ALGORITMOS PARA O PROBLEMA DO SUBGRAFO ACÍCLICO MÁXIMO SOB RESTRIÇÕES DISJUNTIVAS

SÍLVIA MARIA SANTANA MAPA

ALGORITMOS PARA O PROBLEMA DO SUBGRAFO ACÍCLICO MÁXIMO SOB RESTRIÇÕES DISJUNTIVAS

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

ORIENTADOR: SEBASTIÁN ALBERTO URRUTIA

Belo Horizonte - Minas Gerais

Novembro de 2014

Santana Mapa, Sílvia Maria

M297a Algoritmos para o Problema do Subgrafo Acíclico Máximo sob Restrições Disjuntivas / Sílvia Maria Santana Mapa. — Belo Horizonte - Minas Gerais, 2014
xxi, 77 f. : il. ; 29cm

Tese (doutorado) — Universidade Federal de Minas Gerais
Orientador: Sebastián Alberto Urrutia

1. Análise Combinatória. 2. Subgrafo Acíclico Máximo. 3. Restrições Disjuntivas. 4. Algoritmos. I. Título.

CDU 519.6*61(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Algoritmos para o problema do subgrafo acíclico máximo sob restrições disjuntivas

SILVIA MARIA SANTANA MAPA

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. SEBASTIAN ALBERTO URRUTIA - Orientador
Departamento de Ciência da Computação - UFMG

PROF. CELSO DA CRUZ CARNEIRO RIBEIRO
Departamento de Computação - UFF

PROF. GERALDO ROBSON MATEUS
Departamento de Ciência da Computação - UFMG

PROFA. OLGA NIKOLAEVNA GOUSSEVSKAIA
Departamento de Ciência da Computação - UFMG

PROFA. ROSIANE DE FREITAS RODRIGUES
Departamento de Ciência da Computação - UFAM

Belo Horizonte, 21 de outubro de 2014.

Dedico este trabalho ao professor Sebastián Alberto Urrutia, por todo o tempo dedicado à orientação deste trabalho em nossos encontros semanais, pela paciência e pela amizade.

Agradecimentos

Agradeço imensamente à oportunidade concedida pelo projeto de doutorado interinstitucional (DINTER), realizado entre o Instituto Federal de Minas Gerais (IFMG) em parceria com a Universidade Federal de Minas Gerais (UFMG), que culminou na realização deste trabalho, e a todas as pessoas envolvidas no mesmo.

Aos meus pais, Sônia e Luiz, obrigada por me apoiarem sempre nos momentos em que mais precisei. Aos meus irmãos, Michelle e Danilo, obrigada por acreditarem sempre em mim e me darem forças para vencer esta e outras batalhas. Ao meu marido Albano, não tenho como agradecer por toda a paciência, apoio, abdicação de seus ideais em prol dos meus, enfim, por tudo o que passamos juntos nestes quatro últimos anos, bons e não tão bons momentos. Muito obrigada!

Novamente, ao professor Sebastián Alberto Urrutia, meus agradecimentos por toda a ajuda, paciência e dedicação. Costumo dizer que sou uma pessoa de sorte, por ter tido sempre orientadores de tão alta qualidade, a começar pelo professor Marcone Jamilson Freitas Souza, meu orientador de trabalho de conclusão de curso de graduação na Universidade Federal de Ouro Preto (UFOP), que me iniciou com tanta devoção na área da Pesquisa Operacional.

Gostaria de agradecer também aos colegas e amigos do LaPO, por terem me acolhido tão bem e proporcionado um ambiente de trabalho saudável, em especial ao Phillippe Samer, por ter me ajudado na fase inicial de adaptação. Ao DCC e à secretaria, pelo apoio concedido, e à CAPES, pelo apoio financeiro. Por fim, aos membros da banca, professoras Rosiane e Olga e professores Robson e Celso Ribeiro, pelas observações que engrandeceram este trabalho.

"O poema, ao ser feito, deve mudar alguma coisa, nem que seja apenas o próprio poeta. Se, ao poeta, depois de fazer o poema, resta o mesmo que antes, o poema não terá sentido."

(Ferreira Gullar)

Resumo

Neste trabalho são abordados os problemas do Subgrafo Acíclico Máximo sob Restrições Disjuntivas Negativas (SAMRDN) e Subgrafo Acíclico Máximo sob Restrições Disjuntivas Positivas (SAMRDP), ambos pertencentes à classe $\mathcal{NP}$–Difícil. Dado um certo par de entidades, para as restrições disjuntivas negativas, no máximo uma delas poderá compor uma solução viável. Já para as restrições disjuntivas positivas, dado um par de entidades, pelo menos uma delas deverá compor uma solução viável.

Os problemas SAMRDN e SAMRDP possuem como instância um grafo direcionado $G = (V, A)$ e um grafo não direcionado $\bar{G} = (A, E)$, que representa as restrições disjuntivas, positivas ou negativas, sob pares de arcos do grafo G. SAMRDN consiste em determinar um subconjunto máximo $A' \subseteq A$ para o qual o subgrafo $G' = (V, A')$ seja acíclico e A' seja um conjunto de vértices em $\bar{G}$ tal que não existam arestas entre dois de seus vértices, ou seja, um conjunto independente em $\bar{G}$. SAMRDP consiste em determinar um subconjunto máximo $A' \subseteq A$ para o qual o subgrafo $G' = (V, A')$ seja acíclico e A' seja um conjunto de vértices em $\bar{G}$ tal que toda aresta em E seja incidente em algum vértice de A', ou seja, uma cobertura por vértices em $\bar{G}$.

É provado que o problema de decisão sobre a viabilidade de SAMRDP é $\mathcal{NP}$–Completo, por uma redução a partir do problema clássico de Cobertura por Vértices. Para SAMRDN, são apresentados seis algoritmos com fator de aproximação constante igual a 1/2 para classes especiais de grafos de restrições, tal que o cálculo de um conjunto independente máximo em $\bar{G}$ seja polinomial. Dentre os seis algoritmos, três deles seguindo a abordagem denominada *late* geram soluções com no mínimo a mesma cardinalidade das obtidas com os algoritmos equivalentes, segundo a abordagem *early*.

É proposto um pré-processamento de instâncias para SAMRDN e SAMRDP. Testes experimentais sobre modelos de programação linear inteira dos problemas sugerem que, ao se realizar o pré-processamento, soluções melhores são obtidas, no mesmo tempo de execução. É também apresentado um procedimento de melhoria sobre uma solução inicial de SAMRDN, obtida heuristicamente, que é então vinculado a uma metaheurística como um método de busca local. Resultados computacionais evidenciam

que, para instâncias em que $|V| > 200$, a heurística é preferível à resolução do modelo de programação inteira por um software comercial.

Palavras-chave: Subgrafo Acíclico Máximo, Restrições Disjuntivas, Análise Combinatória, Algoritmos.

Abstract

In this work we tackle the Maximum Acyclic Subgraph problem under Negative Disjunctive Constraints (MASNDC) and the Maximum Acyclic Subgraph problem under Positive Disjunctive Constraints (MASPDC), both belonging to the $\mathcal{NP}$–Hard class. Disjunctive constraints can be negative, in which a certain pair of entities cannot be contained simultaneously in a feasible solution; or positive in that, given a pair of entities, at least one of them must compose a feasible solution.

Instances of MASNDC and MASPDC are composed by a directed graph $G = (V, A)$ and a undirected graph $\bar{G} = (A, E)$, which encodes the disjunctive positive or negative constraints over pairs of arcs of graph G. MASNDC consists in determining a maximum subset $A' \subseteq A$ for which the subgraph $G' = (V, A')$ is cycle free and A' is a set of vertices in $\bar{G}$ such that there are no edges between two of its vertices, namely an independent set at $\bar{G}$. MASPDC consists in determining a maximum subset $A' \subseteq A$ for which the subgraph $G' = (V, A')$ is cycle free and A' is a set of vertices in $\bar{G}$ such that every edge in E is incident on some vertex in A', namely a vertex cover in $\bar{G}$.

It is proved that the feasibility decision version of MASPDC is $\mathcal{NP}$–Complete by a reduction from the classic Vertex Cover problem. For MASNDC we propose six approximation algorithms with constant factor equal to 1/2 for special classes of graph $\bar{G}$ in which the maximum independent set is polynomial. Among the six algorithms, three of them following the so-called *late* approach generate solutions with at least the same cardinality of the ones obtained by equivalent algorithms, according to the *early* approach.

We propose a preprocessing of MASPDC and MASNDC instances. Computational experiments on integer linear programming models of both problems suggest that, when performing the preprocessing, solutions of larger cardinalities are obtained, in the same run time. We also proposed an improvement procedure on an heuristically obtained initial solution of MASNDC, which is then linked to a metaheuristic as a local search method. Computational results show that for instances with $|V| > 200$, the heuristic is preferable to solving the integer programming model by using a commercial

software.

Keywords: Maximum Acyclic Subgraph, Disjunctive Constraints, Combinatorial Analysis, Algorithms.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Grafos são estruturas matemáticas usadas para modelar relações entre objetos ou sistemas, e possuem aplicações em diferentes campos da ciência, como matemática, computação, engenharia, física, biologia, ciências sociais, econômicas etc. Problemas com aplicações práticas podem ser modeladas em grafos, como problemas de escalonamento de tarefas [Coffman Jr. & Graham, 1972], roteamento [Reinelt, 1985], análise de fluxo de dados [Ramachandran, 1988] etc. A seguir, serão apresentadas algumas definições básicas usadas ao longo do texto da tese. Para outras definições em teoria dos grafos, vide Bondy & Murty [2008]; Diestel [2005].

Definição 1. *Grafo simples $G = (V, E)$: consiste de um conjunto finito de vértices V e de um conjunto de arestas E, definido como um subconjunto não ordenado de $V \times V$. Os dois vértices formando uma aresta são ditos suas extremidades e a aresta é dita incidente para com os vértices. Um vértice u é dito ser adjacente a outro vértice v se o grafo contem uma aresta $\{u, v\}$. Normalmente, quando se tratar deste tipo de grafo, o adjetivo simples é omitido, estando, no entanto, subentendido.*

Definição 2. *Grafo direcionado $G = (V, A)$: consiste de um conjunto finito de vértices V e de um conjunto de arcos A, definido como um subconjunto ordenado de $V \times V$. Um arco (v_1, v_2) é um arco que sai de v_1 e chega a v_2. O vértice v_1 é dito predecessor do vértice v_2 e v_2 é sucessor de v_1, ou v_1 é o vértice inicial e v_2 é o vértice final.*

Definição 3. *Grafo completo: grafo no qual existe uma aresta entre quaisquer dois vértices distintos. Um grafo completo com n vértices é denotado por K_n.*

Definição 4. *Grafo bipartido: o conjunto de vértices pode ser particionado em dois conjuntos, X e Y, tal que cada aresta possui um extremo em X e o outro em Y. O par (X, Y) é dito uma bipartição do grafo.*

Definição 5. *Grafo bipartido completo: grafo com bipartição (X, Y), tal que cada vértice de X está ligado a todos os vértices de Y. Se $|X| = m$ e $|Y| = n$, então o grafo bipartido completo é denotado por $K_{m,n}$.*

Definição 6. *Subgrafo: um grafo H é subgrafo de G, representado por $H \subseteq G$, se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$.*

Definição 7. *Subgrafo induzido por vértices: seja V' um subconjunto não vazio de V. O subgrafo de G cujo conjunto de vértices é V' e cujo conjunto de arestas é composto pelas arestas de G tal que as duas extremidades pertençam a V', é um subgrafo induzido pelos vértices V', representado por $G[V']$.*

Definição 8. *Subgrafo induzido por arestas: seja E' um subconjunto não vazio de E. O subgrafo de G cujo conjunto de vértices é o conjunto dos extremos das arestas em E', e cujo conjunto de arestas é E', é um subgrafo induzido pelas arestas E', representado por $G[E']$.*

Definição 9. *Grau de um vértice: o grau de um vértice, $\delta(v)$, em um grafo é o número de arestas incidentes a este vértice no grafo. Se o grafo é direcionado, há que se distinguir entre o grau de entrada do vértice $\delta^+(v)$, dado pelo número de arcos que possuem v como vértice final, e o grau de saída do vértice $\delta^-(v)$, dado pelo número de arcos que possuem v como vértice inicial.*

Definição 10. *Caminho: um caminho em um grafo direcionado G é definido por uma sequência finita, não nula, $W = v_0a_1v_1a_2v_2 \ldots a_kv_k$, cujos termos se alternam entre vértices e arcos tal que, para $0 < i \leq k$, a_i tem como vértice inicial v_{i-1} e vértice final v_i. W é um caminho de v_0 a v_k se ele se inicia em v_0 e termina em v_k, e pode ser denotado por $v_0 \rightsquigarrow v_k$. Além disto, em um grafo simples, um caminho pode ser especificado simplesmente pela sequência de seus vértices $v_0v_1v_2 \ldots v_k$.*

Definição 11. *Caminho simples: cada vértice do caminho aparece exatamente uma vez.*

Definição 12. *Comprimento de um caminho: é a soma do número de arcos que o compõe.*

Definição 13. *Ciclo: caminho $C = v_0a_1v_1a_2v_2 \ldots a_kv_ka_{k+1}v_0$, denominado caminho fechado, no qual os vértices de origem e destino são os mesmos. O tamanho de um ciclo é o número de arcos (ou vértices) que o compõe. Um k-ciclo é um ciclo de tamanho k.*

Definição 14. *Árvore: grafo não direcionado em que quaisquer dois vértices estão conectados por exatamente um único caminho simples.*

Definição 15. *Árvore binária: pode ser definida como uma árvore enraizada em que cada vértice têm no máximo dois filhos, representados pelos vértices adjacentes ao vértice de origem, denominado pai. Um vértice folha é um vértice que não possui filhos. O nível de um vértice é a sua distância à raiz.*

Definição 16. *Componente fortemente conexa: dado um grafo direcionado, dois vértices u e v são ditos conectados se existe um caminho de u para v e de v para u em G. O conjunto de vértices conectados definem uma partição de V em subconjuntos $V_1, V_2, \ldots, V_\omega$, tal que dois vértices u e v estão conectados se e somente se ambos pertencem ao mesmo subconjunto V_i. Os subgrafos induzidos $G[V_1], G[V_2], \ldots G[V_\omega]$ são chamados componentes fortemente conexas de G.*

Definição 17. *Grafo fortemente conexo: um grafo direcionado G é dito fortemente conexo se possui uma única componente fortemente conexa, ou seja, se contém um caminho direcionado de u para v e de v para u para cada par de vértices u, v.*

Definição 18. *Clique: dado um grafo $G = (V, E)$, uma clique é um subconjunto $V' \subseteq V$ tal que toda dupla de vértices em V' está ligada por uma aresta em E.*

Definição 19. *Cobertura por vértices: dado um grafo $G = (V, E)$, um subconjunto de vértices $V' \subseteq V$ será uma cobertura se toda aresta de E tem um extremo em um vértice de V'.*

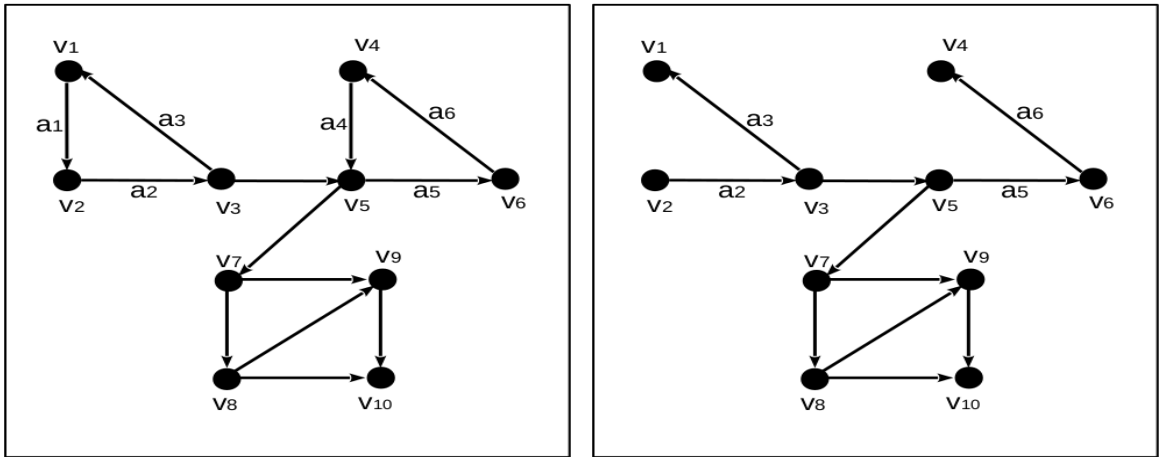
Definição 20. *Conjunto independente: dado um grafo $G = (V, E)$, tem-se um conjunto independente $I \subseteq V$ se nenhum vértice de I é adjacente a outro vértice de I ou, para todo $u, v \in I$, a aresta $\{u, v\} \notin E$. O complemento de um conjunto independente é uma cobertura por vértices.*

1.1 Definição dos problemas

Considerando um grafo direcionado, este pode conter ciclos, os quais podem ser indesejáveis ou não permitidos em algumas aplicações práticas. Pode-se citar como exemplo o trabalho de Jackson et al. [2008], com aplicação na biologia, no qual é proposta a geração de um mapa genético consensual. Um mapa genético pode ser definido como uma ordenação de marcadores genéticos calculados a partir de uma população de linhagem conhecida. Embora, tradicionalmente, um mapa seja gerado a partir de uma

única população de cada espécie, existem também mapas criados a partir de múltiplas populações. Em face destes mapas, surgiu a necessidade de se encontrar um mapa consensual, ou seja, um mapa que combina as informações de vários mapas genéticos de ordens parciais (modelados por grafos acíclicos direcionados) e possivelmente inconsistentes. Sendo assim, os autores modelaram cada mapa de entrada como uma ordem parcial e formularam o problema do mapa genético consensual como encontrar uma ordem parcial mediana dos marcadores genéticos. A solução do problema envolve então encontrar o máximo subgrafo acíclico de um grafo direcionado ponderado, em que os vértices do grafo correspondem aos marcadores genéticos em um mapa e os caminhos correspondem às informações de ordenação.

O problema do Subgrafo Acíclico Máximo (do inglês, *Maximum Acyclic Subgraph*), ou simplesmente *SAM*, consiste em, dado um grafo $G = (V, A)$, determinar um subconjunto máximo $A' \subseteq A$ para o qual o subgrafo $G' = (V, A')$ seja acíclico. A Figura 1.1(a) apresenta uma instância para o *SAM* e a Figura 1.1(b) apresenta uma solução para a instância dada. Note que para esta instância, o grafo G possui dois ciclos disjuntos, $c_1 = (v_1 a_1 v_2 a_2 v_3 a_3 v_1)$ e $c_2 = (v_4 a_4 v_5 a_5 v_6 a_6 v_4)$, portanto pelo menos dois de seus arcos, cada um pertencente a um dos ciclos, não poderão compor o conjunto A'.



(a) Grafo $G = (V, A)$. (b) Grafo $G' = (V, A')$.

Figura 1.1. Exemplo de solução para SAM.

Além de problemas clássicos que podem ser formulados em grafos, pode-se citar generalizações de problemas que possuem certas restrições. Chamam-se restrições disjuntivas binárias aquelas nas quais um cenário ou outro pode ocorrer. Restrições disjuntivas binárias podem ser encontradas sob várias classificações, dentre elas:

1. Restrições Disjuntivas Negativas ou de Conflito: são aquelas nas quais um certo par de entidades não podem compor simultaneamente uma solução viável.

2. Restrições Disjuntivas Positivas ou de Coação: para cada par de entidades, pelo menos uma delas deve compor uma solução viável.

Vários problemas clássicos em teoria dos grafos formulados sob restrições disjuntivas, positivas ou negativas, como os problemas de Árvore Geradora Mínima, Emparelhamento Máximo, Caminho Mínimo e Fluxo Máximo, encontram-se disponíveis na literatura [Kann, 1993; Darmann et al., 2009, 2011; Pferschy & Schauer, 2013; Zhang et al., 2011].

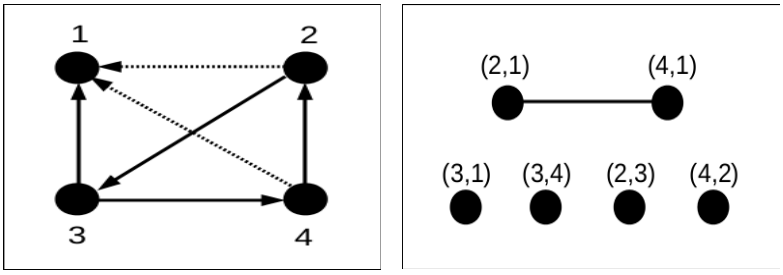
No presente trabalho, serão abordados os problemas do Subgrafo Acíclico Máximo sob Restrições Disjuntivas Positivas (SAMRDP) e Subgrafo Acíclico Máximo sob Restrições Disjuntivas Negativas (SAMRDN), nos quais as restrições disjuntivas incidem sobre pares de arcos do grafo $G = (V, A)$. É conveniente representar as restrições disjuntivas sob um conjunto de arcos em termos de um grafo não direcionado $\bar{G} = (A, E)$, denominado grafo de conflitos para as restrições disjuntivas negativas ou grafo de coação para as restrições disjuntivas positivas. Neste grafo, os vértices correspondem aos arcos do grafo G e as arestas representam as restrições disjuntivas. Seguem as definições formais dos dois problemas.

Definição 21. *SAMRDP: seja $G = (V, A)$ um grafo direcionado e $\bar{G} = (A, E)$ o grafo de coação associado. O problema do Subgrafo Acíclico Máximo sob Restrições Disjuntivas Positivas consiste em determinar um subconjunto $A' \subseteq A$ de cardinalidade máxima, no qual o subgrafo $G' = (V, A')$ seja acíclico e A' seja uma cobertura por vértices em $\bar{G}$.*

Definição 22. *SAMRDN: seja $G = (V, A)$ um grafo direcionado e $\bar{G} = (A, E)$ o grafo de conflitos associado. O problema do Subgrafo Acíclico Máximo sob Restrições Disjuntivas Negativas consiste em determinar um subconjunto $A' \subseteq A$ de cardinalidade máxima, no qual o subgrafo $G' = (V, A')$ seja acíclico e A' seja um conjunto independente em $\bar{G}$.*

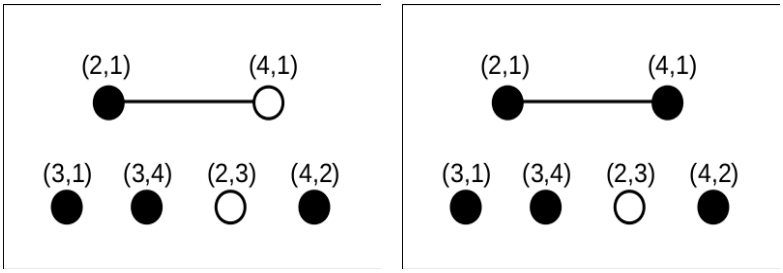
A Figura 1.2 apresenta uma instância para exemplificar a aplicação de SAMRDN e SAMRDP. No grafo G (Figura 1.2(a)), existe uma restrições disjuntiva, seja negativa ou positiva, sobre o par de arcos $\{(2,1),(4,1)\}$, por isto estes arcos estão representados por tracejados. Note que estes arcos são vértices no grafo $\bar{G}$ (Figura 1.2(b)), unidos por uma aresta, que representa a restrição disjuntiva. A Figura 1.3 apresenta os grafos $\bar{G}$ e os conjuntos de vértices A', destacados na cor preta, selecionados para perfazerem um conjunto independente para SAMRDN (Figura 1.3(a)), ou uma cobertura por vértices para SAMRDP (Figura 1.3(b)), tal que A' seja máximo e induza um subgrafo acíclico

em G. Por isto, o vértice $(2,3) \notin A'$, para ambos os problemas, desfazendo o ciclo em G. A Figura 1.4 apresenta as soluções para os problemas, dadas pelos grafos $G' = (V, A')$.



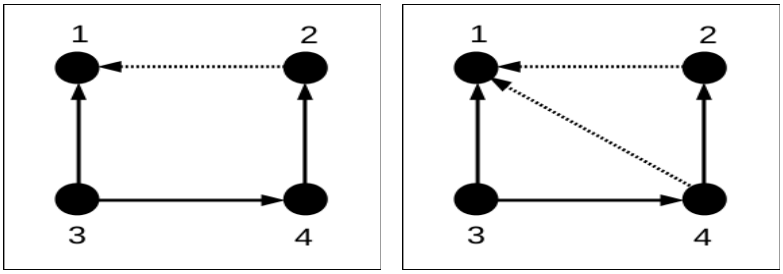
(a) Grafo $G = (V, A)$. (b) Grafo $\bar{G} = (A, E)$.

Figura 1.2. Instância para SAMRDN e SAMRDP.



(a) A' para SAMRDN. (b) A' para SAMRDP.

Figura 1.3. Conjuntos de vértices A' selecionados em $\bar{G}$.



(a) $G' = (V, A')$ para SAMRDN. (b) $G' = (V, A')$ para SAMRDP.

Figura 1.4. Soluções dadas pelos grafos $G' = (V, A')$.

É comum a aplicação prática do *SAM* em projetos nos quais existem preferências de precedência entre tarefas. Estas aplicações podem ser formuladas como o problema do *SAM* com a inclusão de restrições disjuntivas entre pares de arcos. Supondo que os vértices do grafo representam as tarefas e os arcos preferências de precedência, deseja-se atribuir uma ordenação parcial às tarefas de forma a cumprir com o maior número

possível de preferências. Restrições disjuntivas positivas podem constituir uma forma de forçar preferências consideradas importantes no projeto.

Com o desenvolvimento desta pesquisa, foram introduzidos na literatura os problemas do Subgrafo Acíclico Máximo sob Restrições Disjuntivas Negativas e Subgrafo Acíclico Máximo sob Restrições Disjuntivas Positivas, modelados por grafos de conflitos e coação, respectivamente [Mapa & Urrutia, 2015]. Até o presente momento, não se conhecem disponíveis na literatura aplicações diretas para SAMRDN ou SAMRDP, mas tanto o *SAM* quanto as restrições disjuntivas são importantes na prática. O presente trabalho visa focar principalmente na solução do problema teórico, sem mais discursar sobre possíveis aplicações.

1.2 Complexidade computacional

Os três problemas apresentados por Darmann et al. [2011], caminhos mais curtos, árvores geradoras mínimas e emparelhamentos máximos, são resolvidos em tempo polinomial em suas formas clássicas e se tornam $\mathcal{NP}$–Difíceis quando submetidos às restrições disjuntivas. Quanto à sua complexidade, o problema do Subgrafo Acíclico Máximo é $\mathcal{NP}$–Difícil [Karp, 1972; Garey & Johnson, 1979]. Como o problema do Subgrafo Acíclico Máximo sob restrições disjuntivas é uma generalização do problema do *SAM* (os problemas são idênticos quando o conjunto de restrições é vazio), SAMRDN e SAMRDP também pertencem à classe $\mathcal{NP}$–Difícil.

Um problema de decisão é uma pergunta sobre um sistema formal e um conjunto infinito de entradas cuja resposta é binária: *sim* ou *não*. Um problema de decisão D se transforma polinomialmente em um outro problema de decisão D' se sempre é possível, a partir de uma instância de D, I_D, construir uma instância de D', $I_{D'}$, em tempo polinomial em relação ao tamanho da instância de D, tal que I_D tem solução *sim* se e somente se, $I_{D'}$ tem solução *sim*. Se D se transforma (ou se reduz) polinomialmente a D' então, se existe um algoritmo polinomial para resolver D', também existirá um algoritmo polinomial para resolver D [Karp, 1972] .

Um problema de decisão $D \in \mathcal{NP}$ é $\mathcal{NP}$–Completo se todo problema em $\mathcal{NP}$ puder ser reduzido a D. Um problema H é $\mathcal{NP}$–Difícil se existe um problema $\mathcal{NP}$–Completo L que se reduz polinomialmente a H. Segue a definição dos principais problemas $\mathcal{NP}$–Completos referenciados ao longo deste trabalho.

Definição 23. *Satisfabilidade ou SAT (Satisfiability): dado um conjunto C de cláusulas, cada uma formada pela disjunção de literais do conjunto de variáveis U =*

$\{u_1, u_2, \ldots, u_n\}$, a pergunta que se deseja responder é: existe uma atribuição de valores binários às variáveis de forma que todas as cláusulas sejam satisfeitas?

Definição 24. *3-Satisfabilidade ou 3-SAT (3-Satisfiability): dado um conjunto C de cláusulas, cada uma formada pela disjunção de três literais do conjunto de variáveis $U = \{u_1, u_2, \ldots, u_n\}$, a pergunta que se deseja responder é: existe uma atribuição de valores binários às variáveis de forma que todas as cláusulas sejam satisfeitas?*

Os problemas de decisão SAT e 3-SAT foram provados pertencentes à classe $\mathcal{NP}$-Completo por Cook [1971].

Definição 25. *Clique: dado um grafo $G = (V, E)$ e um inteiro positivo $k \leq |V|$ como uma instância, a pergunta que se deseja responder é: existe uma clique em G de tamanho maior que ou igual a k, isto é, um subconjunto $V' \subseteq V$ com $|V'| \geq k$, tal que toda dupla de vértices em V' está ligada por uma aresta em E?*

Definição 26. *Cobertura por Vértices (CV): dado um grafo $G = (V, E)$ e um inteiro positivo $k \leq |V|$ como uma instância, a pergunta que se deseja responder é: existe uma cobertura por vértices em G de tamanho menor ou igual a k, isto é, um subconjunto $V' \subseteq V$ tal que $|V'| \leq k$ e, para cada aresta $\{u, v\} \in E$, $u \in V'$ ou $v \in V'$?*

Definição 27. *Conjunto Independente (CI): dado um grafo $G = (V, E)$ e um inteiro positivo $k \leq |V|$ como uma instância, a pergunta que se deseja responder é: existe um conjunto independente em G de tamanho maior que ou igual a k, isto é, um subconjunto $V' \subseteq V$ tal que $|V'| \geq k$ e, para todo $u, v \in V'$, a aresta $\{u, v\} \notin E$?*

Karp [1972] provou, por meio de uma sequência de transformações (a partir do problema de Satisfabilidade, transformado em Clique que é então reduzido a Cobertura por Vértices), que a versão de decisão do problema de Cobertura por Vértices é $\mathcal{NP}$-Completo. Consequentemente, provou que o problema de decisão de Conjunto Independente é também $\mathcal{NP}$–Completo, uma vez conhecida a relação: V' é uma cobertura por vértices em G, se e somente se, $V - V'$ é um conjunto independente de G.

Definição 28. *Subgrafo Direcionado Acíclico: dado um grafo direcionado $G = (V, A)$ e um inteiro positivo $k \leq |A|$ como uma instância, a pergunta que se deseja responder é: existe um subgrafo acíclico em G de tamanho maior ou igual a k, isto é, um subconjunto $A' \subseteq A$ tal que $|A'| \geq k$, e que o subgrafo induzido $G[A']$ seja acíclico?*

Definição 29. *Conjunto de Arcos de Retorno: dado um grafo direcionado $G = (V, A)$ e um inteiro positivo $k \leq |A|$ como uma instância, a pergunta que se deseja responder*

é: existe um subconjunto $A' \subseteq A$ tal que $|A'| \leq k$, e que A' contenha ao menos um arco de cada ciclo em G?

Karp [1972] provou que o problema de Conjunto de Arcos de Retorno (do inglês *Feedback Arc Set Problem*) é $\mathcal{NP}$-Completo fazendo uma transformação a partir do problema de Cobertura por Vértices. Tem-se a relação: A' é um conjunto de arcos de retorno de G, se e somente se, $G[A \backslash A']$ é um subgrafo acíclico. Assim sendo, o problema do Subgrafo Direcionado Acíclico é equivalente ao problema do Conjunto de Arcos de Retorno.

Logo, o problema de encontrar um Subgrafo Direcionado Acíclico também é provado $\mathcal{NP}$–Completo. O problema correspondente para grafos não direcionados é resolvido trivialmente em tempo polinomial [Karp, 1972].

1.3 Abordagens para problemas $\mathcal{NP}$–Difíceis

Existem pelo menos três abordagens para tratar problemas da classe $\mathcal{NP}$–Difícil. Primeiramente, se as instâncias são pequenas, algoritmos exatos exponenciais, que buscam resolver na otimalidade os problemas, podem ser satisfatórios.

Outra abordagem é encontrar soluções próximas ao ótimo, em tempo polinomial, por meio de técnicas heurísticas ou algoritmos aproximativos. Algoritmos aproximativos são aqueles que retornam uma solução para problemas de otimização cujo valor está a um fator de aproximação conhecido da solução ótima. Se este fator é constante, ou seja, independe do tamanho da instância, tem-se um algoritmo aproximativo que produz uma solução cujo valor está a um certo fator constante da solução ótima.

Heurísticas são técnicas para resolução de problemas que visam encontrar uma solução viável, se possível de boa qualidade, em um tempo de processamento razoável, não garantindo a sua otimalidade e nem sequer um fator de aproximação.

1.4 Objetivos do trabalho

Deseja-se investigar neste trabalho resultados teóricos e experimentais relacionados aos problemas do Subgrafo Acíclico Máximo sob Restrições Disjuntivas Positivas e Negativas. Sendo assim, os seguintes objetivos são propostos:

- Investigar a complexidade computacional dos problemas;
- Propor algoritmos heurísticos e aproximativos para os problemas;

- Fornecer resultados computacionais para os algoritmos propostos;
- Estudar as propriedades dos problemas SAMRDN e SAMRDP, a fim de facilitar a resolução de ambos os problemas;
- Desenvolver modelos de programação linear inteira para SAMRDN e SAMRDP;

É importante atingir estes objetivos para que sejam fornecidos resultados teóricos e experimentais para o problema de Subgrafo Acíclico Máximo sob Restrições Disjuntivas, e que estes sejam difundidos na literatura, incentivando novos trabalhos nesta linha de pesquisa e futuras aplicações práticas para o mesmo. Como consequência deste trabalho foram publicados os seguintes artigos:

- Mapa, S.M.S.; Urrutia, S. *A 1/2-approximation algorithm for the maximum acyclic subgraph problem under negative disjunctive constraints.* APEX 2013: International Workshop on Approximation, Parameterized and Exact Algorithms. Satellite Workshop of ICALP 2013. Riga, Letônia, 06 a 12 de Julho de 2013.
- Mapa, S.M.S.; Urrutia, S. *Um algoritmo (1/2)-aproximativo para o problema do Subgrafo Acíclico Máximo sob restrições disjuntivas negativas.* XLV SBPO – Simpósio Brasileiro de Pesquisa Operacional. Natal, Rio Grande do Norte, 16 a 19 de Setembro de 2013.
- Mapa, S.M.S.; Urrutia, S. On the Maximum Acyclic Subgraph Problem under Disjunctive Constraints. *Information Processing Letters* (2014), http://dx.doi.org/10.1016/j.ipl.2014.07.013

1.5 Organização do trabalho

O restante deste trabalho está organizado da seguinte forma: no próximo capítulo é feita uma revisão da literatura relevante; no Capítulo 3 o problema SAMRDP é formalmente introduzido e é fornecida uma prova de $\mathcal{NP}$–Completude para o problema de decisão sobre sua viabilidade; no Capítulo 4 o problema SAMRDN é formalmente introduzido e são apresentados seis algoritmos (1/2)-aproximativos para casos especiais, mantendo-se o fator de aproximação de algoritmos clássicos para tratar o *SAM*; no Capítulo 5 são introduzidas propriedades de SAMRDP e SAMRDN que permitem a redução do tamanho da instância dos problemas, no Capítulo 6 é tratada uma abordagem heurística para o SAMRDN atrelado a um procedimento de melhoria. Conclui-se o trabalho no Capítulo 7.

Capítulo 2

Trabalhos Relacionados

Esta seção apresenta problemas clássicos combinatoriais e em teoria dos grafos tratados sob restrições disjuntivas positivas e negativas, resolvidos seja por meio de técnicas exatas, heurísticas ou algoritmos aproximativos, além de problemas equivalentes ao SAM, como o *Linear Ordering Problem* e o *Minimum Feedback Arc Set Problem*.

Generalizações de problemas de empacotamento (*bin packing*) [Jansen & Öhring, 1997] e escalonamento (*scheduling*) [Bodlaender et al., 1994], foram estudados sob restrições disjuntivas negativas. Para o primeiro tipo de problema, as restrições de conflito denotam itens que devem ser colocados em recipientes diferentes, e para o segundo problema, tarefas incompatíveis que devem ser executadas em máquinas distintas ou em intervalos de tempos diferentes. No trabalho de Bodlaender et al. [1994] se busca uma atribuição de tarefas sem conflitos atendendo a um limite predeterminado de tempo. Os autores ainda consideram a complexidade do problema de otimização, em que se deseja minimizar o tempo final em que se completa a execução das tarefas. São descritos resultados sobre a aproximabilidade e algoritmos de aproximação em tempo polinomial para casos particulares. Baker & Jr. [1996] estudaram um problema similar (*mutual exclusion scheduling*), no qual tarefas em conflito devem ser executadas em diferentes unidades de tempo.

Considerando o problema de empacotamento, Jansen & Öhring [1997] propõem algoritmos aproximativos para casos especiais de grafos de conflito, baseados em uma fase em que o problema de coloração de vértices é resolvido em tempo polinomial e uma fase de aproximação do *bin-packing*. Jansen [1999] estendeu o trabalho para uma classe maior de grafos de conflitos, incluindo árvores e grafos planares em que, partindo-se de um algoritmo clássico para o *bin-packing*, o algoritmo aproximativo apresenta tempo assintótico polinomial. Gendreau et al. [2004] descrevem seis heurísticas sem qualquer restrição quanto ao grafo de conflitos para o problema do *bin-packing*. Os autores

usaram uma relaxação combinatória em que a cardinalidade de qualquer clique no grafo de conflitos fornece um limite inferior para a solução do problema inteiro.

Trabalhos que investigam o problema clássico da Mochila (*Knapsack Problem*) com restrições disjuntivas também são encontrados na literatura. Yamada et al. [2002] propuseram o problema da mochila sob restrições disjuntivas negativas, em que alguns itens são incompatíveis com outros. Os autores abordaram o problema por meio de uma heurística gulosa e um método de busca local, que fornecem um limite superior ao problema, uma Relaxação Lagrangeana que fornece limites inferiores, e um algoritmo baseado em *branch and bound* que fornece soluções ótimas em tempo razoável. Em um trabalho recente, de Queiroz & Miyazawa [2012] apresentaram uma versão bidimensional do problema, abordando-o por uma metaheurística e uma formulação em programação inteira.

Versões com restrições disjuntivas de problemas clássicos em teoria dos grafos tem sido recentemente estudados, tais como problemas de caminhos mais curtos, árvores geradoras mínimas e emparelhamentos máximos [Darmann et al., 2011]. Estes autores analisaram os problemas sob restrições disjuntivas negativas e positivas, considerando dois tipos de grafos de conflito $\bar{G}$, então denominados: 2-ladder e 3-ladder. No primeiro tipo, todos os componentes conexos de $\bar{G}$ são caminhos de tamanho igual a um. No segundo tipo, todos os componentes conexos de $\bar{G}$ são caminhos de tamanho igual a dois. Em um trabalho anterior, Darmann et al. [2009] investigaram a complexidade do problema de árvores geradoras sob restrições de conflito.

Kann [1993] apresenta um problema de caminho mínimo formulado em grafos com pares de vértices proibidos, cujo objetivo é minimizar a quantidade de arestas no caminho, e para o qual é estabelecida a inaproximabilidade quando submetido às restrições de conflito. Já Darmann et al. [2011] mostram que também o problema sob restrições de coação para caminhos mais curtos é fortemente $\mathcal{NP}$–Difícil e inaproximável, mesmo para grafos não valorados ou quando as restrições correspondem a um grafo de disjunções cujas componentes conexas são caminhos de comprimento unitário.

O recente trabalho de Pferschy & Schauer [2013] apresenta o problema de fluxo máximo sob restrições disjuntivas binárias positivas (denominado *MFFG*) e negativas (denominado *MFCG*). As restrições incidem sobre arcos capacitados de um grafo direcionado conexo e o objetivo é maximizar o fluxo total entre os vértices de origem e destino, não excedendo a capacidade de cada arco, e sujeito às restrições disjuntivas binárias. Para *MFCG*, dado um certo par de arcos, estes não podem ser usados simultaneamente para enviar fluxo em uma solução viável. Já em *MFFG*, dado um certo par de arcos, deverá ser passado fluxo em pelo menos um destes arcos para se ter uma solução viável. Para este último problema, os autores permitem a formação de ciclos

em partes do grafo para que as restrições disjuntivas positivas sejam satisfeitas. Ambas versões são provadas fortemente $\mathcal{NP}$–difíceis, mesmo em instâncias com estrutura simples e grafo de disjunções com arestas isoladas para o *MFCG*, ou quando os valores de fluxo devem ser inteiros para *MFFG*. Como implicação, não poderá existir, para ambos os problemas, um algoritmo aproximativo de tempo polinomial.

Problemas em árvores com restrições sobre o grau de vértices, diâmetro, capacidade, número de arestas, entre outras, têm sido extensivamente estudados [Magnanti & Wolsey, 1995]. Porém, generalizações destes problemas sob restrições disjuntivas foram apresentadas recentemente [Darmann et al., 2009, 2011; Zhang et al., 2011]. Além de definirem o problema de árvore geradora mínima sob restrições disjuntivas positivas e negativas, Darmann et al. [2009] estabelecem sua complexidade teórica e dificuldade de aproximação. Verifica-se que o caso particular de ambos problemas, seja sob restrições disjuntivas positivas ou negativas, em que o grafo de disjunções consiste de arestas isoladas, pode ser resolvido em tempo polinomial. Entretanto, já no caso em que o grafo de conflitos consiste de caminhos simples de comprimento dois ou mais, os problemas são fortemente $\mathcal{NP}$–Difíceis, estabelecendo também a inaproximabilidade da versão negativa por um fator constante, mesmo quando com custos unitários nas arestas.

No trabalho de Zhang et al. [2011], os autores apresentam resultados teóricos e experimentais para o problema da árvore geradora mínima com restrições de conflito. O trabalho inclui casos especiais em que o problema pode ser resolvido em tempo polinomial, além de testes de viabilidade relacionados a problemas de conjunto independente e de cobertura por vértices. São apresentadas formulações em programação inteira e um modelo de relaxação Lagrangeana, além de heurísticas.

O trabalho de Darmann et al. [2011] apresenta também o problema de emparelhamento sob restrições disjuntivas positivas e negativas. Demonstra-se que ambos os problemas são fortemente $\mathcal{NP}$–Difíceis, ainda para grafos de disjunções com arestas isoladas. Um algoritmo (1/2)–aproximativo direto é fornecido para a versão negativa do problema.

2.1 SAM e problemas equivalentes

O *Linear Ordering Problem*, ou simplesmente LOP, é um problema combinatorial que pode ser modelado como um problema em grafos. Os autores Chaovalitwongse et al. [2011] definiram o LOP como: considere um conjunto N de n objetos e uma permutação $\pi : N \to N$. Cada permutação $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ corresponde a uma ordenação linear dos objetos. Seja e_{ij}, com $i, j = 1, 2, \ldots, n$, o custo de se ter i antes de j na

ordenação, e E uma matriz quadrada $n \times n$ de custos. O objetivo do LOP é encontrar uma permutação π que maximiza a soma dos elementos acima da diagonal de uma matriz A, cuja entrada a_{ij} é o resultado da permutação π das linhas e colunas da matriz E. Matematicamente, tem-se:

$$Z(\pi) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} e_{\pi(i)\pi(j)}$$

O problema do Subgrafo Acíclico Máximo é equivalente ao *Linear Ordering Problem* (LOP). Para o *SAM*, dado um grafo direcionado e considerando cada arco de peso igual à unidade, deseja-se encontrar um subgrafo de máximo peso que não contenha ciclos. A solução para este problema gera uma ordenação linear dos vértices no qual todos os arcos do subgrafo obtido são arcos diretos. Arcos diretos são aqueles que, dada uma ordenação dos vértices do grafo, o vértice predecessor possui um índice menor que o vértice sucessor.

A complexidade do LOP pode ser facilmente provada como $\mathcal{NP}$–Difícil transformando-o no problema equivalente *Minimum Feedback Arc Set Problem* [Chaovalitwongse et al., 2011]. Ramachandran [1988] apresentou um algoritmo de tempo polinomial ao MFASP, para casos especiais de grafos (*reducible flow graphs*) com pesos nas arestas, aplicados na modelagem da estrutura de controle de programas de computador, usado extensivamente para otimização de código e análise de fluxo de dados. Na sequência, serão apresentadas abordagens para o *SAM* e problemas relacionados.

2.1.1 Abordagens exatas

No trabalho de Grötschel et al. [1985] os autores investigaram o *SAM* do ponto de vista poliedral e determinaram várias classes de facetas para o politopo do problema. Concluíram que, ao separar a faceta que define desigualdades de ciclos de tamanho igual a dois, o problema pode ser resolvido em tempo polinomial para uma classe de grafos que os autores denominam *weakly acyclic digraphs*, classe esta que inclui grafos planares.

Abordagens exatas para o LOP constam na literatura usando técnicas tais como *Cutting Plane Algorithms* [Reinelt, 1985] e *Interior Point Method* [Mitchell, 1997], trabalho este último no qual o autor integrou o método de pontos interiores com o método de planos de corte. A pesquisa de Reinelt [1985] resultou em facetas induzidas por subgrafos e inequações válidas que melhoram a performance de algoritmos *branch-and-cut*.

2.1.2 Abordagens heurísticas

Métodos heurísticos são capazes de encontrar soluções viáveis para o LOP: *Tabu Search* [Laguna et al., 1999], *Greedy Randomized Adaptive Search Procedure* [Chaovalitwongse et al., 2011], *Variable Neighborhood Search* [Garcia et al., 2006], *Genetic Search* [Schiavinotto & Stützle, 2004] e *Simulated Annealing* [Martí et al., 2012].

Em Laguna et al. [1999], os autores desenvolveram um algoritmo baseado em *Tabu Search* analisando algumas técnicas de intensificação e diversificação e, por meio de extensivos experimentos computacionais, obtiveram resultados para o LOP melhores que os das heurísticas já reportadas na literatura até aquele momento.

Em Chaovalitwongse et al. [2011] os autores propuseram um algoritmo eficiente baseado em *Greedy Randomized Adaptive Search Procedure* para o LOP, fazendo o uso do método *Path Relinking*, introduzida por Glover & Laguna [1997], que integra técnicas de intensificação e diversificação, e propondo um novo procedimento de busca local baseado em programação dinâmica. O algoritmo produziu soluções de boa qualidade, sendo que para as 49 instâncias reais testadas foi resolvido na otimalidade e para as outras 30 instâncias maiores geradas randomicamente obteve-se um *gap* máximo menor que 0.05%.

Martí et al. [2012] fizeram um estudo comparativo de 24 técnicas heurísticas e metaheurísticas aplicadas ao LOP, objetivando encontrar soluções próximas à otimalidade. Segundo estes autores, os algoritmos mais eficientes para abordar o LOP foram, nesta sequência: *Memetic Algorithm* [Goldberg, 1989], *Iterated Local Search* [Mladenović & Hansen, 1997; Lourenço et al., 2002] e *Tabu Search* [Glover & Laguna, 1997].

2.1.3 Algoritmos aproximativos

Berger & Shor [1997] apresentaram em seu trabalho um algoritmo aproximativo polinomial para grafos sem ciclos de tamanho dois e então estenderam os resultados para quaisquer grafos. Os autores introduziram um algoritmo aproximativo polinomial ($O(|V| + |A|)$) alternativo, baseado em *discrepancy-based technique*, para se obter a metade dos arcos de um grafo em *SAM*. Além disto, descreveram um algoritmo randomizado de tempo polinomial ($O(|V| + |A|)$) capaz de alcançar limites melhores para o *SAM*. Dado um grafo $G = (V, A)$ e o conjunto $A' \subseteq A$ tal que o subgrafo induzido por A' seja acíclico, o algoritmo encontra $|A'| \geq |A|/2 + \Omega(\sum_{v \in V} \sqrt{deg(v)})$, em que $deg(v)$ é o grau do vértice v. O algoritmo randomizado é transformado em um algoritmo determinístico de complexidade $O(|V| \times |A|)$ usando-se o método de probabilidades condicionais, mantendo-se os mesmos limites para $|A'|$.

Arora et al. [2002] descrevem um algoritmo aproximativo de tempo polinomial para o *SAM* considerando a classe de grafos *a-densos*. Segundo os autores, um grafo é *a-denso* se o número de arcos $m \geq an^2$, sendo n o número de vértices. Os autores definem o *SAM* como, dado um grafo direcionado G e uma permutação π dos vértices de G, define-se o custo $c(\pi)$ da permutação como o número de arcos (u, v) tal que $\pi(u) < \pi(v)$. O problema de se achar a permutação de máximo custo é exatamente o problema do *SAM*. Os autores concluem que, se o grafo G é *a-denso*, então a cardinalidade, em termos do número de arcos, do Subgrafo Acíclico Máximo é $\Omega(n^2)$.

Em Newman [2000], o autor mostra que o *SAM* não pode ser aproximado por um fator constante melhor que $(1/2)$ por meio de relaxações poliedrais. O autor estudou a classe de grafos na qual o grau máximo de um vértice é três e mostrou que, para qualquer $\epsilon > 0$, se existe um algoritmo $(17/18 + \epsilon)$–aproximativo para o problema do Subgrafo Acíclico Máximo para esta classe de grafos, então existirá um $\delta > 0$ tal que existirá um algoritmo $(1/2 + \delta)$–aproximativo para o *SAM* para grafos gerais. Em Newman [2001], o mesmo autor desenvolveu um algoritmo $(11/12)$–aproximativo para o *SAM* considerando esta mesma classe especial de grafos, melhorando o resultado obtido no trabalho anterior, cujo fator de aproximação era $(8/9)$.

Segundo Guruswami et al. [2008, 2011], assumindo a conjectura de Khot [Khot, 2002], *Unique Games Conjecture*, obter um algoritmo $(1/2 + \epsilon)$–aproximativo, para qualquer $\epsilon > 0$, para o *SAM* não é possível, assumindo $P \neq NP$. Guruswami et al. [2011] provam que qualquer problema sob restrições de ordenação (então denominado *Ordering Constraint Satisfaction Problems*, o qual inclui o *SAM*), tal que as restrições incidam sob um número limitado de variáveis (no caso do *SAM* sob duas variáveis: vértice u deve preceder vértice v), não pode ser aproximado por um algoritmo de tempo polinomial essencialmente melhor do que uma partição aleatória.

Unique Games Conjecture indica o quão complexo é encontrar de forma eficiente uma solução ou mesmo uma boa aproximação para alguns problemas de otimização. Khot e seus colaboradores [Khot, 2002; Khot et al., 2007] demonstraram que a complexidade de *Unique Games* implica em uma caracterização exata dos melhores fatores de aproximação possíveis para uma variedade de problemas de otimização $\mathcal{NP}$–Difíceis. Esta descoberta transformou o *Unique Games* em um grande problema em aberto em teoria da computação e rendeu o prêmio *Rolf Nevanlinna Prize 2014* a Subhash Khot.

Independentemente da validade da *Unique Games Conjecture*, Newman [2000] demonstrou que o *SAM* é $\mathcal{NP}$-inaproximável em $(65/66 + \epsilon)$, para qualquer $\epsilon > 0$, baseado em uma redução a partir de 3-SAT. Em um trabalho recente, Austrin et al. [2013] mostram que *SAM* é $(14/15 + \epsilon)$-inaproximável, melhorando o resultado obtido em Newman [2000]. Os autores concluem relatando que a caracterização feita em seu

trabalho não está completa e que vários problemas interessantes ainda estão em aberto, como por exemplo, se SAM é $(1/2 + \epsilon)$–inaproximável, para qualquer $\epsilon > 0$.

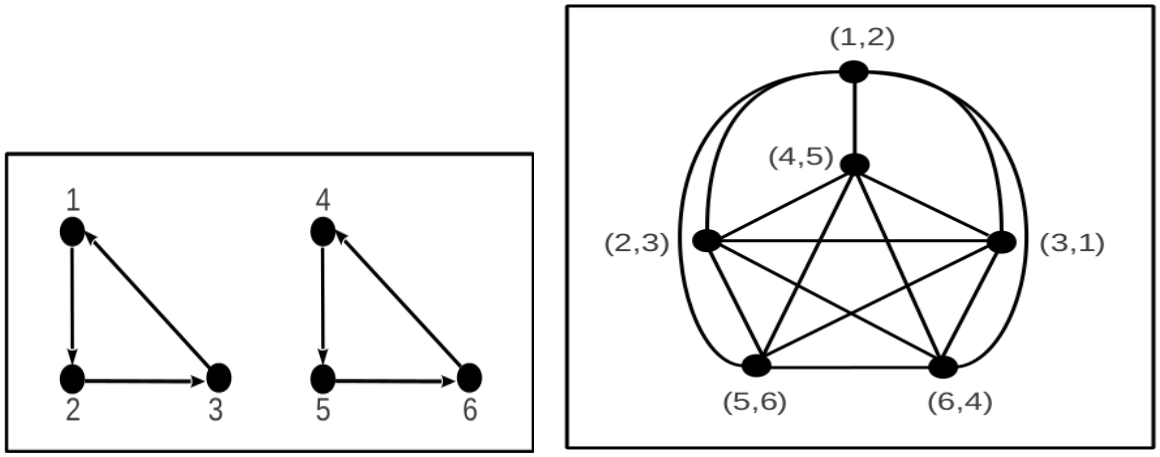
Capítulo 3

O Problema do Subgrafo Acíclico Máximo sob Restrições Disjuntivas Positivas

No problema do Subgrafo Acíclico Máximo sob Restrições Disjuntivas Positivas (SAMRDP), dado um grafo direcionado $G = (V, A)$, uma solução viável deve gerar um subgrafo acíclico $G' = (V, A')$, tal que A' contenha pelo menos um arco de cada restrição. Para SAMRDP as restrições disjuntivas positivas podem ser representadas por meio de um grafo, aqui denominado grafo de coação, $\bar{G} = (A, E)$, não direcionado, cujos vértices correspondem aos arcos do grafo original e cujas arestas representam as restrições disjuntivas positivas.

Em outras palavras, para que uma solução seja viável em SAMRDP, o grafo induzido $G[A']$ deve ser acíclico e A' deve ser uma cobertura por vértices em $\bar{G}$. Note que, perante as restrições disjuntivas positivas, não é sequer garantida a existência de uma solução viável para uma certa instância de SAMRDP, uma vez que a cardinalidade de uma cobertura por vértices mínima em $\bar{G}$ impõe um limite inferior ao número de arcos que deverão compor uma solução viável. Um exemplo de uma instância inviável pode ser observado nos grafos das Figuras 3.1 e 3.2.

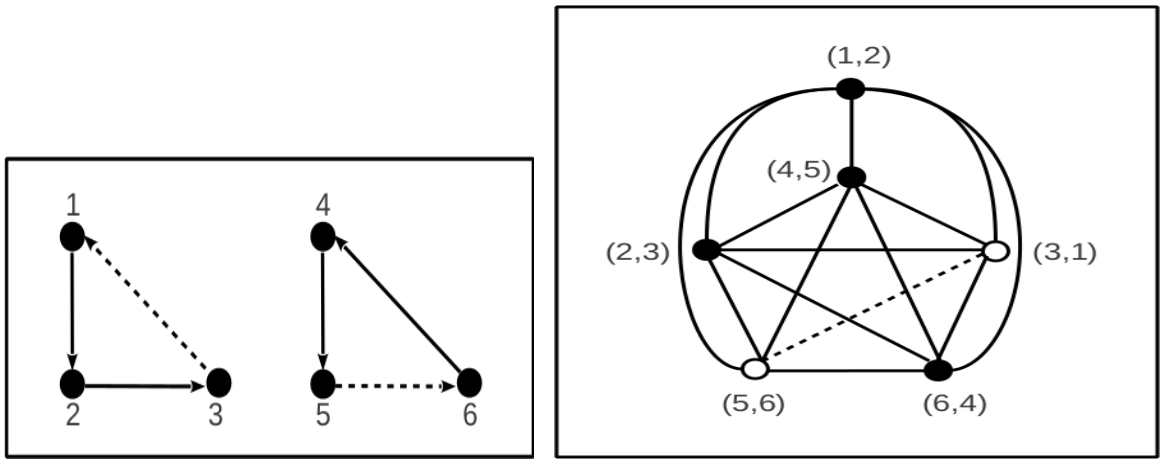
Para o grafo $G = (V, A)$ da figura, considere que cada arco forma uma restrição com todos os demais arcos do grafo, obtendo assim o grafo completo $\bar{G} = (A, E)$. Para que o grafo G seja acíclico percebe-se claramente que dois de seus arcos devem ser retirados, como ilustra a Figura 3.2(a), na qual os arcos selecionados para perfazer uma solução são: $A' = \{(1, 2), (2, 3), (4, 5), (6, 4)\}$. Isto implica, no grafo $\bar{G}$, em uma cobertura por vértices com cardinalidade $|A'| = |A| - 2 = 4$, o que não é possível, como pode ser observado na Figura 3.2(b), em que a aresta $\{(5, 6), (3, 1)\}$ não faz parte da



(a) Grafo $G = (V, A)$. (b) Grafo $\bar{G} = (A, E)$.

Figura 3.1. Instância para SAMRDP.

cobertura. Logo, não existe uma solução viável para esta instância.



(a) Grafo $G' = (V, A')$. (b) Grafo $\bar{G} = (A', E)$.

Figura 3.2. Solução inviável para SAMRDP.

3.1 Complexidade para o teste de viabilidade de SAMRDP

Dada uma instância de SAMRDP, inicialmente deseja-se saber se existe uma solução viável para a mesma, ou seja, dados o grafo $G = (V, A)$ e o grafo de coação correspondente $\bar{G} = (A, E)$, pergunta-se: existe um subconjunto $A' \subseteq A$ no qual $G' = (V, A')$ seja acíclico e A' seja uma cobertura por vértices em $\bar{G}$?

Nesta seção, será demonstrado que o problema de decisão sobre a viabilidade de SAMRDP é $\mathcal{NP}$-Completo. Para isto, será feita uma redução do problema de decisão de cobertura por vértices (CV), problema este pertencente à classe $\mathcal{NP}$-Completo [Garey & Johnson, 1979], ao problema de decisão correspondente à Viabilidade de se obter um Subgrafo Acíclico sob Restrições Disjuntivas Positivas (VSARDP). F

Considere o grafo não-direcionado $G = (V, E)$ e um inteiro k, tal que $0 \leq k \leq |V|$. CV consiste em decidir se existe um subconjunto $V' \subseteq V$, com $|V'| \leq k$, tal que para toda aresta $e = \{u, v\} \in E$, $u \in V'$ ou $v \in V'$.

3.1.1 Construção de uma instância para VSARDP

Para a redução proposta, deve-se construir uma função f que mapeie uma instância arbitrária de CV em uma instância de VSARDP, sendo f uma transformação polinomial. A função f recebe como entrada um grafo não-direcionado $G = (V, E)$, com $V = \{0, 1, \ldots, (n-1)\}$, e um inteiro não-negativo k, representando uma instância de CV, e retorna um grafo direcionado $\hat{G} = (\hat{V}, A)$ e um grafo não-direcionado $\bar{G} = (A, \bar{E})$, representando uma instância de VSARDP. Existirá uma cobertura por vértices em $G = (V, E)$ com não mais que k vértices se e somente se a instância de VSARDP for viável. A função f é descrita na sequência pela definição dos conjuntos $\hat{V}$, A e $\bar{E}$.

$$
\begin{aligned}
\hat{V} =& \{\hat{v}_{i,j} \quad / \quad 0 \leq i \leq k+1; \quad 0 \leq j \leq n\} \\
H =& \{h_{i,j,t} = (\hat{v}_{i,j}, \hat{v}_{i,j+1}) \quad / \quad 0 \leq i \leq k+1; \quad 0 \leq j < n; \quad 0 \leq t \leq 1\} \\
B =& \{b_{(k+1),n,t} = (\hat{v}_{(k+1),n}, \hat{v}_{0,0}) \quad / \quad 0 \leq t \leq 1\} \\
D =& \{d_{i,j} = (\hat{v}_{i,j}, \hat{v}_{i+1,j+1}) \quad / \quad 0 \leq i \leq k; \quad 0 \leq j < n\} \\
A =& H \cup B \cup D \\
\bar{E} =& \{(h_{i,j,0}, h_{i,j,1}) \quad / \quad 0 \leq i \leq k+1; \quad 0 \leq j < n\} \quad \cup \\
& \{(b_{(k+1),n,0}, b_{(k+1),n,1})\} \quad \cup \\
& \{(d_{i,j}, d_{p,q}) \quad / \quad (j,q) \in E; \quad 0 \leq i \leq k; \quad 0 \leq p \leq k\}
\end{aligned}
$$

As cardinalidades dos conjuntos são dadas por:

$$
\begin{aligned}
|\hat{V}| =& \quad (n+1)(k+2) \\
|A| =& \quad 2n(k+2) + n(k+1) + 2 \\
|\bar{E}| =& \quad n(k+2) + |E|(k+1)^2 + 1
\end{aligned}
$$

A fim de exemplificar a transformação a partir da função f, considere o grafo $G = (V, E)$ da Figura 3.3, com $k = 2$, caracterizando uma instância para o problema de CV, e os grafos $\hat{G} = (\hat{V}, A)$ e $\bar{G} = (A, \bar{E})$, gerados aplicando-se a função $f(G, k)$, representados na Figura 3.4, a instância de VSARDP. Percebe-se que a construção pode ser feita em tempo polinomial.

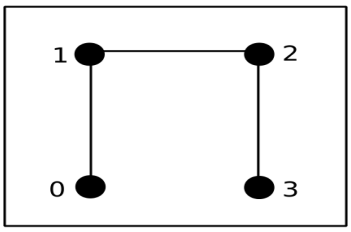


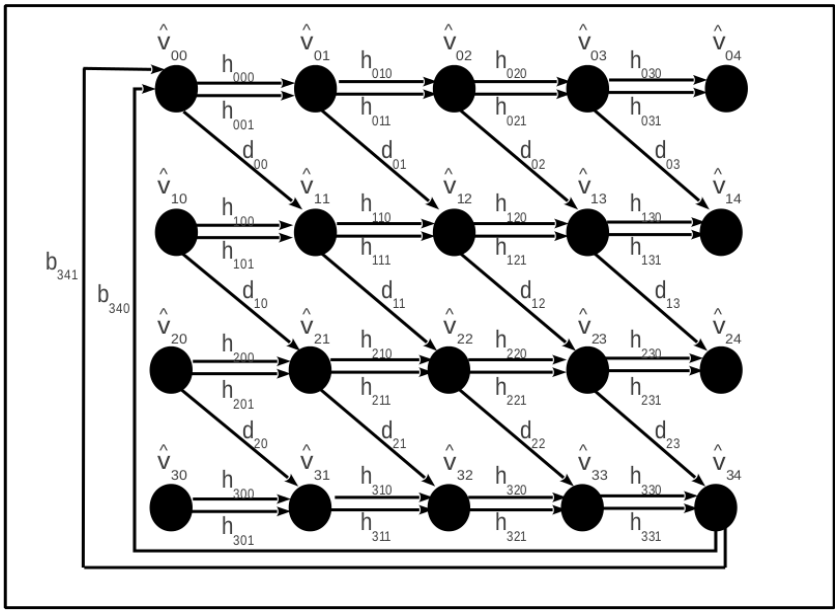
Figura 3.3. Grafo $G = (V, E)$.

Considerando a instância da Figura 3.3 e assumindo $k = 2$, percebe-se que há uma cobertura por vértices em G. Na Figura 3.5 é demonstrado que existe uma cobertura por vértices no grafo $\bar{G}$, que induz um subgrafo acíclico em $\hat{G}$, para a instância de VSARDP correspondente. No grafo da Figura 3.5(b), estão destacados na cor preta os vértices $A' \subset A$ selecionados para compor uma cobertura por vértices no grafo $\bar{G}$. O grafo acíclico apresentado na Figura 3.5(a) é o subgrafo de $\hat{G}$ induzido pelos arcos A'. Logo, a instância de VSARDP é viável.

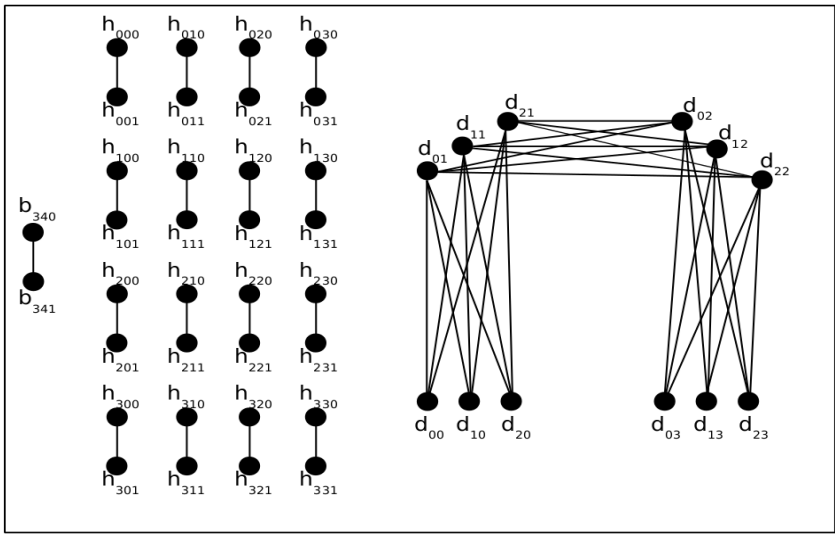
Para o mesmo grafo da Figura 3.3, considerando $k = 1$, tem-se a instância para VSARDP gerada pela função f, representada na Figura 3.6. Nas Figuras 3.7 e 3.8, têm-se a representação de soluções inviáveis. Para a Figura 3.7, embora o grafo $\bar{G} = (A', \bar{E})$ da Figura 3.7(b) tenha uma cobertura por vértices, percebe-se pela Figura 3.7(a) que o subgrafo induzido $\hat{G}[A']$ tem ciclo, e já que isto acontece para toda cobertura de $\bar{G}$, não há solução viável para VSARDP. Logo, para $k = 1$ não poderá haver uma cobertura por vértices no grafo G da Figura 3.3.

Por outro lado, a Figura 3.8 apresenta uma solução inviável em que se tem um subgrafo acíclico em $\hat{G}[A']$, porém não se tem uma cobertura por vértices em $\bar{G} = (A', \bar{E})$. Isto acontece para todo $\hat{G}[A']$ acíclico, logo para esta instância não há solução viável para VSARDP ou para CV.

Na próxima seção será provado que o problema CV, dado um grafo G e um inteiro k, tem solução *sim* se e somente se a solução de VSARDP com os grafos retornados pela função $f(G, k)$ tem solução *sim*.



(a) Grafo $\hat{G} = (\hat{V}, A)$.

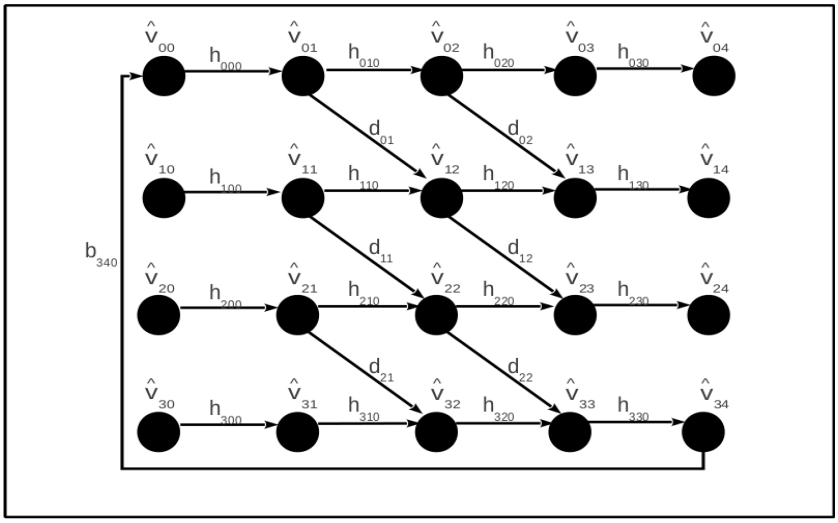


(b) Grafo $\bar{G} = (A, \bar{E})$.

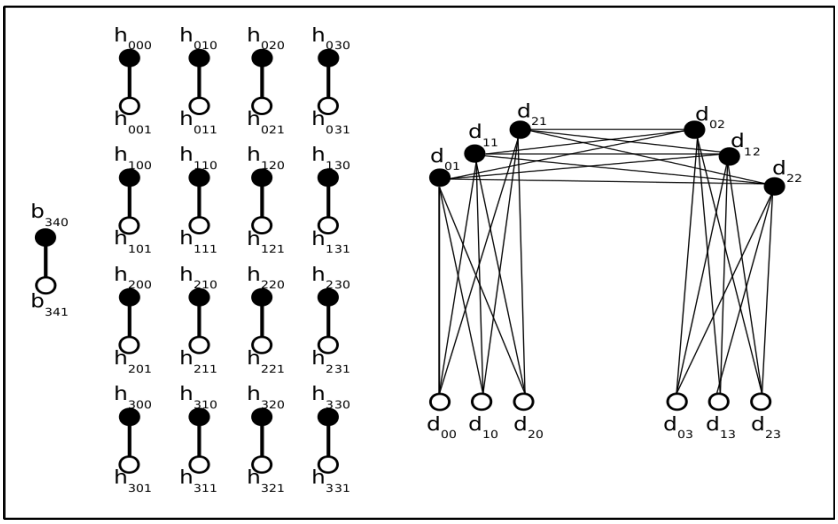
Figura 3.4. Instância para VSARDP.

3.1.2 Prova de NP-Completude para VSARDP

Inicialmente, deve-se provar que VSARDP $\in \mathcal{NP}$. Dada uma instância arbitrária composta pelos grafos $G = (V, A)$ e $\bar{G} = (A, E)$, um algoritmo não-determinístico precisa prover um subconjunto de arcos $A' \subseteq A$ e checar, em tempo polinomial, se o subconjunto selecionado A' é uma cobertura por vértices em $\bar{G}$, e se ele induz um subgrafo $G' = (V, A')$ acíclico.



(a) Grafo $\hat{G}[A']$.



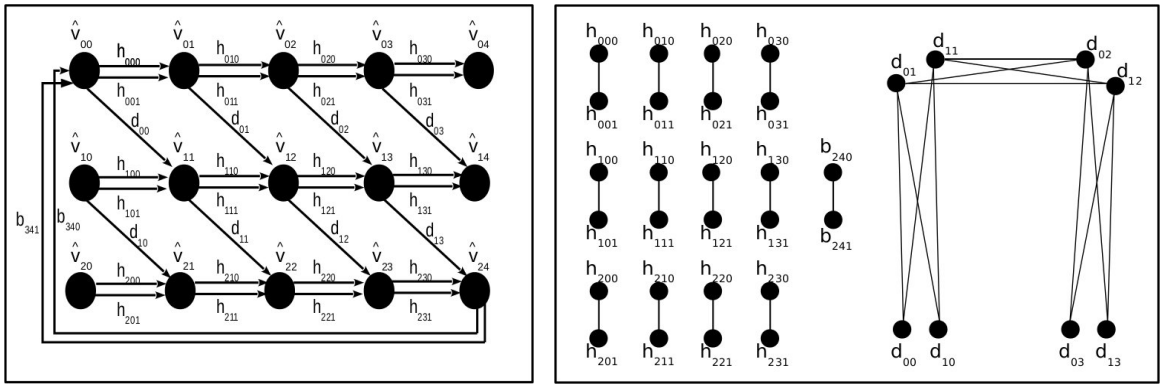
(b) Grafo $\bar{G} = (A', \bar{E})$.

Figura 3.5. Instância para VSARDP com k=2.

Lema 3.1.1. *VSARDP* $\in \mathcal{NP}$.

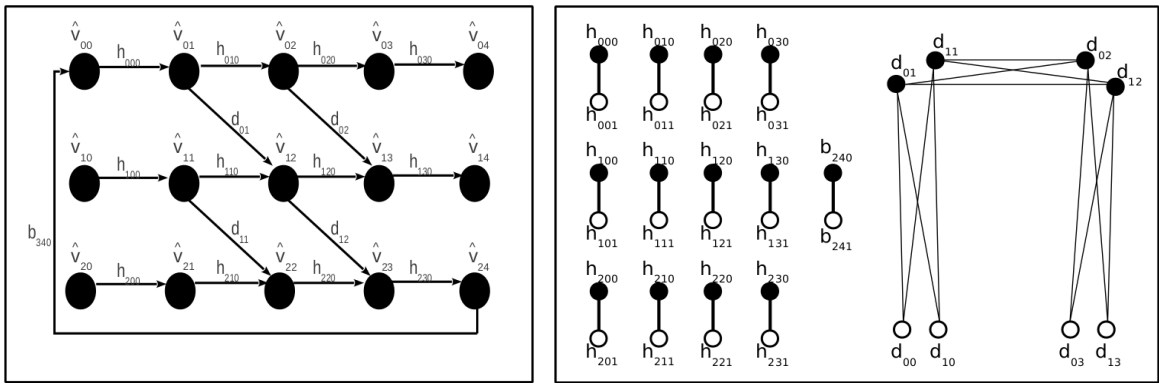
Prova. Verificar, para cada aresta $\{u, v\} \in E$, se $u \in A'$ ou $v \in A'$ pode ser feito em $O(|E|)$. O algoritmo *Depth First Search* (DFS) pode checar se o grafo $G[A']$ é acíclico em $O(|V| + |A|)$, ou seja, linear em relação ao tamanho da entrada. Então, dado um conjunto A' para VSARDP, existe um algoritmo que verifica em tempo polinomial se a solução é viável ou não. □

Lema 3.1.2. *Dados um grafo $G = (V, E)$ e um inteiro k, a solução para CV retorna sim, se e somente se, a solução para VSARDP sobre os grafos retornados pela função $f(G, k)$, $\hat{G} = (\hat{V}, A)$ e $\bar{G} = (A, \bar{E})$, retorna sim.*



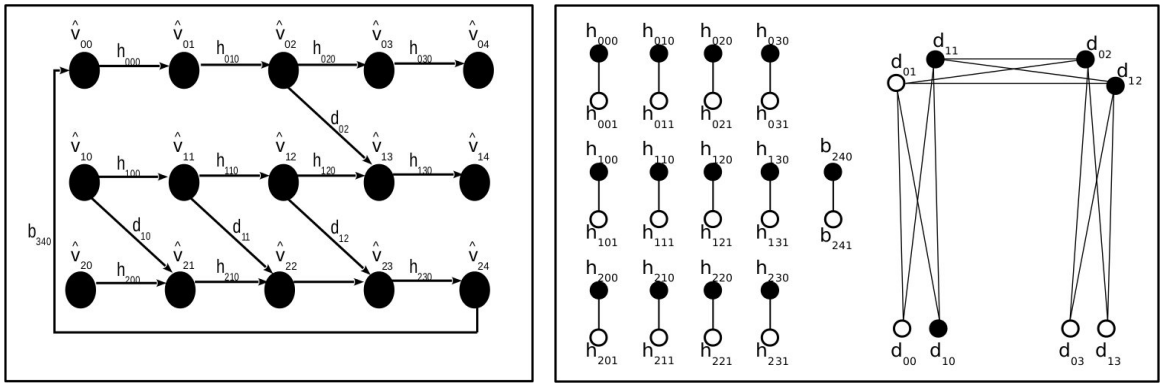
(a) Grafo $\hat{G}[A]$ (b) Grafo $\bar{G} = (A, \bar{E})$.

Figura 3.6. Instância para VSARDP com k=1.



(a) Grafo $\hat{G}[A']$ (b) Grafo $\bar{G} = (A', \bar{E})$.

Figura 3.7. Solução inviável para a instância de VSARDP com k=1.



(a) Grafo $\hat{G}[A']$ (b) Grafo $\bar{G} = (A', \bar{E})$.

Figura 3.8. Outra solução inviável para a instância de VSARDP com k=1.

Prova. Observe que, analisando o grafo $\hat{G} = (\hat{V}, A)$, somente os arcos em B são arcos de retorno, considerando os vértices ordenados pelos seus segundos índices. Todos os

outros arcos estão orientados tal que o vértice inicial possui segundo índice igual a j, e o vértice final possui segundo índice igual a $j+1$. Portanto, para que haja um ciclo, qualquer subgrafo de $\hat{G}$ deve incluir ao menos um arco de B e deve conter um caminho, com exatamente n arcos, a partir do vértice $\hat{v}_{0,0}$ até o vértice $\hat{v}_{k+1,n}$.

Seja $A' \subseteq A$ uma cobertura por vértices mínima em $\bar{G}$. Seja $H' = H \cap A'$, $B' = B \cap A'$ e $D' = D \cap A'$. Uma vez que os vértices associados a cada par de arcos paralelos $h_{i,j,0}$ e $h_{i,j,1}$ em H são adjacentes em $\bar{G}$, pelo menos um dos arcos de cada par estará presente em H'. Portanto, existirá sempre um caminho em $\hat{G}[A']$ a partir do vértice $\hat{v}_{i,j}$ ao vértice $\hat{v}_{i,t}$, para todo $j < t \leq n$. Os vértices associados aos dois arcos em B são adjacentes em $\bar{G}$ e, como anteriormente, ao menos um deles estará sempre presente em B'. Em consequência, o subgrafo induzido $\hat{G}[A']$ é acíclico se e somente se não existe um caminho de $\hat{v}_{0,0}$ a $\hat{v}_{k+1,n}$. Logo, a existência de um ciclo em $\hat{G}[A']$ depende exclusivamente dos arcos pertencentes ao conjunto D'.

Para cada vértice $u \in V$ existem $k+1$ arcos em D, nomeados $d_{i,u}$, tal que $0 \leq i \leq k$. Seja D_u cada um destes subconjuntos disjuntos e observe que $D = \bigcup_{u \in V} D_u$. Se o vértice u é adjacente ao vértice v em G, os arcos associados a u e os arcos associados a v em D são vértices em $\bar{G}$ que induzem um subgrafo bipartido completo. Para cada um destes subgrafos bipartidos completos, ao menos um dos conjuntos da bipartição, por inteiro, deverá pertencer a uma cobertura por vértices de $\bar{G}$. Então, a quantidade mínima de conjuntos D_u que deve compor D' é equivalente à cardinalidade mínima de qualquer cobertura por vértices em G.

Considerando o subgrafo induzido $\hat{G}[A']$, observe que qualquer caminho que comece no vértice $\hat{v}_{0,0}$ e alcance o vértice $\hat{v}_{k+1,n}$ pode usar apenas um dos arcos de cada subconjunto D_u, e que exatamente $k+1$ destes arcos devem ser usados. Portanto, existirá um ciclo em $\hat{G}[A']$ se e somente se D' é composto por no mínimo $k+1$ subconjuntos D_u. Em consequência, é possível encontrar um subconjunto $A' \subseteq A$ que é simultaneamente uma cobertura por vértices em $\bar{G}$ e induz um subgrafo acíclico em $\hat{G}$ se e somente se existe uma cobertura por vértices em G com não mais de k vértices. □

Teorema 1. *VSARDP $\in \mathcal{NP}$–Completo.*

Prova. Uma vez que, pelo Lema 3.1.1, VSARDP está em $\mathcal{NP}$ e, pelo Lema 3.1.2 é possível reduzir polinomialmente o problema CV a VSARDP, conclui-se que VSARDP é $\mathcal{NP}$–Completo. □

3.2 Formulações para SAMRDP

A fim de se atingir os objetivos deste trabalho, o problema SAMRDP será modelado via programação linear inteira. Será proposto, em seções posteriores, um pré-processamento das instâncias de SAMRDP e, por isto, deseja-se averiguar sua eficiência, procedendo-se a uma comparação dos valores das soluções de um resolvedor de programação linear inteira com as instâncias antes e após o pré-processamento.

Apresenta-se a seguir uma formulação para SAMRDP baseada em formulações clássicas para *SAM* e *LOP* [Chaovalitwongse et al., 2011]. Dada uma instância composta pelos grafos $G = (V, A)$ e $\bar{G} = (A, E)$, a variável de decisão x_{ij} definirá os arcos $A' \subseteq A$ que farão parte da solução, modelada na equação 3.1. A variável y_{ij} definirá uma ordenação dos vértices do grafo G, ou seja, dada a relação $u \prec v$, então o vértice u precede o vértice v na ordenação. Esta variável está modelada na equação 3.2.

$$x_{ij} = \begin{cases} 1, & \text{se } (i,j) \in A', \\ 0, & \text{caso contrário.} \end{cases} \tag{3.1}$$

$$y_{ij} = \begin{cases} 1, & \text{se } i \prec j, \\ 0, & \text{caso contrário.} \end{cases} \tag{3.2}$$

O problema do SAMRDP pode ser formulado como:

$$max \sum_{(i,j)\in A} x_{ij} \tag{3.3}$$

sujeito a:

$$y_{ij} + y_{ji} = 1 \qquad \forall \quad i,j \in V, \quad i < j \tag{3.4}$$

$$y_{ij} + y_{jk} + y_{ki} \leq 2 \qquad \forall \quad i,j,k \in V, \quad i \neq j, i \neq k, j \neq k \tag{3.5}$$

$$x_{ij} + x_{kl} \geq 1 \qquad \forall \quad \{(i,j),(k,l)\} \in E \tag{3.6}$$

$$x_{ij} \leq y_{ij} \qquad \forall \quad (i,j) \in A \tag{3.7}$$

$$y_{ij} \in \{0,1\} \qquad \forall \quad i,j \in V \tag{3.8}$$

$$x_{ij} \in \{0,1\} \qquad \forall \quad (i,j) \in A \tag{3.9}$$

A equação 3.3 representa a função objetivo do problema, cujo propósito é maximizar a cardinalidade do conjunto de arcos A', sujeito às restrições presentes em 3.4 – 3.9. A restrição dada em 3.4 força uma ordem entre todo par de vértices do grafo. As restrições 3.4 e 3.5, em conjunto, forçam uma ordenação total dos vértices do grafo

e são suficientes para prevenir a ocorrência de ciclos de qualquer tamanho [Reinelt, 1985]. A restrição em 3.6 força a ocorrência de pelo menos um dos dois arcos que compõem cada restrição disjuntiva positiva no subgrafo obtido. A restrição 3.7 admite que um arco (i, j) faça parte de A' somente se $i \prec j$ na ordenação, garantindo que o subgrafo obtido seja acíclico. As restrições dadas em 3.8 e 3.9 garantem a integralidade das variáveis do problema.

Um outro modelo para SAMRDP pode ser formulado. A variável de decisão x_{ij}, como no modelo anterior, definirá os arcos $A' \subseteq A$ que farão parte da solução, modelada na equação 3.10. A variável y_i atribuirá um número real distinto a cada vértice $i \in V$, que definirá uma ordenação dos vértices do grafo G em que, dada a relação $y_u < y_v$, então o vértice u precede o vértice v na ordenação.

$$x_{ij} = \begin{cases} 1, & \text{se } (i,j) \in A', \\ 0, & \text{caso contrário.} \end{cases} \tag{3.10}$$

$$max \sum_{(i,j)\in A} x_{ij} \tag{3.11}$$

sujeito a:

$$y_j - y_i \geq (1+n)x_{ij} - n \qquad \forall \quad (i,j) \in A, \quad i \neq j \tag{3.12}$$

$$x_{ij} + x_{kl} \geq 1 \qquad \forall \quad \{(i,j),(k,l)\} \in E \tag{3.13}$$

$$y_i = \{y_i \in \mathbb{R} : 0 \leq y_i \leq n\} \qquad \forall \quad i \in V \tag{3.14}$$

$$x_{ij} \in \{0,1\} \qquad \forall \quad (i,j) \in A \tag{3.15}$$

A equação 3.11 representa a função objetivo do problema, cujo propósito é maximizar a cardinalidade do conjunto de arcos A', sujeito às restrições presentes em 3.12 – 3.15. A restrição dada em 3.12 garante que, para todo arco $(i, j) \in A$, se $x_{ij} = 1$ então $y_j - y_i \geq 1$, e é suficiente para prevenir a ocorrência de ciclos em $G[A']$. A restrição em 3.13 força com que pelo menos um dos dois arcos da restrição disjuntiva positiva esteja presente na solução. A restrição dada em 3.14 define a variável y_i como um número real não negativo e a restrição 3.15 garante a integralidade da solução do problema.

Esta nova formulação, quando comparada à formulação anterior, reduz o número de variáveis em $n(n-2)$, e reduz o número de restrições em $(2n^3 - 5n^2 + 3n)/2$. Para os resultados computacionais de programação linear inteira para SAMDRP, apresentados na seção 5.3.2, utilizou-se desta segunda formulação, por gerar soluções para as instâncias testadas com no mínimo a mesma cardinalidade da formulação anterior, no

mesmo tempo de processamento.

Capítulo 4

O Problema do Subgrafo Acíclico Máximo sob Restrições Disjuntivas Negativas

Seja $G = (V, A)$ um grafo direcionado e $\bar{G} = (A, E)$ o grafo de conflitos associado. O problema do Subgrafo Acíclico Máximo sob Restrições Disjuntivas Negativas consiste em determinar um subconjunto máximo $A' \subseteq A$ no qual o subgrafo $G' = (V, A')$ seja acíclico e A' seja um conjunto independente em $\bar{G}$. Como SAMRDN tem o problema SAM como um caso especial (fazendo $E = \emptyset$), SAMRDN também é $\mathcal{NP}$–Difícil.

Assim como SAMRDP, SAMRDN pode ser formulado por meio de programação linear inteira. A única diferença entre os dois problemas está na formulação das restrições disjuntivas, bastando portanto alterar as restrições 3.6 e 3.13 de SAMRDP para:

$$x_{ij} + x_{kl} \leq 1 \qquad \forall \quad \{(i,j),(k,l)\} \in E \tag{4.1}$$

Resultados computacionais para o modelo de programação linear inteira para SAMRDN estão apresentados nas seções 5.3.1 e 6.3.1, e fazem uso das equações 3.11 a 3.15, substituindo a equação 3.13 pela 4.1.

4.1 Algoritmos aproximativos para o problema do Subgrafo Acíclico Máximo

Considerando o problema do *SAM*, serão apresentados três algoritmos da literatura que produzem soluções com no mínimo $|A|/2$ arcos. Uma vez que a cardinalidade de A é um limite superior para o valor da solução ótima do problema, os algoritmos são $(1/2)$-aproximativos [Hassin & Rubinstein, 1994]. Assumindo $P \neq NP$ e a conjectura *Unique Games Conjecture* [Khot, 2002; Austrin et al., 2013; Jünger, 1985], não há um algoritmo aproximativo polinomial de fator constante melhor que $(1/2)$ para o *SAM*. Segue a descrição de cada um dos três algoritmos.

A primeira abordagem é um algoritmo direto, aqui denominado $Sort(G)$. O algoritmo inicialmente sequencia os vértices do grafo arbitrariamente, associando um índice distinto π_i a todo vértice $i \in V$. Então o algoritmo separa os arcos do conjunto A em dois subconjuntos: A_f, que armazena todos os arcos $a = (i, j) \in A$ tais que $\pi_i < \pi_j$ e A_b, o qual armazena todos os arcos $a = (i, j) \in A$ tais que $\pi_i > \pi_j$. O algoritmo então seleciona o subconjunto (A_f ou A_b) de cardinalidade máxima e retorna o subgrafo contendo o conjunto de arcos selecionados como solução.

Teorema 2. *$Sort(G)$ retorna como solução um grafo acíclico com no mínimo $|A|/2$ arcos.*

Prova. Observe que ambos subgrafos $G_f = (V, A_f)$ e $G_b = (V, A_b)$ são acíclicos. Um ciclo $v_1, v_2, \ldots, v_1$ irá implicar na presença de um arco (i, j) com $\pi_i < \pi_j$ e de um arco (k, l) com $\pi_k > \pi_l$. Além disto, como cada arco em A pertence exclusivamente a A_f ou A_b, um destes subconjuntos deverá ter no mínimo $|A|/2$ arcos. □

O segundo algoritmo, denominado $Greedy(G)$, usa uma estratégia gulosa enquanto examina os arcos para formar o subgrafo acíclico. O algoritmo mantém dois subconjuntos, S e T. Então examina gulosamente todos os arcos em A, adicionando um arco a S se com a sua inclusão o subgrafo induzido $G[S]$ permanece acíclico. Caso contrário, o algoritmo adiciona o arco ao conjunto T. Após o processamento de todos os arcos, o conjunto de maior cardinalidade, S ou T, é selecionado para formar o subgrafo acíclico $G' = (V, A')$.

Teorema 3. *$Greedy(G)$ retorna como solução um grafo acíclico com no mínimo $|A|/2$ arcos.*

Prova. $G[S]$ é acíclico por construção. Para mostrar que $G[T]$ é acíclico considere os vértices em V ordenados respeitando-se uma ordenação topológica em $G[S]$. Cada arco

em T forma um ciclo com um subconjunto de arcos em S, logo é um arco de retorno considerando a ordenação topológica de $G[S]$. Desde que todos os arcos em T são de retorno considerando aquela ordenação dos vértices, o argumento para G_b ser acíclico usado no algoritmo $Sort(G)$ também é válido para $G[T]$. Como na prova anterior, cada arco em A está em S ou T, e um destes conjuntos deve ter no mínimo $|A|/2$ arcos. □

Lema 4.1.1. *Se o grafo G é acíclico, o algoritmo $Greedy(G)$ retorna uma solução ótima para o SAM.*

Prova. Todos os arcos $a \in A$ estarão no conjunto S, retornado como solução. Tem-se então $|S| = |A|$. □

O terceiro algoritmo aproximativo, aqui denominado $Degree(G)$, processa todos os vértices de G, em qualquer ordem, analisando os arcos que entram e que saem do vértice. Se o vértice tem o grau de entrada maior que o grau de saída, os arcos que chegam ao vértice são removidos do grafo G e adicionados ao conjunto A', e os arcos que saem são também removidos de G, porém descartados. Se o vértice tem grau de entrada menor que ou igual ao grau de saída, os arcos que saem são adicionados ao conjunto A' e os arcos que entram são descartados. Todos os arcos, após processados, são removidos de G. Quando todos os vértices forem processados, o algoritmo retorna o subconjunto A' como solução.

Teorema 4. *$Degree(G)$ retorna como solução um grafo acíclico com no mínimo $|A|/2$ arcos.*

Prova. Para provar que o subgrafo induzido pelo conjunto A', $G[A']$, é acíclico, considere um ciclo $x_1, \ldots, x_k, x_1$ e, sem perda de generalidade, considere x_1 como o primeiro vértice deste ciclo a ter sido processado pelo algoritmo. Então, os arcos (x_k, x_1) e (x_1, x_2) teriam sido descartados no momento em que x_1 foi processado, impossibilitando a formação de qualquer ciclo. Além disto, note que $|A'| \geq 1/2|A|$, desde que cada arco em A é examinado exatamente uma vez e, enquanto um vértice é processado, o algoritmo sempre adiciona a $|A'|$ no mínimo a mesma quantidade de arcos que são descartados. □

Para estes e outros algoritmos aproximativos para o *SAM*, vide Hassin & Rubinstein [1994]. Na próxima seção, os três algoritmos apresentados serão adaptados para o SAMRDN, considerando-se duas abordagens distintas quanto ao momento em que se considera o cálculo do conjunto independente máximo para decidir quais arcos irão compor a solução.

4.2 Algoritmos aproximativos para SAMRDN

Para os algoritmos aproximativos aqui propostos para SAMRDN, faz-se necessário o cômputo de conjuntos independentes máximos em tempo polinomial nos grafos de conflitos $\bar{G}$. Conjuntos independentes máximos podem ser computados em tempo polinomial para várias classes de grafos, incluindo grafos *claw-free* [Sbihi, 1980] e perfeitos [Grötschel et al., 1993]. Neste trabalho, foram desenvolvidos algoritmos aproximativos para o SAMRDN com estas classes de grafos de conflitos.

A aplicação direta dos algoritmos $Sort(G)$, $Greedy(G)$ e $Degree(G)$ apresentados anteriormente para o *SAM* ao SAMRDN não garante uma solução com no mínimo metade do número de arcos de uma solução ótima. A presença das restrições disjuntivas negativas podem tornar o subconjunto retornado pelo algoritmo inviável, mesmo este sendo acíclico. Sendo assim, alguns dos arcos do subconjunto escolhido deverão ser retirados da solução a fim de torná-la viável, consequentemente podendo reduzir o número de arcos a menos da metade do número de arcos de uma solução ótima.

Como exemplo, considere a aplicação do algoritmo $Sort(G)$ ao grafo da Figura 4.1. Os índices π_i arbitrariamente associados aos vértices do grafo estão representados sobre os mesmos. Para este exemplo, a lista de conflitos é dada por $E = \{\{(0,1),(1,2)\};\{(2,3),(3,4)\}\}$. Seja $|A_f| = m_f$ e $|A_b| = m_b$. Analisando os índices π_i, para este exemplo temos $m_f = m_b = 4$.

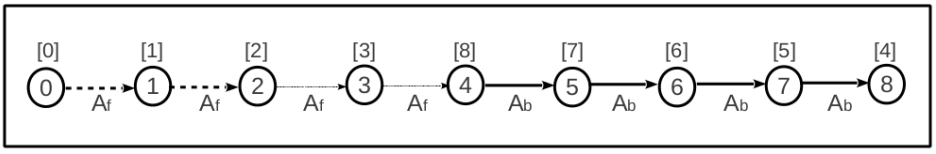


Figura 4.1. Grafo $G = (V, A)$.

O algoritmo $Sort(G)$ então escolhe o subconjunto de maior cardinalidade e em caso de empate, pode selecionar qualquer um dos dois subconjuntos. Para este exemplo, suponha que o algoritmo tenha escolhido o conjunto A_f para compor a solução final. A fim de tornar o subconjunto escolhido viável respeitando-se as restrições disjuntivas negativas, o algoritmo deverá retirar um dos arcos de cada par de conflitos presente em E, uma vez que ambos os arcos dos dois pares de conflitos estão classificados como A_f.

Tome c_f (resp. c_b) como o número de conflitos em que ambos os arcos pertencem a A_f (resp. A_b). Para o grafo do exemplo, tem-se $c_f = 2$ e $c_b = 0$. O número de arcos retornado como solução pela aplicação do algoritmo $Sort(G)$, tendo este algoritmo pré-selecionado o conjunto A_f, e após retirar os arcos que tornam a solução inviável, será

$m_f - c_f = 4 - 2 = 2$ arcos. A solução ótima, facilmente identificada neste grafo, tem valor $m_{opt} = 6$ arcos. Sendo assim, o valor da solução retornada pela aplicação direta do algoritmo para este exemplo é (1/3) da solução ótima. Na sequência, serão propostas pequenas modificações aos três algoritmos apresentados anteriormente, objetivando obter um subconjunto viável de A com no mínimo a metade dos arcos de uma solução ótima, conforme duas abordagens distintas.

A primeira abordagem, denominada *early*, simplesmente retorna a interseção entre um dos subconjuntos de arcos selecionados pelo algoritmo e um conjunto independente máximo calculado sobre o grafo $\bar{G}$. Para os seis algoritmos seguintes, $Sort_e(G, \bar{G})$, $Greedy_e(G, \bar{G})$, $Degree_e(G, \bar{G})$, $Sort_l(G, \bar{G})$, $Greedy_l(G, \bar{G})$ e $Degree_l(G, \bar{G})$, considere I um conjunto independente máximo sobre o grafo $\bar{G}$ e note que o valor da solução ótima é limitada superiormente por $|I|$. Logo, um algoritmo que garante uma solução com no mínimo $|I|/2$ arcos é um algoritmo 1/2-aproximativo.

Para o algoritmo $Sort_e(G, \bar{G})$, dado o grafo $G = (V, A)$, os vértices em V são sequenciados arbitrariamente e então os arcos em A são classificados nos conjuntos A_f e A_b, como no algoritmo anteriormente apresentado para o *SAM*. Então o algoritmo computa os dois subconjuntos $I \cap A_f$ e $I \cap A_b$. $Sort_e(G, \bar{G})$ retorna como solução $I \cap A_f$ se $\max(|I \cap A_f|, |I \cap A_b|) = |I \cap A_f|$ ou $I \cap A_b$ caso contrário.

Teorema 5. *O algoritmo $Sort_e(G, \bar{G})$ é 1/2-aproximativo para SAMRDN*

Prova. O grafo obtido como solução é um subgrafo do grafo obtido executando-se o algoritmo $Sort(G)$, e em consequência é acíclico. Cada arco em I pertence a A_f ou a A_b, logo $|I \cap A_f| \geq |I|/2$ ou $|I \cap A_b| \geq |I|/2$. Então, a solução obtida pelo algoritmo tem no mínimo a metade dos arcos de qualquer solução ótima. □

O algoritmo $Greedy_e(G, \bar{G})$ analisa de forma gulosa todos os arcos de A, adicionando um arco ao conjunto S se com esta adição $G[S]$ permanece acíclico. Caso contrário, o algoritmo adiciona o arco ao conjunto T. Então, o algoritmo computa $I \cap S$ e $I \cap T$, retornando como solução o subconjunto de maior cardinalidade.

Teorema 6. *O algoritmo $Greedy_e(G, \bar{G})$ é 1/2-aproximativo para SAMRDN.*

Prova. Pelo mesmo argumento utilizado na prova anterior, o subgrafo obtido é acíclico. Novamente, $|I \cap S| \geq |I|/2$ ou $|I \cap T| \geq |I|/2$, e o algoritmo é 1/2-aproximativo. □

Lema 4.2.1. *Se o grafo G é acíclico, o algoritmo $Greedy_e(G, \bar{G})$ retorna uma solução ótima para SAMRDN.*

Prova. O conjunto S será composto por todos os arcos $a \in A$. Logo, no subgrafo induzido $\bar{G}[S]$, a interseção $S \cap I$ resulta em I, retornado como solução pelo algoritmo. Tem-se que $|I|$ é o valor ótimo da solução do problema. □

O algoritmo $Degree_e(G, \bar{G})$ processa todos os vértices de G, em qualquer ordem, analisando os arcos que entram e que saem do vértice. Para cada vértice $v \in V$, considere $\delta^+(v)$ o conjunto dos arcos que entram e $\delta^-(v)$ o conjunto dos arcos que saem de v. O algoritmo computa $I \cap \delta^+(v)$ e $I \cap \delta^-(v)$, adicionando ao conjunto A' o subconjunto de maior cardinalidade, e então todos os arcos em $\delta^+(v) \cup \delta^-(v)$ são removidos do grafo G.

Teorema 7. *O algoritmo $Degree_e(G, \bar{G})$ é 1/2-aproximativo para SAMRDN.*

Prova. Seguindo o mesmo argumento utilizado no algoritmo correspondente para o *SAM*, conclui-se que A' induz um subgrafo acíclico em G. O subconjunto obtido é um conjunto independente em $\bar{G}$ pois é um subconjunto de I. Observe que todos os arcos em I são processados exatamente uma vez e que, enquanto processa um certo vértice de G, o algoritmo sempre adiciona a $|A'|$ pelo menos tantos arcos de I quanto o número de arcos de I que são descartados. Então, A' tem no mínimo $|I|/2$ arcos. □

Na segunda abordagem, então denominada *late*, os três algoritmos apresentados para o *SAM* serão readaptados ao SAMRDN, então denominados $Sort_l(G, \bar{G})$, $Greedy_l(G, \bar{G})$ e $Degree_l(G, \bar{G})$. Nesta nova abordagem, um conjunto independente máximo é calculado sobre os subgrafos de $\bar{G}$ induzidos pelos conjuntos de arcos retornados pelos algoritmos. Novamente, tanto esta abordagem quanto a anterior serão polinomiais sempre que o problema do conjunto independente máximo puder ser resolvido em tempo polinomial em $\bar{G}$. Por definição, se o grafo é perfeito (*claw free*), todo subgrafo induzido a partir deste também é perfeito (*claw free*).

Para os três algoritmos, $Sort_l(G, \bar{G})$, $Greedy_l(G, \bar{G})$ e $Degree_l(G, \bar{G})$, seguindo o mesmo argumento utilizado nos algoritmos correspondentes para o *SAM*, $Sort(G)$, $Greedy(G)$ e $Degree(G)$, conclui-se que os conjuntos de arcos selecionados pelos algoritmos induzem um subgrafo acíclico em G.

Para o algoritmo $Sort_l(G, \bar{G})$, considere $\bar{G}[A_f]$ o subgrafo de $\bar{G}$ induzido pelo conjunto de vértices A_f e $\bar{G}[A_b]$ o subgrafo de $\bar{G}$ induzido pelo conjunto de vértices A_b. Tome α_f como a cardinalidade de um conjunto independente máximo em $\bar{G}[A_f]$, e α_b como a cardinalidade de um conjunto independente máximo em $\bar{G}[A_b]$. Observe que no máximo α_f (resp. α_b) vértices de $\bar{G}[A_f]$ (resp. $\bar{G}[A_b]$) podem estar simultaneamente em uma solução viável, considerando as restrições disjuntivas negativas. O algoritmo

seleciona A_f se $\max(\alpha_f, \alpha_b) = \alpha_f$ ou A_b caso contrário. O algoritmo retorna como solução um conjunto independente máximo sobre o grafo induzido pelo subconjunto selecionado no grafo $\bar{G}$ [Mapa & Urrutia, 2013].

Teorema 8. *O algoritmo $Sort_l(G, \bar{G})$ é 1/2-aproximativo para SAMRDN.*

Prova. Note que $I \cap A_f$ é um conjunto independente em $\bar{G}[A_f]$ e $I \cap A_b$ é um conjunto independente em $\bar{G}[A_b]$. Isto implica que $\alpha_f \geq |I \cap A_f|$ e $\alpha_b \geq |I \cap A_b|$. Então, $\alpha_f + \alpha_b \geq |I \cap A_f| + |I \cap A_b| = |I|$. Isto é, $\alpha_f \geq |I|/2$ ou $\alpha_b \geq |I|/2$, logo o algoritmo é 1/2-aproximativo. □

O algoritmo $Greedy_l(G, \bar{G})$ considera os subgrafos induzidos $\bar{G}[S]$ e $\bar{G}[T]$, sobre os quais serão computados conjuntos independentes máximos.

Teorema 9. *O algoritmo $Greedy_l(G, \bar{G})$ é 1/2-aproximativo para SAMRDN.*

Prova. A simples substituição de A_f e A_b por S e T na prova previamente apresentada é suficiente para demonstrar que a adaptação do algoritmo para a abordagem *late* resulta em um algoritmo 1/2-aproximativo. □

Lema 4.2.2. *Se o grafo G é acíclico, o algoritmo $Greedy_l(G, \bar{G})$ retorna uma solução ótima para SAMRDN.*

Prova. O conjunto S será composto por todos os arcos $a \in A$. Logo, o algoritmo retorna como solução um conjunto independente máximo em $\bar{G}[A]$, cuja cardinalidade é o valor ótimo da solução do problema. □

O algoritmo $Degree_l(G, \bar{G})$ computa $|I \cap \delta^-(v)|$ e $|I \cap \delta^+(v)|$ para cada vértice $v \in V$, em qualquer ordem. Se $|I \cap \delta^-(v)| \geq |I \cap \delta^+(v)|$, o algoritmo adiciona ao conjunto $\bar{A}$ todos os arcos em $\delta^-(v)$, ou adiciona a $\bar{A}$ todos os arcos em $\delta^+(v)$ caso contrário. Note que a principal diferença entre as abordagens *early* e *late* é que, na abordagem *late*, todos os arcos do subconjunto selecionado são adicionados ao conjunto $\bar{A}$. Então, o algoritmo descarta todos os arcos do outro subconjunto e apaga do grafo G todos os arcos em $\delta^+(v) \cup \delta^-(v)$. Quando todos os vértices tiverem sido processados, o algoritmo computa um conjunto independente máximo no subgrafo induzido $\bar{G}[\bar{A}]$, retornando este conjunto, nomeado A', como solução.

Teorema 10. *O algoritmo $Degree_l(G, \bar{G})$ é 1/2-aproximativo para SAMRDN.*

Prova. Pelo mesmo argumento utilizado no algoritmo $Degree(G)$, o subgrafo induzido $G[\bar{A}]$ é acíclico. A solução obtida por $Degree_l(G, \bar{G})$ tem pelo menos a metade dos arcos de I uma vez que, por construção, pelo menos a metade dos arcos de I estão

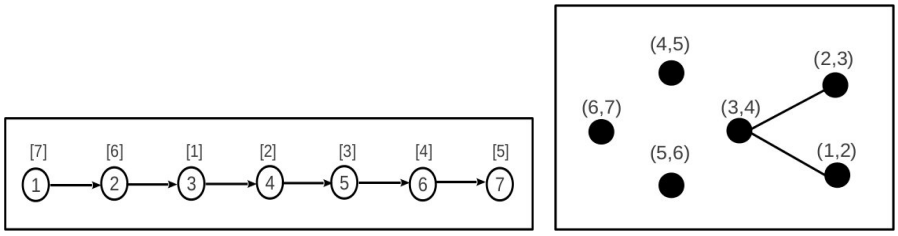
em $\bar{A}$, e estes arcos são um conjunto independente em $\bar{G}[\bar{A}]$. Logo, o algoritmo é 1/2-aproximativo. □

O teorema a seguir demonstra que a abordagem *late* é sempre preferível à abordagem *early*.

Teorema 11. *Se os arcos de G são particionados nos mesmos subconjuntos, um algoritmo seguindo a abordagem late obtém uma solução com no mínimo a mesma cardinalidade que aquela obtida pelo algoritmo equivalente segundo a abordagem early.*

Prova. Após o conjunto A ser dividido em dois subconjuntos, o algoritmo seguindo a abordagem *early* retorna a interseção de um conjunto independente máximo em $\bar{G}$ com um dos subconjuntos. Uma vez que esta interseção é um conjunto independente no grafo induzido pelo subconjunto selecionado em $\bar{G}$, sua cardinalidade é um limite inferior para a cardinalidade de um conjunto independente máximo sobre o grafo induzido pelo subconjunto selecionado em $\bar{G}$. Logo, a solução retornada pelo algoritmo segundo a abordagem *late* possui sempre pelo menos a mesma cardinalidade da solução retornada pelo algoritmo equivalente segundo a abordagem *early*. □

A Figura 4.2 mostra uma instância para o problema no qual $Sort_l(G, \bar{G})$ retorna uma solução melhor que $Sort_e(G, \bar{G})$. Considere os números acima dos vértices na Figura 4.2(a) como sendo os índices indicando a ordem em que foram sorteados. Os conjuntos $A_f = \{(3,4),(4,5),(5,6),(6,7)\}$ e $A_b = \{(1,2),(2,3)\}$ são então obtidos. Note que $I = \{(1,2),(2,3),(4,5),(5,6),(6,7)\}$ é o conjunto independente máximo de $\bar{G}$. O conjunto independente máximo em $\bar{G}[A_f]$ é dado por $I_f = \{(3,4),(4,5),(5,6),(6,7)\}$ e em $\bar{G}[A_b]$ é dado por $I_b = \{(1,2),(2,3)\}$. Logo, $\alpha_f = 4$ e $\alpha_b = 2$.



(a) Graph $G = (V, A)$ (b) Graph $\bar{G} = (A, E)$.

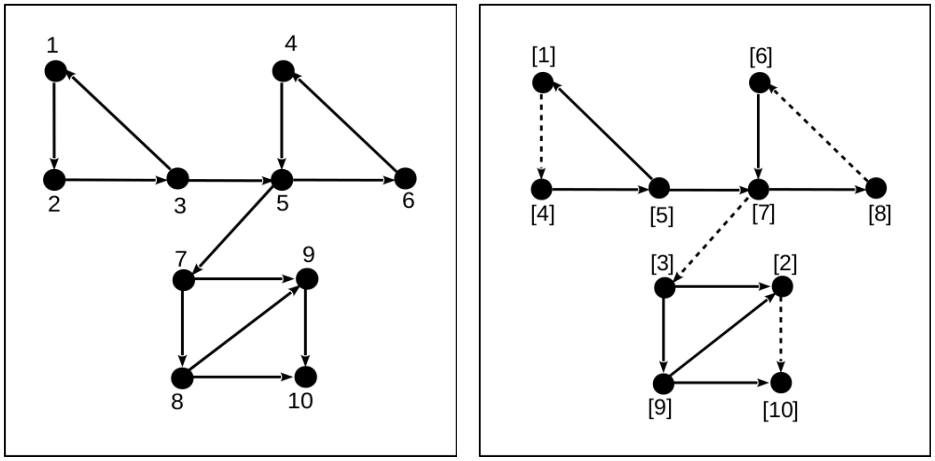
Figura 4.2. Instância para SAMRDN.

O algoritmo $Sort_e(G, \bar{G})$ retorna uma solução com $\max(|I \cap A_f|, |I \cap A_b|) = \max(3, 2) = 3$ arcos e o algoritmo $Sort_l(G, \bar{G})$ retorna uma solução com $\max(\alpha_f, \alpha_b) = \max(4, 2) = 4$ arcos. A solução ótima neste caso possui $|I| = 5$ arcos, coincidindo com o conjunto I.

4.2.1 Exemplo de execução do algoritmo $Degree_l(G, \bar{G})$

Considere como exemplo uma instância dada pelo grafo $G = (V, A)$ da Figura 4.3(a), e pelo grafo de conflitos $\bar{G} = (A, E)$ da Figura 4.4(a). Os números representados logo acima dos vértices de G na Figura 4.3(b) indicam a ordem em que os mesmos serão processados pelo algoritmo $Degree_l(G, \bar{G})$. Ou seja, o vértice de rótulo 1 no grafo $G = (V, A)$ (figura 4.3(a)) será o primeiro a ser processado (representado pelo índice [1] na figura 4.3(b)), assim como o vértice de rótulo 4 (figura 4.3(a)) será o sexto vértice a ser processado (representado pelo índice [6] na figura 4.3(b)).

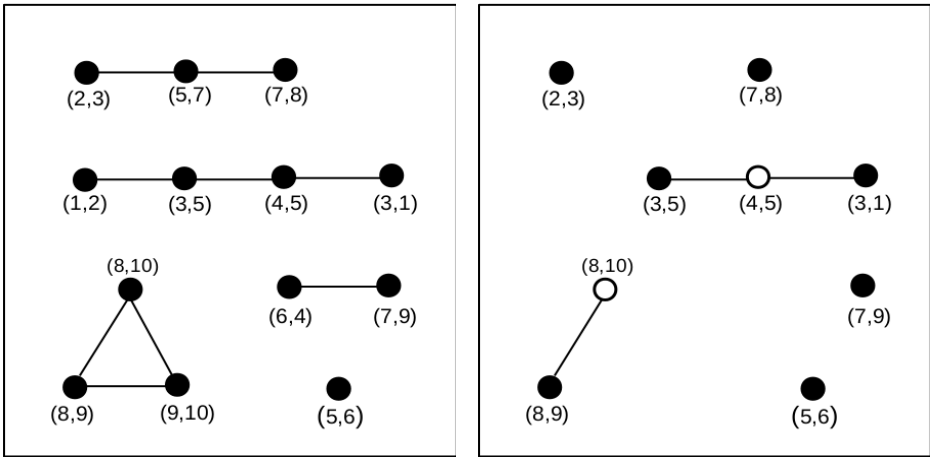
Seja $I = \{(2,3),(7,8),(3,5),(3,1),(8,9),(7,9),(5,6)\}$ o conjunto independente máximo em $\bar{G}$ selecionado para o cômputo de $|I \cap \delta^-(v)|$ e $|I \cap \delta^+(v)|$, para cada vértice $v \in V$. Os arcos em A são então analisados e classificados para comporem o conjunto $\bar{A}$, conforme a descrição do algoritmo. Os arcos que compõem o conjunto $\bar{A}$ podem ser observados na Figura 4.3(b), tendo sido descartados de G os arcos representados por uma linha pontilhada. Por exemplo, ao se processar o vértice de rótulo 1 do grafo $G = (V, A)$, o arco $(3,1) \in I$ e o arco $(1,2) \notin I$, logo $(3,1)$ é selecionado para compor $\bar{A}$ e o arco $(1,2)$ é descartado.



(a) Grafo $G = (V, A)$. (b) Arcos $\bar{A}$ selecionados de G.

Figura 4.3. Grafo para exemplificação do algoritmo $Degree_l(G, \bar{G})$.

Na Figura 4.4(b) está representado o subgrafo induzido $\bar{G}[\bar{A}]$, no qual estão destacados na cor preta os vértices que compõem um conjunto independente máximo, selecionados para compor a solução de SAMRDN. Note que este conjunto independente tem cardinalidade igual à do conjunto $|I| = 7$, inclusive coincidindo com I, sendo portanto uma solução ótima para o problema.



(a) Grafo $\bar{G} = (A, E)$. (b) Subgrafo $\bar{G}[\bar{A}]$.

Figura 4.4. Grafo $\bar{G} = (A, E)$ e subgrafo induzido.

A Figura 4.5 representa o grafo acíclico gerado como solução pela execução do algoritmo $Degree_l(G, \bar{G})$ à instância dada, conforme descrito anteriormente.

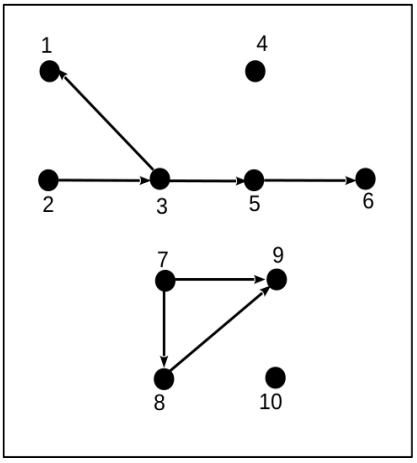


Figura 4.5. Solução ótima para a instância de $Degree_l(G, \bar{G})$.

4.3 Resultados computacionais

A fim de medir o desempenho dos seis algoritmos 1/2–aproximativos para o SAMRDN, usou-se as instâncias disponibilizadas por Zhang et al. [2011]. Estas instâncias foram elaboradas para o problema da árvore geradora mínima com restrições de conflitos e peso inteiro nas arestas. Sendo assim, como os grafos G das instâncias disponibilizadas são não direcionados, para a aplicação ao SAMRDN as arestas foram direcionadas

conforme seus pesos, seguindo a lógica: se o peso é um número par, o arco mantém os vértices inicial e final originais, caso contrário, os vértices de origem e destino do arco são invertidos. Além disto, os grafos $\bar{G}$ disponibilizados pelos autores não foram utilizados nestes testes computacionais, por serem grafos gerais, tendo sido gerados grafos de conflitos conforme as restrições dos algoritmos aproximativos.

Tanto para a abordagem *early* quanto para a *late*, os algoritmos aproximativos calculam subconjuntos independentes máximos sobre o grafo $\bar{G}$ e subgrafos dele e, em consequência, são polinomiais somente se o problema do conjunto independente máximo puder ser resolvido em tempo polinomial no grafo de conflitos. Assim sendo, para estes testes foram gerados grafos de conflitos cujas componentes conexas são compostas por caminhos, ciclos, cliques e árvores, para as quais é possível o cálculo do conjunto independente máximo em tempo polinomial. Para cada instância, na geração de caminhos e ciclos usou-se, para cada tipo de componente conexa, 20% dos arcos do grafo G; para a geração de cliques usou-se outros 30% dos arcos; e para a geração de árvores, usou-se os arcos remanescentes do grafo, sendo os arcos sorteados aleatoriamente e uma vez escolhido, fará parte apenas daquela componente conexa.

Para compor o grafo de conflitos não direcionado $\bar{G}$, os conflitos compostos por caminhos foram gerados com tamanhos $t_{path} = \{1, 2, ..., t_p\}$, até esgotados todos os arcos destinados a comporem esta componente conexa, sendo que o tamanho do último caminho t_p poderá ser repetido. Da mesma forma, os ciclos foram gerados com tamanhos $t_{cycle} = \{3, 4, ..., t_c\}$, cliques com tamanhos $t_{clique} = \{4, 5, ..., t_q\}$, e árvores binárias com o número de níveis $n_{tree} = \{1, 2, ..., n_t\}$. Estas instâncias, compostas pelos grafos $G = (V, A)$ e $\bar{G} = (A, E)$, serão denominadas do tipo T_1, cujas cardinalidades dos conjuntos de vértices, arcos e conflitos estão especificadas na tabela 4.1, assim como as divisões em classes $A, B, ..., I$.

Tabela 4.1. Características das instâncias tipo T_1.

T_1	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**
|V|	50	100	100	200	200	200	300	300	300
|A|	200	300	500	400	600	800	600	800	1000
|E|	261	439	829	610	1027	1503	1027	1503	2026

Para o cálculo de um conjunto independente máximo sobre $\bar{G}$, quando se tem caminhos basta acrescentar todos os vértices ímpares que aparecem ao longo do comprimento do caminho ao conjunto, descartando os demais. Para ciclos, escolhe-se aleatoriamente um primeiro vértice, que será inserido ao conjunto independente e a partir

do qual irá se alternando entre os vértices que serão incluídos ou não, até se atingir novamente o vértice inicial. Se o ciclo é ímpar, os dois vértices adjacentes ao vértice inicial escolhido serão descartados. Para cliques basta selecionar aleatoriamente um único vértice. Já para as árvores o cômputo do conjunto independente máximo foi feito via programação dinâmica.

Para encontrar um conjunto independente máximo em árvores, inicialmente esta será classificada em seus níveis e, a partir do nível mais baixo até a raiz, cada vértice v receberá duas notas: $COM_v = 1 + \sum_{F \in Filhos} SEM_F$ e $SEM_v = \sum_{F \in Filhos} max(COM_F, SEM_F)$, sendo $Filhos$ o conjunto dos filhos do vértice em análise. Para um vértice folha v, $COM_v = 1$ e $SEM_v = 0$. Uma vez atribuídas as notas e iniciando-se da raiz, para cada vértice v, se $COM_v > SEM_v$, então o vértice v é escolhido para perfazer o conjunto independente, e descartado caso contrário ($COM_v \leq SEM_v$).

Nas instâncias de Zhang et al. [2011], para cada grafo G, nove grafos de conflitos $\bar{G}$ distintos foram gerados. Tomando isto como padrão, para cada classe de instância $T_1 = \{A, B, ..., I\}$ foram gerados nove grafos de conflitos distintos, porém com o mesmo número de conflitos. As cardinalidades de um conjunto independente máximo sobre os grafos $\bar{G}$, denotado por $|I|$, que representa um limite superior de arcos que podem compor a solução das instâncias T_1, são dadas por $|I| = \{99, 144, 238, 191, 284, 377, 284, 377, 468\}$, nesta ordem, para as classes $A, B, ..., I$.

Os algoritmos desenvolvidos foram implementados em C++ e executados em uma máquina *Intel Core* i7 980 (3.33GHz), com 24GB de RAM. A tabela 4.2 apresenta, para cada classe de instância T_1 e para cada algoritmo aproximativo, a média do número de arcos que compõem a solução (denominada **S**), e a média da porcentagem do número de arcos na solução (denominada %) com relação ao limite superior de arcos que podem compor esta solução ($|I|$). A tabela 4.3 apresenta estatísticas sobre os valores de mínimo, máximo, média, desvio-padrão e coeficiente de variação, nesta ordem, para as porcentagens de arcos nas soluções.

Ao analisar os valores de mínimo presentes na tabela 4.3, nota-se o fator de aproximação dos seis algoritmos, nunca inferior a 50%. Percebe-se, analisando os coeficientes de variação das soluções geradas, que os algoritmos $Greedy_e$ e $Greedy_l$ são menos estáveis que os demais, por serem dependentes do número de ciclos dos grafos G das instâncias testadas.

Procedendo a uma análise entre as abordagens *early* e *late* dos três algoritmos aproximativos, através do cômputo dos intervalos de confiança sobre a média dos valores das soluções geradas, apresentados na tabela 4.4, conclui-se, com 99% de confiança, que as abordagens *early* e *late* são significativamente diferentes. Para os três algorit-

Tabela 4.2. Resultados para algoritmos aproximativos.

	$Sort_e$		$Sort_l$		$Greedy_e$		$Greedy_l$		$Degree_e$		$Degree_l$	
T_1	**S**	**%**	**S**	**%**	**S**	**%**	**S**	**%**	**S**	**%**	**S**	**%**
A	53.3	53.9	68.8	69.5	64.0	64.6	77.8	78.6	73.0	73.7	81.2	82.0
B	78.9	54.8	102.0	70.8	98.3	68.3	118.6	82.3	108.9	75.6	119.9	83.3
C	124.8	52.4	167.7	70.4	141.4	59.4	178.3	74.9	172.1	72.3	192.8	81.0
D	103.0	53.9	133.4	69.9	149.8	78.4	168.1	88.0	151.7	79.4	163.3	85.5
E	150.6	53.0	197.9	69.7	191.6	67.4	232.4	81.8	217.7	76.6	237.7	83.7
F	194.8	51.7	259.2	68.8	229.7	60.9	291.6	77.3	275.1	73.0	304.3	80.7
G	148.7	52.3	194.3	68.4	235.1	82.8	260.2	91.6	225.7	79.5	244.1	86.0
H	193.2	51.3	260.6	69.1	267.4	70.9	315.7	83.7	288.1	76.4	313.4	83.1
I	245.0	52.4	329.8	70.5	302.9	64.7	375.0	80.1	353.2	75.5	386.7	82.6

Tabela 4.3. Estatísticas sobre as porcentagens de arcos nas soluções.

	$Sort_e$	$Sort_l$	$Greedy_e$	$Greedy_l$	$Degree_e$	$Degree_l$
min	50.00	64.65	55.88	71.01	69.33	77.78
max	62.50	74.75	85.21	93.31	83.10	89.53
med	52.85	69.67	68.62	82.06	75.78	83.10
stdev	2.35	1.96	7.81	5.43	2.94	2.59
covar	0.04	0.03	0.11	0.07	0.04	0.03

mos aproximativos, uma vez que as médias de cada um dos algoritmos, segundo as duas abordagens, não fazem parte do intervalo de confiança de seus pares, revela a abordagem *late* como mais indicada, por gerar soluções com maior número de arcos, como já demonstrado teoricamente.

Fazendo uma análise entre os algoritmos $Sort_l(G, \bar{G})$, $Greedy_l(G, \bar{G})$ e $Degree_l(G, \bar{G})$, para a abordagem *late* que se demonstrou mais eficiente para o SAMRDN, baseada novamente nos intervalos de confiança sobre as médias das soluções apresentadas na tabela 4.4 tem-se, com 99% de confiança, que o algoritmo $Sort_l(G, \bar{G})$ é significativamente diferente dos demais, sendo o menos indicado para o conjunto de instâncias testadas (o valor médio das soluções não pertence aos intervalos de confiança dos demais algoritmos). Porém, ainda observando os intervalos de confiança sobre as médias das soluções, não se pode dizer que os algoritmos $Greedy_l(G, \bar{G})$ e $Degree_l(G, \bar{G})$ são significativamente diferentes, com 99% de confiança, uma vez que a média do algoritmo $Degree_l(G, \bar{G})$ faz parte do intervalo de confiança do algo-

ritmo $Greedy_l(G, \bar{G})$. Assim sendo, um novo intervalo de confiança deve ser feito sobre a média da diferença entre as soluções produzidas pelos algoritmos $Greedy_l(G, \bar{G})$ e $Degree_l(G, \bar{G})$ (teste t emparelhado). Os resultados estão apresentados na tabela 4.5.

Tabela 4.4. Intervalos de Confiança (99%) para a média.

IC	$Sort_e$	$Sort_l$	$Greedy_e$	$Greedy_l$	$Degree_e$	$Degree_l$
min	52.18	69.11	66.38	80.50	74.94	82.36
max	53.52	70.24	70.86	83.61	76.63	83.85

Como já era de se esperar, ao analisar a tabela 4.5, com 99% de confiança os algoritmos $Greedy_l(G, \bar{G})$ e $Degree_l(G, \bar{G})$ não são significativamente diferentes (o intervalo de confiança inclui o ponto zero). Porém, com uma confiança de 95%, pode-se afirmar que $Greedy_l(G, \bar{G})$ e $Degree_l(G, \bar{G})$ são significativamente diferentes, sendo o algoritmo $Degree_l(G, \bar{G})$ capaz de encontrar soluções de maiores cardinalidades para SAMRDN, para as instâncias testadas. O algoritmo $Greedy_l(G, \bar{G})$ possui maior dependência sobre o grafo de entrada, $G = (V, A)$, e o número de ciclos que este possui, para a geração do subconjunto S, sobre o qual é calculado um conjunto independente máximo em $\bar{G}$, retornado como solução.

Tabela 4.5. Estatísticas sobre a média da diferença das soluções.

$(Greedy_l - Degree_l)$		IC_{diff}	99%	95%
med	*1.05*	**min**	*-0.26*	*0.09*
stdev	4.53	**max**	2.36	2.01

Capítulo 5

Pré-processamento de Instâncias

Neste capítulo será proposto um pré-processamento das instâncias visando a redução de seu tamanho em termos de número de arcos do grafo G e número de arestas no grafo $\bar{G}$, para ambos os problemas, SAMRDN e SAMRDP. Após isto, será feita uma comparação via testes computacionais sobre as soluções geradas, antes e após a redução das instâncias, utilizando-se das formulações de programação linear inteira já apresentadas para os problemas.

Para proceder ao pré-processamento das instâncias e posterior comparação das soluções geradas, aquelas instâncias usadas para os algoritmos aproximativos, tratados na seção 4.3, não serão mais utilizadas, visto que já não se faz mais necessário o cálculo de um conjunto independente máximo em $\bar{G}$ em tempo polinomial.

Foram então geradas novas instâncias, denominadas T_2, em que os grafos G e $\bar{G}$ são completamente aleatórios e suas densidades são pré-estabelecidas. Para este novo conjunto de instâncias, o número de vértices do grafo G foi mantido dentro do intervalo das instâncias de Zhang et al. [2011], $n = (50, 100, 150, 200, 250, 300)$. As densidades do grafo G, que irão definir seu número de arcos, foram definidas como $d_G(\%) = (1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3.0)$, para cada conjunto de vértices, totalizando 42 instâncias. A densidade do grafo $\bar{G}$ foi pré-estabelecida em $d_{\bar{G}} = 0.5\%$, e define o número de restrições disjuntivas para as instâncias.

5.1 Redução de instâncias para SAMRDN

Considere o grafo $G = (V, A)$ e o grafo de conflitos $\bar{G} = (A, E)$. Seja $A' \subseteq A$ um conjunto máximo de arcos que forma a solução do problema tal que $G[A']$ seja acíclico e A' seja um conjunto independente em $\bar{G}$. Podem ser definidas algumas propriedades para o problema do Subgrafo Acíclico Máximo sob Restrições Disjuntivas Negativas.

Lema 5.1.1. *Se G não tem ciclos o problema se reduz ao cálculo do conjunto independente máximo em $\bar{G}$.*

Prova. A cardinalidade de um conjunto independente máximo I em $\bar{G}$ é um limite superior para o valor da solução do problema. Dado G acíclico, este conjunto é a solução ótima do problema, visto que $G[I]$ é acíclico. □

Lema 5.1.2. *Se $\bar{G}$ não tem arestas o problema se reduz a determinar o subgrafo acíclico máximo em G.*

Prova. Dado $E = \emptyset$, o problema SAMRDN se reduz ao SAM. Qualquer subconjunto A' tal que $G[A']$ seja acíclico constituirá um conjunto independente em $\bar{G}$, uma vez que todos seus vértices possuem grau nulo. □

Lema 5.1.3. *Se um arco a' não pertence a nenhum ciclo de G e tiver grau nulo em $\bar{G}$, a' pertence a toda solução ótima do problema.*

Prova. Suponha S uma solução ótima tal que $a' \notin S$. Considere outra solução $S' = S \cup \{a'\}$. Tem-se $G[S']$ acíclico, pois a' não cria nenhum ciclo em G, e S' é conjunto independente em $\bar{G}$, pois a' não tem vértices adjacentes. Logo $|S'| > |S|$ e S não é solução ótima. □

Lema 5.1.4. *Se um arco a' não pertence a nenhum ciclo de G e induz, junto com seus vértices vizinhos, uma clique em $\bar{G}$, sempre existirá uma solução ótima que inclui o arco a'.*

Prova. Seja C o conjunto de vértices adjacentes a a' em $\bar{G}$. Seja A' uma solução ótima do problema tal que $a' \notin A'$. Se $\nexists\ c' \in C$ tal que $c' \in A'$, então considere $A'' = A' \cup a'$. Tem-se que A'' é conjunto independente em $\bar{G}$ e $G[A'']$ é acíclico, pois a' não cria nenhum ciclo em G. Logo, tem-se $|A''| > |A'|$, o que é um absurdo. Caso contrário, se $\exists\ c' \in C$ tal que $c' \in A'$, então considere $A'' = A' \cup \{a'\} - \{c'\}$. $G[A'']$ é acíclico e A'' é conjunto independente em $\bar{G}$, já que nenhum outro vértice de C pode estar em A'', e $|A''| = |A'|$. Portanto, A'' é solução ótima do problema.

□

Lema 5.1.5. *Se para todo ciclo $c = v_i a_j v_j a_k v_k \dots v_w a_{w+1} v_i$ de G, existem arcos a_i e a_j pertencentes a c tais que a_i e a_j são adjacentes em $\bar{G}$, o problema se reduz ao cálculo do conjunto independente em $\bar{G}$.*

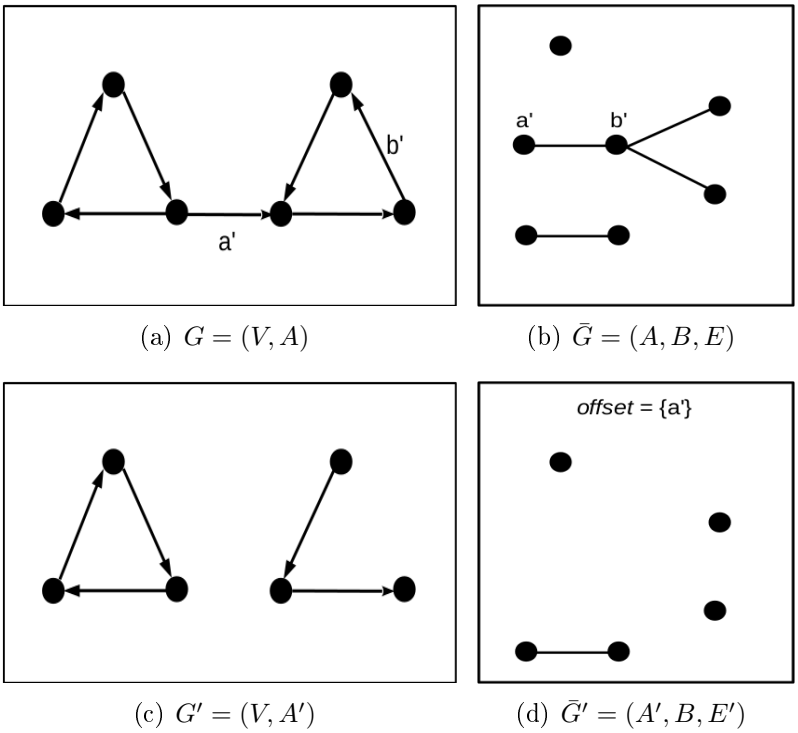
Prova. Um conjunto independente I em $\bar{G}$ incluirá no máximo um dentre os vértices a_i e a_j, por serem adjacentes, logo $G[I]$ será acíclico. □

5.1.1 Testes de Redução

Seja uma instância de SAMRDN composta pelos grafos $G = (V, A)$ e $\bar{G} = (A, E)$. Considere SAMRDNG (Subgrafo Acíclico Máximo sob Restrições Disjuntivas Negativas Generalizado) uma generalização de SAMRDN. SAMRDNG pode ser definido por um grafo direcionado $G = (V, A)$ e um grafo de conflitos $\bar{G} = (A, B, E)$, no qual seu conjunto de vértices é definido por $A \cup B$. O conjunto B inclui todos os vértices de $\bar{G}$ que não possuem arcos correspondentes em G. Se $B = \emptyset$, então a instância de SAMRDNG é a mesma de SAMRDN.

Considere *offset* um conjunto composto por arcos pré-selecionados a fazerem parte de uma solução de SAMRDNG, que são então removidos do conjunto A. Com base nos Lemas 5.1.1–5.1.5, têm-se as definições dos seguintes testes de redução:

(T1) Se um arco $a' \in A$ não pertence a nenhum ciclo de G e a' induz, junto com os seus vértices vizinhos, uma clique em $\bar{G}$, denominada $clique_{\bar{G}}(a') = a' \cup adj_{\bar{G}}(a')$, em que $adj_{\bar{G}}(a')$ representa todos os vértices adjacentes a a' em $\bar{G}$, então a solução de SAMRDNG é equivalente se acrescentarmos previamente o arco a' ao conjunto *offset* e fazer $G' = (V, A')$, $\bar{G}' = (A', B, E')$, em que $A' = A - clique_{\bar{G}}(a')$ e $E' = E - \bar{G}[clique_{\bar{G}}(a')]$. Para um exemplo, veja a Figura 5.1.

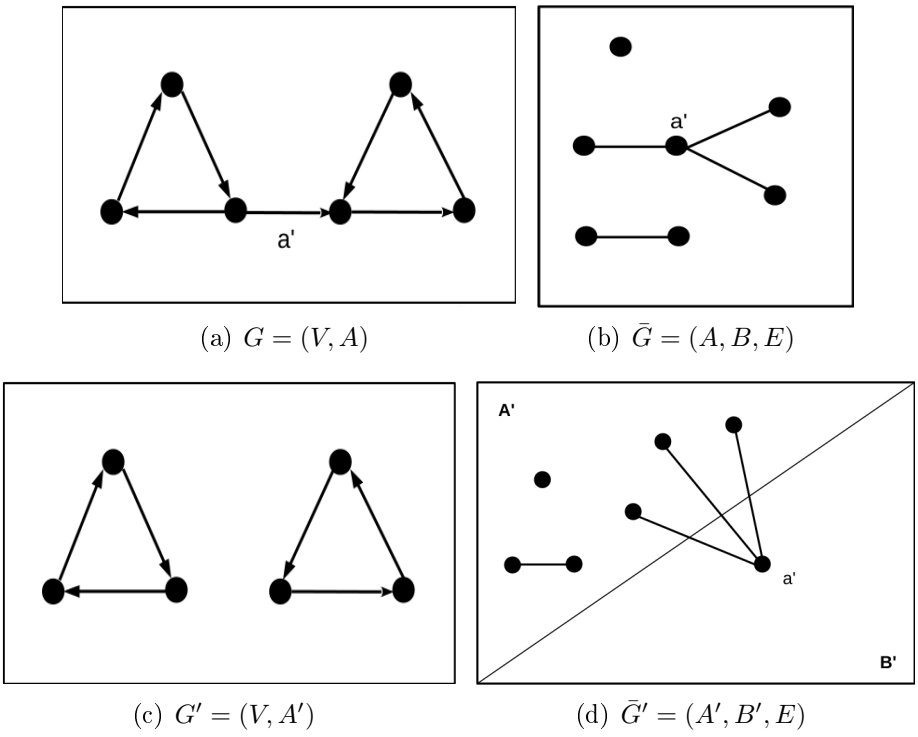


(a) $G = (V, A)$ (b) $\bar{G} = (A, B, E)$

(c) $G' = (V, A')$ (d) $\bar{G}' = (A', B, E')$

Figura 5.1. (T1) aplicado a SAMRDNG.

(T2) Se um arco $a' \in A$ não pertence a nenhum ciclo de G, porém a' não induz, junto com os seus vértices vizinhos, uma clique em $\bar{G}$, então a solução de SAMRDNG é equivalente fazendo-se $G' = (V, A')$ e $\bar{G}' = (A', B', E)$, em que $A' = A - \{a'\}$ e $B' = B \cup \{a'\}$. Para um exemplo, veja a Figura 5.2.



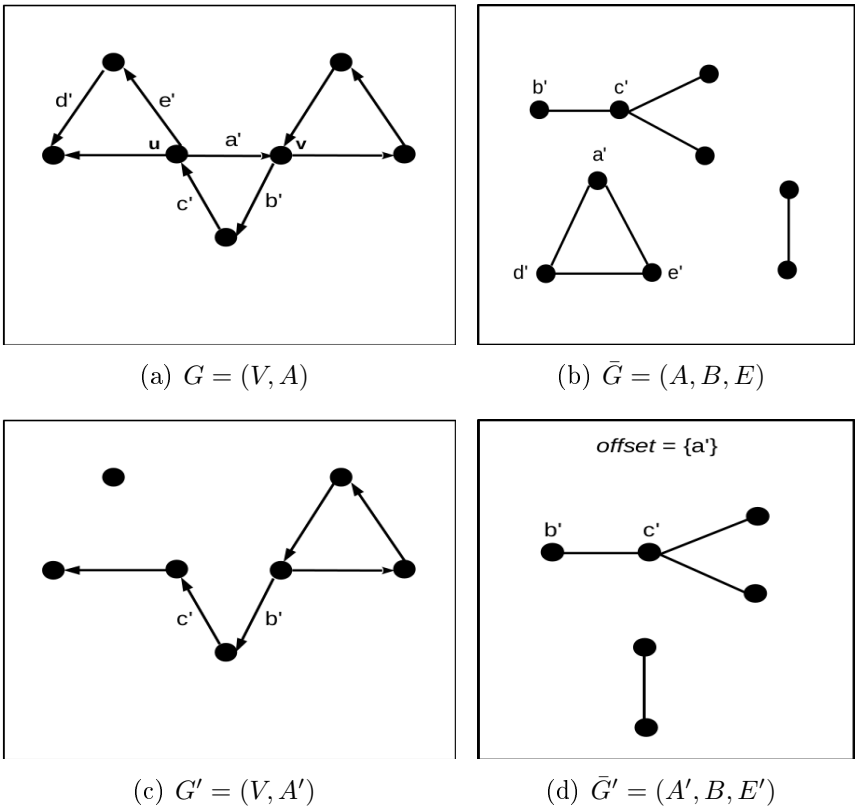
(a) $G = (V, A)$ (b) $\bar{G} = (A, B, E)$

(c) $G' = (V, A')$ (d) $\bar{G}' = (A', B', E)$

Figura 5.2. (T2) aplicado a SAMRDNG.

(T3) Se para um arco $a' = (u, v) \in A$ identificou-se a não existência de caminhos de v para u ($v \rightsquigarrow u$) livres de conflitos entre pares de arcos pertencentes ao caminho, e a' induz, junto com os seus vértices vizinhos, uma clique em $\bar{G}$, denominada $clique_{\bar{G}}(a')$, então a solução de SAMRDNG é equivalente se acrescentarmos previamente o arco a' a *offset* e fazer $G' = (V, A')$, $\bar{G}' = (A', B, E')$, em que $A' = A - clique_{\bar{G}}(a')$ e $E' = E - \bar{G}[clique_{\bar{G}}(a')]$. Para um exemplo, veja a Figura 5.3.

(T4) Se para um arco $a' = (u, v) \in A$ identificou-se a não existência de caminhos de $v \rightsquigarrow u$ livres de conflitos entre pares de arcos pertencentes ao caminho, porém a' não induz, junto com os seus vértices vizinhos, uma clique em $\bar{G}$, então a solução de SAMRDNG é equivalente fazendo-se $G' = (V, A')$ e $\bar{G}' = (A', B', E)$, em que $A' = A - a'$ e $B' = B + a'$. Para um exemplo, veja a Figura 5.4.



(a) $G = (V, A)$ (b) $\bar{G} = (A, B, E)$

(c) $G' = (V, A')$ (d) $\bar{G}' = (A', B, E')$

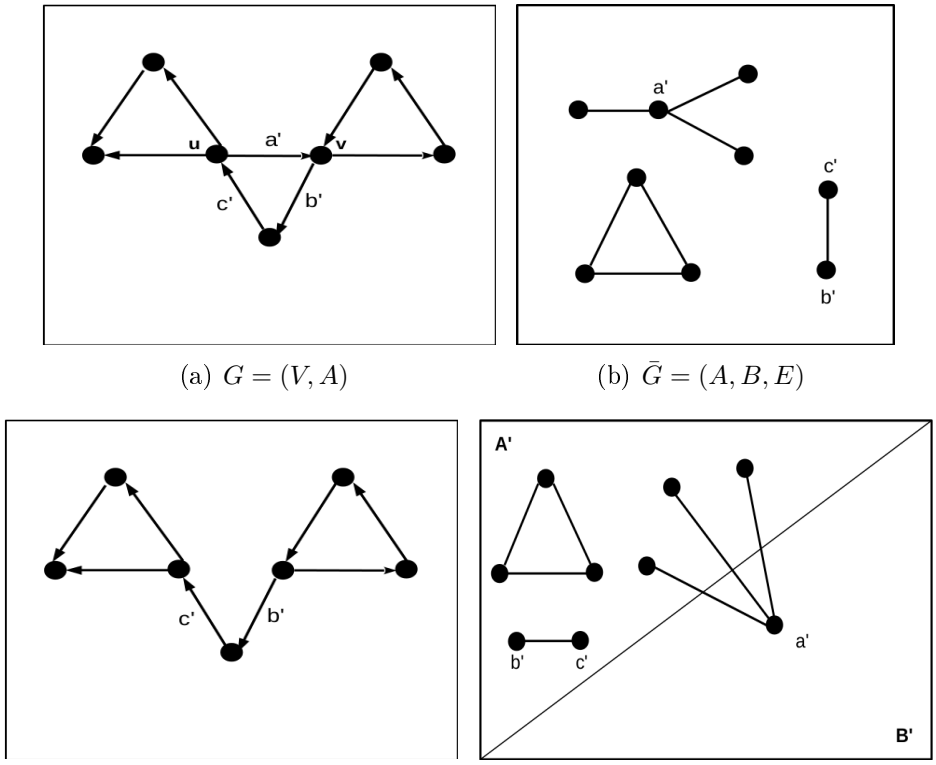
Figura 5.3. (T3) aplicado a SAMRDNG.

(T5) Se um vértice $b' \in B$ induz, junto com os seus vértices vizinhos, uma clique em $\bar{G}$, denominada $clique_{\bar{G}}(b')$, então a solução de SAMRDNG é equivalente acrescentando-se previamente o arco b' a *offset* e fazendo $G' = (V, A')$, $\bar{G}' = (A', B', E')$, em que $A' = A - clique_{\bar{G}}(b')$, $B' = B - clique_{\bar{G}}(b')$ e $E' = E - \bar{G}[clique_{\bar{G}}(b')]$. Para um exemplo, veja a Figura 5.5.

5.1.2 Programação linear inteira para SAMRDNG

Seja $G = (V, A)$ e $\bar{G} = (A, B, E)$, no qual seu conjunto de vértices é definido por $A^+ = A \cup B$, uma instância de SAMRDNG. O modelo de programação linear inteira adaptado para SAMRDNG pode ser expresso pela variável de decisão x_{ij} definindo os arcos $A' \in A^+$ que farão parte da solução, modelada nas equações 5.1 e 5.2, sujeito às restrições presentes em 5.3 – 5.6. Assim como no modelo para SAMRDN, a variável y_i atribuirá um número real a cada vértice $i \in V$.

$$x_{ij} = \begin{cases} 1, & \text{se } (i,j) \in A', \qquad \forall \quad (i,j) \in A^+ \\ 0, & \text{caso contrário.} \end{cases} \tag{5.1}$$



(a) $G = (V, A)$ (b) $\bar{G} = (A, B, E)$

(c) $G' = (V, A')$ (d) $\bar{G}' = (A', B', E)$

Figura 5.4. (T4) aplicado a SAMRDNG.

$$max \sum_{(i,j)\in A^+} x_{ij} \tag{5.2}$$

sujeito a:

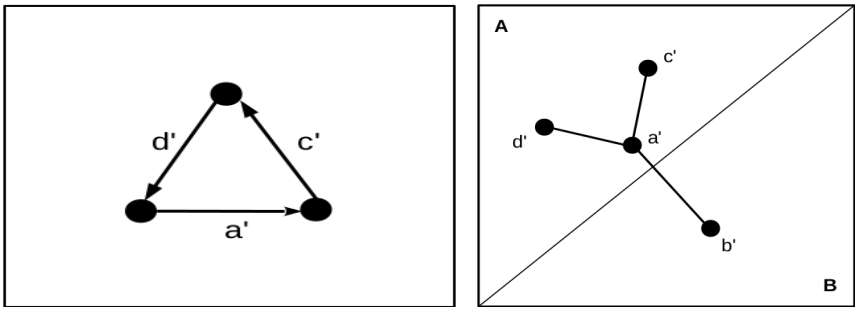
$$y_j - y_i \geq (1+n)x_{ij} - n \qquad \forall \quad (i,j) \in A, \quad i \neq j \tag{5.3}$$

$$x_{ij} + x_{kl} \leq 1 \qquad \forall \quad \{(i,j),(k,l)\} \in E \tag{5.4}$$

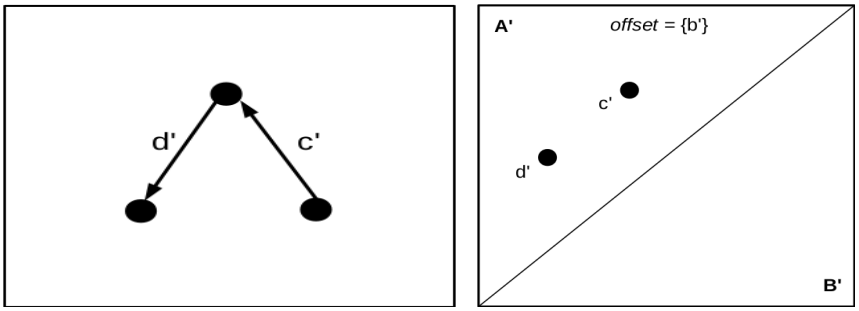
$$y_i = \{y_i \in \mathbb{R} : 0 \leq y_i \leq n\} \qquad \forall \quad i \in V \tag{5.5}$$

$$x_{ij} \in \{0,1\} \qquad \forall \quad (i,j) \in A^+ \tag{5.6}$$

A equação 5.2 representa a função objetivo do problema. A restrição 5.3 garante que, para todo arco $(i,j) \in A$, se $x_{ij} = 1$ então $y_j - y_i \geq 1$, e é suficiente para prevenir a ocorrência de ciclos em G. Repare que para esta restrição, os arcos $a = (i,j) \in B$, que não mais fazem parte de G e que ainda podem fazer parte da solução do problema, são obviamente excluídos da restrição de aciclicidade em G.



(a) $G = (V, A)$ (b) $\bar{G} = (A, B, E)$



(c) $G' = (V, A')$ (d) $\bar{G}' = (A', B', E')$

Figura 5.5. (T5) aplicado a SAMRDNG.

Assim como para SAMRDN, a restrição 5.4 admite que no máximo um dos dois arcos da restrição disjuntiva negativa esteja presente na solução. A restrição dada em 5.5 define a variável y_i como um número real não negativo e a restrição 5.6 garante a integralidade da solução do problema.

5.1.3 Implementação para pré-processamento de instâncias em SAMRDN

Com base nos testes (T1)–(T5) apresentadas anteriormente para SAMRDNG, foi possível proceder à redução das instâncias com relação ao número de arcos e número de conflitos, apresentada no algoritmo $Reduz_{SAMRDNG}(G, \bar{G})$.

Primeiramente, deve-se identificar se um arco $a \in A$ não pertence a nenhum ciclo de G. Para isto, usou-se o conceito de componentes fortemente conexas *(cfc)* sobre o grafo direcionado G. Se para o arco $a = (u, v)$, os vértices u e v pertencem a *cfc* distintas, então o arco a não pertence a nenhum ciclo de G. Para a detecção de componentes fortemente conexas no grafo G, usou-se os algoritmos *DFS(G)* e *STRONGLY-CONNECTED-COMPONENTS(G)*, apresentados por exemplo em Cormen et al. [2009]. No algoritmo $Reduz_{SAMRDNG}(G, \bar{G})$, esta condição será denominada

$cond_1$.

Além disto, deve-se também identificar se um arco $a = (u, v) \in A$ participa somente de caminhos $v \rightsquigarrow u$ nos quais pelo menos um dos arcos que compõem estes caminhos está em conflito com a. Para este procedimento, a partir do mesmo arco $a = (u, v)$, pode ser verificada a existência de um caminho c_v a partir de v, dada a seguinte condição: se v possui grau de saída $\delta^-(v) = 1$, então o arco (v, w) fará parte de todo ciclo que inclua (u, v). O arco (v, w) passa a compor o caminho c_v e o vértice w passa a ser denominado v. Esta condição se repete até que $\delta^-(v) \neq 1$, ou $u = v$.

Da mesma forma, após identificado o caminho c_v e se $u \neq v$, será averiguada a existência de um caminho c_u a partir de u, dada a seguinte condição: se u possui grau de entrada $\delta^+(u) = 1$, então o arco (w, u) fará parte de todo ciclo que inclua $a = (u, v)$. O arco (w, u) passa a compor o caminho c_u e o vértice w passa a ser denominado u. Esta condição se repete até que $\delta^+(u) \neq 1$, ou $u = v$. Tem-se então um caminho $C = c_u \cup \{a\} \cup c_v$.

Se existe conflito entre dois arcos pertencentes a C, conclui-se que não há ciclos que incluam o arco $a = (u, v)$ livres de conflitos entre pares de arcos. Caso contrário, deve-se identificar a não existência de caminhos $P = v \rightsquigarrow u$ livres de conflitos, ou seja, pelo menos um dos arcos de C, denominado c', deverá estar em conflito com no mínimo um dos arcos de P, denominado p'. Em outras palavras, deve-se verificar se existem vértices c' e p' adjacentes em $\bar{G}$, para todo caminho $P = v \rightsquigarrow u$.

Para este procedimento, dado um arco $a = (u, v)$ e determinado o caminho $C = u \rightsquigarrow v$, apaga-se temporariamente do grafo G todos os arcos em conflito com os arcos de C e, a partir do vértice v, aplica-se *DFS(G)*. Se o vértice u foi alcançado na execução de *DFS(G)*, então a condição não foi satisfeita, ou seja, não foi provada a inexistência de caminhos $P = v \rightsquigarrow u$ livres de conflitos entre pares de arcos. Esta condição será denominada $cond_2$ no algoritmo utilizado para a redução de instâncias.

Vale ressaltar que, para o procedimento apresentado acima ($cond_2$), ainda que o vértice u seja alcançado, pode ser que não exista o caminho $P = v \rightsquigarrow u$, pois o procedimento não é capaz de detectar conflitos entre pares de arcos de P ($p'_1 \in P$ e $p'_2 \in P$), tal que estes arcos não pertençam ao caminho C ($p'_1 \notin C$ e $p'_2 \notin C$). Para que esta condição seja detectada, há de se resolver o problema de existência de caminho entre o vértice inicial v e o vértice final u sob restrições disjuntivas negativas. Este problema é provado $\mathcal{NP}$–Completo [Kann, 1994].

O pseudo-código para $Reduz_{SAMRDNG}(G, \bar{G})$ apresenta, em suas linhas 5–16, os testes de redução (T1)–(T4), para cada arco $a = (i, j) \in A$. Se uma das condições, $cond_1$ ou $cond_2$, for satisfeita para algum arco (linha 17), gerando uma redução da instância, então o algoritmo, em suas linhas 19–29, executa o teste (T5) para cada vértice

$b = (i, j) \in B$, iterativamente enquanto algum destes vértices induzirem, juntamente com seus vértices vizinhos, uma clique em $\bar{G}$. O algoritmo executa iterativamente (linhas 3–31) todos os testes de redução enquanto uma das condições, $cond_1$ ou $cond_2$, for satisfeita para algum arco $a \in A$. Então o algoritmo retorna os grafos $G = (V, A)$ e $\bar{G} = (A, B, E)$, após os testes de redução (T1)–(T5), além do conjunto *offset*, que contém arcos pré-selecionados a comporem uma solução de SAMRDNG (linha 32).

Algorithm 1 $Reduz_{SAMRDNG}(G = (V, A), \bar{G} = (A, B, E))$

```
1:  offset ← ∅;
2:  test1 ← true;
3:  while (test1) do
4:      test1 ← false;
5:      for (each arc a = (i, j) ∈ A) do
6:          if (cond_1 || cond_2) then
7:              test1 ← true;
8:              if (a ∪ adj_Ḡ(a) is clique) then
9:                  G ← G − clique_Ḡ(a);
10:                 Ḡ ← Ḡ − clique_Ḡ(a);
11:                 offset ← a;
12:             else
13:                 G ← G − a;
14:                 B ← a;
15:             end if
16:         end if
17:     end for
18:     if (test1) then
19:         test2 ← true;
20:         while (test2) do
21:             test2 ← false;
22:             for (each vertice b = (i, j) ∈ B) do
23:                 if (b ∪ adj_Ḡ(b) is clique) then
24:                     G ← G − clique_Ḡ(b);
25:                     Ḡ ← Ḡ − clique_Ḡ(b);
26:                     B ← B − clique_Ḡ(b);
27:                     offset ← b;
28:                     test2 ← true;
29:                 end if
30:             end for
31:         end while
32:     end if
33: end while
34: return (G, Ḡ, offset)
```

5.1.4 Resultados computacionais para o pré-processamento das instâncias em SAMRDN

Após a execução do Algoritmo $Reduz_{SAMRDNG}(G, \bar{G})$, considerando $G = (V, A)$ e $\bar{G} = (A, B, E)$ uma instância de SAMRDN transformada em uma instância de SAMRDNG, na qual $B = \emptyset$ como entrada para o algoritmo, e considerando $G' = (V, A')$ e $\bar{G}' = (A', B', E')$ uma instância de SAMRDNG retornada pelo algoritmo, além do conjunto *offset*, obteve-se os resultados dos testes de redução sobre as instâncias T_2, relatados na Tabela 5.1.

Nesta tabela, considere $R_A(\%) = 100 * (|A| - |A'|)/|A|$ a porcentagem da redução sobre a cardinalidade do conjunto de arcos A, e $R_E(\%) = 100 * (|E| - |E'|)/|E|$ a porcentagem da redução sobre o número de conflitos $|E|$. O número de arcos inseridos previamente na solução de SAMRDNG com a redução estão especificados em ***offset*** e $t_r(sec)$ é o tempo consumido na execução do algoritmo, em segundos.

Pela análise da Tabela 5.1 percebe-se que, quanto maior o número de vértices da instância, e consequentemente maior o número de arcos e de conflitos (por serem definidos conforme densidades pré-estabelecidas dos grafos G e $\bar{G}$, respectivamente), menor é o efeito da redução. Além disto, no geral, conforme as densidades do grafo G, definidas como $d_G(\%) = (1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3.0)$, vão aumentando, mantendo-se constante o número de vértices, os efeitos da redução também são menores. Os tempos de processamento ficam mais elevados conforme se aumenta o número de vértices e densidades dos grafos das instâncias.

A partir da Tabela 5.2 percebe-se o efeito do pré-processamento, para os conjuntos de instâncias que possuem os conjuntos de vértices com mesmas cardinalidades, com relação às médias das reduções sobre o número de arcos ($\mu_{R_A(\%)}$) e de conflitos ($\mu_{R_E(\%)}$) das instâncias T_2, confirmando a análise feita previamente.

5.2 Redução de instâncias para SAMRDP

Considere o grafo $G = (V, A)$ e o grafo de coação $\bar{G} = (A, E)$ uma instância de SAMRDP. Seja $A' \subseteq A$ um conjunto máximo de arcos que forma a solução do problema tal que $G[A']$ seja acíclico e A' seja uma cobertura por vértices em $\bar{G}$. Podem ser definidas as seguintes propriedades para o problema do Subgrafo Acíclico Máximo sob Restrições Disjuntivas Positivas.

Tabela 5.1. Resultados da redução de SAMRDNG para T_2.

SAMRDN			**SAMRDNG**			**Porcentagens da Redução**		**Solução**	
|V|	**|A|**	**|E|**	**|A'|**	**|B'|**	**|E'|**	$R_A(\%)$	$R_E(\%)$	***offset***	$t_r(sec)$
50	37	4	0	0	0	100.00	100.00	**34**	0.0005
50	43	5	0	0	0	100.00	100.00	**39**	0.0006
50	49	6	2	0	0	95.92	100.00	**41**	0.0008
50	56	8	5	0	0	91.07	100.00	**46**	0.0016
50	62	10	20	1	3	67.74	70.00	**36**	0.0087
50	68	12	0	0	0	100.00	100.00	**60**	0.0012
50	74	14	2	0	0	97.30	100.00	**61**	0.0020
100	149	56	8	1	2	94.63	96.43	**107**	0.0202
100	174	76	0	0	0	100.00	100.00	**126**	0.0521
100	198	98	77	8	31	61.11	68.37	**79**	0.1660
100	223	124	157	11	90	29.60	27.42	**37**	0.5170
100	248	154	164	15	100	33.87	35.06	**43**	0.5280
100	273	186	233	10	164	14.65	11.83	**19**	0.5930
100	297	220	274	6	211	7.74	4.09	**11**	0.8220
150	336	282	199	46	191	40.77	32.27	**56**	1.2300
150	392	384	279	34	297	28.83	22.66	**47**	1.8200
150	447	499	360	45	443	19.46	11.22	**24**	3.0800
150	503	632	470	22	616	6.56	2.53	**6**	3.3500
150	559	780	516	29	756	7.69	3.08	**8**	5.9700
150	615	945	609	2	934	0.98	1.16	**2**	5.5600
150	671	1124	662	7	1116	1.34	0.71	**1**	6.4900
200	597	890	524	56	861	12.23	3.26	**10**	5.4300
200	697	1213	665	28	1208	4.59	0.41	**3**	9.3100
200	796	1583	764	29	1579	4.02	0.25	**2**	11.2000
200	896	2005	888	8	2005	0.89	0.00	**0**	15.1000
200	995	2473	967	24	2462	2.81	0.44	**2**	18.4000
200	1095	2995	1084	11	2995	1.00	0.00	**0**	22.1000
200	1194	3562	1194	0	3562	0.00	0.00	**0**	13.5000
250	934	2179	891	38	2161	4.60	0.83	**3**	19.1000
250	1090	2968	1077	11	2963	1.19	0.17	**1**	28.3000
250	1245	3872	1244	1	3872	0.08	0.00	**0**	38.2000
250	1401	4904	1392	9	4904	0.64	0.00	**0**	47.2000
250	1557	6057	1537	20	6057	1.28	0.00	**0**	56.6000
250	1712	7324	1700	12	7324	0.70	0.00	**0**	67.2000
250	1868	8719	1868	0	8719	0.00	0.00	**0**	40.4000
300	1346	4526	1317	27	4520	2.15	0.13	**1**	49.3000
300	1570	6159	1560	10	6159	0.64	0.00	**0**	68.7000
300	1794	8042	1787	7	8042	0.39	0.00	**0**	88.3000
300	2019	10186	2015	4	10186	0.20	0.00	**0**	117.0000
300	2243	12573	2243	0	12573	0.00	0.00	**0**	74.4000
300	2467	15210	2467	0	15210	0.00	0.00	**0**	85.7000
300	2691	18097	2691	0	18097	0.00	0.00	**0**	100.000

Tabela 5.2. Médias para redução em T_2 para SAMRDNG.

$\|V\|$	**50**	**100**	**150**	**200**	**250**	**300**
$\mu_{R_A(\%)}$	93.15	48.80	15.09	3.65	1.21	0.48
$\mu_{R_E(\%)}$	95.71	49.03	10.52	0.62	0.14	0.02

Lema 5.2.1. *Se G não tem ciclos, o conjunto de arcos A é a solução ótima do problema*

Prova. Conjunto A é a cobertura por vértices máxima em $\bar{G}$ e $G[A]$ é acíclico. □

Lema 5.2.2. *Se $\bar{G}$ não tem arestas o problema se reduz a determinar o subgrafo acíclico máximo em G.*

Prova. Se $E = \emptyset$, SAMRDP se reduz ao *SAM*, pois não há arestas em $\bar{G}$ a serem cobertas. □

Lema 5.2.3. *Se um arco a' não pertence a nenhum ciclo de G, a' sempre pertencerá a uma solução.*

Prova. Suponha S uma solução ótima tal que $a' \notin S$. Considere $S' = S \cup \{a'\}$ outra solução. Tem-se $G[S']$ acíclico, pois a' não cria ciclos, e S' é uma cobertura por vértices em $\bar{G}$. Logo S' é solução de SAMRDP, e $|S'| > |S|$. Absurdo. □

Usando o Lema 5.2.3, é possível promover a redução do tamanho da instância em termos de número de arcos e de conflitos. Pode-se dizer que, se um arco $a' \in A$ não pertence a nenhum ciclo de G, a solução de SAMRDP é equivalente acrescentando-se previamente o arco a' ao conjunto *offset*, assim como para SAMRDNG, e fazendo-se $G' = (V, A')$ e $\bar{G}' = (A', E')$, em que $G' = G - \{a'\}$ e $\bar{G}' = \bar{G} - \{a'\}$.

O algoritmo $Reduz_{SAMRDP}(G, \bar{G})$ apresenta a redução proposta para SAMRDP. O teste $cond_1$ (linha 2) é o mesmo utilizado no algoritmo $Reduz_{SAMRDNG}(G, \bar{G})$.

Algorithm 2 $Reduz_{SAMRDP}(G = (V, A), \bar{G} = (A, E))$

```
1: for (each arc a = (i, j) ∈ A) do
2:     if (cond_1) then
3:         G ← G − a;
4:         Ḡ ← Ḡ − a;
5:         offset ← a;
6:     end if
7: end for
8: return (G, Ḡ, offset)
```

5.2.1 Resultados computacionais para o pré-processamento das instâncias em SAMRDP

Após a execução do algoritmo $Reduz_{SAMRDP}(G, \bar{G})$, considerando $G = (V, A)$ e $\bar{G} = (A, E)$ uma instância de SAMRDP, obtêm-se $G' = (V, A')$, $\bar{G}' = (A', E')$ e *offset*. Os resultados da redução sobre as instâncias T_2 são relatados na Tabela 5.3. Note que $R_A(\%)$, $R_E(\%)$, ***offset*** e $t_r(sec)$ são tais como definidos na seção 5.1.4 para $Reduz_{SAMRDNG}(G, \bar{G})$. Considere RSAMRDP a instância de SAMRDP retornada pelo algoritmo.

Analisando a Tabela 5.3 percebe-se novamente que, assim como para SAMRDNG, quanto maiores os números de vértices das instâncias, e consequentemente maiores os números de arcos e de conflitos, menor é o efeito da redução. Além disto, quanto maiores as densidades dos grafos G, definidas como $d_G(\%) = (1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3.0)$, para o mesmo conjunto de vértices V, menores também são os efeitos da redução. A Tabela 5.4 apresenta as médias das reduções dos números de arcos e de conflitos para cada conjunto de instâncias que possuem as mesmas cardinalidades dos conjuntos de vértices V, que corrobora esta análise.

5.3 Resultados computacionais para a otimização

Para proceder à otimização das instâncias T_2 segundo SAMRDNG e SAMRDP, foram utilizadas as modelagens matemáticas de programação linear inteira já apresentadas para ambos os problemas. Os algoritmos desenvolvidos foram implementados em C++, em interface com o Concert API do CPLEX 12.5. Todas as opções de pré-processamentos, heurísticas e geração de cortes do resolvedor foram desativados. Os experimentos foram realizados em uma máquina *Intel Core* i7 980 (3.33GHz), com 24GB de RAM. Para todos os experimentos, o tempo limite foi estabelecido em 1800 segundos.

Ao analisar as tabelas 5.5 e 5.7, as três primeiras colunas se referem aos dados das instâncias, sendo $|V|$, $|A|$ e $|E|$ as cardinalidades dos conjuntos de vértices, arcos e de restrições disjuntivas, respectivamente. Considere **BI** *(Best Integer)* o melhor valor da solução encontrada pelo resolvedor, **BB** *(Best Bound)* o limite superior para a solução encontrado pelo resolvedor até o momento no qual a execução do algoritmo é interrompida, **GAP** a porcentagem da diferença entre **BI** e **BB**, **t(sec)** o tempo de processamento (em segundos) no software CPLEX e t_{rc}**(sec)** a soma dos tempos de pré-processamento da instância e da otimização no CPLEX. A última coluna **BI+offset** apresenta a soma de **BI** após a redução com os arcos adicionados previamente à solu-

Tabela 5.3. Resultados da redução de SAMRDP para T_2.

SAMRDP			**RSAMRDP**		**Porcentagens da Redução**		**Solução**	
|V|	**|A|**	**|E|**	**|A'|**	**|E'|**	$R_A(\%)$	$R_E(\%)$	**offset**	t_r **(sec)**
50	37	4	0	0	100.00	100.00	**37**	0.0005
50	43	5	0	0	100.00	100.00	**43**	0.0006
50	49	6	2	0	95.92	100.00	**47**	0.0006
50	56	8	13	2	76.79	75.00	**43**	0.0007
50	62	10	33	4	46.77	60.00	**29**	0.0007
50	68	12	6	0	91.18	100.00	**62**	0.0007
50	74	14	12	2	83.78	85.71	**62**	0.0008
100	149	56	40	3	73.15	94.64	**109**	0.0038
100	174	76	102	26	41.38	65.79	**72**	0.0046
100	198	98	137	53	30.81	45.92	**61**	0.0050
100	223	124	189	94	15.25	24.19	**34**	0.0054
100	248	154	217	110	12.50	28.57	**31**	0.0063
100	273	186	244	148	10.62	20.43	**29**	0.0068
100	297	220	280	197	5.72	10.45	**17**	0.0053
150	336	282	241	137	28.27	51.42	**95**	0.0083
150	392	384	308	242	21.43	36.98	**84**	0.0112
150	447	499	387	375	13.42	24.85	**60**	0.0128
150	503	632	475	557	5.57	11.87	**28**	0.0129
150	559	780	525	689	6.08	11.67	**34**	0.0098
150	615	945	611	938	0.65	0.74	**4**	0.0147
150	671	1124	663	1095	1.19	2.58	**8**	0.0154
200	597	890	531	683	11.06	23.26	**66**	0.0191
200	697	1213	667	1116	4.30	8.00	**30**	0.0165
200	796	1583	765	1455	3.89	8.09	**31**	0.0214
200	896	2005	888	1966	0.89	1.95	**8**	0.0212
200	995	2473	969	2338	2.61	5.46	**26**	0.0246
200	1095	2995	1084	2938	1.00	1.90	**11**	0.0274
200	1194	3562	1194	3562	0.00	0.00	**0**	0.0260
250	934	2179	893	2019	4.39	7.34	**41**	0.0254
250	1090	2968	1078	2902	1.10	2.22	**12**	0.0309
250	1245	3872	1245	3872	0.00	0.00	**0**	0.0332
250	1401	4904	1392	4838	0.64	1.35	**9**	0.0376
250	1557	6057	1537	5933	1.28	2.05	**20**	0.0408
250	1712	7324	1700	7241	0.70	1.13	**12**	0.0460
250	1868	8719	1868	8719	0.00	0.00	**0**	0.0487
300	1346	4526	1318	4328	2.08	4.37	**28**	0.0434
300	1570	6159	1560	6081	0.64	1.27	**10**	0.0543
300	1794	8042	1787	7974	0.39	0.85	**7**	0.0610
300	2019	10186	2015	10155	0.20	0.30	**4**	0.0635
300	2243	12573	2243	12573	0.00	0.00	**0**	0.0742
300	2467	15210	2467	15210	0.00	0.00	**0**	0.0777
300	2691	18097	2691	18097	0.00	0.00	**0**	0.0835

Tabela 5.4. Médias para redução em T_2 para SAMRDP.

$\|\mathbf{V}\|$	**50**	**100**	**150**	**200**	**250**	**300**
$\mu_{R_A(\%)}$	84.92	27.06	10.95	3.39	1.16	0.47
$\mu_{R_E(\%)}$	88.67	41.43	20.01	6.95	2.01	0.97

ção pelo pré-processamento da instância, especificados em **offset**, já apresentados em tabelas anteriores.

5.3.1 Comparação das soluções de SAMRDNG antes e após o pré-processamento da instância

Os resultados da formulação matemática de programação linear inteira para SAMRDNG sobre as instâncias T_2 estão presentes na Tabela 5.5. Para algumas instâncias, sequer é necessário a execução de SAMRDNG após a redução no CPLEX, visto que durante o pré-processamento da instância já se tornou possível conhecer o valor ótimo da solução, como por exemplo para a instância $|V| = 100$, $|A| = 174$ e $|E| = 76$. Os valores de **offset** são aqueles já apresentados na Tabela 5.1.

Ao analisar os intervalos de confiança (IC_{dif}) para a média da diferença entre as soluções de SAMRDNG, antes e após a redução, obtidos pelo teste t emparelhado, presentes na tabela 5.6, com 95% de confiança as soluções obtidas não são significativamente diferentes, pois o intervalo inclui o ponto nulo. Porém, com uma confiança de 90%, pode-se afirmar que as otimizações para SAMRDNG, antes e após a redução das instâncias, são significativamente diferentes, sendo a modelagem de SAMRDNG após o pré-processamento de instâncias mais eficiente para encontrar soluções de maiores cardinalidades e em menores intervalos de tempo para o problema. Os valores médios de GAP são 28.42 e 27.66, para antes e após a redução, respectivamente. Porém, analisando individualmente algumas instâncias, como para $|V| = 150$, $|A| = 615$ e $|E| = 945$, o valor da solução antes da redução (309) é maior que após (308).

5.3.2 Comparação das soluções de SAMRDP antes e após o pré-processamento da instância

Os resultados da formulação de programação linear inteira para SAMRDP sobre as instâncias T_2 estão presentes na Tabela 5.7. Para boa parte destas instâncias, sequer a existência de uma solução é conhecida, dentro do tempo limite de processamento,

Tabela 5.5. Resultados de otimização para SAMRDNG sobre instâncias T_2.

Instância			**CPLEX (antes da redução)**				**CPLEX(após a redução)**				**BI+**
|V|	**|A|**	**|E|**	**BI**	**BB**	**GAP**	**t(sec)**	**BI**	**BB**	**GAP**	$t_{rc}(sec)$	**offset**
50	37	4	**34**	34.00	0.00	0.00	0	-	0.00	0.001	**34**
50	43	5	**39**	39.00	0.00	0.00	0	-	0.00	0.001	**39**
50	49	6	**42**	42.00	0.00	0.00	1	1.00	0.00	0.001	**42**
50	56	8	**49**	49.00	0.00	0.00	3	3.00	0.00	0.002	**49**
50	62	10	**53**	53.00	0.00	0.01	17	17.00	0.00	0.009	**53**
50	68	12	**60**	60.00	0.00	0.00	0	-	0.00	0.001	**60**
50	74	14	**62**	62.00	0.00	0.00	1	1.00	0.00	0.002	**62**
100	149	56	**114**	114.00	0.00	0.00	7	7.00	0.00	0.020	**114**
100	174	76	**126**	126.00	0.00	0.00	0	-	0.00	0.052	**126**
100	198	98	**140**	140.00	0.00	0.02	61	61.00	0.00	0.176	**140**
100	223	124	**159**	159.00	0.00	0.00	122	122.00	0.00	0.517	**159**
100	248	154	**167**	167.01	0.01	3.46	124	124.01	0.01	4.438	**167**
100	273	186	**175**	175.00	0.00	19.62	156	156.00	0.00	8.053	**175**
100	297	220	**191**	191.02	0.01	478.42	180	180.02	0.01	471.292	**191**
150	336	282	**213**	213.00	0.00	0.02	157	157.00	0.00	1.240	**213**
150	392	384	**235**	235.00	0.00	0.05	188	188.00	0.00	1.860	**235**
150	447	499	**262**	262.75	0.29	1794.00	238	238.02	0.01	95.350	**262**
150	503	632	**278**	278.03	0.01	191.37	272	272.02	0.01	57.700	**278**
150	559	780	**288**	299.76	4.08	1793.80	283	291.26	2.92	1793.97	**291**
150	615	945	**309**	319.89	3.52	1791.50	306	318.21	3.99	1794.06	**308**
150	671	1124	**324**	347.98	7.40	1792.20	317	347.18	9.52	1791.09	**318**
200	597	890	**325**	325.00	0.00	254.15	315	315.03	0.01	34.10	**325**
200	697	1213	**344**	346.83	0.82	1794.10	341	344.71	1.09	1794.71	**344**
200	796	1583	**370**	400.50	8.24	1794.00	365	396.50	8.63	1794.00	**367**
200	896	2005	**384**	444.50	15.76	1794.10	386	442.50	14.64	1794.10	**386**
200	995	2473	**380**	490.50	29.08	1794.10	365	488.50	33.84	1794.20	**367**
200	1095	2995	**390**	534.50	37.05	1794.00	399	533.00	33.58	1791.90	**399**
200	1194	3562	**396**	584.00	47.48	1793.80	396	584.00	47.48	1793.50	**396**
250	934	2179	**402**	462.50	15.05	1794.10	404	456.00	12.87	1792.70	**407**
250	1090	2968	**408**	532.50	30.52	1793.80	427	531.00	24.36	1794.60	**428**
250	1245	3872	**411**	609.50	48.30	1794.00	424	603.50	42.34	1793.50	**424**
250	1401	4904	**444**	685.50	54.39	1793.70	448	684.00	52.68	1796.00	**448**
250	1557	6057	**457**	760.00	66.30	1793.70	466	761.00	63.31	1795.80	**466**
250	1712	7324	**444**	839.00	88.96	1793.80	475	840.00	76.84	1791.30	**475**
250	1868	8719	**491**	915.50	86.46	1791.40	491	915.50	86.46	1793.30	**491**
300	1346	4526	**456**	657.00	44.08	1793.70	472	654.00	38.56	1791.60	**473**
300	1570	6159	**471**	765.00	62.42	1793.60	470	768.50	63.51	1791.90	**470**
300	1794	8042	**515**	880.50	70.97	1793.40	522	879.00	68.39	1791.50	**522**
300	2019	10186	**533**	993.00	86.30	1793.50	522	993.00	90.23	1793.20	**522**
300	2243	12573	**534**	1102.50	106.46	1793.40	534	1103.00	106.55	1798.30	**534**
300	2467	15210	**520**	1213.50	133.37	1794.00	520	1213.50	133.37	1791.00	**520**
300	2691	18097	**538**	1325.50	146.38	1794.10	538	1326.00	146.47	1793.40	**538**

Tabela 5.6. Comparação dos resultados de otimização de SAMRDNG para T_2.

Diferença das soluções		IC_{dif}	**95.0%**	**90.0%**
med	*2.02*	**min**	*-0.29*	*0.10*
stdev	7.41	**max**	4.33	3.95

estabelecido em 1800 segundos, como por exemplo, para a instância $|V| = 150$, $|A| = 671$ e $|E| = 1124$, para ambos os casos, antes e após o pré-processamento da instância. Nesta tabela, este fato fica caracterizado quando se tem traços (-) nas colunas de *BI*, *BB* e *GAP* da instância.

Para outras instâncias, sequer é necessário a execução da otimização no CPLEX após o teste de redução para SAMRDP, visto que durante o pré-processamento da instância já foi possível conhecer os valores ótimos das soluções, como para a instância $|V| = 50$, $|A| = 37$ e $|E| = 4$.

O intervalo de confiança *(IC)* para a média da diferença entre as soluções de otimização de SAMRDP, antes e após a redução, usando-se o teste t emparelhado, é expresso por: $IC_{99\%} = [0.94, 8.63]$. Ou seja, com 99% de confiança, pode-se dizer que aplicar a otimização após a redução das instâncias é mais indicado para encontrar soluções de maiores cardinalidades para SAMRDP, dentro do limite de tempo de processamento. Os valores da média e desvio padrão para este teste são, respectivamente, 4.79 e 9.21.

Tabela 5.7. Resultados de otimização para SAMRDP sobre instâncias T_2.

Instância			**CPLEX (antes da redução)**				**CPLEX (após a redução)**				**BI+**
|V|	**|A|**	**|E|**	**BI**	**BB**	**GAP**	**t(sec)**	**BI**	**BB**	**GAP**	$t_{rc}(sec)$	**offset**
50	37	4	**37**	37	0.00	0.00	0	-	0.00	0.00	**37**
50	43	5	**43**	43	0.00	0.00	0	-	0.00	0.00	**43**
50	49	6	**48**	48	0.00	0.00	1	1.00	0.00	0.00	**48**
50	56	8	**52**	52	0.00	0.01	9	9.00	0.00	0.00	**52**
50	62	10	**56**	56	0.00	0.15	27	27.00	0.00	0.10	**56**
50	68	12	**67**	67	0.00	0.00	5	5.00	0.00	0.00	**67**
50	74	14	**71**	71	0.00	0.00	9	9.00	0.00	0.00	**71**
100	149	56	**145**	145	0.00	0.15	36	36.00	0.00	0.10	**145**
100	174	76	**166**	166	0.00	138.00	94	94.00	0.00	93.90	**166**
100	198	98	**183**	189	3.04	1800.00	122	127.73	4.69	1793.80	**183**
100	223	124	**205**	215	4.90	1800.00	171	181.26	5.99	1793.30	**205**
100	248	154	**217**	241	10.90	1800.00	186	209.60	12.69	1793.70	**217**
100	273	186	**233**	264	13.40	1800.00	208	235.41	13.17	1793.90	**237**
100	297	220	**252**	289	14.80	1800.00	239	272.07	13.83	1793.80	**256**
150	336	282	**314**	330	5.08	1800.00	218	233.93	7.31	1793.60	**313**
150	392	384	**347**	385	10.90	1800.00	266	300.62	13.01	1793.80	**350**
150	447	499	**386**	440	14.00	1800.00	326	379.62	16.45	1785.60	**386**
150	503	632	**418**	494	18.30	1800.00	385	466.94	21.28	1790.80	**413**
150	559	780	**452**	551	21.80	1800.00	414	516.44	24.74	1793.50	**448**
150	615	945	**492**	606	23.10	1800.00	485	602.65	24.26	1794.10	**489**
150	671	1124	**-**	-	-	1800.00	**-**	-	-	0.02	**8**
200	597	890	**522**	591	13.20	1800.00	457	524.45	14.76	1793.90	**523**
200	697	1213	**569**	690	21.30	1800.00	544	659.52	21.24	1794.00	**574**
200	796	1583	**633**	789	24.70	1800.00	603	757.79	25.67	1794.00	**634**
200	896	2005	**-**	-	-	1800.00	**-**	-	-	0.02	**8**
200	995	2473	**-**	-	-	1800.00	**-**	-	-	0.02	**26**
200	1095	2995	**-**	-	-	1800.00	**-**	-	-	0.03	**11**
200	1194	3562	**-**	-	-	1800.00	**-**	-	-	0.03	**0**
250	934	2179	**-**	-	-	1800.00	**-**	-	-	0.03	**41**
250	1090	2968	**-**	-	-	1800.00	**-**	-	-	0.03	**12**
250	1245	3872	**-**	-	-	1800.00	**-**	-	-	0.03	**0**
250	1401	4904	**-**	-	-	1800.00	**-**	-	-	0.04	**9**
250	1557	6057	**-**	-	-	1800.00	**-**	-	-	0.04	**20**
250	1712	7324	**-**	-	-	1800.00	**-**	-	-	0.05	**12**
250	1868	8719	**-**	-	-	1800.00	**-**	-	-	0.05	**0**
300	1346	4526	**-**	-	-	1800.00	**-**	-	-	0.04	**28**
300	1570	6159	**-**	-	-	1800.00	**-**	-	-	0.05	**10**
300	1794	8042	**-**	-	-	1800.00	**-**	-	-	0.06	**7**
300	2019	10186	**-**	-	-	1800.00	**-**	-	-	0.06	**4**
300	2243	12573	**-**	-	-	1800.00	**-**	-	-	0.07	**0**
300	2467	15210	**-**	-	-	1800.00	**-**	-	-	0.08	**0**
300	2691	18097	**-**	-	-	1800.00	**-**	-	-	0.08	**0**

Capítulo 6

Heurística e Procedimento de Melhoria para SAMRDN

Inicialmente, será proposto um procedimento de melhoria para SAMRDN em que, dada uma solução inicial A', a aplicação deste procedimento obterá uma solução A''. Na sequência, propõe-se o uso deste procedimento como uma Busca Local para uma heurística aplicada ao SAMRDN.

6.1 Geração de uma solução inicial A'

Para a geração de uma solução inicial A', propõe-se o uso de um dos algoritmos aproximativos apresentados na seção 4.1 apenas para proceder à seleção de um subconjunto de arcos, então denominados A_f, tal que $G[A_f]$ seja acíclico. Como o procedimento de melhoria aqui proposto é para aplicação sobre grafos de conflito $\bar{G}$ gerais, um conjunto independente em $\bar{G}[A_f]$ será obtido heuristicamente, não se tendo mais a garantia do fator de aproximação do algoritmo utilizado.

O cálculo do conjunto independente sobre o subgrafo induzido $\bar{G}[A_f]$ será determinado por uma heurística gulosa, então denominada $Greedy\_CI(\bar{G}[A_f])$, que considera como candidatos aqueles vértices do subgrafo com menor grau, escolhendo-se aleatoriamente um deles para compor uma solução. Todos os vértices vizinhos a um vértice escolhido ficam proibidos de fazerem parte da solução. A heurística repete iterativamente este procedimento até que não haja mais candidatos, formando então uma solução A' para SAMRDN.

6.2 Procedimento de melhoria

Os trabalhos de Felsner & Reuter [1999] e Brightwell & Massow [2013] apresentam uma abordagem para se obter ordenações topológicas (ou extensões lineares) baseada no conceito de diâmetro de uma extensão linear (*linear-extension-diameter*) de um *poset* $P = (X, \leq)$. A *distância* entre duas permutações que contém o mesmo conjunto X é o número de pares de elementos em ordens de precedência distintas nas duas permutações. A distância maximal é o *diâmetro* da extensão linear de P, denotada por $led(P)$. Um *par crítico* são elementos que possuem ordem de precedência já definida na ordenação, ou seja, $\{u, v\}$ é um *par crítico* de P se $(\downarrow u \subseteq \downarrow v)$ e $(\uparrow v \subseteq \uparrow u)$ procede, sendo $(\downarrow u)$ o conjunto de predecessores de u e $(\uparrow u)$ o conjunto de sucessores de u. Um grafo de extensões lineares $G(P)$ é aquele em que seus vértices são extensões lineares de P e dois vértices serão adjacentes se as extensões lineares correspondentes têm *distância = 1*, ou seja, apenas um par de elementos adjacentes estão permutados. Felsner & Reuter [1999] identificam limites inferiores para $led(P)$ e Brightwell & Massow [2013] provam que determinar $led(P)$ é $\mathcal{NP}$–Completo para casos gerais.

A relação entre esta abordagem e SAMRDN é estabelecida quando se cria uma ordenação total dos vértices de G a partir de ordenações parciais, fazendo a analogia em que $P = (V, \leq)$, sendo V o conjunto de vértices do grafo G. Uma ordenação parcial para SAMRDN será gerada a partir de uma solução inicial A', em que todos os arcos $a = (i, j) \in A'$ passam a formar um par crítico na ordenação, ou seja, tem-se que i precede j $(i \prec j)$ na ordenação topológica, sendo $a = (i, j)$ classificado como um arco $forward$. Dado um arco $a = (i, j)$, tal que $j \prec i$ na ordenação, o arco é classificado como *backward*.

Uma ordenação total dos vértices de G é então gerada respeitando-se a ordenação parcial dos pares críticos dados em A'. Note que, além dos arcos $a = (i, j) \in A'$, outros arcos também poderão ficar $forward$ após uma ordenação total dos vértices. Uma nova solução A'' é então obtida computando-se um conjunto independente em $\bar{G}$ sobre todos arcos que ficam $forward$ em G após uma ordenação total de seus vértices, via heurística gulosa $Greedy\_CI(\bar{G}[A_f])$, previamente apresentada, em que A_f denota o conjunto de todos os arcos classificados como $forward$.

6.2.1 Ordenação topológica dos vértices de G

Seja $G = (V, A)$ e $\bar{G} = (A, E)$ uma instância de SAMRDN, e A' uma solução inicial para o problema. Para realizar o procedimento de melhoria aqui proposto, então denominado *Top_ Sort_ Rand*$(A', G, \bar{G})$, será gerada uma ordenação total dos vértices de

G a partir de $G[A']$, tal que, se um arco $a = (i, j) \in A'$, então $i \prec j$ na ordenação. Além dos arcos de A', todos os arcos (i, j) tal que $i \prec j$ na ordenação obtida, então pertencentes ao conjunto denominado F, serão considerados para o cômputo de um conjunto independente em $\bar{G}[A_f]$, sendo $A_f = A' \cup F$. Este conjunto independente constituirá então uma nova solução A'' para SAMRDN.

Obtida uma solução inicial A', têm-se uma ordenação parcial dos vértices tal que, se $a = (i, j) \in A'$, então $i \prec j$ na ordenação. Para uma ordenação total dos vértices de G, considere que os vértices de $G[A']$ serão inseridos em uma ordenação da direita para a esquerda, um por vez à frente do outro. Dado um grafo $G = (V, A)$ acíclico, sempre existirá um ou mais vértices com grau de saída $\delta^-(v) = 0$ (ou grau de entrada $\delta^+(v) = 0$).

Tem-se que $G[A']$ é acíclico. Para proceder à ordenação total dos vértices de G, inicialmente todos os vértices que possuem grau de saída nulo, $\delta^-(v) = 0$, em $G[A']$ serão considerados vértices candidatos a serem inseridos na ordenação. A fim de randomizar a escolha de um vértice dentre o conjunto de candidatos, a cada um deles será atribuída uma nota. Esta nota expressará a probabilidade de um vértice ser escolhido para entrar na ordenação. Quanto mais atrativo o vértice for, maior será esta probabilidade.

Para que a escolha do vértice seja feita de forma tal que um vértice mais atrativo tenha maior probabilidade de ocorrer, alguns critérios serão estabelecidos. O grau de atração de um vértice candidato v_c será medido em função de seus graus de entrada e saída em G e do grau médio dos vértices de $\bar{G}$, que correspondem a arcos $(v_i, v_c) \in G$, tal que v_i não tenha sido ainda inserido na ordenação. Além disto, deve-se considerar como vértices adjacentes a (v_i, v_c) em $\bar{G}$ apenas aqueles arcos que ficam ou podem ficar $forward$ com a inclusão de v_c na ordenação. O objetivo é fazer com que arcos $a = (i, j) \in A'$ e arcos mais atrativos fiquem $forward$ na ordenação, garantindo que os arcos presentes na solução induzam um subgrafo acíclico em G.

Sendo assim, vértices com maior grau de entrada em G serão privilegiados para que sejam inseridos primeiramente na ordenação. Como novos vértices são sempre inseridos à frente na ordenação, isto possibilitará maiores chances de se ter arcos $forward$. Seguindo a mesma lógica, quanto maior o grau de saída do vértice, menos atrativo ele será, pois fará com que arcos cujos vértices de destino ainda não foram inseridos na ordenação fiquem *backward*. Este critério pode ser visto na equação 6.1, em que n_G denota a nota do vértice analisando-se o grafo G. Seja $\delta^+(v_c)$ o grau de entrada do vértice candidato v_c considerando apenas os arcos $forward$ $a = (v_i, v_c)$ cujos vértices de origem v_i ainda não foram inseridos na ordenação. Da mesma forma, $\delta^-(v_c)$ define o grau de saída do vértice v_c considerando os arcos *backward* $a = (v_c, v_f)$ cujos vértices

de destino v_f ainda não foram inseridos na ordenação. Um exemplo pode ser visto na Figura 6.1.

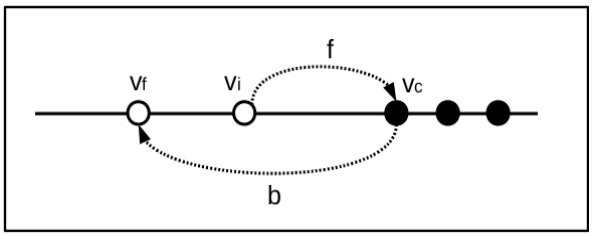
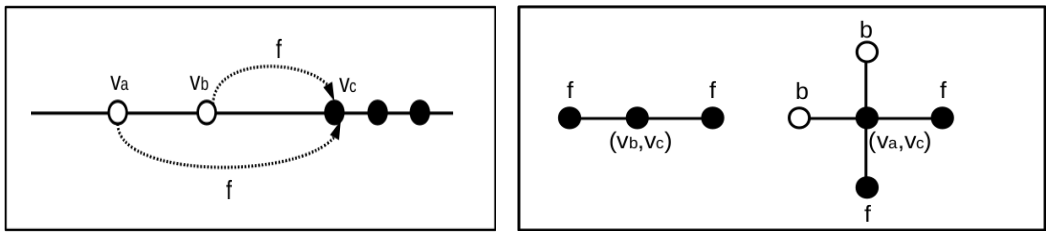


Figura 6.1. Ordenação topológica inserindo vértice v_c.

Deve-se considerar também o grau médio dos vértices de $\bar{G}$ que correspondem a arcos em G que ficarão $forward$ na ordenação com a inserção do vértice candidato v_c, ou seja, aqueles arcos que, se inserido o vértice v_c na ordenação, tem-se que o vértice de origem v_i ainda não foi inserido, fazendo com que o arco $a = (v_i, v_c)$ fique $forward$. Além disto, ao analisar os graus dos vértices (v_i, v_c) em $\bar{G}$, deve-se contabilizar para efeito de sua nota apenas os vértices adjacentes que já são arcos $forward$ em $\bar{G}$, ou que tem a possibilidade de ficarem $forward$, ou seja, arcos $a = (v_i, v_f)$ para os quais v_i e v_f ainda não foram inseridos na ordenação. Este critério está descrito na equação 6.2, em que $n_{\bar{G}}$ denota a nota do vértice no grafo $\bar{G}$, e um exemplo pode ser visualizado na Figura 6.2. Para este exemplo, se o vértice v_c é inserido na ordenação (Figura 6.2(a)), dois outros arcos, (v_a, v_c) e (v_b, v_c), ficarão $forward$. Analisando estes arcos na Figura 6.2(b), que são vértices em $\bar{G}$, nota-se que (v_a, v_c) é adjacente a dois outros arcos $forward$ (destacados na cor preta), e (v_b, v_c) também é adjacente a dois outros arcos $forward$. A média, para este exemplo, é dois ($n_{\bar{G}} = 2$).



(a) Ordenação Topológica (b) $\bar{G} = (A, E)$

Figura 6.2. Grau médio de vértices em $\bar{G}$.

Para a equação 6.2, tome cf_v como o número de vértices adjacentes ao vértice $v = (v_i, v_c)$ em $\bar{G}$ que serão $forward$ na ordenação, ou seja, os vértices adjacentes que fazem parte da solução inicial A', ou os vértices $a = (v_i, v_f)$ tal que v_i e v_f já foram ordenados e $v_i \prec v_f$ na ordenação, ou os vértices $a = (v_i, v_f)$ cujo vértice v_f já foi

ordenado, porém v_i não. Da mesma forma, considere ci_v como o número de vértices adjacentes ao vértice $v = (v_i, v_c)$ em $\bar{G}$, que poderão ficar $forward$ na ordenação, ou seja, os vértices adjacentes $a = (v_i, v_f)$ tal que v_i e v_f ainda não foram ordenados. A nota de um vértice candidato a ser inserido na ordenação será então dada pela equação 6.3.

$$n_G = \delta^+(v_c) - \delta^-(v_c) \tag{6.1}$$

$$n_{\bar{G}} = \left\lceil \frac{\sum_{v=(v_i,v_c)} cf_v + \frac{1}{2} ci_v}{\delta^+(v_c)} \right\rceil \tag{6.2}$$

$$n_{v_c} = n_G - n_{\bar{G}} \tag{6.3}$$

A equação 6.1 contabiliza os graus de entrada e saída do vértice candidato v_c em G e a equação 6.2 contabiliza o grau médio em $\bar{G}$ (número de conflitos médio) dos arcos que ficarão $forward$ com a inserção do vértice candidato na ordenação. Como ci_v contabiliza arcos incertos de ficarem $forward$ na ordenação, este deve ter um peso menor (expresso pela razão $\frac{1}{2}$ na equação 6.2) se comparado aos arcos que certamente ficam $forward$. Tem-se que, quanto maior $n_{\bar{G}}$, menos atrativo é o vértice candidato $v_c \in V$ a ser inserido na ordenação, uma vez que deseja-se privilegiar arcos que possuem menor número de conflitos, para posterior cálculo de conjunto independente em $\bar{G}$.

Considere $C_v = \{v_1, v_2, ..., v_k\}$ o conjunto dos vértices candidatos à ordenação. Seja $N_v = \{n_{v_1}, n_{v_2}, ..., n_{v_k}\}$ o conjunto das notas associadas aos vértices candidatos. Tome $n_{v_-} = min\{n_{v_1}, n_{v_2}, ..., n_{v_k}\}$. Para que as notas sejam não-negativas, se $n_{v_-} < 0$ então, para toda nota $n_{v_c} \in N_v$ tal que $1 \leq c \leq k$, tem-se $n_{v_c} = n_{v_c} + |n_{v_-}|$. Seja $P_v = \{p_{v_1}, p_{v_2}, ..., p_{v_k}\}$ o conjunto das probabilidades associadas aos vértices candidatos. A partir das notas dos vértices candidatos, calcula-se as probabilidades p_{v_c}, tal que $1 \leq c \leq k$, para a ocorrência de cada vértice candidato como sendo:

$$p_{v_c} = \frac{n_{v_c}}{\sum_{v_c \in C_v} n_{v_c}} \tag{6.4}$$

Após todos os candidatos iniciais tiverem sido identificados ($\delta^-(v_c) = 0$) e contabilizadas suas respectivas notas e probabilidades, um deles será escolhido para compor a ordenação. Uma vez que um vértice v_c é inserido, ele e todos os arcos incidentes $a = (v_i, v_c)$ são apagados de $G[A']$. Sendo assim, novos vértices candidatos podem surgir e em caso positivo, as notas e probabilidades dos candidatos são recalculadas. Este processo se repete até que todos os vértices de G tenham sido inseridos na ordenação.

Uma vez que se têm uma ordenação topológica, todos os arcos que ficaram $forward$ na ordenação, ou seja, todos os arcos $a = (v_i, v_j)$ tal que $v_i \prec v_j$, expresso por $A_f = A' \cup F$, serão considerados para o cômputo de um conjunto independente em $\bar{G}[A_f]$, como já descrito anteriormente, formando uma nova solução A'' para SAMRDN.

Considerando o procedimento de melhoria aplicado às instâncias do tipo T_1, para as quais é possível o cômputo de um conjunto independente máximo em tempo polinomial, se $F = \emptyset$, então $|A''| = |A'|$; caso contrário, tem-se $|A''| \geq |A'|$, uma vez que $|A'|$ já é um limite inferior para a nova solução A''. Como para as instâncias tipo T_2 usa-se uma heurística para o cômputo de um conjunto independente e não há garantia de que este seja máximo, esta mesma conclusão não é válida.

6.3 Heurística ILS aplicada ao SAMRDN

A metaheurística *Iterated Local Search*, ou simplesmente ILS, tem se mostrado eficiente para uma variedade de aplicações, inclusive para o LOP [Lourenço et al., 2002; Martí et al., 2012]. O Algoritmo 3 descreve a heurística ILS aplicada ao SAMRDN, tendo como entrada os grafos $G = (V, A)$, $\bar{G} = (A, E)$ e uma solução inicial A' gerada conforme descrito na seção 6.1.

Algorithm 3 $ILS\_SAMRDN(A', G, \bar{G})$

1: $A'' \leftarrow$ Top_Sort_Rand$(A', G, \bar{G})$;
2: **while** ($\neg$ *Critério de Parada*) **do**
3: $\quad A''' \leftarrow$ *Perturbação(A'')*;
4: $\quad A''' \leftarrow$ *Busca_Local(A''')*;
5: $\quad A'' \leftarrow$ *Critério_Aceitação(A'', A''')*;
6: **end while**
7: **return** A''

Na linha 1 do algoritmo $ILS\_SAMRDN(A', G, \bar{G})$, uma nova solução A'' é gerada através da execução do algoritmo $Top\_Sort\_Rand(A', G, \bar{G})$, apresentado na seção anterior. Nas linhas 2–6, enquanto um critério de parada não seja satisfeito ($\neg$ *Critério de Parada*), a heurística executa as funções: *Perturbação(A'')*, *Busca_Local(A''')* e *Critério_Aceitação(A'', A''')*. O critério de parada será estabelecido como o tempo máximo de execução do algoritmo.

A função *Perturbação(A'')* é feita violando-se a imposição de que todos os arcos da solução de A'' devam ficar $forward$ em uma ordenação topológica. Sendo assim, um dos arcos de A'', sorteado aleatoriamente, ficará *backward* na nova ordenação.

O Algoritmo 4 descreve a função *Busca_Local(A''')*, na qual gera-se um número pré-estabelecido (n_iter) de ordenações topológicas a partir de

Algorithm 4 Busca_Local (A''')

```
1: iter ← 0;
2: while (iter < n_iter) do
3:     Â ← Top_Sort_Rand(A''', G, Ḡ);
4:     if (|Â| ≥ |A'''|) then
5:         A''' ← Â;
6:     end if
7:     iter ← iter + 1;
8: end while
9: return A'''
```

$Top\_Sort\_Rand(A''', G, \bar{G})$. A melhor solução A''' é atualizada sempre quando outra solução $\hat{A}$ com maior ou a mesma cardinalidade de A''' é encontrada (linhas 4–6). A busca local aqui sugerida se difere da clássica, na qual são estabelecidos movimentos, como de troca ou inserção, e em que toda a vizinhança é explorada.

A função *Critério_ Aceitação(A'', A''')* descrita na linha 5 do Algoritmo 3 fará com que, se $|A'''| \geq |A''|$, então $A'' \leftarrow A'''$, atualizando a melhor solução. O algoritmo retorna então a solução A''.

6.3.1 Resultados computacionais

Novamente, os algoritmos foram implementados em C++ e executados em uma máquina *Intel Core* i7 980 (3.33GHz), com 24GB de RAM. Para a execução da heurística $ILS\_SAMRDN(A', G, \bar{G})$, considerando a função *Busca_ Local(A''')*, foi definido $n\_iter = 10$. O critério de parada foi estabelecido em 1800 segundos, assim como nos testes anteriores sobre as instâncias T_2, para fins comparativos. Utilizou-se o algoritmo $Degree(G)$ para a geração de um conjunto de arcos A_f tal que $G[A_f]$ seja acíclico, conforme descrito na seção 6.1.

A Tabela 6.2 apresenta os resultados de otimização de SAMRDN sobre as instâncias T_2 (que são os mesmos presentes na Tabela 5.5 antes da redução), versus os resultados da heurística $ILS\_SAMRDN(A', G, \bar{G})$, sobre as mesmas instâncias. Nesta tabela, Sol_ILS é o valor da solução encontrada para cada instância aplicando-se $ILS\_SAMRDN(A', G, \bar{G})$ e $t^*ILS(s)$ especifica o tempo (em segundos) em que a melhor solução foi encontrada pela heurística. A sua última coluna (Sol_ILS - BI) apresenta a diferença entre as soluções heurísticas e de otimização.

Percebe-se pela análise da Tabela 6.2 que a heurística $ILS\_SAMRDN(A', G, \bar{G})$, para instâncias maiores ($|V| > 250$), fornece resultados melhores que aqueles obtidos pelo Cplex. Porém, para conjunto de instâncias com $|V| \leq 200$ e densidades em G, $d_G(\%) \leq 2.25$, as soluções heurísticas são inferiores às de otimização.

Procedendo ao teste t–emparelhado, a Tabela 6.1 apresenta uma análise estatística. Ao analisar o intervalo de confiança para a média da diferença entre as soluções exatas de SAMRDN e da heurística $ILS\_SAMRDN(A', G, \bar{G})$ para as instâncias T_2, com 99.8% pode-se afirmar que as soluções da otimização versus heurísticas são significativamente diferentes, sendo o algoritmo $ILS\_SAMRDN(A', G, \bar{G})$ mais eficiente para encontrar soluções de boa qualidade para o problema. A média para as soluções utilizando o resolvedor *Cplex* é 298,4 (arcos) e a média para a heurística Sol_ILS é 311,48 (arcos). Porém, ao se analisar as instâncias individualmente, como para a instância $|V| = 100$, $|A| = 248$ e $|E| = 154$, o valor da solução de SAMRDN alcançada pelo Cplex (167) é maior que a fornecida por $ILS\_SAMRDN(A', G, \bar{G})$ (165). Além disto, o Cplex não se limita apenas a achar uma solução para o problema, como a heurística proposta, como também verifica a otimalidade desta solução.

Tabela 6.1. Comparação dos resultados exatos e heurísticos para T_2.

(CPLEX–ILS)		**IC_diff**	**99.80%**
med	*13.07*	**min**	*0.62*
stdev	24.41	**max**	25.53

Tabela 6.2. Resultados de otimização X heurísticos para instâncias T_2.

Instância			**CPLEX**		*ILS_SAMRDN*		**Sol_ILS**
|V|	**|A|**	**|E|**	**BI**	**t(sec)**	**Sol_ILS**	**t*ILS(s)**	**- BI**
50	37	4	**34**	0.00	**34**	0.0007	0
50	43	5	**39**	0.00	**39**	0.0006	0
50	49	6	**42**	0.00	**42**	0.0007	0
50	56	8	**49**	0.00	**49**	0.0008	0
50	62	10	**53**	0.01	**53**	0.0007	0
50	68	12	**60**	0.00	**60**	0.0003	0
50	74	14	**62**	0.00	**62**	0.0008	0
100	149	56	**114**	0.00	**114**	0.0010	0
100	174	76	**126**	0.00	**126**	0.0010	0
100	198	98	**140**	0.02	**140**	13.8920	0
100	223	124	**159**	0.00	**157**	0.1680	-2
100	248	154	**167**	3.46	**165**	458.2200	-2
100	273	186	**175**	19.62	**171**	1749.0000	-4
100	297	220	**191**	478.42	**187**	454.5200	-4
150	336	282	**213**	0.02	**213**	347.4700	0
150	392	384	**235**	0.05	**233**	379.9600	-2
150	447	499	**262**	1794.00	**258**	23.0360	-4
150	503	632	**278**	191.37	**274**	352.0000	-4
150	559	780	**288**	1793.80	**285**	109.9900	-3
150	615	945	**309**	1791.50	**303**	492.9000	-6
150	671	1124	**324**	1792.20	**323**	250.8800	-1
200	597	890	**325**	254.15	**316**	1170.5000	-9
200	697	1213	**344**	1794.10	**332**	1146.5000	-12
200	796	1583	**370**	1794.00	**358**	653.6300	-12
200	896	2005	**384**	1794.10	**381**	47.0760	-3
200	995	2473	**380**	1794.10	**404**	95.3570	24
200	1095	2995	**390**	1794.00	**426**	1058.9000	36
200	1194	3562	**396**	1793.80	**440**	651.3500	44
250	934	2179	**402**	1794.10	**387**	98.0380	-15
250	1090	2968	**408**	1793.80	**432**	1747.4000	24
250	1245	3872	**411**	1794.00	**444**	257.9900	33
250	1401	4904	**444**	1793.70	**475**	197.2000	31
250	1557	6057	**457**	1793.70	**496**	1252.2000	39
250	1712	7324	**444**	1793.80	**520**	662.8500	76
250	1868	8719	**491**	1791.40	**536**	1036.6000	45
300	1346	4526	**456**	1793.70	**472**	1304.0000	16
300	1570	6159	**471**	1793.60	**502**	1365.5000	31
300	1794	8042	**515**	1793.40	**541**	1548.7000	26
300	2019	10186	**533**	1793.50	**546**	789.7900	13
300	2243	12573	**534**	1793.40	**581**	1154.6000	47
300	2467	15210	**520**	1794.00	**589**	609.3000	69
300	2691	18097	**538**	1794.10	**616**	930.8800	78

Capítulo 7

Conclusão

Foram tratados neste trabalho variações do problema clássico em Teoria dos Grafos, Subgrafo Acíclico Máximo (SAM), então denominadas Subgrafo Acíclico Máximo sob Restrições Disjuntivas Negativas (SAMRDN) e Subgrafo Acíclico Máximo sob Restrições Disjuntivas Positivas (SAMRDP). Estes problemas foram introduzidos na literatura por meio das publicações geradas neste trabalho [Mapa & Urrutia, 2013, 2015].

Como resultados, foi demonstrado que o problema de decisão sobre a viabilidade de SAMRDP é $\mathcal{NP}$–Completo, a partir de uma redução do problema clássico de Cobertura por Vértices. Considerando o SAMRDN, foram desenvolvidos seis algoritmos 1/2–aproximativos para o problema, sendo que três deles, em suas versões então denominadas *late*, são mais indicadas para tratar o problema, por gerarem soluções com no mínimo as mesmas cardinalidades dos algoritmos equivalentes, segundo a abordagem *early*. Os algoritmos mantém o fator de aproximação 1/2 de três outros algoritmos clássicos para tratar o SAM, e poderão ser resolvidos em tempo polinomial sempre que o problema do conjunto independente máximo seja polinomial no grafo de conflitos. Além disto, testes computacionais apontam o algoritmo $Degree_l(G, \bar{G})$ como preferível dentre os três algoritmos em suas versões *late*, por produzir soluções de melhor qualidade, dentre o conjunto de instâncias testadas.

Com o objetivo de viabilizar a obtenção de soluções de forma mais eficiente, foi proposto um pré-processamento das instâncias, tanto para SAMRDN quanto para SAMRDP, visando a redução do número de arcos em $G = (V, A)$ e do número de restrições em $\bar{G} = (A, E)$. Testes computacionais comprovam que o efeito da redução diminui, para ambos os problemas, à medida em que se tem grafos $G = (V, A)$ maiores com relação ao número de vértices e mais densos com relação ao número de arcos.

Foi proposta uma heurística *Iterated Local Search* integrada a um procedimento de melhoria para SAMRDN como busca local, com o intuito de maximizar o conjunto

de arcos $A' \subseteq A$ do grafo G tal que o subgrafo $G[A']$ seja acíclico e A' seja um conjunto independente em $\bar{G} = (A, E)$. Para o conjunto de instâncias em que se tem $|V| > 200$, a heurística se mostrou como uma opção mais viável que a abordagem exata, gerada por um modelo de programação linear inteira, para execuções com tempo limite de 1800 segundos, dentre o conjunto de instâncias testadas.

Indicações de pesquisas futuras podem incluir o desenvolvimento de heurísticas cujo objetivo consiste em, dada uma uma solução inicial viável para SAMRDP, proceder à maximização do conjunto de arcos $A' \subseteq A$ do grafo $G = (V, A)$ tal que o subgrafo $G'[A']$ seja acíclico e A' seja uma cobertura por vértices em $\bar{G} = (A, E)$. Para os algoritmos 1/2–aproximativos para SAMRDN, outras classes de grafos de conflitos que sejam polinomiais para o cômputo de um conjunto independente máximo ou que tenham um fator de aproximação constante conhecido podem ser estudadas.

Embora o pré-processamento de instâncias para SAMRDP e SAMRDN tenha possibilitado a redução do número de arcos no grafo G e do número de restrições em $\bar{G}$, em ambos os problemas, para algumas instâncias, ao se resolver o problema via programação linear inteira, o valor da solução da instância após a redução não se difere ou mesmo fica inferior à solução obtida antes do pré-processamento da instância. Por isto, para trabalhos futuros propõe-se investigar a possibilidade de se obter reduções mais fortes para ambos os problemas, além de incluí-las como técnica de corte para os modelos de programação linear inteira.

Ao se tratar o SAMRDN heuristicamente, quando se tem grafos de conflitos gerais e faz-se necessário o cômputo de um conjunto independente em $\bar{G}$, usou-se neste trabalho uma heurística gulosa. Outra indicação de pesquisa é o uso de heurísticas mais eficientes para o cálculo de um conjunto independente em grafos simples ou de algoritmos exatos para maximizar a cardinalidade de um conjunto independente no grafo de conflitos. Ainda para SAMRDN, foi proposta a heurística ILS integrada a um procedimento de melhoria para a obtenção de soluções. Outras metaheurísticas também podem ser aplicadas ao problema, além de heurísticas híbridas.

Outra indicação de trabalho futuro inclui investigar as propriedades dos problemas SAMRDN e SAMRDP considerando o grafo $G = (V, A)$ um grafo simples. Embora o problema de decisão sobre a aciclicidade em G seja tratável polinomialmente, a inclusão das restrições disjuntivas incita a investigação das complexidades dos problemas.

Referências Bibliográficas

Arora, S.; Frieze, A. & Kaplan, H. (2002). A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. *Math. Program.*, Ser. A:1–36.

Austrin, P.; Manokaran, R. & Wenner, C. (2013). On the np-hardness of approximating ordering constraint satisfaction problems. Em Raghavendra, P.; Raskhodnikova, S.; Jansen, K. & Rolim, J. D. P., editores, *APPROX-RANDOM*, volume 8096 of *Lecture Notes in Computer Science*, pp. 26–41. Springer.

Baker, B. S. & Jr., E. G. C. (1996). Mutual exclusion scheduling. *Theoretical Computer Science*, 162:225--243. ISSN 0304-3975.

Berger, B. & Shor, P. W. (1997). Tight bounds for the maximum acyclic subgraph problem. *J. Algorithms*, 25:1--18.

Bodlaender, H. L.; Jansen, K. & Woeginger, G. J. (1994). Scheduling with incompatible jobs. *Discrete Applied Mathematics*, 55:219 – 232. ISSN 0166-218X.

Bondy, A. & Murty, U. (2008). *Graph Theory*. Graduate Texts in Mathematics. Springer. ISBN 9781846289699.

Brightwell, G. & Massow, M. (2013). Diametral pairs of linear extensions. *SIAM J. Discrete Math.*, 27:634–649.

Chaovalitwongse, W.; Oliveira, C.; Chiarini, B.; Pardalos, P. & Resende, M. (2011). Revised grasp with path-relinking for the linear ordering problem. *J. Comb. Optim.*, 22:572–593.

Coffman Jr., E. G. & Graham, R. L. (1972). Optimal scheduling for two-processor systems. *Acta Inf.*, 1:200–213.

Cook, S. A. (1971). The complexity of theorem-proving procedures. Em *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pp. 151–158, New York, USA. ACM.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L. & Stein, C. (2009). *Introduction to Algorithms*. The MIT Press, 3ª edição. ISBN 0262033844, 9780262033848.

Darmann, A.; Pferschy, U. & Schauer, J. (2009). Determining a minimum spanning tree with disjunctive constraints. Em Rossi, F. & Tsoukias, A., editores, *Algorithmic Decision Theory*, volume 5783 of *Lecture Notes in Computer Science*, pp. 414–423. Springer Berlin Heidelberg.

Darmann, A.; Pferschy, U.; Schauer, J. & Woeginger, G. J. (2011). Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics*, 159:1726–1735.

de Queiroz, T. A. & Miyazawa, F. K. (2012). Problema da mochila 0-1 bidimensional com restrições de disjunção. Em *Anais do XVI CLAIO/XLIV SBPO-Simpósio Brasileiro de Pesquisa Operacional/Congreso Latino-Iberoamericano de Investigación Operativa*, pp. 1–12.

Diestel, R. (2005). *Graph Theory (Graduate Texts in Mathematics)*. Springer.

Felsner, S. & Reuter, K. (1999). The linear extension diameter of a poset. *SIAM J. Discrete Math.*, 12:360–373.

Garcia, C.; Pérez-Brito, D.; Campos, V. & Martí, R. (2006). Variable neighborhood search for the linear ordering problem. *Comput. Oper. Res.*, 33:3549–3565.

Garey, M. & Johnson, D. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, New York.

Gendreau, M.; Laporte, G. & Semet, F. (2004). Heuristics and lower bounds for the bin packing problem with conflicts. *Computers & Operations Research*, 31:347 – 358. ISSN 0305-0548.

Glover, F. & Laguna, M. (1997). *Tabu Search*. Kluwer Academic, Dordrecht.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., New York, USA.

Grötschel, M.; Jünger, M. & Reinelt, G. (1985). On the acyclic subgraph polytope. *Mathematical Programming*, 33:28–42.

Grötschel, M.; Lovász, L. & Schrijver, A. (1993). *Geometric algorithms and combinatorial optimization*. Algorithms and Combinatorics. Springer-Verlag. ISBN 9780387136240.

Guruswami, V.; HÅstad, J.; Manokaran, R.; Raghavendra, P. & Charikar, M. (2011). Beating the random ordering is hard: Every ordering csp is approximation resistant. *SIAM Journal on Computing*, 40:878–914.

Guruswami, V.; Manokaran, R. & Raghavendra, P. (2008). Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph. Em *FOCS*, pp. 573–582.

Hassin, R. & Rubinstein, S. (1994). Approximations for the maximum acyclic subgraph problem. *Inf. Process. Lett.*, 51:133–140. ISSN 0020-0190.

Jackson, B. N.; Schnable, P. S. & Aluru, S. (2008). Consensus genetic maps as median orders from inconsistent sources. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5:161–171.

Jansen, K. (1999). An approximation scheme for bin packing with conflicts. *Journal of Combinatorial Optimization*, 3:363–377.

Jansen, K. & Öhring, S. (1997). Approximation algorithms for time constrained scheduling. *Information and Computation*, 132:85 – 108. ISSN 0890-5401.

Jünger, M. (1985). Polyhedral combinatorics and the acyclic subdigraph problem. Em *Research and Exposition in Mathematics*, volume 7. Heldermann Verlag, Berlin.

Kann, V. (1993). Polynomially bounded minimization problems which are hard to approximate. Em Lingas, A.; Karlsson, R. & Carlsson, S., editores, *Automata, Languages and Programming*, volume 700 of *Lecture Notes in Computer Science*, pp. 52–63. Springer Berlin/Heidelberg.

Kann, V. (1994). Polynomially bounded minimization problems that are hard to approximate. *Nord. J. Comput.*, 1:317–331.

Karp, R. M. (1972). Reducibility among combinatorial problems. Em Miller, R. E. & Thatcher, J. W., editores, *Complexity of Computer Computations*, pp. 85–103. Plenum Press, New York.

Khot, S. (2002). On the power of unique 2-prover 1-round games. Em *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, STOC '02, pp. 767–775, New York, USA. ACM Press.

Khot, S.; Kindler, G.; Mossel, E. & O'Donnell, R. (2007). Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM J. Comput.*, 37:319–357. ISSN 0097-5397.

Laguna, M.; Martí, R. & Campos, V. (1999). Intensification and diversification with elite tabu search solutions for the linear ordering problem. *Comput. Oper. Res.*, 26:1217–1230.

Lourenço, H. R.; Martin, O. & Stützle, T. (2002). Iterated local search. Em *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pp. 321–353. Kluwer Academic Publishers, Norwell, MA.

Magnanti, T. L. & Wolsey, L. A. (1995). Chapter 9 optimal trees. Em M.O. Ball, T.L. Magnanti, C. M. & Nemhauser, G., editores, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pp. 503–615. Elsevier.

Mapa, S. M. S. & Urrutia, S. (2013). Um algoritmo 1/2-aproximativo para o problema do subgrafo acíclico máximo sob restrições disjuntivas negativas. Em *Anais do XLV SBPO-Simpósio Brasileiro de Pesquisa Operacional*, pp. 3095–3101.

Mapa, S. M. S. & Urrutia, S. (2015). On the maximum acyclic subgraph problem under disjunctive constraints. *Information Processing Letters*, 115:119 – 124. ISSN 0020-0190.

Martí, R.; Reinelt, G. & Duarte, A. (2012). A benchmark library and a comparison of heuristic methods for the linear ordering problem. *Comput. Optim. Appl.*, 51:1297–1317.

Mitchell, J. (1997). Computational experience with an interior point cutting plane algorithm. Relatório técnico, Mathematical Sciences, Rensellaer Polytechnic Institute, Troy, NY, USA.

Mladenović, N. & Hansen, P. (1997). Variable neighborhood search. *Comput. Oper. Res.*, 24:1097–1100.

Newman, A. (2000). Approximating the maximum acyclic subgraph. Dissertação de mestrado, Department of Electrical Engeneering and Computer Science, Massachusetts Institute of Technology.

Newman, A. (2001). The maximum acyclic subgraph problem and degree-3 graphs. Em *Proceedings of the 4th International Workshop on Approximation Algorithms*

for Combinatorial Optimization Problems and 5th International Workshop on Randomization and Approximation Techniques in Computer Science: Approximation, Randomization and Combinatorial Optimization, APPROX '01/RANDOM '01, pp. 147–158, London, UK. Springer-Verlag.

Pferschy, U. & Schauer, J. (2013). The maximum flow problem with disjunctive constraints. *Journal of Combinatorial Optimization*, 26:109–119.

Ramachandran, V. (1988). Finding a minimum feedback arc set in reducible flow graphs. *J. Algorithms*, 9:299–313.

Reinelt, G. (1985). The linear ordering problem: algorithms and applications. Em *Research and Exposition in Mathematics*, volume 8. Heldermann, Berlin.

Sbihi, N. (1980). Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans étoile. *Discrete Mathematics*, 29:53–76.

Schiavinotto, T. & Stützle, T. (2004). The linear ordering problem: instances, search space analysis and algorithms. *J. Math. Model. Algorithms*, 3:367–402.

Yamada, T.; Kataoka, S. & Watanabe, K. (2002). Heuristic and exact algorithms for the disjunctively constrained knapsack problem. *Transactions of Information Processing Society of Japan*, 43:2864 – 2870. ISSN 0387-5806.

Zhang, R.; Kabadi, S. N. & Punnen, A. P. (2011). The minimum spanning tree problem with conflict constraints and its variations. *Discrete Optimization*, 8:191 – 205.