@dcc.ufmg.br, joao@decom.cefetmg.br

Abstract. *Gamma Deployment is a metric to evaluate the quality of service in vehicular ad hoc networks (VANETs). Gamma Deployment Problem consists in applying that metric to minimize the amount of RSUs deployed in a VANET. In this work, we present a proof that the that problem is NP-Complete.*

Resumo. *Deposição Gamma é uma métrica usada para avaliar a qualidade de serviço oferecida por redes veiculares (VANETs). O Problema da Deposição Gamma consiste no emprego dessa métrica na minimização de RSUs para compor VANETs. Neste trabalho, prova-se que esse problema é NP-Completo.*

1. Introdução

Deposição Gamma ($\Gamma_D\binom{\tau}{\rho}$) (SILVA et al., 2016) é uma métrica para avaliar a qualidade de serviço de deposições de RSUs em redes veículares (VANETs) a partir de seu tráfico veicular. Ele disponibiliza dois parâmetros para o arquiteto da VANET atingir seus objetivos de projeto: o tempo de intercontato τ e cobertura percentual mínima exigida ρ. Um veículo é considerado coberto pela VANET caso não permaneça mais que τ segundos sem cruzar com alguma RSU. O parâmetro ρ fixa a fração mínima de veículos a se cobrir. Embora $\Gamma_D\binom{\tau}{\rho}$ tenha sido proposta como uma métrica, sua aplicação para minimizar a deposição de RSUs define o Problema da Deposição Gamma (PDG). Silva et al. (2016) propuseram uma formulação de programação linear inteira e uma heurística determinística para o PDG. Faraj et al. (2017) propõem um algoritmo memético para o PDG. Neste trabalho, prova-se que o PDG é NP-Completo, o que era apenas uma conjectura.

2. O Problema da Deposição Gamma

Seja um passeio T_i uma sequência de vértices tal que haja aresta entre cada par de vértices consecutivos. Seja $L(T_i)$ o número de vértices (distintos ou não) em T_i e seja $T_i[j]$ o vértice na posição j de T_i, $j \in \{1, ..., L(T_i)\}$. Considere-se a operação de subtrair um conjunto R de um passeio T_i como resultando em uma coleção de subpasseios de T_i, a qual é obtida pela remoção em T_i de todos os elementos em R. Essa operação será denotada por $T_i \setminus R$, como no exemplo: $(1, 7, 2, 3, 1, 4, 5, 6, 7) \setminus \{1, 5\} = \{(7, 2, 3), (4), (6, 7)\}$.

Definição 1 (PDG). *Seja $G = (V, E)$ um grafo e $T = \{T_1, ..., T_p\}$ uma coleção de passeios em G. Seja $F : T \times \mathbb{Z}_+^* \to \mathbb{R}_+^*$ uma função associando um número positivo a cada etapa j de cada passeio $T_i \in T$, $j \in \{1, ..., L(T_i)\}$. Seja $\tau \in \mathbb{R}_+^*$, $\rho \in [0\%, 100\%]$ e $K \in \mathbb{Z}_+^*$ parâmetros adicionais ao problema. O PDG consiste em determinar se existe ou não um conjunto $R \subseteq V$ tal que $|R| \leq K$ e, para pelo menos uma fração ρ dos passeios $T_i \in T$, é satisfeita a desigualdade $\sum_{c=1}^{L(C)} F(C, c) < \tau$ para todo $C \in T_i \setminus R$.*

Para certificar a corretude de uma instância SIM, deve-se verificar se cada $C \in T_i \setminus R$ satisfaz $\sum_{c=1}^{L(C)} F(C,c) < \tau$, o que pode ser executado em tempo polinomial, uma vez que $|T_i \setminus R| < L(T_i)$ Assim, fica claro que o PDG pertence à classe NP. Silva et al. (2016) trataram da situação em que G possuía topologia de grade e propuseram uma discretização simples para expressar qualquer rede rodoviária como grade. Por isso, este trabalho dá um enfoque especial ao Problema da Deposição Gamma em Grades (PDGG).

3. As Demonstrações

A partir do Problema da Cobertura de Arestas por Vértices em Grafos Planares com Grau Máximo 3 (PCAVGPGM3), que é NP-Completo (GAREY; JOHNSON, 1977), será feita uma redução polinomial ao Problema da Cobertura de Caminhos por Vértices em Grades (PCCVG), não definido na literatura. Dele, faz-se uma redução polinomial para o PDGG. Seguem-se definições para os problemas intermediários utilizados nas demonstrações.

Definição 2 (PCAVGPGM3)**.** *Dado um grafo plano $G = (V, E)$, no qual todos os vértices tem grau menor ou igual a 3, e um número inteiro positivo K, existe um conjunto $R \subseteq V$ tal que todas as arestas em E têm pelo menos uma extremidade em R e $|R| \leq K$?*
Definição 3 (PCCVG)**.** *Dada uma grade $G = (V, E)$, uma coleção S de caminhos simples em G e um número inteiro positivo K, existe um conjunto $R \subseteq V$ tal que todo caminho $P \in S$ possua ao menos um vértice de R e $|R| \leq K$?*

Lema 1. *O PCCVG é NP-Completo.*
Demonstração. Dado um grafo plano $G = (V, E)$ com grau máximo menor ou igual a 3 e com $K \in \mathbb{Z}_+^*$, busca-se construir uma grade G', uma coleção de caminhos S e um número $K' \in \mathbb{Z}_+^*$. Deve haver uma cobertura R de arestas por vértices em G com $|R| \leq K$ se, e somente se, houver uma cobertura R' dos caminhos em S por vértices, no grafo $G' = (V', E')$, com $|R'| \leq K'$. Seja $V = \{1, ..., n\}$ e $|E| = m$.

Em uma Incorporação em Livro de um grafo $G = (V, E)$, cada vértice é posto linearmente na espinha do livro seguindo a ordem dada por $F_V : V \to \mathbb{Z}_+^*$ e cada aresta é atribuída a uma de suas páginas por $F_E : E \to \mathbb{Z}_+^*$. Cada página é um semiplano que parte da espinha do livro. As arestas atribuídas a uma página estão nela contidas e não se tocam. Heath (1985) mostrou como incorporar qualquer grafo plano com grau máximo menor ou igual a 3 em um livro de 2 páginas polinomialmente. Na primeira etapa da nossa transformação, incorpora-se G em um livro de 2 páginas, como exemplifica a Figura 1.

Seja $G'{=}(V', E')$ uma grade com $V'{=}\{1, ..., 3n\}{\times}\{1, ..., 2n - 1\}$ e $E'{=}\{((a, b), (c, d)), \forall (a, b), (c, d){\in}V' : (|a{-}c|{=}1 \wedge b{=}d) \vee (a{=}c \wedge |b{-}d|{=}1)\}$. Seja $H = \{(3(F_V(v)){-}1, n), \forall v{\in}V(G)\}{\subset}V'$. Seja $S{=}\{S_1, ..., S_m\}$ uma coleção com m caminhos, cada um deles subgrafo de G', correspondente a uma das arestas em $E(G)$ e com vértices extremos contidos em H. Cada caminho de $(a, b){\in}H$ para $(c, d){\in}H$ é construído segundo as regras seguintes. Regra (1): Cada caminho tem uma componente vertical. Se a aresta $e{\in}E(G)$ correspondente for tal que $F_E(e)$ é a página esquerda, essa componente passa por vértices na coluna $n{-}\frac{|b-d|}{3}$. Caso contrário, na coluna $n{+}\frac{|b-d|}{3}$. Regra (2): Se uma página de $v{\in}V(G)$ contiver 1 aresta, o caminho correspondente em S tem componente horizontal iniciada no vértice $(3(f_V(v)){-}1, n)$ que toca a componente vertical feita na regra (1). Regra (3): Se uma página de $v{\in}V(G)$ contiver 2 ou 3 arestas, cada caminho correspondente deixa o vértice $(3(f_V(v)){-}1, n)$ em uma direção diferente (norte, sul ou horizontal) com base na regra (1), evitando cruzamento. Após passar por um vértice, o caminho prossegue horizontalmente até encontrar a componente feita na regra (1).

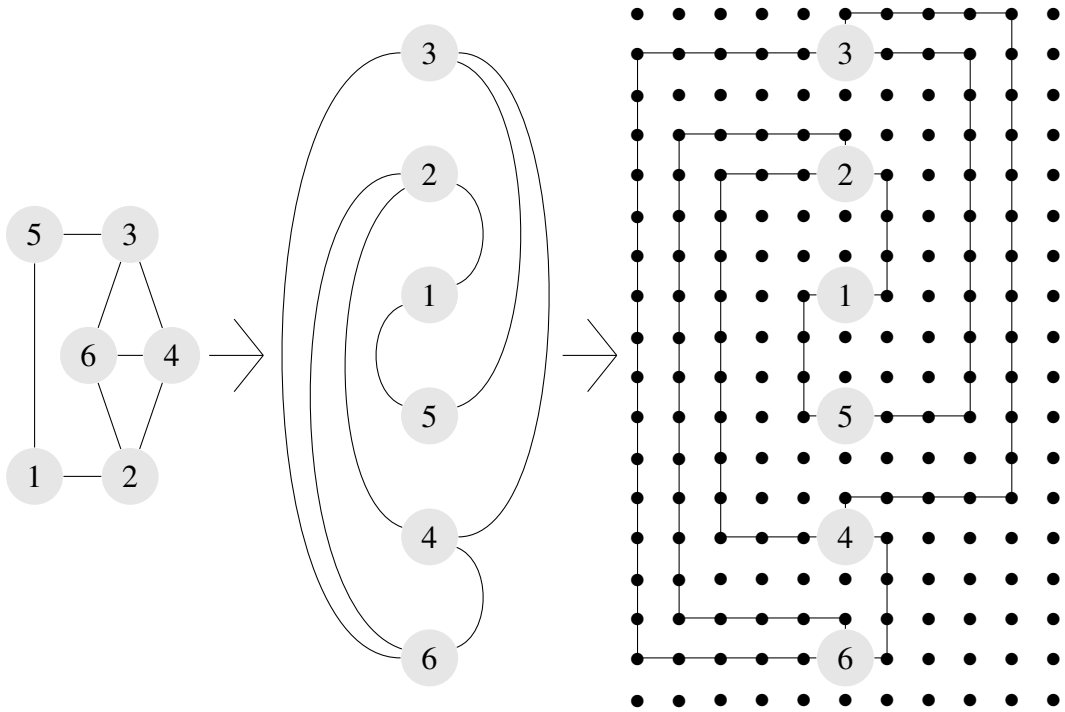


Figura 1. Incorporação de um grafo planar com grau máximo menor ou igual a 3 em um livro de 2 páginas e subsequente transformação em caminhos em grade.

Será provado que $\forall S_1, S_2 \in S, S_1 \cap S_2 \subseteq H$. Suponha-se que existam $S_1 = (u_1, ..., u_2)$ e $S_2 = (w_1, ..., w_2)$ com $S_1 \cap S_2 \not\subseteq H$ e, sem perda de generalidade, seja $F_V(u_1) < F_V(u_2)$, $F_V(w_1) < F_V(w_2)$ e $F_V(u_1) \leq F_V(w_1)$. Para simplificar, serão usados os mesmos símbolos (u_1, u_2, w_1 e w_2) para se referir aos vértices extremos de S_1 e S_2 e aos vértices correspondentes em $V(G)$. Seja o caso em que S_1 e S_2 tenham uma extremidade coincidente. Pela regra (3), os subcaminhos horizontais dessa extremidade de S_1 e S_2 ocorrem disjuntos nas três linhas em torno da extremidade correspondente, o que é sempre possível devido ao grau máximo de G. A regra (1), adicionalmente, garante que não ocorrerá intercepção em qualquer ponto de S_1 e S_2 que não pertença a H. Considere-se agora o caso em que u_1, u_2, w_1 e w_2 são vértices distintos. São disjuntos os caminhos caso $F_V(u_1)$ e $F_V(u_2)$ sejam ambos maiores ou menores do que $F_V(w_1)$ e $F_V(w_2)$. No caso em que $F_V(u_1) < F_V(w_1) < F_V(w_2) < F_V(u_2)$, não ocorre intersecção de caminhos devido à regra (1). A única opção restante é $F_V(u_1) < F_V(w_1) < F_V(u_2) < F_V(w_2)$. Nesse caso, as arestas $(u_1, u_2), (w_1, w_2) \in E(G)$ devem certamente pertencer a páginas diferentes na incorporação em livro do grafo, ou seja, $F_E(u_1, u_2) \neq F_E(w_1, w_2)$. Logo, $S_1 \cap S_2 \subseteq H$.

A Figura 1 exemplifica toda a transformação. É possível encontrar uma cobertura R' de caminhos por vértices com $|R'| \leq K' = K$? No parágrafo seguinte, prova-se por absurdo que a resposta para o PCAVGPGM3 é SIM se, e somente se, também for SIM para o PCCVG em (G', S, K'). O PCCVG está em NP, pois, dado um certificado R' para uma instância SIM (G', S, K'), basta checar se $|R'| \leq K'$ e se $S_i \cap R' \neq \emptyset, \forall S_i \in S$.

Suponha-se que $\exists R \subseteq V(G)$, com $|R| \leq K$, que é cobertura de arestas por vértices em (G, K) e que a resposta para o PCCVG em (G', S, K') é NÃO. Como cada vértice de H é associado biunivocamente a um vértice de $V(G)$, é possível obter um conjunto $R' \subseteq H$ a partir dos elementos de H associados a R. Como cada caminho em S tem as extremidades em H e iguais às extremidades das arestas de $E(G)$, R' é uma cobertura para S em G'

e tem-se $|R'| \leq K'$, o que contradiz a hipótese. Suponha-se, agora, que $\exists R' \subseteq V(G')$, com $|R'| \leq K'$, que é uma cobertura de caminhos por vértices para (G', S, K') e que a resposta para o PCAVGPGM3 em (G, K) é NÃO. Pela construção proposta, os vértices em $V(G') \backslash H$ pertencem a, no máximo, um caminho em S. Se $\exists S_i \in S$ com $S_i \cap (R' \setminus H) \neq \emptyset$, $u \in S_i \cap (R' \backslash H)$ cobre exclusivamente S_i. Logo, pode-se remover u de R' e substituí-lo por um vértice em $H \cap S_i$. Repetindo isso exaustivamente, obtém-se cobertura de caminhos por vértices $R'' \subseteq H$, $|R''| = K'$. Assim, pode-se resolver o PCAVGPGM3 com $R \subseteq V(G)$, $|R| = K'$, obtendo cada vértice de V associado a R'', o que contradiz a hipótese. □

Teorema 1. *O PDG é NP-Completo*
Demonstração. Seja $G=(V, E)$ uma grade com $V=\{v_{ab}, \forall (a, b) \in \{1, ..., m\} \times \{1, ..., n\}\}$ e $E=\{(v_{ab}, v_{cd}), \forall \{v_{ab}, v_{cd}\} \subseteq V : (a-c=1 \wedge b=d) \vee (a=c \wedge b-d=1)\}$. Seja $S=\{S_1, ..., S_p\}$ uma coleção de subgrafos de G tal que $S_i \in S$ é um caminho em G. Com $K \in \mathbb{Z}_+^*$, pode-se obter uma cobertura de caminhos por vértices $R \subseteq V$ com $|R| \leq K$? Dada essa instância do PCCVG, que será referida por $PCCVG(G,S,K)$, constrói-se a instância $PDGG(G', T,F,\tau,\rho,K')$, do PDGG. Em seguida, mostra-se que $PCCVG(G,S,K)$ é uma instância SIM se, e somente se, $PDGG(G',T,F,\tau,\rho,K')$ também é. O PDG pertence a NP, pois um certificado R^* de SIM pode ser checado em tempo polinomial como mostrado na seção 2.

Seja $G'=G$, $T=S$, $K'=K$, $\tau=1$ e $\rho=100\%$. Para cada passeio $T_i \in T$, F é função definida como $F(T_i, x)=\frac{1}{L(T_i)}, \forall x \in \{1, ..., L(T_i)\}$. Após essa transformação, a seguinte igualdade é válida para cada $T_i \in T$: $\sum_{c=1}^{L(T_i)} F(T_i, c)=1=\tau$ Suponha-se que $\exists R \subseteq V$ que é solução para $PCCVG(G, S, K)$ e não para $PDGG(G', T, F, \tau, \rho, K')$. Assim, $\exists T_i \in T$ contendo subpasseio $C \in T_i \backslash R$ que satisfaça $\sum_{c=1}^{L(C)} F(C, c) \geq \tau$. Como $\sum_{c=1}^{L(T_i)} F(T_i, c)=\tau$, pode-se deduzir que $T_i \cap R=\emptyset \rightarrow S_i \cap R=\emptyset$. Logo, R não resolve $PCCVG(G,S,K)$ já S_i não está coberto, o que é uma contradição. Seguindo, suponha-se que $\exists R \subseteq V(G')$ que é solução para $PDGG(G', T, F, \tau, \rho, K')$ mas não para $PCCVG(G, S, K)$. Assim, há um caminho $S_i \in S$, $S_i \cap R=\emptyset$. Como $T_i=S_i$, então $T_i \cap R=\emptyset$ e $T_i \backslash R=\{T_i\}$. Logo, $\exists C \in T_i \backslash R$, $\sum_{c=1}^{L(C)} F(C, c) = \tau$, o que contradiz a hipótese de que R resolve o PDGG. □

Corolário 1. *O PDGG é NP-Completo.*
Corolário 2. *O PDGG com $\rho = 1$ é NP-Completo.*

Referências

FARAJ, M. F.; SARUBBI, J. a. F. M.; SILVA, C. M. da; MARTINS, F. V. C. A hybrid genetic algorithm for deploying RSUs in VANETs based on inter-contact time. In: **Proceedings of the Genetic and Evolutionary Computation Conference Companion**. New York, NY, USA: ACM, 2017. (GECCO '17), p. 193–194. ISBN 978-1-4503-4939-0. Disponível em: ⟨http://doi.acm.org/10.1145/3067695.3076032⟩.

GAREY, M. R.; JOHNSON, D. S. The rectilinear steiner tree problem is NP-complete. **SIAM Journal on Applied Mathematics**, SIAM, v. 32, n. 4, p. 826–834, 1977.

HEATH, L. S. **Algorithms for embedding graphs in books**. Tese (Doutorado) — University of North Carolina at Chapel Hill, 1985.

SILVA, C. M.; GUIDONI, D. L.; SOUZA, F. S.; PITANGUI, C. G.; SARUBBI, J. F.; PITSILLIDES, A. Gamma deployment: Designing the communication infrastructure in vehicular networks assuring guarantees on the V2I inter-contact time. In: IEEE. **Mobile Ad Hoc and Sensor Systems (MASS), 2016 IEEE 13th International Conference on**. [S.l.], 2016. p. 263–271.

Dominação Vetorial na Família dos Grafos *Split*-Indiferença

Rodrigo Lamblet Mafort[*1], **Fábio Protti**[*1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói – RJ – Brasil

{rmafort,fabio}@ic.uff.br

Abstract. *This work presents a polynomial time algorithm capable of solving the Vector Domination Problem for Split-Indifference graphs. The proposed method is based on two characteristics of this class of graphs: the restriction about the number of simplicial vertices and the division in at most three maximal cliques.*

Resumo. *Este trabalho apresenta um algoritmo polinomial capaz de solucionar o Problema da Dominação Vetorial para grafos Split-Indiferença. O método proposto decorre de duas características inerentes a esta classe de grafos: a limitação do número de vértices simpliciais e a divisão em no máximo três cliques maximais.*

1. Introdução

O Problema da Dominação Vetorial (do inglês *Vector Domination Problem*) busca, dados um grafo $G = (V, E)$, um vetor de inteiros $R = (R[v] \in \{0, 1, \ldots, d(v)\} : v \in V\}$ determinar um conjunto $S \subseteq V$ mínimo tal que todo vértice $v \in V$ ou está contido em S ou possui $R[v]$ vizinhos neste conjunto. Um conjunto S capaz de dominar todos os vértices de G é denominado um conjunto R-dominante.

Existem outras variações deste problema, formadas pela aplicação de restrições adicionais. A variação mais conhecida é o Conjunto Dominante, onde todos os vértices possuem requisitos unitários ($R[v] = 1 \ \forall v \in V$).

A demonstração da complexidade computacional do problema da Dominação Vetorial decorre justamente desta variação. Considerando que o problema do Conjunto Dominante é NP-Completo para grafos em geral [Garey and Johnson 1990] e constitui uma restrição ao problema da Dominação Vetorial conclui-se que o Dominação Vetorial também é NP-Completo para tais grafos.

O estudo da complexidade computacional do problema da dominação vetorial na classe dos grafos *split*-indiferença foi motivado pela intratabilidade deste mesmo problema quando restrito aos grafos *split*, conforme demonstrado por [Bertossi 1984]. Outro ponto que instigou esta pesquisa foi a indeterminação da complexidade computacional deste problema no escopo dos grafos de intervalo próprio. Este trabalho é parte de uma análise mais aprofundada que busca identificar os limiares da intratabilidade do problema da dominação vetorial em algumas classes de grafos.

[*]Os autores agradecem a CAPES e a FAPERJ o apoio recebido para execução deste trabalho.

2. Grafos *Split*-Indiferença

Os grafos Split-Indiferença são constituídos pela interseção entre a classe dos Grafos *Split* e os Grafos de Intervalo Próprio, isto é, atendem em simultâneo as restrições inerentes a ambas as classes. A primeira definição formal desta classe foi apresentada por [Ortiz et al. 1998] no seguinte teorema:

Teorema 1. *[Ortiz et al. 1998] Seja G um grafo conexo, $G = (V, E)$ é um grafo split-indiferença se e somente se*

1. *G é um grafo completo, ou*
2. *G possui duas cliques maximais C_1 e C_2, tais que $|C_1 \setminus C_2| = 1$ ou*
3. *G possui três cliques maximais C_1, C_2 e C_3, tais que*
   - *$|C_1 \setminus C_2| = |C_3 \setminus C_2| = 1$*
   - *$C_1 \cap C_3 = \emptyset$ ou $C_1 \cup C_3 = V$*

O Teorema 1 prova a existência de quatro casos possíveis de grafos split-indiferença. Sua definição é baseada no número de cliques e em suas interseções.

3. Algoritmo proposto

O algoritmo desenvolvido para o problema da dominação vetorial em grafos split-indiferença é baseado no Teorema 1, uma vez que para cada caso apresentado no teorema, um método distinto é aplicado. Sejam o grafo $G = (V, E)$ e o vetor de requisitos R as entradas do algoritmo e seja S o conjunto R-dominante em construção.

O primeiro caso do Teorema 1 consiste de apenas uma única clique e possui solução trivial: inicialmente, os vértices são ordenados de forma decrescente por seus requisitos. Em seguida, essa ordenação é percorrida e a cada iteração um vértice $x \in V$ é analisado. Se $R[x] > |S|$, o algoritmo adicionará x ao conjunto S. Do contrário, x já foi dominado e o algoritmo pode ser encerrado (os demais vértices estão dominados por S).

No segundo caso, o grafo é constituído de duas cliques maximais, sendo que uma delas possui exatamente um vértice simplicial v. Considerando que v pode ou não estar contido na solução, os dois casos são analisados. O primeiro teste consiste de incluir v em S, atualizar os requisitos dos vizinhos de v para sinalizar que já possuem um vizinho em S e resolver a outra clique, que, por definição, contém todos os vértices da primeira, exceto v. Finalmente, o conjunto obtido para esta clique é incorporado à S. O segundo teste considera que v não será incluído na solução, implicando que $R[v]$ vizinhos de v devem estar contidos em S. Os requisitos dos vizinhos dos vértices adicionados devem ser debitados em $|S|$ unidades para considerar que já possuem $|S|$ vizinhos em S e, em seguida, a clique induzida por $(V \setminus (S \cup \{v\}))$ é resolvida pelo Caso 1 e conjunto obtido é incorporado à S. A solução para este caso consiste do conjunto de menor cardinalidade dentre os dois obtidos.

Já no terceiro caso descrito existem três cliques C_1, C_2 e C_3 tais que $|C_1 \setminus C_2| = |C_3 \setminus C_2| = 1$ e $C_1 \cap C_3 = \emptyset$. Tendo em vista que C_1 e C_3 não possuem vértices em comum, é possível utilizar o mesmo procedimento aplicado ao caso anterior, considerando não apenas um vértice simplicial, mas dois (v e w). Desta forma, existem quatro casos que devem ser analisados isoladamente: (a) $v \notin S$ e $w \notin S$; (b) $v \in S$ e $w \notin S$; (c) $v \notin S$ e $w \in S$; (d) $v \in S$ e $w \in S$. A solução para este caso também consiste do menor conjunto dentre os obtidos.

O quarto e último caso apresentado pelo Teorema 1 também é constituído por três cliques C_1, C_2 e C_3, tais que $|C_1 \setminus C_2| = |C_3 \setminus C_2| = 1$. Contudo, difere do caso anterior pela existência de vértices comuns as cliques C_1 e C_3, o que inviabiliza o estudo isolado do caso onde v e w não estão presentes em S, uma vez que ao adicionar vizinhos de v neste conjunto, o requisito de w também é impactado. Para este caso uma nova abordagem foi aplicada. Os demais casos podem ser resolvidos através do método apresentado para o terceiro caso do teorema, pois v e w não são adjacentes.

Inicialmente assume-se que v e w não participarão do conjunto S, e, portanto, devem ser dominados por seus vizinhos. Na abordagem desenvolvida para este caso, são construídas soluções S_1 e S_3 para as cliques C_1 e C_3. Considerando que $C_1 \cup C_3 = V(G)$, sabe-se que $S = S_1 \cup S_3$ é capaz de dominar G, contudo, não se pode afirmar ainda que S é mínimo. Desta forma, o algoritmo inicia uma fase que busca reduzir S através de sucessivas remoções e trocas.

Na fase de remoções, os vértices contidos em S são estudados, em ordem crescente de requisitos, para avaliar se sua exclusão transformaria um vértice já dominado em não dominado. Caso essa exclusão seja possível, esse vértice é imediatamente removido de S. Quando nenhum vértice puder ser removido, o algoritmo passará a procurar formas de diminuir S utilizando trocas de vértices.

As trocas são movimentos que removem dois vértices de S e acrescentam apenas um. Para que isso seja possível, três condições devem ser satisfeitas: (a) São necessários dois vértices x_1 e x_3 tal que $x_1 \in ((C_1 \setminus C_3) \cap S)$ e $x_3 \in ((C_3 \setminus C_1) \cap S)$; (b) Todos os vértices de $(C_2 \setminus (S \setminus \{x_1, x_3\}))$ devem possuir requisitos estritamente menores que o tamanho de S; (c) Existência de um vértice z tal que z não esteja contido em S e seja adjacente à v e à w. Se essas três condições forem satisfeitas, a troca é executada: x_1 e x_3 são removidos e z é adicionado ao conjunto S. Dentre todos os vértice que atendam as restrições, x_1 e x_3 correspondem aos de menores requisitos. Já z corresponde ao vértice de maior requisito que atenda as restrições.

Uma vez que todas as trocas possíveis forem executadas, os vértices de S são analisados novamente em busca de novas remoções. Quando nenhuma remoção for possível, S é um conjunto R-dominante para o grafo G.

3.1. Corretude do algoritmo

Teorema 2. *O algoritmo proposto encontra um Conjunto R-Dominante S para um grafo split-indiferença G e um vetor de requisitos R.*

Demonstração. Caso 1: Considerando que nenhum conjunto de cardinalidade inferior à S é suficiente para converter o último vértice adicionado e que todos os anteriores a esse possuem requisitos superiores, conclui-se que S é mínimo.

Caso 2: Sejam C_1 e C_2 as duas cliques maximais de G e seja v o vértice simplicial de C_1. Ainda, seja S_{C_2} o conjunto de todos os conjuntos R-dominantes para C_2 e seja A o conjunto dos $R[v]$ vizinhos de v de maiores requisitos. Supondo que existe um conjunto $S \in S_{C_2}$ capaz de converter v, sabe-se que um destes conjuntos contém A. Desta forma, seja S um conjunto de S_{C_2} que contenha A. Além disso, sejam G' o subgrafo induzido por $(V \setminus (A \cup \{v\}))$ e $R'[u] = R[u] - |A|\ \forall u \in V(G')$ o vetor de requisitos atualizado. Aplicando o Caso 1 do algoritmo é possível obter um conjunto R'-dominante para G'.

Seja B esse conjunto. Sabe-se que $A \cup B$ é capaz de dominar todos os vértices de G (A domina v e B, os demais), contudo, resta demonstrar que este conjunto é mínimo. Tendo em vista como A e B foram construídos, pode-se concluir que $S \subseteq (A \cup B)$. Desta forma, supondo, por absurdo, que $A \cup B$ não seja mínimo, seja u o vértice que pode ser removido de $A \cup B$. Considerando que A possui os vértices de maiores requisitos dentre os vizinhos de v e que nenhum subconjunto de A domina v, conclui-se que $u \in B$. Como B contém os vértices de maiores requisitos de G' pode-se deduzir que $R'[u] \geq |B|$. Em contrapartida, considerando que u pode ser removido de $A \cup B$, conclui-se que $R[u] < |A| + |B|$, o que é absurdo, pois $R'[u] = R[u] - |A|$. Sendo assim, fica demonstrado que $A \cup B$ é mínimo.

Resta demonstrar o caso em que nenhum conjunto $S \in S_{C_2}$ é capaz de dominar v, isto é, para dominar v é necessário pelo menos mais um vértice. Desta forma, o próprio v será adicionado ao conjunto R-dominante. Seja R' o vetor de requisitos atualizado para considerar que v já contribui na dominação de seus vizinhos ($R'[u] = R[u] - 1 \ \forall u \in adj(v)$) e seja B um conjunto R'-dominante para C_2. Considerando que B é mínimo, domina C_2, mas não domina v ($B \subseteq S$), conclui-se que $B \cup \{v\}$ é um conjunto R-dominante para G.

Caso 3: Sejam C_1, C_2 e C_3 as três cliques maximais de G, tais que $C_1 \cap C_3 = \emptyset$ e sejam v e w os vértices simpliciais de C_1 e C_3 respectivamente. Considerando que $adj(v) \cap adj(w) = \emptyset$, pode-se aplicar a mesma lógica do caso anterior para os quatro casos possíveis.

Caso 4: Sejam C_1, C_2 e C_3 as três cliques maximais de G, tais que $C_1 \cup C_3 = V$ e sejam v e w os vértices simpliciais de C_1 e C_3 respectivamente. Nos casos em que v ou w estão contidos no conjunto R-dominante, a demonstração segue o caso anterior. Entretanto, o caso em que v e w não estão contidos no conjunto R-dominante deve ser demonstrado. Considerando que nos demais casos o algoritmo encontra um conjunto R-dominante (demonstrado no Caso 3), supõe-se que não existe um conjunto R-dominante que contenha v ou w, pois do contrário este seria localizado pelos casos anteriores. Contudo, tendo em vista as limitações relativas ao tamanho deste resumo, esta parte da prova foi omitida.

Tendo em vista que em todos os casos o algoritmo localizou um Conjunto R-Dominante para o grafo G, conclui-se que o método apresentado está correto. □

A complexidade do algoritmo proposto é $O(n^2)$, pois no pior caso, ao selecionar um vértice, todo o vetor de requisitos deve ser percorrido para atualização.

Referências

Bertossi, A. A. (1984). Dominating sets for split and bipartite graphs. *Information Processing Letters*, 19(1):37–40.

Booth, K. S. (1980). Dominating Sets in Chordal Graphs. Technical report, University of Waterloo.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co.

Ortiz, C. Z., Maculan, N., and Szwarcfiter, J. L. (1998). Characterizing and edge-colouring split-indifference graphs. *Discrete Applied Mathematics*, 82(1-3):209–217.

Implementações Eficientes da Heurística Min-Min para o HCSP em GPU

Rafael F. Schmid[1], **Edson N. Cáceres**[1]

[1]Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
Campo Grande, MS – Brasil

rafaelfschmid@gmail.com, edson@facom.ufms.br

Abstract. *Min-min heuristic is one of the most indicated algorithm to solve HCSP, when we are looking for performance. A previous result presented an efficient implementation to this problem with assymptotic complexity $O(t \log(t)m)$. In this work, we implement this problem with linear complexity. Our implementation is competitive with the previous results.*

1. Introdução

A utilização de grades computacionais e computação em nuvem têm obtido bons resultados na solução de problemas complexos. De uma forma geral, grades computacionais e computação em nuvem são compostas de vários comutadores heterogêneos que são capazes de trabalhar de forma cooperativa.

Considerando que temos várias tarefas a serem executadas num conjunto heterogêneo de computadores, onde o tempo de execução de uma tarefa pode variar de acordo com o computador a ser utilizado, temos que resolver o problema de como escalonar as tarefas para serem resolvidas nos computadores no menor tempo possível. Nesses casos, temos uma competição entre as tarefas para utilizar o computador que seja capaz de resolvê-las no menor tempo [Nesmachnow and Canabé 2011]. Esse problema é denominado o Problema de Escalonamento em Ambientes de Computação Heterogênea (HCSP - Heterogeneous Computing Scheduling Problem).

Como trata-se de um problema NP-completo, muitas heurísticas tem sido aplicadas para resolvê-lo, como min-min, max-min, Algoritmos Genéticos (GA), *Genetic Simulated Annealing* (GSA), *Sufferage*, etc [Braun et al. 2001]. Em função de suas características, a heurística min-min se mostrou como uma das mais rápidas para esse problema e apresenta soluções quase tão boas quanto GA, que é uma das heurísticas mais demoradas [Wu et al. 2000].

Nesse trabalho, baseado na implementação híbrida de [Pedemonte et al. 2016] e no trabalho de [Schmid et al. 2016], desenvolvemos uma implementação usando a heurística min-min para o HCSP. Nosso resultado tem complexidade linear e a sua implementação se mostrou competitiva em relação aos resultados anteriores.

2. O problema do HCSP e a Heurística Min-Min

No modelo apresentado nesse trabalho, todas as tarefas podem ser executadas independentemente, sem se preocupar com a ordem de execução, e quando uma tarefa inicia ela é executada até o final. As aplicações a serem executadas são compostas por uma

coleção de tarefas indivisíveis, que não tem dependência entre elas, comumente chamadas de *metatask*. Problemas de escalonamento consistem em tentar minimizar o tempo gasto para executar todas essas tarefas. A formalização abaixo apresenta o modelo matemático para o HCSP com o intuito de minimizar o *makespan* (tempo gasto a partir do momento em que a primeira tarefa começa até o momento em que a última é completada [Nesmachnow and Canabé 2011]):

- Dado um sistema de computação heterogênea composto de um conjunto de M máquinas $P = m_1, m_2, ..., m_M$ e uma coleção de N tarefas $T = t_1, t_2, .., t_N$ a serem executadas.
- Considere uma função de tempo de execução $ET : P \times T \rightarrow R^+$, onde $ET(t_i, m_j)$ é o tempo necessário para executar a tarefa t_i na máquina m_j.
- O objetivo do HCSP é encontrar uma atribuição de tarefas às máquinas (uma função $f : T^N \rightarrow P^M$) que minimize o *makespan*, conforme definido na Equação 1.

$$\max_{m_j \in P} \sum_{\substack{t_i \in T \\ f(t_i)=m_j}} ET(t_i, m_j) \tag{1}$$

A heurística min-min começa com o conjunto U de todas as tarefas não mapeadas. Então, o conjunto de máquinas de menor tempo de conclusão M para cada tarefa em U é encontrado. A tarefa cujo tempo de conclusão seja o menor, é escolhida e atribuída à máquina correspondente. A nova tarefa mapeada é removida do conjunto U, e o processo se repete até que todas as tarefas sejam mapeadas [Braun et al. 2001].

3. Trabalhos Relacionados

Nesmachnow e Canabé (2011) apresentaram uma implementação em GPU para a heurística min-min e obtiveram *speedups* de aproximadamente seis vezes em relação a uma implementação sequencial. Ezzatti et al. (2013) propuseram um algoritmo min-min, que chamamos de **min-min sort**, que efetua a ordenação segmentada sobre os tempos que cada máquina leva para executar cada uma das tarefas, e depois verifica, nessa ordem, as tarefas de cada máquina, selecionando a menor para aquela iteração. Se utilizado um algoritmo de ordenação linear, esse algoritmo alcança complexidade assintótica também linear: $O(tm)$, onde t é o número de tarefas e m o número de máquinas, mas quando da implementação os autores utilizaram um algoritmo $O(t \log(t)m)$. Posteriormente, Pedemonte et al. (2016) apresentaram uma implementação híbrida GPU-CPU do *min-min sort* com resultados melhores que os anteriores.

4. Solução Proposta e Resultados Experimentais

Os experimentos deste artigo foram realizados em uma máquina com o Sistema Operacional Ubuntu 16.04, 16 GB de memória RAM, processador Intel Core^{tm} i5-4460 CPU @ 3.20 GHz x 4 e placa gráfica GeForce GTX 1050 Ti com 4 GB de memória, e os cenários de teste utilizados foram obtidos do endereço `https://par-cga-sched.gforge.uni.lu/instances/etc/`.

Baseado no trabalho de [Schmid et al. 2016], apresentamos uma implementação da heurística *min-min sort* utilizando o *radix-sort*. O principal objetivo é utilizar uma

ordenação segmentada que garanta complexidade linear para a heurística min-min. Para efetuar as comparações também foi implementada: a min-min padrão em CPU e GPU; e a *min-min sort* em CPU, GPU e Híbrida (GPU-CPU).

A implementação em GPU da heurística min-min padrão foi testada utilizando duas estruturas diferentes. A primeira utiliza as linhas da matriz para representar as máquinas e as colunas para representar as tarefas. Cada thread CUDA é responsável por processar uma tarefa, ou seja, encontrar a máquina em que a tarefa possui o menor tempo de conclusão. Cada thread armazena seu resultado em uma posição do vetor resultante, onde é realizada uma redução paralela para encontrar a tarefa que deve ser escalonada naquela iteração. Dessa forma, as threads acessam a mesma linha da matriz a cada iteração (Figura 4), favorecendo a coalescência. Ao inverter essa estrutura perdemos a coalescência e os tempos sobem bastante, por isso optamos não abordá-la aqui.

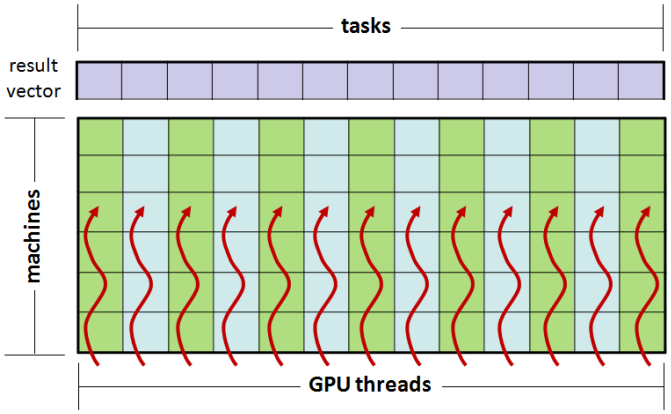


Figura 1. Estratégia de implementação em GPU da heurística min-min [Nesmachnow and Canabé 2011].

A implementação paralela do *min-min sort* de [Ezzatti et al. 2013], efetua a ordenação segmentada utilizando uma biblioteca CUDA, e depois utiliza uma redução paralela para identificar as máquinas que possuem o menor tempo de conclusão na iteração corrente. Para essa estratégia representamos as máquinas em linhas e as tarefas em colunas, estrutura necessária para efetuarmos a ordenação segmentada de forma eficiente.

A nossa implementação do *min-min sort* híbrido CPU-GPU usa a mesma estrutura da versão paralela do algoritmo *min-min sort*. Na primeira etapa, a ordenação das linhas da matriz, é efetuada em GPU. Posteriormente, os dados são copiados de volta para CPU e a mesma implementação da versão sequencial do *min-min sort* é executada. Para a ordenação utilizamos a biblioteca CUB (*radix-sort*).

A Figura 2 apresenta o *speedup* de cada uma das implementações: baseadas no *merge-sort* (Pedemont et al.) e baseadas no *radix-sort* (implementação proposta) quando comparadas com a implementação sequencial do min-min padrão. Com isso, podemos observar que, apesar de estarmos utilizando um algoritmo linear, em termos de desempenho, o comportamento é semelhante ao algoritmo de complexidade $O(t \log(t)m)$ da biblioteca MGPU.

Podemos notar também, que a implementação sequencial do *min-min sort* apresenta melhores resultados para matrizes menores, já que o acesso não é coalescido, devido

a ordenação segmentada do início do algoritmo. O motivo de apresentar melhores resultados é que a ordenação segmentada em GPU possui desempenho superior à ordenação sequencial, prova disso são os desempenhos bem melhores das implementações híbridas.

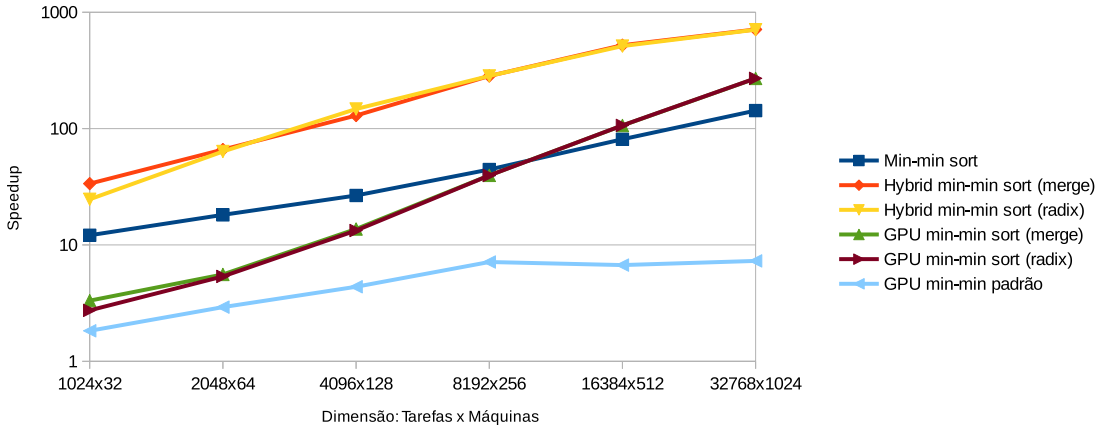


Figura 2. Speedup das implementações do Min-Min.

5. Conclusões

A implementação que apresentamos possui complexidade linear e os resultados obtidos na GPU e GPU-CPU são competitivos.

Referências

Braun, T. D., Siegel, H. J., Beck, N., Bölöni, L. L., Maheswaran, M., Reuther, A. I., Robertson, J. P., Theys, M. D., Yao, B., Hensgen, D., and Freund, R. F. (2001). A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 61(6):810 – 837.

Ezzatti, P., Pedemonte, M., and Martín, Á. (2013). An efficient implementation of the min-min heuristic. *Computers & Operations Research*, 40(11):2670–2676.

Nesmachnow, S. and Canabé, M. (2011). Gpu implementations of scheduling heuristics for heterogeneous computing environments. In *XVII Congreso Argentino de Ciencias de la Computación*.

Pedemonte, M., Ezzatti, P., and Martín, Á. (2016). Accelerating the min-min heuristic. In *Parallel Processing and Applied Mathematics*, pages 101–110. Springer.

Schmid, R., Pisani, F., Borin, E., and Cáceres, E. (2016). An evaluation of segmented sorting strategies on gpus. In *IEEE 18th International Conference on High Performance Computing and Communications (HPCC)*, pages 1123–1130. IEEE.

Wu, M.-Y., Shu, W., and Zhang, H. (2000). Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems. In *Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th*, pages 375–385. IEEE.

Limite Superior para o Problema da Diversidade Máxima

Pablo Luiz B. Soares[1], **Manoel Campêlo**[2]

[1]Universidade Federal do Ceará
Campus de Russas, Bloco 1, CEP 62900-000, Russas - CE, Brasil.

[2]Universidade Federal do Ceará, Departamento de Estatística e Matática Aplicada.
Campus do Pici, Bloco 910, CEP 60440-900, Fortaleza - CE, Brasil.

pablo.soares@ufc.br, mcampelo@lia.ufc.br

Resumo. *Aplicamos a t–linearização ao modelo quadrático* $0-1$ *natural para o problema da diversidade máxima. Comparamos computacionalmente o modelo obtido com duas outras formulações lineares da literatura. Experimentos computacionais mostram que a t–linearização gera em média menos restrições que as formulações, além de conseguir alcançar um limite superior mais apertado em todas as instâncias usadas. Por outro lado, uma das formulações consegue ser ligeiramente mais rápida na obtenção dos limites.*

Abstract. *We apply the t–linearization to the straightforward* $0-1$ *quadratic model for the maximum diversity problem. We computationally compare the obtained model with two other linear formulations from the literature. Computational experiments show that the t–linearization generates, on average, fewer constraints than the formulations, in addition to achieving a tighter upper bound in all used instances. On the other hand, one of the formulations can be slightly faster in obtaining the bounds.*

1. Introdução

Pertencente a uma vasta família de problemas de dispersão e diversidade, o problema da diversidade máxima (*Maximum Diversity Problem* – MDP) tem como objetivo identificar, em um dado conjunto, um subconjunto S de cardinalidade fixa, de tal forma que a soma das distâncias entre os pares de elementos em S seja máxima [Kuo et al. 1993, Zhou et al. 2017]. Trata-se de um problema NP-Difícil, que aparece com diferentes denominações na literatura [Martí et al. 2013].

Matematicamente, MDP pode ser formulado a partir de um grafo não-direcionado $G = (V, E)$, com $n = |V|$ vértices e $m = |E|$ arestas ponderadas, e um inteiro $k < n$, que representa a cardinalidade do subconjunto (de vértices) desejado. Considerando $V = \{v_1, v_2, \ldots, v_n\}$ e $c_{ij} \in \mathbb{R}_+$ o peso da aresta entre v_i e v_j ($c_{ij} = c_{ji}$ se $(v_i, v_j) \in E$, e $c_{ij} = 0$ se $(v_i, v_j) \notin E$), MDP é naturalmente modelado como o seguinte problema quadrático binário [Kuo et al. 1993]:

$$(F_1) \quad \max \left\{ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_i x_j : e^T x = k, x \in \{0,1\}^n \right\},$$

onde $e^T = (1, \ldots, 1)$, e a variável binária $x_i = 1$ se o vértice $v_i \in S$ e $x_i = 0$ se $v_i \notin S$.

A não-linearidade em problemas de programação inteira é normalmente tratada com o uso de técnicas envolvendo uma aproximação linear por partes [Dantzig 1963, Hu 1969] ou a transformação da função não-linear em uma função

polinomial, a seguir convertida em uma função linear de variáveis 0–1 [Balas 1964, Hammer and Rudeanu 1968]. Na maioria das vezes, a não-linearidade em problemas de programação inteira aparece já na forma polinomial, sendo que um número significativo dos casos envolve apenas termos de segunda ordem [Glover 1975].

Em [Glover and Woolsey 1974], os autores mostram como converter um problema de programação quadrática $0-1$ em um problema de programação linear $0-1$. Esse procedimento, conhecido como linearização clássica, consiste em substituir cada produto $x_i x_j$ por uma variável não negativa y_{ij} e adicionar as restrições $y_{ij} \leq x_i$, $y_{ij} \leq x_j$, $y_{ij} \geq x_i + x_j - 1$. Logo após, [Glover 1975] introduziu inequações que servem para lidar com problemas quadráticos inteiros. Usando esses procedimentos, [Kuo et al. 1993] propuseram as seguintes formulações lineares (F_2 e F_3) para o MDP:

$$(F_2)\ \max\left\{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij}y_{ij} : e^T x = k, y_{ij} \leq x_i, y_{ij} \leq x_j, y_{ij} \geq x_i + x_j - 1, x \in \{0,1\}^n\right\}$$

$$(F_3)\ \max\left\{\sum_{i=1}^{n-1} u_i : e^T x = k, u_i \leq \left(\sum_{j=i+1}^{n} c_{ij}\right) x_i, u_i \leq \sum_{j=i+1}^{n} c_{ij}x_j, x \in \{0,1\}^n\right\}.$$

Como os custos são não-negativos, note que, em (F_3), ocorre $u_i = x_i \sum_{j=i+1}^{n} c_{ij}x_j$, para todo $1 \leq i \leq n-1$, em uma solução ótima. Já em (F_2), as restrições $y_{ij} \geq x_i + x_j - 1$, para todo $1 \leq i < j \leq n$, são desnecessárias.

[Rodrigues 2010] propôs uma nova abordagem de linearização, no contexto do Problema Quadrático da Mochila (PQM), que consiste basicamente de duas etapas. Primeiro, substitui-se o termo quadrático da função objetivo por uma variável real t, que é limitada superiormente pela expressão quadrática, com a inclusão de uma restrição adicional. Depois, essa restrição quadrática é substituída por um conjunto exponencial de restrições lineares, que definem os mesmos pontos inteiros.

Na próxima seção, revisitamos a *t*–linearização para termos quadráticos com coficientes não-negativos, apresentando outras demonstrações para resultados de [Rodrigues et al. 2012], além de aplicarmos tal desenvolvimento ao MDP. Experimentos computacionais para MDP, comparando a *t*–linearização e as formulações F_2 e F_3, são apresentados e analisados na Seção 3.

2. *t*–Linearização

Com o auxílio de uma variável adicional t, podemos reescrever F_1 como $(F_1)_t$, dada por:

$$\max\left\{t : e^T x = k, (x,t) \in Z\right\}, \text{ onde } Z = \{(x,t) \in \mathbb{B}^n \times \mathbb{R} : t \leq \textstyle\sum_{i=1}^{n}\sum_{j=1}^{i-1} c_{ji}x_i x_j\}.$$

Seja S_n o conjunto de todas as permutações de $\{1,\ldots,n\}$. O conjunto Z pode ser expresso por restrições lineares, como segue:

Teorema 1. $Z = Z' := \left\{(x,t) \in \mathbb{B}^n \times \mathbb{R} : t \leq \sum_{i=1}^{n}\sum_{j=1}^{i-1} c_{\pi(j)\pi(i)}x_{\pi(i)}\ \forall \pi \in S_n\right\}$.

Demonstração. Primeiro, note que

$$\textstyle\sum_{i=1}^{n}\sum_{j=1}^{i-1} c_{ji}x_i x_j = \sum_{i=1}^{n}\sum_{j=i}^{i-1} c_{\pi(j)\pi(i)}x_{\pi(i)}x_{\pi(j)}\ \forall \pi \in S_n. \quad (1)$$

Se $(x,t) \in Z$, então temos que $(x,t) \in Z'$ devido a (1) e

$$\sum_i \sum_{j<i} c_{\pi(j)\pi(i)} x_{\pi(i)} x_{\pi(j)} \leq \sum_i \sum_{j<i} c_{\pi(j)\pi(i)} x_{\pi(i)} \; \forall \pi \in S_n.$$

Suponha agora $(x,t) \in Z'$ e seja $\pi \in S_n$ que ordena as componentes de x em ordem não-crescente, ou seja, $x_{\pi(i)} \geq x_{\pi(j)}$ se $i \leq j$. Observe que $(x_{\pi(1)}, \ldots, x_{\pi(n)}) = (1, \ldots, 1, 0, \ldots, 0)$. Como $(x,t) \in Z$, temos que

$$t \leq \sum_i \sum_{j<i} c_{\pi(j)\pi(i)} x_{\pi(i)} = \sum_i \sum_{j<i} c_{\pi(j)\pi(i)} x_{\pi(i)} x_{\pi(j)},$$

onde a igualdade decorre da ordenação das entradas de x por π. □

Pelo Teorema (1), podemos substituir Z por Z' na descrição de $(F_1)_t$. Mais ainda, a separação das restrições lineares que definem Z' pode ser feita em tempo polinomial, como mostrado em [Rodrigues 2010]. A extensão desses resultados para coeficientes c_{ij} arbitrários pode ser encontrada em [Soares et al. 2017].

Na verdade, a relaxação linear de Z' coincide com sua envoltória convexa [Rodrigues 2010]. Todavia, a presença da restrição de cardinalidade em $(F_1)_t$ permite fortalecer as restrições que definem Z'.

Teorema 2. *Para $i = 1, \ldots, n$ e $\pi \in S_n$, seja $s_{\pi(i)}$ a soma dos $r := \min\{i-1, k-1\}$ maiores valores do conjunto $\{c_{\pi(j)\pi(i)} : j = 1, \ldots, i-1\}$. Então as desigualdades $t \leq \sum_{i=1}^{n} s_{\pi(i)} x_{\pi(i)}$, para todo $\pi \in S_n$, é válida para $(F_1)_t$.*

Demonstração. Seja $(x,t) \in Z$, com $e^T x = k$. Então, se $x_{\pi(i)} = 1$, temos $\sum_{j=1}^{i-1} x_{\pi(j)} \leq r$. Logo, $t \leq \sum_{i=1:x_{\pi(i)}=1}^{n} (\sum_{j=1}^{i-1} c_{\pi(j)\pi(i)} x_{\pi(j)}) x_{\pi(i)} \leq \sum_{i=1}^{n} s_{\pi(i)} x_{\pi(i)}$. □

3. Experimentos Computacionais

Os experimentos computacionais foram conduzidos para testar a qualidade do limite superior LS, quando relaxamos a integralidade nas formulações F_2, F_3 e $(F_1)_t$ (esta descrita pelas desigualdades do Teorema 2). Para isso utilizamos um conjunto de 50 instâncias denominadas *Glover2*, disponíveis em www.optsicom.es/mdp. Esse conjunto consiste de 5 instâncias para cada par (n,k), onde $n \in \{25, 50, 100, 125, 150\}$ e $k = 0.1n, 0.3n$. Para cada uma das formulações relaxadas, calculamos o GAP%$= \frac{LS-ML}{ML} \times 100$, onde ML é a melhor solução, disponível em www.optsicom.es/binaryss/, encontrada até o momento, o tempo de cpu (CPU) e a quantidade de restrições geradas na obtenção de LS (NRG). Para implementar as formulações lineares relaxadas, usamos o software comercial IBM/ILOG CPLEX 12.7 integrado com Ambiente de Desenvolvimento Integrado (IDE - Integrated Development Environment) *Code::Blocks* através da linguagem C++. Os experimentos foram realizados em um processador Intel® Core™ i$5-4570$ com 3.20 GHz, 16 GB RAM e sistema operacional Ubuntu 16.04 LTS.

A Tabela 1 sumariza os resultados obtidos por cada formulação, onde cada linha a partir da 3 representa a média das 5 instâncias para cada par (n,k). Destacamos em negrito os melhores resultados. Podemos observar que a relaxação de F_3 mostrou-se competitiva quanto ao tempo, sendo cerca de duas vezes mais rápida que a relaxação da *t*–linearização. Por outro lado, os resultados mostram que a *t*–linearização obteve em média limites cerca de cinco vezes mais apertados com relação a F_2 e F_3, em todas as combinações de (n,k), além de utilizar em média uma quantidade menor de restrições.

Tabela 1. Média GAP, CPU e NRG obtido pelas relaxações F_2, F_3 e $(F_1)_t$

Instâncias	n	k	F_2			F_3			$(F_1)_t$		
			GAP(%)	CPU(s)	NRG	GAP(%)	CPU(s)	NRG	GAP(%)	CPU(s)	NRG
Glover2	25	2	1513.84	0.01	600	1519.32	**0.00**	48	**223.96**	0.007	**40.2**
	25	7	228.161	0.012	600	228.77	**0.002**	**48**	**80.66**	0.013	49
	50	5	814.64	0.210	2450	816.92	**0.018**	98	**119.36**	0.035	**62.8**
	50	15	187.40	0.210	2450	188.97	**0.018**	**98**	**73.40**	0.073	98.8
	100	10	747	1.074	9900	75082	**0.22**	198	**103.97**	0.0414	**132.4**
	100	30	184.90	1.294	9900	186.44	**0.22**	**198**	**66.18**	1.056	210.2
	125	12	733.67	1.998	15500	741.8	**0.51**	248	**106.44**	1.057	**184**
	125	37	198.47	2.428	15500	199.37	**0.514**	**248**	**67.69**	3.488	285.8
	150	15	735.13	3.46	22350	738.68	**1.02**	298	**99.56**	2.065	**211.8**
	150	45	196.48	4.502	22350	197.30	**1.022**	**298**	**67.34**	5.257	305.8
Média Geral			461.64	1.266	10160	464.03	**0.295**	178	**84.05**	1.112	**131.73**

Referências

Balas, E. (1964). Extension de l'algorithme additif à la programmation en nombres entiers et à la programmation non linéaire. *Comptes Rendus Hebdomadaires des Seances de L'Academie des Sciences*, 258(21):5136–5139.

Dantzig, G. B. (1963). *Linear Programming and Extensions*. Princeton University Press.

Glover, F. (1975). Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22(4):455–460.

Glover, F. and Woolsey, E. (1974). Converting the 0-1 polynomial programming problem to a 0-1 linear program. *Operations research*, 22(1):180–182.

Hammer, P. L. and Rudeanu, S. (1968). *Boolean Methods in Operations Research*. Springer-Verlag.

Hu, T. C. (1969). *Integer programming and network flows*.

Kuo, C.-C., Glover, F., and Dhir, K. S. (1993). Analyzing and modeling the maximum diversity problem by zero-one programming. *Decision Sciences*, 24(6):1171–1185.

Martí, R., Gallego, M., Duarte, A., and Pardo, E. G. (2013). Heuristics and metaheuristics for the maximum diversity problem. *Journal of Heuristics*, 19(4):591–615.

Rodrigues, C., Quadri, D., Michelon, P., and Gueye, S. (2012). 0-1 quadratic knapsack problems: an exact approach based on a t-linearization. *SIAM Journal on Optimization*, 22(4):1449–1468.

Rodrigues, C. D. (2010). *Abordagens híbridas na solução de problemas de programação inteira da teoria e prática*. PhD thesis, Universidade Federal do Ceará e Université d'Avignon.

Soares, P., Campêlo, M., Rodrigues, C. D., and Michelon, P. (2017). t-linearização de funções quadráticas de variáveis binárias. *Anais do XLIX SBPO*.

Zhou, Y., Hao, J.-K., and Duval, B. (2017). Opposition-based memetic search for the maximum diversity problem. *IEEE Transactions on Evolutionary Computation*, 21(5):731–745.

Método de pontos interiores para estimar os parâmetros de um modelo probabilístico usando o *corpus Thyco Brahe*

Esther S. Mamián Lopez[1], **Aurelio Ribeiro Leite Oliveira**[1]

[1]Inst. de Matemática, Estatística e Computação Científica – Universidade Estadual de Campinas (UNICAMP)
Rua Sérgio Buarque de Holanda, 651– 13083-859– Campinas – SP – Brazil

esmamian@gmail.com, aurelio@ime.unicamp.br

Abstract. *Statistical methods research for natural language processing and other important applications, have been presenting fast growth in the recent years. In this work, we propose a primal-dual interior point method for trainning stochastic context free grammar. For that purpose, we use a Portuguese based corpus Tycho Brahe [IEL-UNICAMP and IME-USP].*

Resumo. *Nos anos recentes têm acontecido um importante crescimento no interesse sobre os métodos estatísticos para processamento de linguagens e as diferentes aplicações que se derivam. Propomos um método de pontos interiores barreira logarítmica primal-dual para abordar o problema de atribuir valores ótimos de probabilidade às regras de uma gramática probabilística livre de contexto (GPLC) baseados no corpus da linguagem portuguesa Tycho Brahe [IEL-UNICAMP and IME-USP].*

1. Introdução

Dentre os modelos estatísticos para modelar as linguagens naturais estão as gramáticas probabilísticas livres de contexto, que na sua forma mais simples podem ser decompostas numa parte estrutural e numa parte estocástica. Abordamos o problema de modelar uma linguagem natural através de uma GPLC treinando a mesma, ou seja, encontrar as probabilidades ótimas associadas às regras da gramática [Manning and Schutze. 2003]. Este processo é realizado em base a um *corpus* que contém sentenças da linguagem que desejamos modelar. Para o processo do treino precisamos de uma função critério da amostra[1] e um marco para otimizá-la: usamos a função de máxima verossimilhança da amostra e o método de pontos interiores barreira logarítmica primal-dual, respectivamente.

Assim, o problema de treinamento de uma gramática é resumido como o problema de otimização descrito a seguir:

$$\begin{array}{ll} \text{maximizar} & f(x) \\ \text{sujeito a} & \sum\limits_{x_i \in \Psi_A} x_i = 1, \quad \forall\ \Psi_A : A \in \Sigma \\ & 0 \leq x_i \leq 1, \qquad i = 1, ..., |P| \end{array} \tag{1}$$

onde $f(x) = Pr(\Omega|G_p)$ é a função de verossimilhança que, depende da amostra Ω e da GPLC G_p. Esta função é um polinômio nas variáveis x_i, $i = 1...|P|$, sendo que

[1]Usamos a palavra *corpus* e amostra indistintamente para nos referir ao conjunto de sentenças de uma linguagem natural que estamos desejando modelar.

x_i correspondem aos valores de probabilidade associados às regras de G_p. Denotamos por Ψ_A como os valores de probabilidade associados às regras que possuem o mesmo antecedente. Denotamos por Σ o conjunto finito de símbolos não terminais, e P como o conjunto finito das regras da G_p.

2. Metodologia

Uma vez definido o problema (1), implementamos o método barreira logarítmica primal-dual baseados na formulação para problemas não lineares, descrita em [El-Bakry et al. 1996]. Este método foi implementado na linguagem C++, versão 4.8.4. O *corpus* usado é o *Tycho Brahe*, para o português do Brasil. Para analisar a viabilidade da proposta extraímos do *corpus* algumas GPLCs de diferentes tamanhos e as treinamos usando o método barreira logarítmica primal-dual. Seguem os detalhes das gramáticas extraídas para o treinamento e algumas discussões associadas às implementações.

2.1. Tamanho dos problemas

Trabalhamos com seis sub-problemas detalhados na Tabelas 1 e 2.

	Nro símbolos não terminais m	Nro símbolos terminais n	Nro de regras $\|P\|$	Símbolo inicial
Sub-Gramática 1	5	2.500	12.580	S
Sub-Gramática 2	7	2.500	17.752	S
Sub-Gramática 3	9	2.500	23.076	S

Tabela 1. Características da gramática para cada sub-problema do 1 até 3.

	Nro símbolos não terminais m	Nro símbolos terminais n	Nro de regras $\|P\|$	Símbolo inicial
Sub-Gramática 4	3	5.500	16.512	S
Sub-Gramática 5	3	7.500	22.512	S
Sub-Gramática 6	3	7.500	22.512	S

Tabela 2. Características da gramática para cada sub-problema do 4 até 6.

2.2. Cálculo da função objetivo $f(x)$

Note que, o cálculo da função objetivo $f(x)$ deve ser feita a cada iteração do método de pontos interiores barreira logarítmica primal-dual. E, dado que a função objetivo depende das probabilidades das sentenças do *corpus*, observe que não é eficiente calcular a probabilidade de uma sentença como a soma das probabilidades de todas suas possíveis árvores sintáticas. Para isso, implementamos um método baseado no método Cocke-Younger-Kassami, desenvolvido em [López 2018] para calcular o valor da função objetivo sem comprometer as capacidades físicas das máquinas, e utilizamos a biblioteca Open Multi-processing (OpenMP 2.5) para melhorar os tempos computacionais deste método.

2.3. Cálculo do gradiente e da Hessiana de $f(x)$

Os valores do gradiente e da Hessiana devem ser calculados a cada iteração. Usamos o método das diferenças finitas para calculá-lo, pois a expressão das derivadas é muito cara.

2.4. Resolução do sistema linear

Além do custo computacional para calcular o valor da função objetivo e suas derivadas a cada iteração, o trabalho computacional dos métodos de pontos interiores também é dominado pela resolução de um sistema linear [Gondzio 2012]. Portanto uma forma eficiente de solução desses sistemas lineares é indispensável para resolver problemas de grande porte.

A matriz de coeficientes do sistema [Gondzio 2012] é uma matriz indefinida, mal condicionada, esparsa, portanto usamos um método iterativo precondicionado: o método do gradiente bi-conjugado estabilizado para matrizes esparsas, com precondicionador baseado nas entradas da diagonal [Saad 2003].

2.5. Reescalonamento do problema

Uma grande dificuldade é quando, no método de pontos interiores barreira logarítmica primal-dual, os diferentes valores das variáveis com as quais estamos trabalhando atingem valores pequenos causando *underflow*, levando a problemas estabilidade e arredondamento [Trefethen and Bau III 1997, Ruggiero and da Rocha Lopes. 1997]. Dada a ordem do polinômio do problema, os principais valores que devemos analisar e acompanhar são a função de verossimilhança, o valor do gradiente e o valor da Hessiana. Portanto, nossa proposta é manter os valores dessas grandezas e os cálculos que os envolvem controlados, tanto para diminuir os erros de arredondamento, como para evitar problemas de *underflow*. Assim, quando for preciso, multiplicamos a função objetivo por uma constante $C \gg 0$. Basicamente estamos re-escalando os valores numéricos para resolver estes inconvenientes.

3. Resultados e discussões

Nas tabelas a seguir apresentamos os resultados obtidos para as seis sub-gramáticas detalhadas nas Tabelas 1 e 2. Apresentamos o tamanho da amostra usada, o número de iterações e o tempo que utilizou o método até convergir, assim como o valor da constante C. Para as primeiras três sub-gramáticas (ver Tabela 1) usamos uma amostra de tamanho 67, com comprimento das sentenças entre quatro e cinco. Para o segundo grupo de sub-gramáticas (ver Tabela 2) usamos amostras de diferentes tamanhos e sentenças de comprimento entre quatro e quinze.

	$\|\Omega\|$	Iterações até convergir	Tempo	Vlr da constante C
Sub-Problema 1	67	9	34m45,830s	10^{20}
Sub-Problema 2	67	9	335m13,625s	10^{15}
Sub-Problema 3	67	9	4558m27,323s	10^{10}

Tabela 3. Resultados obtidos para o primeiro grupo de sub-problemas.

Da Tabela 3 podemos evidenciar que, uma vez obtida a convergência para um problema com m símbolos não terminais, podemos aumentar este valor e utilizando a estratégia para controlar problemas de *underflow*, atingimos a convergência do método.

A Tabela 4 apresenta um aumento do tamanho da amostra e do comprimento das sentenças da mesma, havendo um maior tamanho dos problemas. O método atinge convergência e

	$\|\Omega\|$	Iterações até convergir	Tempo	Vlr da constante C
Sub-Problema 4	161	9	127m19,235s	10^{25}
Sub-Problema 5	388	9	407m19,538s	10^{25}
Sub-Problema 6	310	9	9519m2,029s	10^{15}

Tabela 4. Resultados obtidos para o segundo grupo de sub-problemas.

como no caso do primeiro grupo de testes, o acompanhamento garante que não se gerem problemas de *underflow*, isto basicamente é controlado com o parâmetro C.

4. Conclusão

O método de pontos interiores mostrou-se um método viável para estimar as gramáticas probabilísticas livres do contexto. Um dos detalhes relevantes é que o número de iterações até atingir convergência é mantido em nove iterações, isto quer dizer que o método é estável para diferentes tamanhos tanto da gramática como do *corpus*. Isto é importante porque uma das maiores desvantagens do Método *Inside-Outside* [Baker 1979] em aplicações práticas é o elevado número de iterações requeridas para atingir a convergência.

O método de pontos interiores implementado é robusto e bem comportado na presença de novos dados, sendo uma proposta que sugere continuar fazendo testes aumentando o tamanho da amostra, até conseguir modelar uma linguagem natural.

Referências

Baker, J. K. (1979). Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132.

El-Bakry, A., Tapia, R., Tsuchiya, T., and Zhang, Y. (1996). On the formulation and theory of the newton point-pnterior for nonlinear programming. *Journal of Optimization Theory and Applications*.

Gondzio, J. (2012). Interior point methods 25 years later. *European Journal of Operational Research*, 218(3):587–601.

IEL-UNICAMP and IME-USP. *Corpus anotado do Português histórico Tycho Brahe*. http://www.tycho.iel.unicamp.br/corpus/index.html, acessado em 2017.

López, E. S. M. (2018). *Método de pontos interiores para estimar os parâmetros de uma gramática probabilística livre do contexto.* Disertação doutorado, Universidade Estadual de Campinas. Instituto de matemática, Estatística e Computação Científica.

Manning, C. D. and Schutze., H. (2003). *Foundations of statistical natural language processing*. Cambridge, MA : MIT.

Ruggiero, M. A. G. and da Rocha Lopes., V. L. (1997). *Cálculo numérico. Aspectos teóricos e computacionais.* São Paulo, SP : Makron.

Saad, Y. (2003). *Iterative Methos for Sparse Linear Systems*. SIAM Publications, SIAM, Philadelphia, PA, USA.

Trefethen, L. N. and Bau III, D. (1997). *Numerical linear algebra*. Siam.

Expected Emergence of Algorithmic Information from a Lower Bound for Stationary Prevalence

Felipe S. Abrahão, Klaus Wehmuth, Artur Ziviani

[1]National Laboratory for Scientific Computing (LNCC)
Av. Getúlio Vargas, 333, Quitandinha – CEP 25651-075 – Petrópolis, RJ – Brazil

fsa@lncc.br, klaus@lncc.br, ziviani@lncc.br

Abstract. *We study emergent information in populations of randomly generated networked computable systems that follow a Susceptible-Infected-Susceptible contagion (or infection) model of imitation of the fittest neighbor. These networks have a scale-free degree distribution in the form of a power-law following the Barabási-Albert model. We show that there is a lower bound for the stationary prevalence (or average density of infected nodes) that triggers an unlimited increase of the expected emergent algorithmic complexity (or information) of a node as the population size grows.*

1. Introduction

The general scope of this work involves complex systems, complex networks, information theory, and computability theory. In particular, we aim at studying the general problem of emergence of complexity or information when complex systems are networked compared with when they are isolated. This issue has a pervasive importance in the literature about complex systems with applications on investigating systemic properties of biological, economical, or social systems. Following a theoretical approach on this subject, we present a study on the emergence of irreducible information in networked computable systems that follow an information-sharing (or communication) protocol based on contagion or infection models, as described in [Pastor-Satorras and Vespignani 2001a, Pastor-Satorras and Vespignani 2001b, Pastor-Satorras and Vespignani 2002]. As supported by these references, such spreading models, using the approach from complex networks, have been shown to be relevant to study epidemic and disease spreading, computer virus infections, or the spreading of polluting agents. Consequently, the study of such systems has helped on immunization strategies, epidemiology, or pollution control.

Nevertheless, instead of focusing on the pathological properties of such complex networks' contagion dynamics, we show a preliminary result on how such contagion dynamics might trigger an unlimited potential of optimization through diffusion. That is, diffusing the best solution (or the largest integer when one uses the Busy Beaver Game [Abrahão et al. 2017] as a toy model) through the network may trigger an unlimited increase of expected emergent algorithmic information of the nodes as the randomly generated population of computable systems (i.e., nodes) grows. Thus, we aim at mathematically investigating under which conditions this phenomenon is expected to happen. For this purpose, we use the theoretical framework for networked computable systems developed in [Abrahão et al. 2017] and the network model studied in [Pastor-Satorras and Vespignani 2001a, Pastor-Satorras and Vespignani 2001b, Pastor-Satorras and Vespignani 2002].

As a toy model, such theoretical approach to the study of emergence of complexity or information in networked computable systems may help understand and establish foundational properties on why an information dynamics within a system displaying synergistic or emergent behavior might be advantageous from a computational, evolutionary, or game-theoretical point of view. Moreover, as it is our goal to suggest in the present work, these phenomena may be also related to infection dynamics (either from computer viruses or diseases) and scale-free networks.

2. Model

We define a model for randomly generated Turing machines that are networked with a scale-free degree distribution that obeys a power law of the form $P(k) \sim 2m^2\,k^{-3}$. The topology and construction of the networks are defined by a random process connecting new nodes under a probability distribution given by preferential attachment as in [Barabási et al. 1999, Barabási and Bonabeau 2003]. That is, new nodes are more likely to establish connections to higher degree nodes. The diffusion or "infection" scheme is ruled by the Susceptible-Infected-Susceptible model (SIS), in which susceptible nodes have a constant probability ν of being "infected" by an "infected" neighbor and an "infected" node has a constant probability δ of becoming "cured". We also assume, as in [Pastor-Satorras and Vespignani 2001a, Pastor-Satorras and Vespignani 2001b, Pastor-Satorras and Vespignani 2002], that the prevalence of "infected" nodes (i.e., the average density of "infected" nodes) becomes stationary after sufficient time.

However, nodes are now Turing machines that can send and receive information (partial outputs) as each node runs its computations until returning a final output. We have defined this population of randomly generated Turing machines and a more general mathematical model for networked computable systems which we have called as *algorithmic networks* [Abrahão 2016, Abrahão et al. 2017]. This population plays the Busy Beaver Imitation Game (BBIG) in which each node always imitates the fittest neighbor only. However, differently from the one in [Abrahão et al. 2017], we here present a variation on the information-sharing protocol. The difference in respect to this previous work comes from allowing nodes to become "cured" (with rate δ). Besides, now nodes also get "infected" with rate ν — which may have a different value from 1 — while in [Abrahão et al. 2017] one has that $\nu = 1$ always holds. While still playing a Busy Beaver Imitation Game, susceptible nodes follow a rule of imitating the neighbor that had output the largest integer, but they follow this rule with probability ν. Thus, the effective spreading rate $\lambda = \nu/\delta$ defined in [Pastor-Satorras and Vespignani 2001a, Pastor-Satorras and Vespignani 2001b, Pastor-Satorras and Vespignani 2002] assumes a direct interpretation of the rate in which the imitation-of-the-fittest protocol [Abrahão et al. 2017] was applied on a node—and this is the reason why we are using the words "infection" and "cure" between quotation marks.

3. Results

Our proofs follow mainly from information theory, computability theory, and graph theory applied on a variation on the information-sharing (communication) protocol of the model in [Abrahão et al. 2017]. From this model, we have proven results for general

dynamic networks and for dynamic networks with small diameter, i.e., $\mathbf{O}(\log(N))$ compared to the network size N. We have shown that there are topological conditions that trigger a phase transition in which eventually the algorithmic network $\mathfrak{N}_{BB}$ begins to produce an unlimited amount of bits of emergent algorithmic complexity (i.e., emergent algorithmic information). These conditions come from a positive trade-off between the average diffusion density and the number of cycles (or communication rounds). Therefore, the diffusion power of a dynamic network has proven to be paramount with the purpose of optimizing the average fitness/payoff of an algorithmic network that plays the Busy Beaver Imitation Game. Besides, this diffusion power may come either from the cover time [Costa et al. 2015] or from a small diameter compared to the network size.

Therefore, our developed proofs[1] for a network following the SIS diffusion model, as in [Pastor-Satorras and Vespignani 2001a, Pastor-Satorras and Vespignani 2001b, Pastor-Satorras and Vespignani 2002], also stems from the main idea of combining an estimation of a lower bound for the average algorithmic complexity/information of a networked node and an estimation of an upper bound for the expected algorithmic complexity/information of an isolated node. The estimation of the latter still comes from the strong law of large numbers, Gibb's inequality, and algorithmic information theory applied on the randomly generated population. However, now the estimation of the former comes from the SIS model with a stationary prevalence (i.e., the average density of infected nodes), which gives directly this lower bound by the fact that the prevalence $\rho \sim 2\exp(-1/m\lambda)$ in [Pastor-Satorras and Vespignani 2001a, Pastor-Satorras and Vespignani 2001b, Pastor-Satorras and Vespignani 2002] becomes equal to the average diffusion density $\tau_{\mathbf{E}}$ in [Abrahão et al. 2017].

Thus, let condition $\tau_{\mathbf{E}}|_{t_0}^{c(N)} \sim 2\exp(-1/m\lambda) > \Omega(w, c(N))$ holds where $c(N)$ is an upper bound for the time in which the network achieves stationary prevalence (as in [Pastor-Satorras and Vespignani 2001b, Pastor-Satorras and Vespignani 2002, Pastor-Satorras and Vespignani 2001a]). Additionally, we have that $c(N)$ is a computable function of the network size N[2], t_0 is the first time instant, w is the network input available to every node at the beginning of the first cycle, and $\Omega(w, c(N))$ is the probability that a node (i.e., a Turing machine) halts, with w as initial input, in every cycle until the last cycle $c(N)$. Note that $\Omega(w, c(N))$ is a value that depends only on the chosen self-delimiting programming language in which one defines a universal Turing machine. Then, one can prove that the expected emergent algorithmic complexity/information of a node goes to infinity as the network size (i.e., the population size) N goes to infinity.

In other words, the average irreducible information that emerges when nodes are networked, compared with when they are isolated, is expected to always increase for large enough populations of randomly generated Turing machines that compute during $\leq c(N)$ cycles (or communication rounds). Moreover, since it is an emergent phenomenon that arises depending only on the network/population size, one can show that such scale-free networks of randomly generated Turing machines are also expected to display, for large enough populations, a complexity/information phase transition, which we have called *expected emergent open-endedness* [Abrahão et al. 2017].

[1]The article presenting these and other results and complete and extended versions of these proofs is available at https://doi.org/10.5281/zenodo.1228505

[2]In fact, as a function of N^C, where C is a constant that may have a value ≤ 1.

4. Conclusions

First, we have defined a family of Barabási-Albert scale-free networks with a power law on the degree distribution of the form $P(k) \sim 2m^2\, k^{-3}$ that follow a contagion (or infection) dynamics described by the Susceptible-Infected-Susceptible model as in [Pastor-Satorras and Vespignani 2001a, Pastor-Satorras and Vespignani 2001b, Pastor-Satorras and Vespignani 2002]. Their set of nodes are randomly generated Turing machines. Additionally, every node plays the SIS Busy Beaver Imitation Game, which ensures that each node always imitates the fittest neighbor whenever an "infection" would occur with probability ν. Nodes also may return to its initial partial output—being "cured"—with probability δ.

Then, we have shown that, for large enough values of $m\,\lambda$, if the time for achieving a stationary prevalence of "infected" nodes $\rho \sim 2\exp(-1/m\lambda)$ is upper bounded by a value given by a computable function of the network (or population) size N, then these networks of randomly generated Turing machines will have the property of expected emergent open-endedness for large enough network/population sizes. That is, the expected emergent algorithmic complexity/information of a node will eventually start to increase undefinitely as the population grows. Thus, this characterizes the complexity (or information) phase transition introduced in [Abrahão et al. 2017].

Acknowledgements

This work is partially funded by CAPES, CNPq, FAPESP, and FAPERJ.

References

Abrahão, F. S. (2016). Emergent algorithmic creativity on networked Turing machines. In *The 8th International Workshop on Guided Self-Organization at the Fifteenth International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*, Cancún.

Abrahão, F. S., Wehmuth, K., and Ziviani, A. (2017). Algorithmic Networks: central time to trigger expected emergent open-endedness.

Barabási, A.-L., Albert, R., and Jeong, H. (1999). Mean-field theory for scale-free random networks. *Physica A: Statistical Mechanics and its Applications*, 272(1-2):173–187.

Barabási, A.-L. and Bonabeau, E. (2003). Scale-Free Networks. *Scientific American*, 288(5):60–69.

Costa, E. C., Vieira, A. B., Wehmuth, K., Ziviani, A., and da Silva, A. P. C. (2015). Time Centrality in Dynamic Complex Networks. *Advances in Complex Systems*, 18(07n08).

Pastor-Satorras, R. and Vespignani, A. (2001a). Epidemic dynamics and endemic states in complex networks. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, 63(6).

Pastor-Satorras, R. and Vespignani, A. (2001b). Epidemic spreading in scale-free networks. *Physical Review Letters*, 86(14):3200–3203.

Pastor-Satorras, R. and Vespignani, A. (2002). Immunization of complex networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 65(3):036104.

Algoritmo Eficiente para Quebra de Privacidade em Redes Sociais Anonimizadas

Pamela Tabak, Daniel Ratton Figueiredo

[1]Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro (UFRJ) – Rio de Janeiro – RJ – Brasil

{pamelatabak, daniel}@cos.ufrj.br

Abstract. *Companies that control social networks often release data, including the users' relationship network, removing their identity to preserve their privacy. This work defines and evaluates a two-step methodology to reveal relationships between users of a social network. In the first phase, artificial nodes (users) are inserted and connected to a set of victim users. The second phase identifies the artificial nodes in the anonymized network and reveals relationships between the victim users. The algorithms were implemented and the results indicate that they are effective in revealing relationships between victims, breaking their privacy.*

Resumo. *Empresas que controlam redes sociais muitas vezes disponibilizam dados, incluindo a rede de relacionamento entre seus usuários, removendo a identidade deles para preservar sua privacidade. Este trabalho define e avalia uma metodologia de duas fases para revelar relacionamentos entre usuários de uma rede social. Na primeira, nós (usuários) artificiais são inseridos e conectados a um grupo de usuários vítimas. A segunda fase identifica os nós artificiais na rede anonimizada e revela os relacionamentos entre os usuários vítimas. Os algoritmos foram implementados e os resultados indicam que eles são eficientes em revelar os relacionamentos entre vítimas, quebrando sua privacidade.*

1. Introdução

Redes sociais vêm sendo usadas por uma quantidade crescente de serviços na *Web*. Um aspecto importante é a privacidade dos usuários e, em particular, os relacionamentos formados entre eles. Tal informação (arestas da rede social) geralmente não é pública e pertence às empresas que operam o serviço.

Entretanto, dados de usuários de redes sociais são ocasionalmente disponibilizados para a realização de estudos e melhorias dos serviços prestados (como o caso do Prêmio Netflix). Porém, para que a privacidade dos usuários seja preservada, os dados são anonimizados, removendo a identificação dos usuários (como trocar o nome dos usuários por um código aleatório) e gerando uma rede anonimizada isomórfica a rede original.

Neste contexto surgem ataques de quebra de privacidade, que visam desvendar a identidade de alguns usuários em redes sociais anonimizadas e, consequentemente, os relacionamentos entre eles. Este trabalho define e analisa uma metodologia de ataque ativo de duas fases, generalizando e avaliando empiricamente uma metodologia proposta recentemente [Backstrom et al., 2007], que vem sendo muito estudada. A primeira fase ocorre antes da rede ser disponibilizada, na qual o atacante cria nós (usuários) artificiais na rede social e conecta tais nós aos usuários vítimas, além de criar conexões entre os

próprios nós artificiais. Na segunda fase o atacante procura pelos nós criados na rede anonimizada, identificando os nós vítimas a partir de suas conexões com os nós criados e, consequentemente, revelando os relacionamentos (arestas) entre os nós vítimas. Existem, portanto, três conjuntos de nós importantes: nós vítimas, usuários da rede social que foram escolhidos para ter a privacidade violada; nós não vítimas, outros usuários da mesma rede social; nós atacantes, nós artificiais adicionados na rede para conduzir o ataque.

2. Criação e conexão dos nós atacantes

Nós atacantes são usados para a identificação dos relacionamentos entre nós vítimas na rede anonimizada, sendo criados e inseridos na rede social antes da anonimização. Iremos assumir que o atacante conhece a identidade de todos os nós antes da rede ser anonimizada, entretanto não conhece o grau destes nós ou as arestas incidentes a eles. Ou seja, o atacante conhece apenas as arestas que ele cria, que incidem sobre nós por ele criado (nós atacantes) e em outros nós (usuários) da rede. A Figura 1 ilustra os grafos após a criação dos nós e arestas artificiais, antes (esquerda) e depois (direita) da anonimização. O grafo da esquerda apresenta os nós atacantes $a_0, ..., a_3$, os nós vítimas v_0, v_1, vértices da rede não vítimas u_0, u_1 e as novas arestas, incidentes aos nós atacantes, em linhas pontilhadas.

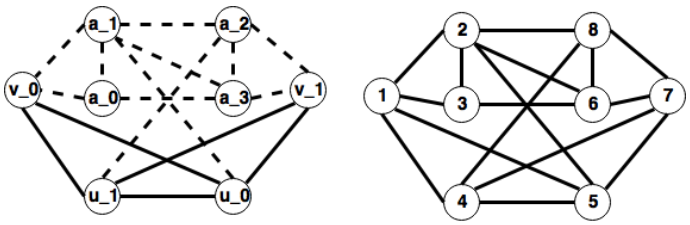


Figura 1. Grafo após a criação de nós e arestas artificiais, antes e depois da anonimização, que gera um identificador aleatório para cada nó.

2.1. Criação dos nós

Cada nó vítima precisa estar conectado a um subconjunto único de nós atacantes, composto por ao menos dois nós, para ser reconhecido. Ou seja, é preciso que nós vítimas se conectem a nós atacantes de maneira única, de forma que ao identificar o conjunto de nós atacantes identificamos também cada nó vítima. Considerando o caso em que todos os nós do grafo original, de tamanho n, estão sendo atacados, o número de nós atacantes precisa ser ao menos $\lceil \log_2 n \rceil + 1$, de forma a termos n subconjuntos distintos de nós atacantes, de tamanho mínimo 2. Neste trabalho, são criados $s = 2\lceil \log_2 n \rceil$ nós atacantes e o fator de 2 foi escolhido arbitrariamente para facilitar a busca por subconjuntos distintos.

2.2. Criação das arestas

É preciso identificar os nós atacantes no grafo anonimizado, formado pelo grafo original mais os nós e arestas artificialmente adicionados, para o ataque funcionar. Portanto, o subgrafo induzido pelos atacantes (a_0,..., a_{s-1}) será criado com propriedades específicas.

1. Criação das arestas entre nós atacantes: toda aresta do tipo (a_i, a_{i+1}) é criada. As demais arestas são criadas aleatoriamente: a aresta entre cada par (a_i, a_j), sendo $i < (j + 1)$, é criada com probabilidade 1/2. Essa configuração foi escolhida devido a propriedade do modelo de Erdõs-Rényi, pois com $p = 1/2$ todos os possíveis grafos têm a mesma probabilidade de serem gerados.

2. Criação das arestas entre nós atacantes e vítimas: um número binário b de s dígitos é criado e inicializado com 0, em que cada dígito representa um nó atacante. Para cada nó vítima é verificado se existem ao menos dois dígitos 1 em b. Caso positivo, para cada posição contendo 1, uma aresta é criada entre o nó atacante correspondente e o nó vítima em questão. Em seguida, b é acrescido em uma unidade binária. Caso não possua pelo menos duas posições com o valor 1, apenas a soma acontece. Esse processo é repetido para cada nó vítima. Dessa forma, é garantido que todo nó vítima estará conectado a pelo menos dois nós atacantes e que nenhum outro nó vítima estará conectado a exatamente os mesmos nós atacantes.
3. Criação das arestas entre nós atacantes e não vítimas: esse processo é feito para aumentar a aleatoriedade do subgrafo de nós atacantes, diminuindo as chances de encontrar um isomorfismo deste subgrafo induzido na rede divulgada. Para cada nó atacante, um número g maior ou igual ao seu grau atual é gerado aleatoriamente. Cada nó atacante se conecta a nós não vítimas até atingir grau igual a g. Entretanto, cada nó não vítima só pode se conectar a um nó atacante, de modo a jamais ser confundido com um nó vítima, que é conectado a pelo menos dois nós atacantes. Este processo é interrompido caso não existam mais nós não vítimas para conectar aos nós atacantes.

2.3. Detecção de Automorfismos

Ao final do procedimento acima, o algoritmo faz uma busca para verificar a possível existência de automorfismos no subgrafo de atacantes que foi criado, o que impossibilita a identificação correta destes nós no grafo anonimizado. Caso um possível automorfismo seja identificado, a criação de nós e arestas descrita acima é refeita.

O algoritmo para procurar por possíveis automorfismos possui uma complexidade polinomial e foi inspirado em ideias propostas recentemente [Hay et al., 2008]: utilizar o grau e a sequência parcial de grau dos vizinhos de cada nó atacante. Este algoritmo analisa se existem dois nós atacantes com mesmo grau e com sequência de grau dos vizinhos atacantes que seja majorada pela outra. Esta análise é uma condição necessária para a existência de um automorfismo entre os nós atacantes, mas não suficiente. Caso positivo, o processo de criação de arestas incidentes a nós atacantes é refeito.

3. Identificação dos nós atacantes e vítimas

Para encontrar os relacionamentos entre nós vítimas na rede anonimizada basta identificar todos os nós atacantes, uma vez que cada nó vítima está conectado a um subconjunto distinto de nós atacantes. Consequentemente, nós vítimas também serão identificados, assim como seus relacionamentos, presentes na rede anonimizada.

O algoritmo procura nós atacantes respeitando o ciclo de arestas (a_i, a_{i+1}), iniciando por a_0 e utilizando duas informações de cada nó atacante: seu grau e sequência de grau dos vizinhos que são nós atacantes. Uma lista de árvores é utilizada e cada árvore representa um possível subgrafo formado por nós atacantes. Para encontrar a_0, o processo identifica todos os nós u do grafo anonimizado que tem o mesmo grau de a_0 e sequência de grau dos nós atacantes adjacentes a a_0 contida na sequência de grau dos nós vizinhos de u. Caso positivo, uma nova árvore com raiz u é criada. Os filhos v de u são todos os vértices vizinhos de u que tem o mesmo grau de a_1 e sequência de grau dos vizinhos

atacantes de a_1 contida na sequência de grau dos vizinhos de v. A busca então avalia cada nó folha f, que representa um possível a_i, sendo i sua altura na árvore. Para cada f, o processo segue as regras de busca definidas nos passos anteriores nos vizinhos de f, procurando por a_{i+1}. Se não encontrar nos vizinhos de f um possível a_{i+1}, f é removido da árvore. Uma vez removido, o algoritmo analisa se o pai deste nó na árvore virou folha. Caso positivo, ele também é removido e o processo continua subindo na árvore até encontrar um nó não folha ou a raiz.

O processo é bem sucedido se ao final da busca existe uma única árvore com s nós, na forma de um grafo linha, identificando assim os nós atacantes $a_0, \ldots, a_{s-1}$. Caso contrário, o subgrafo de nós atacantes possui um automorfismo ou isomorfismo com outro subgrafo induzido do grafo e o processo de identificação dos nós atacantes falha.

4. Resultados

Os algoritmos descritos neste artigo foram testados em redes geradas a partir de dois modelos de grafos aleatórios, Erdös-Rényi e Barabási-Albert, e em três redes reais [Leskovec and Krevl, 2018], de tamanhos variando entre 1000 nós e 5000 arestas até 7115 nós e 100762 arestas. Foram definidos diversos conjuntos de nós vítimas, de tamanhos entre 2 e o número de nós na rede. Para cada rede e conjunto de vítimas, 50 execuções do algoritmo foram realizadas. No total, o procedimento de quebra de privacidade foi executado 20800 vezes em um Macbook Pro, com processador Intel core i7 e 8 Gb de memória RAM, sendo apresentadas algumas médias de tempos de execução na tabela 1. Em apenas 28 casos não foi possível identificar os relacionamentos entre os nós vítimas, tendo sido todas as falhas causadas por isomorfismos entre o subgrafo de nós atacantes e subgrafos do grafo anonimizado, isto é, nós do grafo original foram confundidos com nós atacantes. Esse resultado indica que o processo de detecção de automorfismos foi capaz de evitar este tipo de falha, uma vez que não aconteceu em nenhuma das execuções feitas.

Número de Nós no Grafo Original - Nós Vítimas	Média do Tempo de Execução
1000 - 500	1.7 segundos
1000 - 900	1.9 segundos
4039 - 1000	1.3 segundos
4039 - 4039	7.5 segundos
6301 - 1000	0.8 segundos
6301 - 4250	7.2 segundos
6301 - 6301	14.2 segundos

Tabela 1. Tempo de execução do algoritmo proposto em diferentes cenários

Referências

Backstrom, L., Dwork, C., and Kleinberg, J. (2007). Wherefore art thou r3579x? anonymized social networks, hidden patterns, and structural steganography. In *Proc. of Intern. Conf. WWW*.

Hay, M., Miklau, G., Jensen, D., Towsley, D., and Weis, P. (2008). Resisting structural re-identification in anonymized social networks. *Proc. VLDB Endow.*, 1(1):102–114.

Leskovec, J. and Krevl, A. (2018). SNAP Datasets: Stanford large network dataset collection.

Algoritmos FPT para reconhecer grafos bem cobertos

Rafael T. Araújo[1], Sulamita Klein[2], Rudini Sampaio[1]

[1]Universidade Federal do Ceará, Fortaleza, Brazil

[2]Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil

{rafaelteixeira,rudini}@lia.ufc.br, sula@cos.ufrj.br

Resumo. *Dado um grafo G, sejam $vc(G)$ e $vc^+(G)$ os tamanhos de uma cobertura mínima de vértices e de uma máxima cobertura minimal de vértices, respectivamente. Dizemos que G é bem coberto se $vc(G) = vc^+(G)$ (ou seja, todas as coberturas minimais são mínimas). É coNP-completo decidir se um grafo é bem coberto. Nesse artigo, obtemos algoritmos FPT de tempos $O^*(2^{vc})$ e $O^*(1.4656^{vc^+})$ para decidir se um grafo é bem coberto, parametrizados por $vc(G)$ e $vc^+(G)$, respectivamente, melhorando resultados de Boria et al. em 2015. Também obtemos algoritmo FPT parametrizado por $\alpha(G) = n-vc(G)$ em grafos d-degenerados, que inclui grafos com genus limitado (como grafos planares) e grafos com grau máximo limitado. Finalmente usamos a decomposição primeval para obter algoritmo linear para grafos P_4-laden estendidos e grafos $(q, q-4)$, que é FPT parametrizado por q, melhorando resultados de Klein et al. em 2013.*

1. Introdução

Seja $G = (V, E)$ um grafo e $C, I \subseteq V$. Dizemos que C é *cobertura* se toda aresta de G tem extremidade em C e que I é *independente* se todos os vértices em I são não adjacentes entre si. Sabe-se que C é cobertura se e somente se $V - C$ é independente. Seja $vc(G)$ o tamanho de uma cobertura mínima e $\alpha(G) = n-vc(G)$ o tamanho do maior conjunto independente de G. Um grafo é *bem coberto* se todas coberturas minimais são mínimas (ou também se todos conjuntos independentes maximais são máximos). Grafos bem cobertos foram introduzidos em 1970 por [Plummer 1970]. São interessantes pois o algoritmo guloso que gera um conjunto independente maximal sempre gera um conjunto independente máximo (e também uma cobertura mínima). Porém, decidir se um grafo é bem coberto é coNP-completo mesmo em grafos livres de $K_{1,4}$ [Caro et al. 1996].

Nesse artigo, investigamos esse problema com relação à sua complexidade parametrizada e obtemos algoritmos FPT de tempos $O^*(2^{vc})$ e $O^*(1.4656^{vc^+})$ para decidir se um grafo é bem coberto, parametrizados por $vc(G)$ e $vc^+(G)$, respectivamente, melhorando resultados de Boria et al. em 2015 [Boria et al. 2015]. Também obtemos algoritmo FPT parametrizado por $\alpha(G) = n - vc(G)$ em grafos d-degenerados, que inclui grafos com genus limitado (como grafos planares) e grafos com grau máximo limitado.

Finalmente estudamos a boa cobertura de grafos com poucos P_4's. Em [Klein et al. 2013], a boa cobertura de muitas classes de grafos com poucos P_4's foi investigada, como os cografos, P_4-esparsos, P_4-lite e P_4-tidy. Nesse artigo, também obtemos algoritmo linear para grafos P_4-laden estendidos e grafos $(q, q-4)$, que são superclasses dessas classes (ver Figura 1). Esse algoritmo é FPT parametrizado por q.

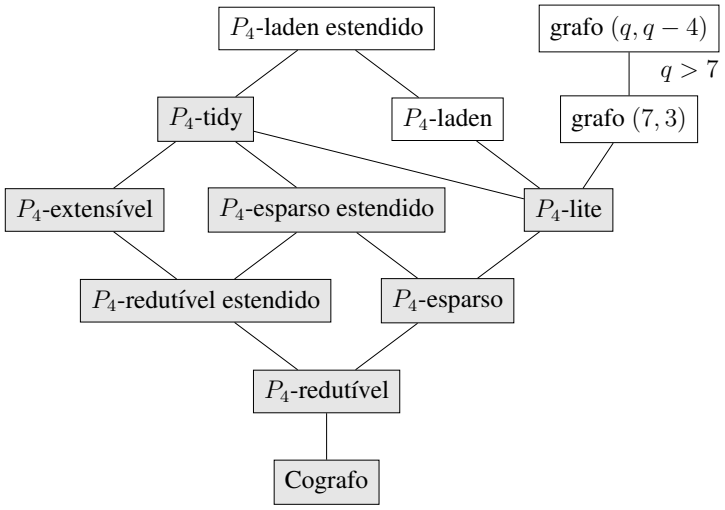


Figure 1. Hierarquia das classes investigadas. Em cinza, as classes investigadas em [Klein et al. 2013].

2. Boa cobertura de grafos com poucos P_4's

Cografo é um grafo sem P_4 induzido. Um grafo é P_4-*esparso* se todo grupo de 5 vértices induz no máximo um P_4. Dado $q \geq 4$, um grafo é $(q, q-4)$ se todo grupo de no máximo q vértices induz no máximo $q-4$ P_4's. Cografos e P_4-esparsos são exatamente os grafos $(4, 0)$ e $(5, 1)$. Em [Babel et al. 2001], foram obtidos algoritmos lineares para vários problemas de otimização em grafos $(q, q-4)$. Um grafo é P_4-laden estendido se todo subgrafo induzido de no máximo 6 vértices é livre de $\{2K_2, C_4\}$ ou contém dois P_4's induzidos [Giakoumakis 1996]. Uma motivação para obter algoritmos em P_4-laden estendidos e grafos $(q, q-4)$ é por eles estarem no topo de uma hierarquia bem conhecida de grafos com poucos P_4's, como P_4-lite, P_4-laden e P_4-tidy (Figura 1). Em [Klein et al. 2013], foram obtidos algoritmos lineares para reconhecer grafos P_4-tidy bem cobertos. Usando a decomposição primeval dos P_4-laden estendidos e dos grafos $(q, q-4)$, obtemos:

Teorema 1 *Seja G um grafo e $q \geq 4$. Se G é $(q, q-4)$ ou P_4-laden estendido, é possível decidir se G é bem coberto em tempo linear $O(2^q q^2 \cdot (m+n))$.*

3. FPT parametrizado por $vc(G)$ e $vc^+(G)$

Um problema de decisão em uma classe de grafos $\mathcal{C}$ é *tratável com parâmetro fixo* (FPT) em algum parâmetro $k = k(G)$ em tempo $O(n^c)$ (para uma constante c) se existe algoritmo em tempo $O(f(k) \cdot n^c)$ que o resolve para qualquer grafo G de $\mathcal{C}$, onde f é uma função que depende apenas de $k = k(G)$. Nesse caso, dizemos que o tempo é $O^*(f(k))$.

Em 2015, [Boria et al. 2015] obtiveram algoritmo FPT de tempo $O^*(1.5397^{vc^+})$ para computar $vc^+(G)$, que pode ser usado para decidir se um grafo é bem coberto. No teorema abaixo, nós estendemos esse resultado.

Teorema 2 *É possível decidir se um grafo é bem coberto em tempo $O^*(1.4656^{vc^+})$.*

Em 2015, [Boria et al. 2015] também obtiveram algoritmo FPT de tempo $O^*(2.8284^{vc})$ para computar $vc^+(G)$. No teorema abaixo, estendemos esse resultado para um algoritmo FPT de tempo melhor $O^*(2^{vc})$ que enumera todas as coberturas minimais de um grafo.

Teorema 3 *É possível enumerar todas as coberturas minimais de vértices de um grafo em tempo* $O^*(2^{vc})$.

Prova: [Esboço] Seja C uma cobertura mínima de vértices de G. Logo toda aresta tem uma extremidade em C. Então, para toda partição de C em dois conjuntos A e B ($A \cup B = C$, $A \cap B = \emptyset$), temos que $(A \cup N(B)) \setminus B$ é uma cobertura de G se não existe aresta com ambas as extremidades em B. Além disso, para toda cobertura minimal C' de G, $A = C \cap C'$ e $B = C \setminus C'$ formam uma partição de C tal que $C' = (A \cup N(B)) \setminus B$, pois $C' \setminus C \subseteq N(B)$ (visto que C' é uma cobertura e é minimal). Portanto, podemos enumerar todas as coberturas minimais de G verificando para cada partição (A, B) de C se $(A \cup N(B)) \setminus B$ é uma cobertura minimal de G. Note que é possível verificar se um conjunto é uma cobertura minimal em tempo $O(m + n)$. Como existem $2^{|C|}$ partições de C, $|C| = vc(G)$ e é possível obter uma cobertura mínima C em tempo $O(2^{vc} \cdot (m + n))$, o resultado segue. □

4. FPT parametrizado por $\alpha(G) = n - vc(G)$

A treewidth local [Eppstein 2000] de um grafo G é a função $ltw_G : \mathbb{N} \to \mathbb{N}$ que associa à cada $r \in \mathbb{N}$ a treewidth máxima de uma r-vizinhança de G. Ou seja, $ltw_G(r) = \max_{v \in V(G)}\{tw(G[N_r(v)]\}$, onde $N_r(v)$ é o conjunto de vértices à distância no máximo r de v. Dizemos que uma classe $\mathcal{C}$ de grafos tem treewidth local limitada se existe uma função $f_{\mathcal{C}} : \mathbb{N} \to \mathbb{N}$ tal que, para todo $G \in \mathcal{C}$ e $r \in \mathbb{N}$, $ltw_G(r) \leq f_{\mathcal{C}}(r)$. Sabe-se que grafos com genus limitado e grafos com grau máximo limitado tem treewidth local limitada [Eppstein 2000]. Em particular, $ltw_G(r) \leq \Delta(G)^r$ e, se G é planar, $ltw_G(r) \leq 3r - 1$ [Bodlaender 1998]. Nós provamos o seguinte:

Teorema 4 *Decidir se um grafo é bem coberto é FPT parametrizado por* $\alpha(G) = n - vc(G)$ *em tempo* $O(n^2)$ *para grafos com treewidth local limitada.*

Prova: [Esboço] Seja $WellCov_k$ a seguinte fórmula lógica de 1^o-ordem que é verdadeira se e só se G não tem dois conjuntos independentes X e Y com $|X| = k$, $|Y| = k - 1$ e Y sendo maximal:

$$WellCov_k := \forall x_1, \ldots, x_k \, \forall y_1, \ldots, y_{k-1} \left(\bigwedge_{1 \leq i < j \leq k} x_i \neq x_j \right) \wedge Indep(\{x_1, \ldots, x_k\})$$

$$\rightarrow \neg\Big(Indep(\{y_1, \ldots, y_{k-1}\}) \wedge Maximal(\{y_1, \ldots, y_{k-1}\}\Big),$$

onde $Indep(X) := \forall x, y((x \in X \wedge y \in X) \rightarrow \neg E(x, y)$ e $Maximal(X) := \forall y \exists x(y \notin X) \rightarrow (x \in X) \wedge E(x, y)$. Note que, se G não é bem coberto, então existem conjuntos independentes X e Y com $2 \leq |X| \leq \alpha$, $|Y| = |X| - 1$ e Y sendo maximal. Assim, seja $WellCov$ a fórmula lógica de 1^o-ordem que é verdadeira se e só se G é bem coberto:

$$WellCov := \bigwedge_{2 \leq k \leq \alpha} WellCov_k.$$

Como $WellCov$ contém no máximo α^2 variáveis, temos pelo Teorema de Frick-Grohe (veja [Flum and Grohe 2006]) que o problema de decidir a boa cobertura é FPT parametrizado por $\alpha(G)$ em tempo $O(n^2)$ para grafos com treewidth local limitada. □

Podemos obter algoritmos FPT específicos (parametrizados por $\alpha(G)$) para grafos d-degenerados, como grafos planares ou grau máximo limitado. Um grafo é *d-degenerado* se todo subgrafo induzido tem um vértice com grau no máximo d. A *degenerância* de G é o menor d tal que G é d-degenerado. Por exemplo, grafos periplanares, grafos planares e grafos com grau máximo Δ tem degenerância no máximo 2, 5 e Δ, respectivamente.

Teorema 5 *O problema de decidir se um grafo é bem coberto é FPT parametrizado por $\alpha = \alpha(G) = n - vc(G)$ em tempo $O((d+1)^\alpha \cdot (m+n))$ para grafos d-degenerados, para todo $d > 0$. Além disso, o tempo é $O(7^\alpha \cdot (m+n))$, se G tem genus limitado.*

Prova: [Esboço] Seja G um grafo d-degenerado. O algoritmo usa uma árvore de busca com altura $\alpha(G)$ em que cada nó tem um grafo associado. A raiz da árvore representa o grafo G. Uma folha é um nó com altura $\alpha(G)$ ou com grafo associado vazio. Seja h um nó não-folha. Ramificamos h de acordo com um vértice v com grau mínimo no grafo associado de h. Seja $N[v] = \{u_1, \ldots, u_\ell\}$, onde $\ell = |N[v]| \leq d+1$. O nó h terá $\ell + 1$ filhos $h_1, h_2, \ldots, h_\ell$ na árvore. No nó filho h_i ($1 \leq i \leq \ell$), remova $N[u_i]$ do grafo associado, que também é d-degenerado. Se existem duas folhas com diferentes alturas, retorne NÃO (pois G tem conjunto independente maximal que não é máximo). Caso contrário, retorne SIM. Note que a altura da árvore é no máximo $\alpha(G)$ e cada nó tem no máximo $d+1$ filhos. Portanto, a árvore tem no máximo $(d+1)^\alpha$ nós e o tempo total é $O((d+1)^\alpha \cdot (m+n))$, pois cada nó leva tempo $O(m+n)$. □

References

Babel, L., Kloks, T., Kratochvil, J., Kratsch, D., Muller, H., and Olariu, S. (2001). Efficient algorithms for graphs with few p4's. *Discrete Mathematics*, 235(1):29 – 51.

Bodlaender, H. L. (1998). A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1 – 45.

Boria, N., Croce, F. D., and Paschos, V. T. (2015). On the max min vertex cover problem. *Discrete Applied Mathematics*, 196:62 – 71.

Caro, Y., Sebő, A., and Tarsi, M. (1996). Recognizing greedy structures. *Journal of Algorithms*, 20(1):137 – 156.

Eppstein, D. (2000). Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27:275–291.

Flum, J. and Grohe, M. (2006). *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc.

Giakoumakis, V. (1996). P4-laden graphs: A new class of brittle graphs. *Information Processing Letters*, 60(1):29 – 36.

Klein, S., de Mello, C. P., and Morgana, A. (2013). Recognizing well covered graphs of families with special p4-components. *Graphs and Combinatorics*, 29:553–567.

Plummer, M. D. (1970). Some covering concepts in graphs. *Journal of Combinatorial Theory*, 8(1):91 – 98.

Tratabilidade por Parâmetro Fixo para Largura em Árvore de Grafos Direcionados*

A. Karolinna Maia[1], **Raul Lopes**[1,2], **Victor Campos**[1,3]

[1]Departamento de Computação – Universidade Federal do Ceará (UFC)
Fortaleza – CE – Brazil
Grupo de Pesquisa ParGO

karolmaia@lia.ufc.br, raul.wayne@gmail.com,campos@lia.ufc.br

Abstract. *The tree-width of an undirected graph G is a measure of how similar G is to a tree. An analogous definition is also known for directed graphs. For constant k, we improve on a known result and show an FPT algorithm that decides whether the tree-width of a directed graph D is at most $3k - 2$ or if it admits a haven of order k.*

Resumo. *A largura em árvore de um grafo não-direcionado G é uma medida de quão similar G é de uma árvore. Uma definição análoga também é conhecida para grafos direcionados. Para k constante, nós melhoramos um resultado conhecido mostrando um algoritmo FPT, com parâmetro k, que decide se a largura em árvore de um grafo direcionado D é no máximo $3k - 2$ ou se D admite um refúgio de ordem k.*

Parâmetros de largura em grafos podem ser vistos como uma estimava de quão similar um grafo é de uma estrutura típica. Muitos problemas conhecidamente difíceis podem ser eficientemente resolvidos em grafos com parâmetros de largura limitados, seja fazendo uso de técnicas clássicas de construção de algoritmos, como programação dinâmica, ou enunciando o problema em lógica monádica de segunda ordem, como afirmado pelo Teorema de Courcelle [Courcelle 1990]. O foco deste trabalho é em largura em árvore de grafos direcionados. Aplicações de algoritmos baseados em decomposição em árvore vão de problemas de alocação de frequência [Koster et al. 1999] ao problema do caixeiro viajante [Cook and Seymour 2003].

Foi mostrado que grades são estruturas bloqueadores para largura em árvore em grafos não-direcionados. Como a largura em árvore de uma grade completa $k \times k$ é k, a largura em árvore de um grafo não-direcionado deve ser pelo menos a ordem da maior grade nele contido como um grafo menor. [Robertson and Seymour 1986] mostraram que existe uma função computável $f : \mathbb{N} \to \mathbb{N}$ tal que todo grafo não-direcionado de largura em árvore pelo menos $f(k)$ contém uma grade $k \times k$ como um grafo menor. Este resultado, conhecido como o Teorema da grade, é fundamental no estudo de parâmetros de largura em grafos não-direcionados. A partir dele, foi criado um campo de estudo conhecido como teoria da bidimensionalidade, introduzido

[2]Financiado pela CAPES.

[3]Parcialmente financiado pelo CNPq.

*Este trabalho foi parcialmente apoiado pelo CNPq - projeto Universal 425297/2016-0 e FUNCAP - PRONEM PNE-0112-00061.01.00/16.

em [Demaine et al. 2005], pelo qual os autores receberam o prêmio Nerode 2015. A teoria da bidimensionalidade foi usada como base para algoritmos aproximativos, subexponenciais, esquemas de aproximação em tempo polinomial e kernelizações para um grande número de problemas difíceis em grafos que excluem um grafo menor fixo. Estes incluem o problema de conjunto de vértices de retroalimentação, conjunto dominante, emparelhamento mínimo maximal e conjunto independente, entre outros. Para alguns exemplos, encaminhamos o leitor para [Demaine et al. 2005, Demaine and Hajiaghayi 2004, Fomin et al. 2010, Fomin et al. 2011].

Dado o sucesso alcançado no estudo de parâmetros de largura para problemas em grafos não-direcionados, não é surpresa que haja interesse em encontrar definições análogas para grafos direcionados. Em particular, algumas foram propostas para largura em árvore em grafos direcionados ([Johnson et al. 2001, Reed 1997], por exemplo). Um resultado análogo ao Teorema da grade, que permaneceu como uma conjectura aberta de [Reed 1999] e [Johnson et al. 2001] por quase 20 anos, foi provado recentemente para grafos direcionados. Foi mostrado que as estruturas bloqueadoras para largura em árvore grande em grafos direcionados são grades cilíndricas [Kawarabayashi and Kreutzer 2015]. Um exemplo de grade cilíndrica de ordem 4 é dado na Figura 1. Neste trabalho consideramos as definições contidas em [Johnson et al. 2001]. Detalhamos estas abaixo para fins de completude. Deste ponto em diante, utilizamos D para representar um grafo direcionado.

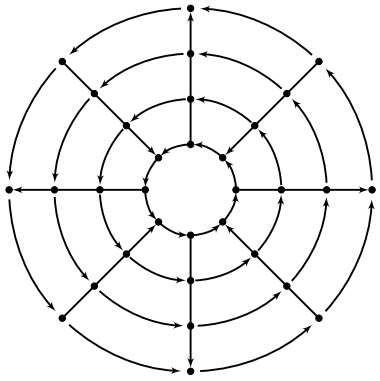


Figura 1. Grade cilíndrica de ordem 4.

Definição (Conjuntos Z-normais)**.** *Seja Z um subconjunto de $V(D)$ e $S \subseteq V(D) - Z$. Dizemos que S é Z-*normal *se não existe passeio direcionado em $D \setminus Z$ com ambas as extremidades em S que utilize um vértice de $D \setminus (Z \cup S)$.*

Seja R uma arborescência. Para $\{r, r'\} \subseteq V(R)$, escrevemos $r' > r$ se $r' \neq r$ e existe um caminho direcionado em R de r para r'. Ademais, se e é uma aresta de R com cabeça r, escrevemos $r' > e$ se $r' = r$ ou $r' > r$. Se e incide em r, escrevemos $e \sim r$.

Definição (Decomposição arbórea)**.** *Uma* decomposição arbórea *de D é um trio (R, X, W) onde R é um arborescência, $X = \{X_e : e \in E(R)\}$ e $W = \{W_r : r \in V(R)\}$ são tais que*

1. *X_e e W_r são subconjuntos de $V(D)$, para toda $e \in E(R)$ e $r \in V(R)$;*
2. *$\{W_r : r \in V(R)\}$ é uma partição de $V(D)$ em conjuntos não-vazios;*
3. *se $e \in E(R)$, então $\bigcup\{W_r : r \in V(R), r > e\}$ é X_e-normal.*

A largura de (R, X, W) *é o menor inteiro* k *tal que, para todo* $r \in V(R)$,

$$|W_r \cup (\bigcup_{e \sim r} X_e)| \leq k + 1.$$

A largura em árvore de D *é o menor inteiro* k *tal que* D *admite uma decomposição arbórea de largura* k.

Definição (Refúgio)**.** *Um* refúgio *de ordem* k *em* D *é uma função* β *atribuindo a todo conjunto de vértices* $Z \subseteq V(D)$*, com* $|Z| \leq k - 1$*, o conjunto de vértices* $\beta(Z)$ *de uma componente fortemente conexa de* $D \setminus Z$ *de tal forma que se* $Z' \subseteq Z \subseteq V(D)$*, então* $\beta(Z) \subseteq \beta(Z')$.

Em [Johnson et al. 2001] foi mostrada uma relação min-max entre refúgios e decomposições em árvore de grafos não-direcionados.

Teorema 1. *Seja* G *um grafo não-direcionado e* $k \geq 0$ *um inteiro. Então* G *admite um refúgio de ordem* k *se e somente se* G *tem largura em árvore maior ou igual a* $k - 1$.

Para o caso direcionado, sabemos que apenas a condição necessária do Teorema 1 apresenta uma cota inferior tão justa quanto no caso não-direcionado [Johnson et al. 2001]. Para a suficiência, um resultado aproximativo é dado também em [Johnson et al. 2001].

Teorema 2. *Seja* D *um grafo direcionado e* k *um inteiro não-negativo. Existe um algoritmo polinomial, para* k *constante, que decide se* D *tem largura em árvore no máximo* $3k - 2$ *ou admite um refúgio de ordem* k.

Adaptando o algoritmo do Teorema 2 com técnicas baseadas em separadores introduzidas em [Chen et al. 2007, Chitnis et al. 2011, Erbacher et al. 2014], melhoramos este resultado mostrando que o problema é tratável por parâmetro fixo com parâmetro k.

Teorema 3 (Resultado principal)**.** *Seja* D *um grafo direcionado e* k *um inteiro não-negativo. Existe um algoritmo FPT com parâmetro* k *que decide se* D *tem largura em árvore no máximo* $3k - 2$ *ou admite um refúgio de ordem* k.

Referências

Chen, J., Liu, Y., and Lu, S. (2007). Directed feedback vertex set problem is fpt. In *Structure Theory and FPT Algorithmics for Graphs, Digraphs and Hypergraphs*, number 07281 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

Chitnis, R., Hajiaghayi, M., and Marx, D. (2011). Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. abs/1110.0259.

Cook, W. and Seymour, P. (2003). Tour merging via branch-decompositions. *J. on Computing*, 15:233–248.

Courcelle, B. (1990). The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and Computation*, 85(1):12 – 75.

Demaine, D., Fomin, V., Hajiaghayi, M., and Thilikos, M. (2005). Subexponential parameterized algorithms on bounded-genus graphs and h-minor-free graphs. *J. ACM*, 52(6):866–893.

Demaine, E. and Hajiaghayi, M. (2004). Fast algorithms for hard graph problems: Bidimensionality, minors, and local treewidth. *Graph drawing*, pages 517–533.

Erbacher, R., Jaeger, T., Talele, N., and Teutsch, J. (2014). Directed multicut with linearly ordered terminals. *CoRR*, abs/1407.7498.

Fomin, F., Lokshtanov, D., Raman, V., and Surabh, S. (2011). Bidimensionality and eptas. *ACM-SIAM Symposium on discrete algorithms (SODA)*, pages 748–759.

Fomin, F., Lokshtanov, D., Surabh, S., and Thilikos, D. (2010). Bidimensionality and kernels. *ACM-SIAM Symposium on discrete algorithms (SODA)*, pages 503–510.

Johnson, T., Robertson, N., Seymour, P., and Thomas, R. (2001). Directed tree-width. *Journal of combinatorial theory, series B*, 82(01):138–154.

Kawarabayashi, K. and Kreutzer, S. (2015). The directed grid theorem. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 655–664, New York, NY, USA. ACM.

Koster, A., van Hoesel, S., and Kolen, A. (1999). Solving frequency assignment problems via tree-decomposition. *Broersma, H.J. et al 6th Twente workshop on graphs and combinatorial optimization. Univ. of Twente, Enschede, Netherlands*, 3.

Reed, B. (1997). Tree width and tangles: A new connectivity measure and some applications. *R. Bailey, editor, Surveys in Combinatorics*, pages 87–162.

Reed, B. (1999). Introducing directed tree-width. *Electronic notes in discrete mathematics*, 03:222–229.

Robertson, N. and Seymour, P. (1986). Graph minors v. excluding a planar graph. *Journal of combinatorial theory, series B*, 41(01):92–114.

Complexidade Parametrizada de Cliques e Conjuntos Independentes em Grafos Prismas Complementares

Priscila Camargo, Alan D. A. Carneiro, Uéverton S. Santos

[1]Programa de Pós-Graduação em Computação - IC UFF, Niterói, Brasil

{pcamargo,aaurelio,usouza}@ic.uff.br

Resumo. *Um grafo do tipo prisma complementar $G\bar{G}$ surge da união disjunta do grafo G e do seu complementar $\bar{G}$ e a adição de um emparelhamento perfeito entre os vértices correspondentes em G e $\bar{G}$. Os problemas clássicos determinar se no grafo existe uma clique de ordem k e determinar se no grafo existe um conjunto independente de ordem k são provados NP-completos quando o grafo de entrada é um prisma complementar. Neste trabalho, estudamos a complexidade de ambos os problemas em grafos prismas complementares sob a ótica da complexidade parametrizada. Nós provamos que estes problemas possuem um núcleo e, portanto são Tratáveis por Parâmetro Fixo (FPT). Em seguida, mostramos que ambos não admitem núcleo polinomial.*

Abstract. *The complementary prism $G\bar{G}$ arises from the disjoint union of the graph G and its complement $\bar{G}$ by adding the edges of a perfect matching joining pairs of corresponding vertices of G and $\bar{G}$. The classical problems of graph theory, clique and independent set were proved NP-complete when the input graph is a complemantary prism. In this work, we study the complexity of both problems in complementary prisms graphs from the parameterized complexity point of view. First, we prove that these problems have a kernel and therefore are Fixed-Parameter Tractable (FPT). Then, we show that both problems do not admit polynomial kernel.*

1. Introdução

Fazendo uma alusão ao conhecido prisma, um **prisma complementar** é composto por duas bases: um grafo G e o seu complementar $\bar{G}$. A união destas bases é dada através de um emparelhamento perfeito entre os vértices correspondentes em G e $\bar{G}$. Os prismas complementares foram introduzidos na última década por [Haynes et al. 2007]. Em [Duarte et al. 2017] foram analisadas as complexidade de cliques, conjuntos independentes, k-dominação e P_3-convexidade em prismas complementares. Em [Duarte et al. 2017] foi provado que tanto o problema da clique quanto o problema dos conjuntos independentes são NP-completos para esta classe de grafos.

Neste artigo, investigamos os problemas das cliques e dos conjuntos independentes em grafos prismas complementares sob a ótica da complexidade parametrizada. Devido a restrições de espaço, todas as provas serão omitidas.

2. Definições e Notações

Nesta seção serão apresentados os principais conceitos e notações utilizados neste trabalho.

Grafos: Utilizamos as notações e conceitos padrões de teoria dos grafos e quaisquer notações ou conceitos não definidos podem ser encontrados em [Bondy and Murty 1976]. *Grafo* G é um par ordenado ($V(G)$, $E(G)$), onde $V(G)$ é um conjunto finito de vértices e $E(G)$ um conjunto de arestas formadas por pares de vértices. Denotaremos por $\bar{G}$ o complemento de um grafo G. $\bar{G}$ é composto pelo mesmo conjunto de vértices de G e pelo conjunto de arestas complementares de G. Isto é, se a aresta uv existir em G, ela não existirá em $\bar{G}$, logo os vértices u e v não serão adjacentes em $\bar{G}$. Entretanto, se os vértices u e v não são adjacentes em G, existirá a aresta uv em $\bar{G}$. O número de vértices de um grafo G é dito ser a *ordem* ou *cardinalidade* de G. Um *subgrafo* de um grafo G é um grafo H tal que $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$, denotado por $H \subseteq G$. Um subgrafo H é um subgrafo induzido de G por um subconjunto $X \subseteq V(G)$ (representado por $H[X]$) se, para qualquer par de vértices x e y de H, xy é uma aresta de H se e somente se xy for uma aresta de G. *Clique* é um subgrafo induzido completo, isto é, cada par de vértices desse subgrafo é conectado por uma aresta. Denotaremos por k-CLIQUE o problema de encontrar uma clique de tamanho k em um grafo G. *Conjunto independente* de um grafo G é um subconjunto S de vértices de G tal que em S não existem dois vértices adjacentes. Denotaremos por k-CONJUNTO INDEPENDENTE o problema de encontrar um conjunto independente de tamanho k em um grafo G. *Grafos split* são grafos cujo conjunto de vértices pode ser particionado em 2 subconjuntos distintos: um conjunto independente e uma clique. Denotamos como k-SPLIT o problema de determinar se um grafo G possui um subgrafo split H. Denotaremos por k-RAMSEY o problema de encontrar uma clique de tamanho k ou um conjunto independente de tamanho k em um grafo G. Tal problema não admite núcleo polinomial a menos que $NP \subseteq coNP/poly$.

Complexidade Parametrizada: As definições e conceitos básicos de complexidade parametrizada podem ser encontrados em [Cygan et al. 2015]. Um problema parametrizado $\Pi(k)$ é chamado Tratável por Parâmetro Fixo (*Fixed Parameter Tratable* (FPT)) se existe um algoritmo A (chamado algoritmo FPT) que computa corretamente qualquer instância $I = (\chi, k)$ onde χ denota o tamanho da entrada e decide se I é uma instância *sim* ou *não* em tempo $f(k).|\chi|^{O(1)}$ para alguma função computável f. Dessa maneira, se k é definido com um valor pequeno, o crescimento da função em relação a χ é pequeno.

A **kernelização** ou **redução a um núcleo** é uma poderosa técnica comumente utilizada na obtenção de algoritmos FPT para problemas parametrizados. A técnica consiste em reduzir uma instância I parametrizada a uma nova instância equivalente I' tal que I' de alguma forma é limitada pelo parâmetro k. A instância reduzida I' é chamada de **kernel** ou **núcleo**. O núcleo de um problema parametrizado é formalmente definido a seguir:

Definição 1. Núcleo: *Seja* $\Pi(k)$ *um problema parametrizado.* $\Pi(k)$ *possui um núcleo se para toda instância* $I = (\chi, k)$ *existe um algoritmo* A *executável em tempo polinomial que fornece como saída uma instância equivalente* $I' = (\chi, k')$ *tal que* $k' \leq g(k)$ *para alguma função computável* g.

O ponto chave na técnica de redução a um núcleo é o tamanho do núcleo em relação ao parâmetro onde busca-se encontrar o menor núcleo já que possibilita o desenvolvimento de um algoritmo FPT mais eficiente. Para demonstrar o limite inferior no tamanho do kernel, usamos uma Transformação Polinomial Parametrizada, chamada PPT-redução. Tal redução é definida a seguir.

Definição 2. PPT-redução: *Seja* $\Pi(k)$ *e* $\Pi'(k')$ *dois problemas parametrizados onde*

$k' \leq g(k)$ *para alguma função polinomial* $g : \mathbb{N} \to \mathbb{N}$. *Uma PPT-redução de* $\Pi(k)$ *a* $\Pi'(k')$ *é uma redução* R *tal que:*

(i) Para toda instância $I = (\chi, k)$*, temos que* $I \in \Pi(k)$ *se e somente se* $R(I) \in \Pi'(k')$*;*
(ii) R *é computável em tempo polinomial em relação a* k*.*

3. Resultados

Nesta seção provamos que tanto k-CLIQUE quanto k-CONJUNTO INDEPENDENTE são FPT's e não admitem núcleo de tamanho polinomial.

3.1. Tratabilidade Parametrizada

Afim de provar que k-CLIQUE e k-CONJUNTO INDEPENDENTE são FPT quando o grafo de entrada é $G\bar{G}$ e o parâmetro é o tamanho da solução k, utilizamos o conhecido limite superior do número de Ramsey (Teorema 1). No Teorema de Ramsey, dados inteiros positivos k e l existe um inteiro positivo $R = r(k, l)$ tal que, em qualquer grafo com R de vértices, sempre existe uma clique de tamanho k ou conjunto independente de tamanho l.

Teorema 1. *[Erdös and Szekeres 1935] Para todos inteiros positivos* k *e* l*, vale que:* $r(k, l) \leq \begin{pmatrix} k+l-2 \\ k-1 \end{pmatrix}$.

Teorema 2. k-CLIQUE *e* k-CONJUNTO INDEPENDENTE *são FPT quando o grafo de entrada é* $G\bar{G}$ *e o parâmetro é o tamanho da solução* k*.*

O Teorema 2, portanto, apresenta um núcleo para o problema k-CLIQUE ou k-CONJUNTO INDEPENDENTE quando o grafo de entrada é um prisma complementar. Dessa forma, ambos os problemas são solucionáveis em tempo $f(k).n^{O(1)}$ e pertencem à classe FPT.

3.2. Limite Inferior de Redução a um Núcleo

Em complexidade parametrizada, um problema $\Pi(k)$ pertence a classe FPT se e somente se $\Pi(k)$ possuir um núcleo. Em geral esses núcleos obtidos possuem tamanhos exponenciais, ou até pior, em relação ao parâmetro. Dessa forma, é natural para qualquer problema FPT questionarmos quão bom é o núcleo, ou mesmo se é possível a obtenção de um núcleo menor. Em geral busca-se um núcleo de tamanho polinomial, embora existam diversos resultados positivos sobre a existência de núcleos polinomiais ou até mesmo lineares. Recentemente apareceram os primeiros resultados a respeito da inviabilidade destes núcleos [dos Santos and dos Santos Souza 2015].

Mostramos que embora k-CLIQUE e k-CONJUNTO INDEPENDENTE pertençam à classe FPT quando parametrizados pelo tamanho da solução k, ambos *não* admitem núcleo polinomial. Primeiramente provamos que problema k-CLIQUE não admite núcleo polinomial através de uma PPT-redução do problema k-RAMSEY.

Teorema 3. k-CLIQUE *quando o grafo de entrada é* $G\bar{G}$ *e o parâmetro é o tamanho da solução* k *não admite núcleo polinomial, a menos que* $NP \subseteq coNP/poly$*.*

Mostramos que o problema dos conjuntos independentes em grafos prismas complementares também não admite núcleo polinomial (Corolário 6). Entretanto, diferentemente de cliques, os conjuntos independentes não são tão bem comportados nos prismas

complementares. Sendo assim, dividiremos essa demonstração em duas etapas. Na primeira (Lema 4), é demonstrada uma PPT-redução do problema k-SPLIT para o problema k-CONJUNTO INDEPENDENTE. A segunda (Teorema 5) demonstrada uma PPT-redução do problema k-RAMSEY para o problema k'-SPLIT onde $k' = 3k$.

Lema 4. *k-SPLIT quando parametrizado pelo tamanho da solução k é PPT-redutível ao problema k'-CONJUNTO INDEPENDENTE tendo como grafo de entrada um prisma complementar $G\bar{G}$ e parametrizado pelo tamanho da solução $k' = k$.*

A redução do problema k-RAMSEY para o problema k-SPLIT é formalmente apresentada a seguir:

Teorema 5. *k-RAMSEY parametrizado pelo tamanho da solução k é PPT-redutível ao k'-SPLIT parametrizado pelo tamanho da solução $k' = 3k$.*

Transitivamente a partir do Lema 4 e do Teorema 5 temos que k-CONJUNTO INDEPENDENTE não admite núcleo polinomial.

Corolário 6. *k-CONJUNTO INDEPENDENTE quando o grafo de entrada é $G\bar{G}$ e o parâmetro é o tamanho da solução k não admite núcleo polinomial, a menos que $NP \subseteq coNP/poly$.*

4. Conclusões e Considerações Finais

Neste trabalho, estudamos o problema de k-CLIQUE e k-CONJUNTO INDEPENDENTE tendo como entrada a classe de grafos prisma complementar. Utilizando como ferramenta a Teoria de Ramsey, provamos que tanto o problema k-CLIQUE quanto o problema k-CONJUNTO INDEPENDENTE em grafos prismas complementares pertencem à classe FPT quando parametrizados pelo tamanho da solução k, ou seja, são Tratáveis por Parâmetro Fixo. Além disso, através de PPT-reduções mostramos que ambos os problemas não admitem núcleo polinomial. Finalmente, é interessante evidenciar que a inviabilidade de núcleo polinomial para o problema k-CONJUNTO INDEPENDENTE quando parametrizado pelo tamanho da solução k foi obtida de forma transitiva a partir do problema intermediário k-SPLIT. Sendo assim, o problema de determinar se um grafo possui um subgrafo split de cardinalidade k é FPT, e a menos que $NP \subseteq coNP/poly$, não admite núcleo polinomial.

Referências

Bondy, J. A. and Murty, U. S. R. (1976). *Graph theory with applications*, volume 290. Macmilan.

Cygan, M., Fomin, F. V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., and Saurabh, S. (2015). *Parameterized algorithms*, volume 3. Springer.

dos Santos, V. F. and dos Santos Souza, U. (2015). Uma introdução à complexidade parametrizada.

Duarte, M. A., Penso, L., Rautenbach, D., and dos Santos Souza, U. (2017). Complexity properties of complementary prisms. *Journal of Combinatorial Optimization*, 33(2):365–372.

Erdös, P. and Szekeres, G. (1935). A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470.

Haynes, T. W., Henning, M. A., Slater, P. J., and van der Merwe, L. C. (2007). The complementary product of two graphs. *Bulletin of the Institute of Combinatorics and its Applications*, 51:21–30.

On Tuza's conjecture for graphs with treewidth at most 6

Fábio Botler[1], **Cristina G. Fernandes**[2], **Juan Gutiérrez**[2]

[1] CIMFAV, Facultad de Ingeniería
Universidad de Valparaíso, Valparaíso – Chile
fbotler@dii.uchile.cl

[2] Departamento de Ciência da Computação
Universidade de São Paulo – São Paulo, SP – Brazil
{cris,juanguti}@ime.usp.br

Abstract. *Tuza (1981) conjectured that the size $\tau(G)$ of a minimum set of edges that meets every triangle of a graph G is at most twice the size $\nu(G)$ of a maximum set of edge-disjoint triangles of G. In this paper we verify this conjecture for graphs with treewidth at most* 6.

1. Introduction

In this paper, all graphs considered are simple and the notation and terminology are standard. A *triangle transversal* of a graph G is a set of edges of G whose deletion results in a triangle-free graph; and a *triangle packing* of G is a set of edge-disjoint triangles of G. We denote by $\tau(G)$ (resp. $\nu(G)$) the size of a minimum triangle transversal (resp. triangle packing) of G. In [Tuza 1981] the following conjecture was posed:

Conjecture (Tuza, 1981)**.** *For every graph G, we have $\tau(G) \leq 2\nu(G)$.*

This conjecture was verified for many classes of graphs, in particular for planar graphs in [Tuza 1990], and tripartite graphs in [Haxell and Kohayakawa 1998]. A set of tools for dealing with graphs that contain vertices of small degree was introduced in [Puleo 2015] (Lemma 4), and Tuza's Conjecture was verified for graphs with maximum average degree less than 7, i.e., for graphs in which every subgraph has average degree less than 7. In this paper, we extend this technique (Lemma 5) in order to prove Tuza's Conjecture for graphs with treewidth at most 6 (Theorem 6). The following example (Figure 1) shows that there are graphs with treewidth at most 6 whose maximum average degree is at least 7. So our result is not implied by the previous ones. (Conversely, note that the $k \times k$ grid has treewidth at least k and maximum average degree at most 4.) Due to space limitations, we present only a sketch of some proofs.

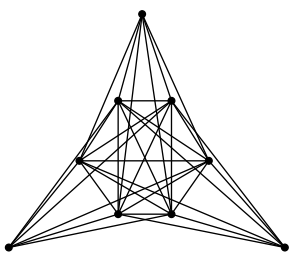


Figure 1. A graph with treewidth 6 and average degree 22/3.

A *rooted tree* is a pair (T, r), where T is a tree and r is a vertex of T. Given $t \in V(T)$, let $P_T(t)$ be the unique path in T that joins r and t. If t' is a vertex in $V(P_T(t))$, then t' is an *ancestor* of t. Every vertex in T that has t as its ancestor is called a *descendant* of t. If $t \neq r$, then the *parent* of t, denoted by $p(t)$, is the ancestor of t that is adjacent to t. The *successors* of t are the vertices whose parent is t, and we denote the set of sucessors of t by $S_T(t)$. The *height* of t, denoted by $h_T(t)$, is the length of a longest path that joins t to a descendant of t. When T is clear from the context, we simply write $S(t)$ and $h(t)$. For a graph G and a vertex v of G, we denote by $N_G(v)$ the set of neighbors of v in G. When G is clear from the context, we simply write $N(v)$.

A *tree decomposition* of a graph G is a pair $(T, \mathcal{V})$, consisting of a tree T and a collection $\mathcal{V} = \{V_t : t \in V(T)\}$ of *bags* $V_t \subseteq V(G)$, satisfying the following conditions:

(T1) $\bigcup_{t \in V(T)} V_t = V(G)$;
(T2) for every $uv \in E(G)$, there exists a bag V_t such that $u, v \in V_t$;
(T3) if a vertex v is in two different bags V_{t_1}, V_{t_2}, then $v \in V_t$ for every t in the path of T that joins t_1 and t_2.

The *width* of $(T, \mathcal{V})$ is the number $\max\{|V_t| - 1 : t \in V(T)\}$, and the *treewidth* $tw(G)$ of G is the minimum width of any tree decomposition of G. We refer to the vertices of T as *nodes*. If G is a graph with treewidth k, then we say that $(T, \mathcal{V})$ is a *full tree decomposition* of G if $|V_t| = k + 1$ for every $t \in V(T)$, and $|V_t \cap V_{t'}| = k$ for every $tt' \in E(T)$. Every graph has a full tree decomposition (see [Bodlaender 1998, Gross 2014]).

We say that a triple $(\mathcal{V}, T, r)$ is a *rooted tree decomposition* of a graph G if $(\mathcal{V}, T)$ is a full tree decomposition of G, (T, r) is a rooted tree, and $V_t \cap V_{p(t)} \neq V_t \cap V_{t'}$ for every $t \in V(T) \setminus \{r\}$ and $t' \in S(t)$. Given a rooted tree decomposition $(\mathcal{V}, T, r)$ of a graph G, and a node $t \in V(T)$ with $t \neq r$, we say that the (unique) vertex $v_t \in V_t \setminus V_{p(t)}$ is the *representative* of t.

Proposition 1. *Every graph has a rooted tree decomposition.*

Proposition 2. *If t is a leaf of a rooted tree decomposition of a graph G and y is the representative of t, then $N_G(y) \subseteq V_t$.*

2. Graphs with treewidth at most 6

A nonempty set $V_0 \subseteq V(G)$ is called *reducible* if there is a set $X \subseteq E(G)$ and a set Y of edge-disjoint triangles in G such that the following conditions hold: (i) $|X| \leq 2|Y|$; (ii) every triangle containing a vertex of V_0 has an edge in X; and (iii) if $uv \in E(A)$ for some $A \in Y$, and $u, v \notin V_0$, then $uv \in X$. When V_0, X, and Y satisfy the definition above, we say that V_0 is *reducible using* X and Y. When G has no reducible set, we say that G is *irreducible*. The following lemma comes naturally.

Lemma 3 ([Puleo 2015, Lemma 2.2])**.** *Let G be a graph and $V_0 \subseteq V(G)$ be reducible using X and Y. Let $G' = (G - X) - V_0$. If $\tau(G') \leq 2\nu(G')$, then $\tau(G) \leq 2\nu(G)$.*

We say that a graph G is *robust* if, for every $v \in V(G)$, every component of $G[N(v)]$ has order at least 5. The following lemma (see [Puleo 2015, Lemma 2.7]) is an important tool in our proof. In what follows, the closed neighborhood $N(u) \cup \{u\}$ of a vertex $u \in V(G)$ is denoted by $N[u]$ and $\Delta(G)$ is the maximum degree in G.

Lemma 4. *If G is an irreducible robust graph and x, $y \in V(G)$, then the following holds.*

(a) *if* $d(x) \leq 6$*, then* $\Delta(\overline{G[N(x)]}) \leq 1$ *and* $|E(\overline{G[N(x)]})| \neq 2$*;*
(b) *if* $d(x) \leq 6$ *and* $d(y) \leq 6$ *then* $xy \notin E(G)$*;*
(c) *if* $d(x) = 7$ *and* $d(y) = 6$ *then* $N[y] \not\subseteq N[x]$*;*
(d) *if* $d(x) \leq 8$ *and* $d(y) = 5$*, then* $N[y] \not\subseteq N[x]$.

In this paper we extend the result above to the following lemma.

Lemma 5. *If* G *is an irreducible robust graph and* x*,* $y \in V(G)$ *are such that* $xy \notin E(G)$*,* $d(x)$*,* $d(y) \leq 6$*, and* $|N(x) \cup N(y)| \leq 7$*, then* $d(x) = d(y) = 5$*,* $|N(x) \cap N(y)| = 3$*, and* $G[N(x)], G[N(y)] \simeq K_5$.

The following theorem is the main result of this paper.

Theorem 6. *If* G *is a graph with treewidth at most* 6*, then* $\tau(G) \leq 2\nu(G)$.

Proof. Suppose, for a contradiction, that the statement does not hold, and let G be a minimal counterexample. It is not hard to check that $|V(G)| \geq 8$. We claim that G is irreducible. Indeed, suppose that there is a set $V_0 \subseteq V(G)$ that is reducible using X and Y, and let $G' = (G - X) - V_0$. Since $G' \subseteq G$, we have $tw(G') \leq 6$, and by the minimality of G, we have $\tau(G') \leq 2\nu(G')$. Thus, by Lemma 4, $\tau(G) \leq 2\nu(G)$, a contradiction. It is not hard to check that G is also robust, and hence has minimum degree at least 5.

By Proposition 1, G has a rooted tree decomposition $(T, \mathcal{V}, r)$. Since $|V(G)| \geq 8$, we have $|V(T)| > 1$, and hence there is a node $t \in V(T)$ with $h(t) = 1$. Suppose that $S(t) = \{t'\}$, and $V_t = \{v_t, v_1, v_2, v_3, v_4, v_5, v_6\}$, where v_t is the representative of t if $t \neq r$. Since $(T, \mathcal{V}, r)$ is a rooted tree decomposition of G, we may assume that $V_{t'} = \{v_t, v_1, v_2, v_3, v_4, v_5, x\}$, with $x \neq v_6$. By Proposition 2, $d(x) \leq 6$, which implies, that $d(x) \in \{5, 6\}$. Also, by Proposition 2 applied to $G - x$ and t, we have that $N_{G-x}(v_t) \subseteq V_t$. Thus $d(v_t) = |N(v_t)| \leq |N_{G-x}(v_t)| + 1 \leq |V_t| = 7$.

Suppose that $d(v_t) = 7$. Now $N[x] \not\subseteq N[v_t]$ since $d(x) \in \{5, 6\}$, either by Lemma 4(c) or by Lemma 4(d). But, since $d(v_t) = 7$, note that v_t is adjacent to x and to every vertex in $V_t \setminus \{v_t\}$, hence $N[x] \subseteq N[v_t]$, a contradiction. So we may assume that $d(v_t) \leq 6$, and hence $v_t x \notin E(G)$ by Lemma 4(b). Since $d(v_t) \geq 5$, there are at least four neighbors of v_t in $\{v_1, v_2, v_3, v_4, v_5\}$, say v_1, v_2, v_4, v_5. Now, since $v_t x \notin E(G)$, we have $d(x) = 5$. Thus, by Lemma 4(a), we have that $|E(G[N(x)])| \in \{9, 10\}$. If $|E(G[N(x)])| = 10$, then put

$X = (\{v_t v_6\} \cap E(G)) \cup E(G[\{v_1, v_2, v_3, v_4, v_5\}])$ and $Y = \{v_1v_2v_3, v_3v_4v_5, v_1v_4x, v_2v_5x, v_1v_5v_t, v_2v_4v_t\}$.

Note that $|X| \leq 11 \leq 12 = 2|Y|$. It is not hard to check that $V_0 = \{v_t, x\}$ is reducible using X and Y, a contradiction. So we may assume that $|E(G[N(x)])| = 9$. Note that the missing edge $e \notin E(G[N(v)])$ must be incident to a neighbor, say v_5, of v_t. Thus we may assume that $e \in \{v_3v_5, v_4v_5\}$. Put

$X = (\{v_t v_6\} \cap E(G)) \cup E(G[\{v_1, v_2, v_3, v_4, v_5\}])$ and $Y = \{v_1v_2v_3, v_1v_5x, v_3v_4x, v_1v_4v_t, v_2v_5v_t\}$.

Note that $|X| \leq 10 = 2|Y|$. It is not hard to check that $V_0 = \{v_t, x\}$ is reducible using X and Y, a contradiction.

Therefore we may assume that $|S(t)| > 1$. First, suppose that $|S(t)| \geq 3$, and let t_1, t_2, $t_3 \in S(t)$, and vertices x, y, z be the representatives of t_1, t_2, t_3, respectively.

Note that, by Proposition 2, $xy, xz, yz \notin E(G)$ and $|N(x) \cup N(y)|$, $|N(x) \cup N(z)|$, and $|N(y) \cup N(z)| \leq |V_t| = 7$. Thus, by Lemma 5, $d(x) = d(y) = d(z) = 5$, $|N(x) \cap N(y)|$=$|N(x) \cap N(z)|$=$|N(y) \cap N(z)| = 3$, and $G[N(x)], G[N(y)], G[N(z)] \simeq K_5$. Assume without loss of generality that $N(x) = \{v_1, v_2, v_3, v_4, v_5\}$. Since $|N(x) \cap N(y)| = 3$, we may assume, without loss of generality, that $N(y) = \{v_3, v_4, v_5, v_6, v_7\}$. It is not hard to check that, since $|N(x) \cap N(z)| = |N(y) \cap N(z)| = 3$ and $|N(z)| = 5$, then $N(z)$ contains exactly one vertex in $N(x) \cap N(y)$. So we may assume, without loss of generality, that $N(z) = \{v_1, v_2, v_4, v_6, v_7\}$. Let $H = G[\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}]$ and note that every pair of vertices of H is contained in at least one $N(v)$ for $v \in \{x, y, z\}$. Thus, since $G[N(v)]$ is isomorphic to K_5 for $v \in \{x, y, z\}$, H is isomorphic to K_7. Let $X = E(H)$ and note that $|X| = 21$. Put

$$\begin{aligned} Y_1 = &\ \{xv_1v_3, xv_2v_5, yv_3v_4, yv_5v_6, zv_1v_6, zv_2v_4\}; \\ Y_2 = &\ \{v_1v_2v_7, v_2v_3v_6, v_4v_6v_7, v_3v_5v_7, v_1v_4v_5\}, \end{aligned}$$

and note that if $Y = Y_1 \cup Y_2$, then $|Y| = 11$ and $|X| \leq 2|Y|$. Hence $V_0 = \{x, y, z\}$ is reducible using X and Y, a contradiction. We conclude that $|S(t)| \leq 2$.

If $t \neq r$, we let w be the representative of t, otherwise we let w be an arbitrary vertex of V_t. Let $t_1, t_2 \in S(t)$ and let x, y be the representatives of t_1, t_2, respectively. Again, by Lemma 5, we have $d(x) = d(y) = 5$, $|N(x) \cap N(y)| = 3$, and $G[N(x)], G[N(y)] \simeq K_5$. Note that t is a leaf of $(T', \mathcal{V}', r)$, where $T' = T - t_1 - t_2$ and $\mathcal{V}' = \mathcal{V} \setminus \{V_{t_1}, V_{t_2}\}$, hence $d_{G-x-y}(w) \leq 6$. Thus, we have $d_G(w) \leq 8$. Note that $w \in N(x) \cup N(y)$ and assume, without loss of generality, that $w \in N(x)$. Since $G[N(x)]$ is a complete graph, we have $N[x] \subseteq N[w]$, a contradiction to Lemma 4(d). This concludes the proof. □

3. Concluding remarks

This work has benefited greatly from [Puleo 2015]. Nevertheless, there were still gaps that we were able to explore. As we can see, the tree decomposition, specially under bounded treewidth, provides a suitable structure for problems of this nature. We believe that these techniques may be further improved by studying the behaviour of nodes from the tree decomposition with different heights.

References

Bodlaender, H. (1998). A partial k-arboretum of graphs with bounded treewidth. Theoretical Computer Science, 209(1):1–45.

Gross, J. (2014). Embeddings of graphs of fixed treewidth and bounded degree. ARS Mathematica Contemporanea, 7:379–403.

Haxell, P. E. and Kohayakawa, Y. (1998). Packing and covering triangles in tripartite graphs. Graphs and Combinatorics, 14(1):1–10.

Puleo, G. (2015). Tuza's conjecture for graphs with maximum average degree less than 7. European Journal of Combinatorics, 49:134–152.

Tuza, Z. (1981). Conjecture in: finite and infinite sets. In Proc. Colloq. Math. Soc. J. Bolyai (Eger, Hungary, 1981), volume 37, page 888.

Tuza, Z. (1990). A conjecture on triangles of graphs. Graphs and Combinatorics, 6(4):373–380.

Patrocinador Diamante

GOVERNO
DO RIO GRANDE DO NORTE

Patrocinadores Bronze

ESiG
Software & consultoria

dynavideo

CABOTELECOM

Sistema FIERN

PREFEITURA DO NATAL
A NOSSA CIDADE

PARNAMIRIM
Cuidando de você.

Apoio Financeiro

CAPES

MINISTÉRIO DA EDUCAÇÃO

BRASIL
GOVERNO FEDERAL

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

nic.br
Núcleo de Informação e Coordenação do Ponto BR

cgi.br
Comitê Gestor da Internet no Brasil