Extensões Induzidas de Altura Mínima de um Conjunto Parcialmente Ordenado

Este exemplar corresponde a redação final da Dissertação devidamente corrigida e defendida por Italo Siqueira Lima e aprovada pela Banca Examinadora.

Fortaleza, 30 de Agosto de 2007

Manoel B. Campêlo Neto (MDCC/UFC)

Ricardo C. Corrêa (MDCC/UFC)

Dissertação apresentada ao programa de Mestrado e Doutorado em Ciência da Computação, UFC, como requisito para obtenção do título de Mestre em Ciência da Computação.

Mestrado e Doutorado em Ciência da Computação (MDCC)
Universidade Federal do Ceará (UFC)

Extensões Induzidas de Altura Mínima de um Conjunto Parcialmente Ordenado

Italo Siqueira Lima
italo@lia.ufc.br

Agosto de 2007

Banca Examinadora:

- Manoel Bezerra Campelo Neto (MDCC/UFC)
- Ricardo Cordeiro Corrêa (MDCC/UFC)
- Philippe Michelon (Université d'Avignon)
- Luiz Satoru Ochi (IC/UFF)

Agradecimentos

Agradeço primeiro a Deus, fonte de tudo que tenho hoje, inclusive das pessoas maravilhosas que estão ao meu lado. Agradeço à minha família pelo amor, carinho e compreensão pelos momentos em que não pude estar com eles. Também os agradadeço por estarem sempre ao meu lado desde o começo da minha vida e por me tornarem a pessoa que sou hoje. Agradeço à Flávia por todo o carinho, cuidado, preocupação e compreensão para comigo durante esta caminhada difícil para nós dois. Aos meus amigos, que sempre acreditaram em mim e à sua disposição em me ajudar nos momentos mais difíceis, tenha sido com uma simples conversa descontraída ou com debates teóricos. Agradeço aos meus mestres até aqui, que ajudaram a moldar o estudante que sou hoje e, principalmente, aos meus orientadores pelo precioso auxílio neste trabalho. Sem vocês eu não teria conseguido superar toda a pressão e todas as dificuldades durante o percurso. Agradeço à Funcap, pelo apoio financeiro dado durante o desenvolvimento desta dissertação, e ao projeto ALFA, pela oportunidade de estudar em um país estrangeiro e vivenciar um pouco de uma rotina diferente, tanto acadêmica quanto pessoal.

"The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!', but 'That's funny...' "
Isaac Asimov

Sumário

Lista de Tabelas

Lista de Figuras

1

Introdução

Neste trabalho estudamos o problema de extensões induzidas de altura mínima de conjuntos parcialmente ordenados (*poset*). Este problema consiste em, dado um poset e um conjunto de pares de elementos que não se relacionam, atribuir uma relação entre os pares citados de modo que o poset original acrescido destas novas relações não deixe de ser um poset e que tenha a menor altura possível.

Este problema surgiu da generalização do problema de *t-gathering* em redes. Na literatura, encontramos trabalhos que buscam, de forma exata, uma solução para este problema em diferentes topologias de rede, como grades quadradas ou caminhos ([7]), ao mesmo tempo que tentam estabelecer limites para uma solução. Além disso, sabemos da existência de um algoritmo aproximativo de fator de aproximação 4 para qualquer topologia de rede e para quaisquer parâmetros do problema ([9]).

Para nosso conhecimento, o aspecto de otimização combinatória do problema ainda não foi explorado, apesar da semelhança deste problema com outros de escalonamento de tarefas com restrições de precedência e disponibilidade de recursos ([1, 4, 5, 6, 14, 16]), estes sim bastante explorados na literatura. *Branch-and-bound* ([5, 16]), decomposição e propagação de restrições ([4]), planos-de-corte ([17]) e metaheurísticas ([24, 31]) são algumas das técnicas já utilizadas para atacar estes problemas. Além disso, a modelagem do problema, por parte da nossa proposta de generalização, utilizando conceitos da teoria de ordens parciais, induz um problema específico no campo de extensões de ordens parciais, que parece não ter sido abordado ainda.

Nesta dissertação, estudamos várias heurísticas, propostas por nós, para obtenção de limites superiores e inferiores para o problema, explorando a estrutura do poset dado, a relação do problema com coloração de grafos e a técnica de programação por restrições. Apresentamos também dois modelos de programação linear inteira e algumas desigualdades que podem ser utilizadas na resolução da relaxação dos modelos e, consequentemente, na obtenção de limites inferiores.

Os capítulos seguintes estão estruturados como segue. No Capítulo 3 definimos ordens parcias e conjuntos parcialmente ordenados, além de apresentarmos outros conceitos necessários durante a leitura do texto. Também introduzimos a definição do problema, realizamos a demonstração de sua complexidade e enumeramos alguns problemas relacionados. No Capítulo 4 apresentamos nossas heurísticas para obtenção de limites superiores baseadas na estrutura do poset e no conceito de coloração de grafos. O Capítulo 5 é dedicado à apresentação de duas propostas de obtenção de limites inferiores: a primeira, baseada na resolução do problema de número cromático fracionário em grafos, e a segunda, que adapta a técnica de programação por restrições ao nosso caso, técnica essa que tem apresentado bons resultados no tratamento de problemas de escalonamento com restrições de recursos ([17, 6]). No Capítulo 6, os dois modelos propostos são descritos e fortalecidos através das idéias descritas na seção 6.3. No Capítulo 7 apresentaremos os resultados do estudo computacional das heurísticas e modelos propostos e o procedimento utilizado para a geração das nossas instâncias de teste. Finalmente, o Capítulo 8 contém algumas conclusões e direções para trabalhos futuros.

2

Definições e Notações

Este capítulo se destina a exposição das definições e notações adotadas durante a dissertação e alguns problemas na literatura passíveis de serem reformulados utilizando os conceios apresentados aqui.

Na Seção 2.1 serão apresentados os conceitos fundamentais em grafos necessários durante a exposição de nosso trabalho, bem como a definição de um problema clássico da teoria dos grafos. Já na Seção 2.2 serão vistos conceitos de ordens parciais e algumas definições utilizadas na descrição do problema objeto deste estudo e nos métodos desenvolvidos para a resolução do mesmo. A Seção 2.3 apresenta alguns problemas que podem ser modelados utilizando a noção de conjuntos parcialmente ordenados.

2.1 Teoria dos Grafos

Um grafo $G = (V, E)$ é uma dupla ordenada de conjuntos, onde V é o *conjunto de vértices* e E é um conjunto de pares não ordenados de V chamados de *arestas*. Utilizamos $V(G)$ e $E(G)$, respectivamente, para o conjunto de vértices e o conjunto de arestas de G, respectivamente. A aresta definida pelos vértices u e v é denotada por uv, e os vértices u e v são chamados de extremidades da aresta uv. Em um *grafo direcionado*, consideramos que, se $u \in V$ e $v \in V$, então $(u, v) \neq (v, u)$. Neste caso, um elemento em E será chamado de um *arco*. Um grafo é dito *simples* se não for direcionado e não possuir *arestas múltiplas* ou *laços*, onde duas arestas são múltiplas se elas coincidem em ambas as extremidades e onde um laço é uma aresta cujas extremidades são iguais.

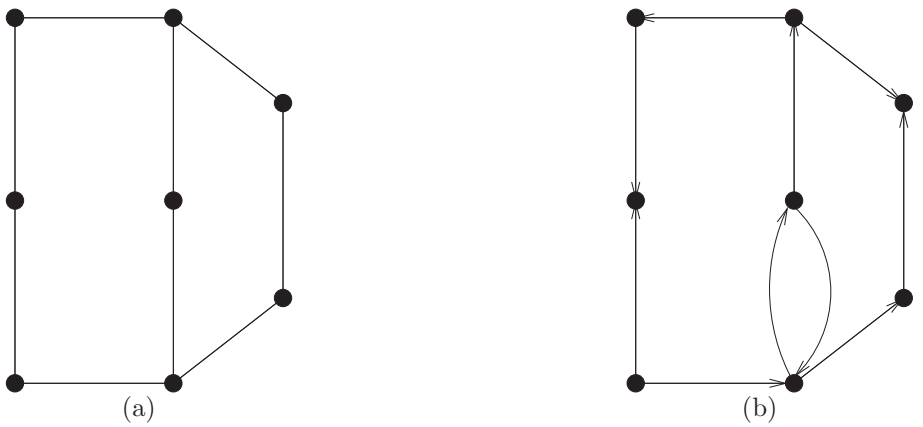


Figura 2.1: Representação gráfica de grafo simples, na figura 2.1(a), e de um grafo direcionado, na figura 2.1(b)

Dois vértices u e v são *adjacentes* em G se $uv \in E$. A *vizinhaça* de u é definida por $N_G(u) = \{v \in G \,|\, uv \in E\}$. Além disso, utilizamos $N_G[u] = N_G(u) \cup \{u\}$. Em ambos os casos, o subscrito G pode ser omitido quando o grafo estiver claro pelo contexto. O *grau* de u, denotado por $d(u)$, é a cardinalidade de $N_G(u)$.

Um grafo G' é um subgrafo de G se $V(G') \subseteq V(G)$ e $E(G') \subseteq E(G)$, o que pode ser denotado por $G' \subseteq G$. Um subgrafo $G' \subseteq G$ é um *subgrafo induzido* se, para toda aresta $uv \in E(G)$, uv pertence a $E(G')$ sempre que $u, v \in V(G')$. O subgrafo induzido por $V' \subseteq V(G)$ é definido por $G[V'] = (V', E(G) \cap V' \times V')$.

Uma *clique* em G é um subconjunto de vértices dois a dois adjacentes. Uma *clique maximal* em G é uma clique que não está estritamente incluída em nenhuma outra clique de G. A maior clique do grafo G, isto é, a clique de maior cardinalidade, é chamada de *clique máxima*. Um *conjunto independente* é qualquer subconjunto de vértices $S \subseteq V$ tal que não há dois vértices neste subconjunto adjacentes em G. A noção de conjunto independente também pode ser visto como o dual da noção de clique.

Um conjunto de arestas $C = \{v_0v_1, v_1v_2, \ldots, v_{k-1}v_k\}$, com $v_i \neq v_j$, para $i \neq j$, será chamado de *caminho*. Dizemos que C é um caminho de v_0 a v_k e tem *comprimento* k. A *distância* entre dois vértices u e v é definida como o menor comprimento de um caminho entre u e v. Caso v_0 seja adjacente a v_k em um caminho C, com comprimento maior ou igual a 2, chamamos o conjunto $C \cap \{v_0v_k\}$ um *ciclo* de tamanho $k + 1$. Um grafo que não possui ciclos é chamado de *acíclico*.

Uma *orientação* de G é uma função $\sigma : E \to V$ tal que $\sigma(uv) \in \{uv, vu\}$. O grafo direcionado obtido a partir de G com a orientação σ é denotado por G_σ. Neste contexto, temos que a *vizinhaça positiva* de u será definida por $N^+(u) = \{v \,|\, \sigma(uv) = uv\}$ e a *vizinhaça negativa* por $N^-(u) = \{v \,|\, \sigma(uv) = vu\}$. Esta notação pode ser estendida de (u) para $[u]$ adicionando o vértice u como elemento destes conjuntos. Um *caminho direcionado* é um conjunto de arcos $\overrightarrow{p} = \{u_0u_1, u_1u_2, \ldots, u_{k-1}u_k\}$ tal que, para $i \neq j$, temos que $u_i \neq u_j$. O tamanho de um caminho direcionado equivale à cardinalidade de $\overrightarrow{p}$ e pode ser denotado por $|\overrightarrow{p}|$. Uma orientação é *acíclica* se G_σ não possui ciclos orientados.

Dois arcos de um grafo formam um *par disjuntivo* se qualquer caminho neste grafo só poderá passar por um deles. Um grafo D contendo arcos disjuntivos é um *grafo disjuntivo*, denotado por $D = (N; Z, W)$, onde N é o conjunto de vértices, Z é o conjunto de arcos que não são disjuntivos e W o conjunto de arcos disjuntivos.

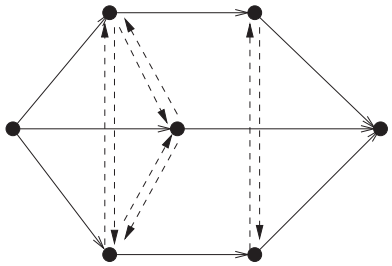


Figura 2.2: Grafo disjuntivo. Note que qualquer caminho $C = \{v_0v_1, v_1v_2, \ldots, v_{k-1}v_k\}$, com $v_i \neq v_j$, só poderá contar com uma aresta de cada par de arcos disjuntivos (linhas tracejadas)

Dado um inteiro positivo k, uma *coloração* ou *k-coloração* de G é uma atribuição de valores do conjunto $\{1, 2, \ldots, k\}$ aos vértices de G tal que cada vértice recebe um valor e as extremidades de cada aresta recebem valores diferentes. Uma coloração de G também pode ser vista como uma família $W_1, W_2, \ldots, W_k$ de $k \geq \chi(G)$ conjuntos independentes de G, cada conjunto independente W_i na família definindo uma classe de cor associada à cor i (ver Figura 2.3).

O *problema de coloração de grafos* é definido como o problema de encontrar o menor número de cores $\chi(G)$, conhecido como *número cromático*, tal que G possui uma $\chi(G)$-coloração. Os valores do conjunto $\{1, 2, \ldots, k\}$ podem ser chamados de cores quando associados ao problema de coloração. Uma coloração *first-fit* é um procedimento para colorir os vértices de um grafo G que consiste em ordenar tais vértices em uma lista segundo uma certa regra e colori-los nesta mesma ordem, um de cada vez. Cada vértice é colorido com a cor de menor índice dentre aquelas possíveis para ele (isto é, nenhum vértice adjacente a ele possui aquela cor). O problema de coloração é $\mathcal{NP}$-difícil ([22]).

Nesta dissertação utilizaremos, frequentemente, dois tipos de grafos: grafos simples e grafos direcionados acíclicos (GDA).

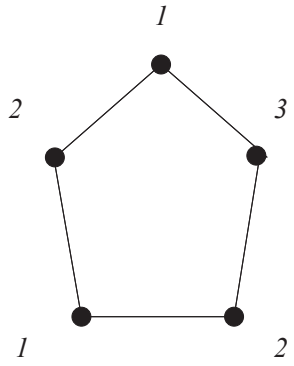


Figura 2.3: Coloração própria de um grafo. O número ao lado de cada vértice indica o índice da cor a ele atribuída.

2.2 Ordens Parciais e Posets

Uma *relação* é qualquer subconjunto de um produto cartesiano entre conjuntos. Particularmente, uma *relação binária* é um subconjunto de $A \times B$, onde A e B são dois conjuntos quaisquer, ou seja, uma coleção de pares ordenados (a, b) com o primeiro elemento em A ($a \in A$) e o segundo em B ($b \in B$). Em uma relação binária R, escrevemos geralmente $a\,R\,b$ para mostrar que $(a, b) \in R$. Se $A = B$, R é uma relação binária em A.

Seja R uma relação binária em um conjunto A. Chamamos R de relação *irreflexiva* se nenhum elemento em A está relacionado consigo mesmo em R ($a \in A \Rightarrow (a, a) \notin R$); quando tal ocorrer, para todo elemento em A, então R é dita *reflexiva* ($a \in A \Rightarrow (a, a) \in R$). Já a relação R é chamada *simétrica* quando, para todos $x, y \in A$, $x\,R\,y$ se, e somente se, $y\,R\,x$; será *anti-simétrica* caso não existam $x, y \in A$ tal que $x\,R\,y$ e $y\,R\,x$. A relação R é *transitiva* se, para todos $x, y, z \in A$, $x\,R\,y$ e $y\,R\,z$ implica que também teremos $x\,R\,z$.

Entende-se por *fecho transitivo* de R a menor relação transitiva R^+ em A que contenha R, ou seja, $R^+ \supseteq R$ é transitiva e toda relação $R' \subseteq R^+$, $R' \neq R^+$, não é transitiva. Em sentido inverso, a *redução transitiva* de R é a relação de menor cardinalidade $R^\downarrow$ em A tal que o fecho transitivo de $R^\downarrow$ é igual ao fecho transitivo de R. No grupo de figuras 2.4, a relação 2.4(a) possui como fecho transitivo a relação 2.4(b). Já a relação em 2.4(d) é a redução transitiva da relação original. A relação 2.4(c) não pode ser considerada como redução transitiva de 2.4(a) por haver outra relação menor (ver Figura 2.4(d)) com fecho transitivo igual à relação em 2.4(a).

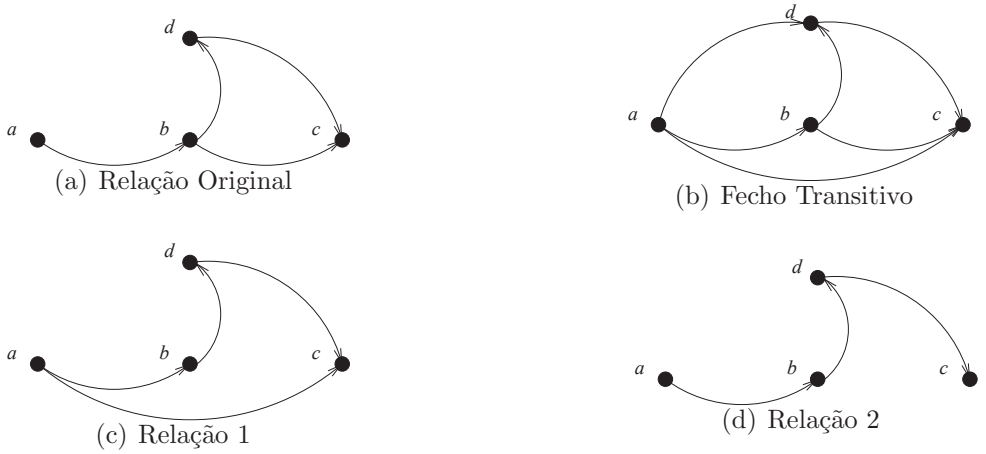
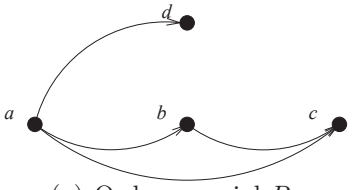


Figura 2.4: A relação em 2.4(d) é a redução transitiva da relação original

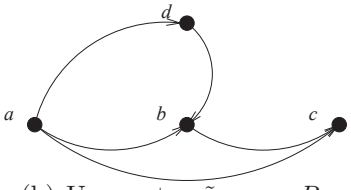
Uma *ordem parcial* em um conjunto A é uma relação binária em A irreflexiva, anti-simétrica e transitiva. Em alguns contextos, a ordem parcial definida deste modo é chamada de *ordem parcial estrita*,

enquanto que uma relação binária reflexiva, anti-simétrica e transitiva é chamada de *ordem parcial fraca (ou reflexiva)*. No nosso contexto, ao nos referirmos a uma ordem parcial, ela será sempre irreflexiva.

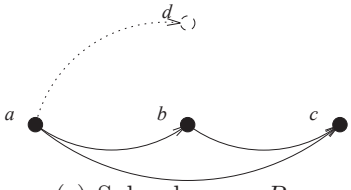
Seja R uma ordem parcial em A. Uma *extensão* de R é uma ordem parcial R' em A contendo R. Por outro lado, $R' \subseteq R$ é uma *subordem* de R quando R' for uma ordem parcial em A', onde $A' \subseteq A$. Em particular, quando $x\ R\ y$ implicar $x\ R'\ y$, para $x, y \in A'$, dizemos que R' é uma *subordem induzida por A' em R*, denotada por $R[A']$. Por exemplo, seja a ordem parcial $R = \{(a,b),(a,c),(a,d),(b,c)\}$ definida em $A = \{a,b,c,d\}$ (ver Figura 2.5(a)). Uma possível extensão para R pode ser vista na Figura 2.5(b). Tomando $A' = \{a,b,c\}$, obtemos a subordem induzida por A' em R apresentada na Figura 2.5(c).



(a) Ordem parcial R



(b) Uma extensão para R



(c) Subordem em R

Figura 2.5: Uma possível extensão para R e subordem induzida por A' em R

Um conjunto finito A sobre o qual se define uma ordem parcial R é chamado de *conjunto parcialmente ordenado (poset)*, representado pelo par $Z = (A, R)$. Como temos visto, no caso de ordens parciais, o poset Z também pode ser visto como um GDA, no qual os elementos de A são os vértices e os pares ordenados $i\ R\ j$, os arcos (ver Figura 2.6). Finalmente, o *poset induzido* por $A' \subseteq A$ é $Z' = (A', R[A'])$.

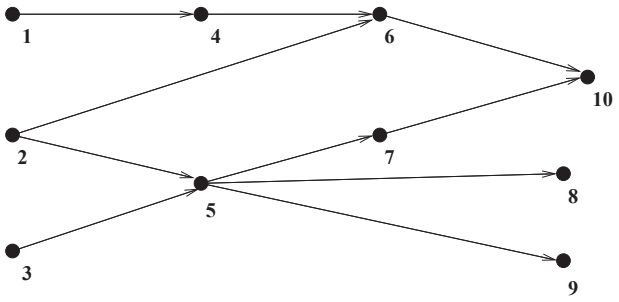


Figura 2.6: GDA que representa um conjunto parcialmente ordenado (somente a redução transitiva da ordem parcial está indicada pelos arcos do GDA)

Dado o poset $Z = (A, R)$, chamamos o poset $Z' = (A, R')$ de uma *extensão* de Z, se $R' \supseteq R$ é extensão de R. Os elementos $i \in A$ e $j \in A$ são *relacionados* em Z se $i\ R\ j$ ou $j\ R\ i$. Caso contrário, eles são *incomparáveis* (ver Figura 2.7), denotados por $i \parallel j$. Se $i\ R\ j$, dizemos que i é *predecessor* de j, que por sua vez é *sucessor* de i. Neste contexto, assim como visto para grafos, podemos definir os conjuntos $N_R^-(i) = \{j \in A \,|\, j\ R\ i\}$ e $N_R^+(i) = \{j \in A \,|\, i\ R\ j\}$. Estes conjuntos denotam, respectivamente, a vizinhaça negativa e positiva de i, obtidas a partir de R. Esta notação também pode ser estendida de (i) para $[i]$ (ver Figura 2.8).

Um elemento i é *minimal* em Z se $N_R^-(i) = \emptyset$. De forma análoga, um elemento *maximal* em Z é qualquer $i \in A$ tal que $N_R^+(i) = \emptyset$. Na Figura 2.6 os elementos minimais são $\{1,2,3\}$, enquanto que os maximais são $\{8,9,10\}$.

Define-se uma *cadeia* $a_i\ R\ a_{i+1}$ em $Z = (A, R)$, para todo $i = 1, \ldots, k-1$, como um conjunto de elementos $B = \{a_1, a_2, \ldots, a_k\}$, $B \subseteq A$, relacionados dois a dois em R. A *altura* de Z, a ser denotada

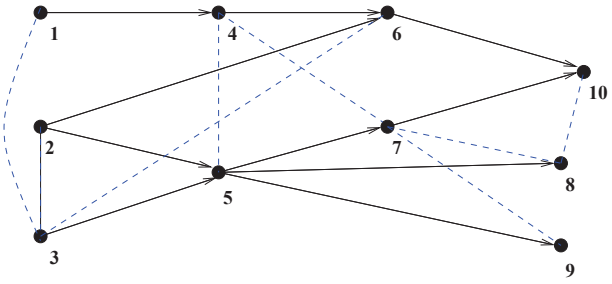
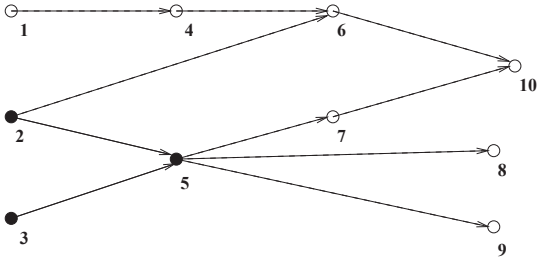
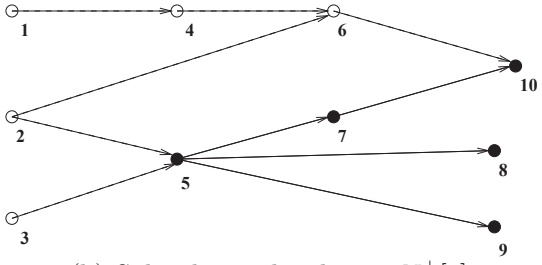


Figura 2.7: Alguns pares incomparáveis de Z representados por arestas tracejadas.



(a) Subordem induzida por $N_R^-[5]$



(b) Subordem induzida por $N_R^+[5]$

Figura 2.8: Exemplos de vizinhaça em Z

por $h(Z)$, é a maior cardinalidade de uma cadeia sua. Assim, a altura de Z pode ser vista também como uma unidade a mais que o comprimento do maior caminho orientado entre dois vértices distintos no grafo representativo de Z. A altura de um elemento $i \in A$ é a altura do poset induzido por $N_R^-[i]$.

Devido à estreita relação entre um poset $Z = (A, R)$ e a ordem parcial R que o define, iremos utilizar, sem perda de entendimento do texto, uma mesma notação para indicar tanto um quanto outro. Comumente usaremos R para representar $Z = (A, R)$. O contexto se encarregará de realizar a devida distinção. Por exemplo, usaremos alternativamente $h(Z)$ ou $h(R)$ e, por isso mesmo, vamos nos referir indistintamente à altura do poset ou da ordem que o define.

2.3 Problemas descritos via posets

Como veremos no Capítulo 3, o nosso problema modela um certo tipo de aplicação de escalonamento de tarefas com restrições de precedência e compartilhamento de recursos. Estas aplicações se caracterizam pela existência de um conjunto de tarefas que levam um certo intervalo de tempo para serem executadas e um conjunto de recursos com disponibilidade limitada. Estes conjuntos estão relacionados da seguinte maneira:

i) cada tarefa só pode ser executada após o término de um conjunto (possivelmente vazio) de tarefas predecessoras;

ii) subgrupos de tarefas demandam um certo recurso para sua execução, mas estas não podem usá-lo simultaneamente devido à relação demanda/disponibilidade.

As relações enumeradas em *(i)* pré-definem uma ordem parcial entre as tarefas. Por outro lado, devido a *(ii)*, duas ou mais tarefas que compartilham um recurso insuficiente para atender às suas demandas devem ser relacionadas de alguma forma que ainda não está definida (assumindo que estas tarefas não se encontrem relacionadas na ordem parcial). O que desejamos é relacionar as tarefas que se encontram na situação apresentada em *(ii)* de tal maneira que a ordem presente em *(i)* seja respeitada e, ao mesmo tempo, garantir que a nova relação criada entre as tarefas seja a melhor de acordo com certos critérios, como por exemplo, minimizar o tempo total de execução das tarefas.

Apresentaremos a seguir três problemas de otimização e os seus respectivos conjuntos parcialmente ordenados. Estes problemas, como veremos, se encaixam na descrição presente no parágrafo anterior. No capítulo seguinte, a descrição completa de cada um destes problemas como uma instância do nosso

problema será realizada. Por enquanto, nos limitamos a definir cada problema e mostrar como modelar um conjunto parcialmente ordenado relativo a cada um deles.

2.3.1 Escalonamento de programas em processadores

Imaginemos um problema de escalonamento de módulos de um programa em processadores, onde vamos supor os seguintes fatos: já existe uma prévia alocação dos módulos aos processadores e a entrada de certos módulos é saída de alguns outros, criando deste modo relações de precedência entre eles (possivelmente todos). Assim, é preciso estabelecer uma ordem de execução entre cada par de módulos alocados a um mesmo processador, mesmo que eles não estejam ligados por alguma relação de precedência. Logo, o problema consiste em estabelecer um cronograma de execução dos módulos, de modo que se respeite as relações de precedência e a alocação dos processadores, isto é, módulos que devam utilizar um mesmo processador não podem utilizá-lo ao mesmo tempo.

É possível criar o seguinte poset, relacionado ao problema, a partir do conjunto de módulos que temos e à sua ordem de dependência entre si. Os módulos deste programa formam um conjunto A, enquanto as relações de dependência entre a saída e entrada destes módulos podem definir uma ordem parcial R da seguinte maneira:

$$x\,R\,y \leftrightarrow y \text{ depende da saída de } x,\ \forall y \in A, x \in A$$

Deste modo, podemos definir o poset $Z = (A, R)$.

2.3.2 *t-gathering problem* em caminhos

Uma outra aplicação, que motivou nossos esforços iniciais e está relacionada ao nosso problema, é o problema de *gathering* em redes, visto em [9, 7]. Suponha que temos à nossa disposição uma rede composta por nós de comunicação. Estes nós se comunicam somente por meio de transmissões de rádios, ou *chamadas*. Cada chamada envolve dois nós, um emissor e um receptor. Uma chamada está sujeita às seguintes restrições:

a) cada nó tem um alcance de transmissão limitado e, portanto, só poderá enviar mensagens para interfaces próximas;

b) ao transmitir uma mensagem, um nó provoca interferência em outros que se encontrem a uma certa distância dele. Estes outros nós não podem receber nenhuma mensagem enquanto a primeira transmissão citada não se completar.

Logo, considerando estas duas restrições, uma mensagem sendo transmitida por um nó só poderá ser corretamente recebida se o destinatário da mensagem estiver no raio de alcance do transmissor e não há interferência por outra mensagem sendo transmitida simultaneamente. Neste contexto, teremos o seguinte problema:

Problema de *t*-gathering : considere que cada nó da rede possui uma mensagem. O problema de *t-gathering* consiste em coletar todas estas mensagens em um nó especial t da rede, chamado de *gathering node*.

É possível estruturar o problema como um conjunto parcialmente ordenado, no caso da rede em estudo ser um caminho, com base na ordem das chamadas ([7]). Denote por $\Pi_{-p}\Pi_p$ o caminho de comprimento $2p$, com $2p+1$ vértices $-p, -(p-1), \ldots, -1, 0, 1, 2, \ldots, p$, arestas $(-i, -(i-1))$ e $(i, i-1)$ e *gathering node* $t = 0$. Uma chamada do nó i $(-i)$ para o nó $i-1$ $(-(i-1))$, transmitindo a mensagem x $(-x)$, é referenciada por X_i $(-X_i)$. Uma *rodada* é um conjunto de chamadas que não interferem com nenhuma outra. Em [7] as seguintes propriedades foram enunciadas:

- X_i $(-X_i)$ ocorre em uma rodada anterior à rodada onde X_j $(-X_j)$ ocorrer se $i > j$;
- X_i $(-X_i)$ ocorre em uma rodade anterior à rodade onde Y_i $(-Y_i)$ ocorrer se $x > y$.

As propriedades acima definem claramente uma ordem parcial R, onde

$$\begin{aligned} X_i &\ R\ Y_j \text{ se } x \leq y \text{ e } i \geq j \\ -X_i &\ R\ -Y_j \text{ se } x \leq y \text{ e } i \geq j \end{aligned}$$

quando $x \neq y$ ou $i \neq j$.

2.3.3 *Job-shop scheduling*

A terceira aplicação é o clássico problema de sequenciamento em máquinas, conhecido também como *job-shop scheduling*. Na sua forma mais conhecida, consiste em encontrar uma sequência para processar m itens em q máquinas, onde cada item consiste em uma cadeia de operações e cada operação precisa ser executada durante um certo intervalo de tempo ininterrupto em uma máquina. Cada máquina só pode realizar uma operação por vez. Além do tempo de duração de cada operação, sabe-se que cada operação pertinente a um item deve ser executada em uma sequência previamente estabelecida e cada operação, independente a que item pertença, está alocada para uma máquina específica. Há liberdade de escolha para as operações a serem executadas em cada máquina. A sequência ótima será aquela que minimize o tempo total de execução (tempo necessário para executar todas as operações de todos os itens).

Em [2], o autor mostra como representar uma instância deste problema através de um grafo disjuntivo $D = (N; Z, W)$, que é definido associando-se:

- um nó $j \in N$ a cada operação, incluindo dois nós artificiais: nó 1 ("início") para ser a fonte de D e nó n ("final") para ser o sumidouro de D;
- um arco $(i, j) \in Z$ a cada par de operações relativas a um mesmo item e adjacentes na sequência de operações daquele item; também, um arco $(1, h) \in Z$ para cada nó h que represente a primeira operação a ser executada em um item. De forma análoga, um arco $(k, n) \in Z$ para cada nó k que represente a última operação a ser executada em um item;
- um par de arcos disjuntivos $(i, j) \in W$, $(j, i) \in W$, para cada par de operações i e j a serem executadas em itens diferentes, mas na mesma máquina;
- um valor (real não negativo) para cada arco $(i, j) \in W \cup Z$ igual à menor diferença de tempo entre o início das operações i e j.

Cada par de arcos disjuntivos $[(i, j), (j, i)]$ expressa a condição de que uma das duas operações, i ou j, deve ser executada antes da outra iniciar. Com isso, uma solução para o job-shop é definida como o comprimento do caminho máximo em uma orientação para D que respeite a sequência de operações de cada item.

Particurlamente, uma instância deste problema terá o seguinte poset relacionado $Z = (A, R)$. O conjunto A será formado pelo conjunto de operações existentes, independente da sua relação com os m itens. A ordem parcial R é definida como:

$$x \, R \, y \leftrightarrow \text{ a operação } x \text{ deve ser executada antes da operação } y, \, \forall y \in A, x \in A$$

3

Extensões Induzidas de Altura Mínima

Neste capítulo definimos o *problema da extensão induzida de altura mínima de um conjunto parcialmente ordenado* e o relacionamos com o problema de coloração de grafos. Esta relação é utilizada para mostrar que este problema faz parte da classe de problemas de difícil resolução. Além disso, a relação da resolução deste problema com aqueles apresentados no Capítulo 2 é estabelecida.

Na Seção 3.1 o problema é descrito formalmente. Na Seção seguinte mostramos como a resolução dos problemas apresentados na Seção 2.3 pode ser obtida através de uma extensão de altura mínima induzida por elementos específicos de seus posets. A relação deste problema com o problema de coloração de grafos e a prova de sua NP-Completude, utilizando esta relação, são descritas nas seções 3.3 e 3.4, respectivamente.

3.1 Descrição do problema de extensões induzidas de altura mínima

O nosso problema pode ser descrito da seguinte forma. Seja V um conjunto não vazio de $|V|$ elementos. Defina uma ordem parcial $\prec$ em V, que indica relações de precedência entre os elementos, e denote por $P = (V, \prec)$ o conjunto parcialmente ordenado definido por V e $\prec$. Defina também um subconjunto E de pares de elementos distintos de V, incomparáveis em P. O *problema da extensão induzida de altura mínima de um conjunto parcialmente ordenado* (EIP) consiste em encontrar uma extensão de $\prec$ em V, $\prec^*$, induzida pelos pares em E, de tal modo que

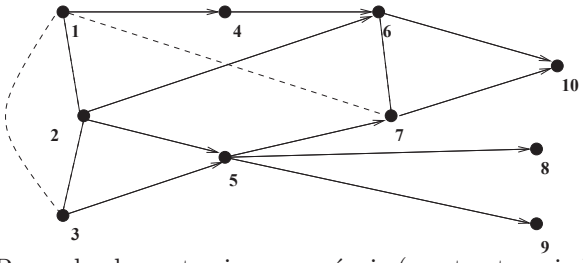
$$(i,j) \in E \;\Rightarrow\; i \;\prec^*\; j \text{ ou } j \;\prec^*\; i,$$

e que a altura da extensão $P^* = (V, \prec^*)$ de P seja a menor possível (ver Figura 3.1). Em outros termos, deseja-se atribuir uma ordem a cada par de elementos em E, de tal maneira que esta ordenação não cause conflitos com a ordem parcial já estabelecida por $\prec$, minimizando a altura do novo poset.

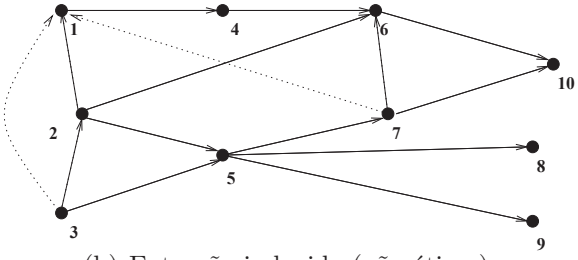
Outra forma interessante de definir o EIP é através do uso da noção de arcos disjuntivos, reduzindo-se o problema a encontrar um caminho maximal de comprimento mínimo em um grafo disjuntivo ([2]), como visto na seção anterior.

Neste contexto, para o EIP dado por $(V, \prec)$ e E, defina o grafo disjuntivo $D = (V; \vdash, W)$, onde $\vdash$ é a redução transitiva de $\prec$ e W contém o par de arcos disjuntivos (i,j) e (j,i) para todo $(i,j) \in E$. Assim, encontrar $\prec^*$ equivale a encontrar o menor caminho maximal em D.

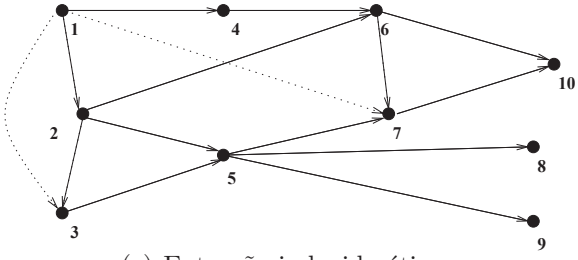
Em referência ao EIP, vamos usar a seguinte notação. Denotamos por $\vdash$ a redução transitiva de $\prec$. Consideremos ainda os posets P_{i^-} e P_{i^+} induzidos por $N^-_{\prec}[i]$ e $N^+_{\prec}[i]$, respectivamente, para todo $i \in V$, que utilizaremos ao longo deste texto. Chamamos de *EIP induzido por* $V' \subseteq V$, o subproblema definido pelo poset $(V', \prec [V'])$ e pelo conjunto $E[V'] = \{(i,j) \in E : i \in V', j \in V'\}$.



(a) Pares de elementos incomparáveis (arestas tracejadas)



(b) Extensão induzida (não ótima)



(c) Extensão induzida ótima

Figura 3.1: Exemplos de extensões induzidas. A extensão em 3.1(c) é ótima.

Vamos supor ao longo do texto que o *poset* P de qualquer instância para o EIP possua somente um elemento maximal e um elemento minimal. Caso isto não ocorra, podemos facilmente incluir dois elementos artificais para representar o único elemento minimal e maximal de P, que denotaremos por 1 e n, respectivamente, após realizar as devidas correções quanto à representação numérica dos demais elementos de V. Os elementos 1 e n são adicionados a V com $1 \in N_{\prec}^{-}[i]$ e $n \in N_{\prec}^{+}[i]$, para todo $i \in V$. Para isso, faça $V = V \cup \{1, n\}$ e $\prec = \prec \cup \{(1, i), (i, n) : i \in V\}$ (ver Figura 3.2).

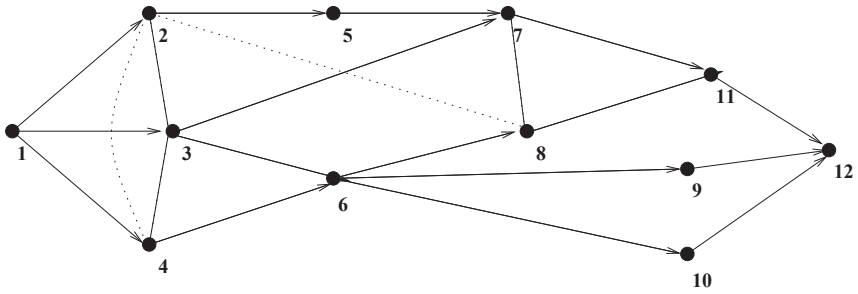


Figura 3.2: O grafo induzido pela instância do problema (P, E) com os elementos artificiais 1 e 12.

3.2 Aplicações

Esta seção busca complementar o que já foi visto na Seção 2.3, definindo o conjunto E necessário para a descrição completa de uma instância para o EIP. Os mesmos problemas apresentados naquela seção serão revisitados aqui com o objetivo de complementar a descrição de suas instâncias utilizando os conceitos vistos na descrição de uma instância para o EIP.

Escalonamento de programas em processadores Este problema pode ser modelado como um EIP se tomarmos módulos de tempo unitário de execução, juntamente com sua ordem de precedência, para definir um poset, enquanto que o conjunto E será composto pelos pares de módulos que necessitem de uma orientação, ou seja, aqueles alocados para um mesmo processador. Claramente esta situação é um

exemplo da classe de problemas descritos na Seção 2.3, onde os módulos são as tarefas a serem realizadas e o recurso a compartilhar é o processador alocado para o módulo em questão.

Uma solução para o EIP, neste caso, representaria uma solução ótima para o problema, já que iria definir uma ordem de execução entre todos os módulos de tal maneira que o tempo total seria o menor possível. Além disso, dois módulos que necessitem utilizar um mesmo processador seriam alocados em momentos diferentes.

***t-gathering problem* em caminhos** Já vimos como modelar este problema como um poset, através das rodadas onde as chamadas devem ocorrer. As restrições de interferência nos fornecem os meios para definir que chamadas não podem ser efetuadas ao mesmo tempo, se elas não se encontram relacionadas pela ordem original. Deste modo, estes pares de chamadas irão compor o conjunto E necessário para definir uma instância do EIP.

A resolução do EIP, neste caso, fornecerá a menor quantidade de rodadas necessária para resolver o problema de *t-gathering*. Os elementos que possuem a mesma altura na extensão final ótima representam chamadas que não interferem entre si. Note que estes elementos definem uma rodada. Logo, uma extensão induzida de altura mínima gera a menor sequência de rodadas para que todas as mensagens chegem ao nó t.

***Job-shop scheduling* com tempos unitários** As operações que estejam programadas para serem executadas em uma mesma máquina são escolhas naturais para compor o conjunto E. Entretanto, elas devem possuir tempo unitário de execução. Cada par de operações programadas para serem executadas em uma mesma máquina induzem um *único* par em E. O conjunto E, juntamente com o poset Z definido na Seção 2.3.3, formam uma instância completa para o EIP.

Uma solução ótima para o EIP leva a uma ordenação das operações agendadas em um mesmo processador, mas com o detalhe de que, claramante, esta ordenação será aquela que levará executando a menor quantidade de tempo possível, ou seja, uma solução ótima para o problema de *job-shop scheduling*.

3.3 Extensões induzidas e número cromático

Nesta seção discutimos a relação existente entre uma extensão de um poset e o número cromático do seu grafo representativo. Denote por G_P o GDA induzido por $P = (V, \prec)$ e por G_E o grafo simples obtido retirando a orientação dos arcos de G_P e adicionando as arestas presentes no conjunto de pares de elementos incomparáveis E. Estes dois grafos serão importantes ao longo do texto.

Uma observação trivial é dada a seguir.

Proposição 1 *: Dada uma solução viável $\prec'$ para o EIP, os elementos $i \in V$ que possuem a mesma altura formam um conjunto independente em G_E.*

Portanto, qualquer solução viável $\prec$ para o EIP consiste também em um particionamento de V em $h(\prec')$ conjuntos independentes, que por sua vez definem uma $h(\prec')$-coloração para o grafo G_E (ver Figura 3.3).

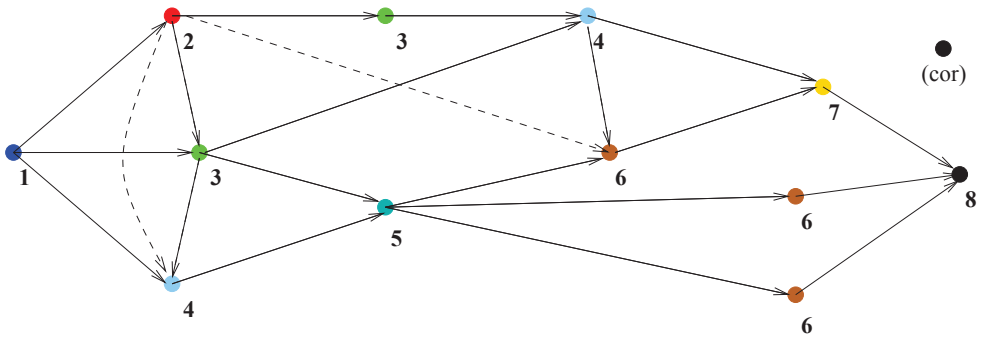


Figura 3.3: Uma 8-coloração em $G_{P'}$. $\chi(G_{P'}) = 8$.

A seguir mostramos que o número cromático do grafo G_E é um limite inferior para o EIP.

Proposição 2 *: Se $\prec^*$ é uma solução ótima para o EIP $(\prec, E)$, então $\chi(G_E) \leq h(\prec^*)$.*

Prova: Pela Proposição (1) temos $h(\prec^*)$ conjuntos independentes em G_E. Como a menor partição em conjuntos independentes em G_E é dada por $\chi(G_E)$, temos que $\chi(G_E) \leq h(\prec^*)$.□

Corolário 3 *: Qualquer limite inferior para o número cromático de G_E é um limite inferior para o EIP.*

Dois limites inferiores conhecidos para o número cromático $\chi(G)$ de um grafo G são o tamanho da maior clique $w(G)$ e seu número cromático fracionário $\chi_F(G)$. Calcular esses valores, entretanto, são problemas $\mathcal{NP}$-difíceis. Por outro lado, existem algoritmos que determinam boas aproximações para $w(G)$ e $\chi_F(G)$ ([10, 12]). Baseados em resultados computacionais da literatura, decidimos usar o método proposto em [12], que determina bons limites inferiores para o número cromático fracionário através de um procedimento de planos de corte. Com isso esperamos ter resultados que possam ser utilizados na avaliação da qualidade das soluções que propomos neste trabalho.

3.4 NP-Completude

Neste seção mostramos que o EIP pertence à classe de problemas $\mathcal{NP}$-completo através de uma redução do problema de coloração de um grafo.

Primeiro devemos mostrar que o EIP faz parte da classe de problemas $\mathcal{NP}$. Este fato é facilmente comprovado tomando-se uma extensão $P' = (V, \prec')$ para o EIP definido pelo poset $P = (V, \prec)$ e por um conjunto E de pares de elementos incomparáveis segundo P. Chamamos P' de um *certificado*. Além deste certificado, é necessário um procedimento capaz de aferir, em tempo polinomial no tamanho da entrada, se P' é realmente uma extensão induzida por E em P. Tal procedimento terá que verificar em P' a presença de ciclos e a correta ordenação do conjunto (isto é, as novas relações criadas respeitam a ordem já existente entre os elementos). O simples fato de procurar por ciclos já leva tempo polinomial para ser realizada, fazendo com que o procedimento de verificação também leve tempo polinomial para validar qualquer certificado. Portanto, EIP faz parte de $\mathcal{NP}$.

O próximo passo é reduzir um problema $\mathcal{NP}$-completo já conhecido para o EIP. O problema a ser utilizado nesta redução será o problema de coloração de grafos. Antes, porém, devemos definir os dois problemas como *problemas de decisão*, isto é, problemas cujas respostas são *sim* ou *não*. O EIP pode ser reformulado como um problema de decisão EIP_k da seguinte maneira:

Problema da extensão induzida de altura k : Dados um inteiro k, um conjunto parcialmente ordenado $P = (V, \prec)$ e um subconjunto $E_\prec$ de pares de elementos distintos de V, incomparáveis em P, existe uma extensão de P induzida por $E_\prec$ de altura até k?

Do mesmo modo, o problema de decisão para coloração de um grafo, $\chi_k(G)$, pode ser enunciado como:

Problema do número cromático k **de um grafo** : Dados um inteiro k e um grafo $G = (V, E)$, existe uma coloração própria para G que utilize até k cores?

A redução do problema de coloração para o EIP é baseada no seguinte resultado acerca de coloração e orientações acíclicas. Considere uma orientação σ e o grafo direcionado G_σ.

Deming ([19]) mostrou que se pode determinar o número cromático $\chi(G)$ de um grafo G a partir do conjunto de orientações acíclicas de G. Precisamente, o valor de $\chi(G)$ é a solução ótima do seguinte problema de otimização:

Proposição 4 $\chi(G) = \min_{\sigma \in \Omega} \max_{\overrightarrow{p} \in \mathcal{P}_\sigma} |\overrightarrow{p}| + 1$*, onde Ω é o conjunto de todas as orientações acíclicas de G e $\mathcal{P}_\sigma$ é o conjunto de caminhos em G_σ.*

A Proposição 4 permite caracterizar a complexidsade de EIP como a seguir.

Proposição 5 *: O problema EIP_k é $\mathcal{NP}$-completo.*

Prova: Nossa demonstração consiste em mostrar uma redução do problema de coloração de grafos $\chi_k(G)$ para o EIP_k. Considere uma instância $G = (V, E)$ para o problema $\chi_k(G)$. Podemos criar em tempo polinomial uma instância $\{P = (V_\prec, \prec), E_\prec\}$ para o EIP_k:

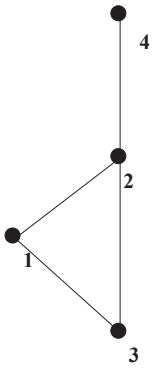


Figura 3.4: Grafo $G = (V, E)$ com 4 vértices e 4 arestas

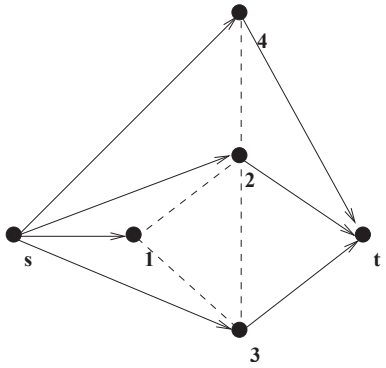


Figura 3.5: Poset derivado de G com 6 elementos e $|E_{\prec}| = 4$

- $V_{\prec} = V \cup \{s, t\}$, onde s e t são dois elementos artificiais que desempenharão o papel de vértices minimal e maximal, respectivamente, em P;
- $\prec = \{(s, i) \,|\, \forall i \in V\} \cup \{(j, t) \,|\, \forall j \in V\}$;
- $E_{\prec} = E$.

Agora note que toda orientação de G define uma extensão de P induzida por $E_{\prec}$, e vice-versa. Sendo assim, pela Proposição 4, a extensão de altura $k + 2$, P^*, induzida por $E_{\prec}$, fornece uma k-coloração própria para G. Mais precisamente, considerando a existência dos elementos artificiais s e t, temos que $k = h(P^*) - 2$.

Portanto, o problema de coloração de grafos $\chi_k(G)$ pode ser reduzido para o EIP$_k$, mostrando assim que EIP pertence à classe $\mathcal{NP}$-completo.□

4

Limites Superiores

Este capítulo se destina à exposição de algoritmos para cálculo de limites superiores para o EIP propostos neste trabalho. Os primeiros métodos se baseiam na altura dos posets induzidos pelos predecessores e sucessores de cada elemento em V, enquanto que os últimos são adaptações de procedimentos de coloração de grafos.

Apresentamos a descrição de cada método e alguns detalhes de implementação quando for necessário, deixando os resultados computacionais e sua análise para o Capítulo 7.

4.1 Limites por subordens induzidas

Descrevemos a seguir um método heurístico para se obter uma solução viável para o EIP. Este método atua sobre os pares em E, escolhendo, de forma gulosa, cada um destes elementos, de forma iterativa. Uma vez escolhido um elemento em E, resta decidir que orientação dar ao novo par a ser criado, usando novamente um critério guloso. O procedimento geral está descrito no Algoritmo 1.

Após a exposição do algoritmo geral, faremos uma análise dos principais pontos relacionados à obtenção de uma boa solução, propondo maneiras diferentes de se trabalhar com eles.

Algoritmo 1 *Greedy*

1: $P' = P$
2: $h_i^+ = h(P'_{i^+}), h_i^- = h(P'_{i^-}), \forall i \in V$ tal que $(i,j) \in E$ ou $(j,i) \in E$
3: **enquanto** $E \neq \emptyset$ **faça**
4: Remova um par $(i,j) \in E$ e oriente-o, utilizando os valores de $h_i^-, h_i^+, h_j^-, h_j^+$
5: Acrescente o par orientado a P'
6: $h_i^+ = h(P'_{i^+}), h_i^- = h(P'_{i^-}), \forall\, (i,k) \in E$ onde $i,k \in V$
7: **fim enquanto**
8: **retorne** P' e $h(P')$

O método itera sobre os pares $(i,j) \in E$ até que não reste em E arestas sem orientação. A cada iteração, tanto a escolha do par $(i,j) \in E$ como o processo de decisão de sua orientação seguem um critério guloso, baseado nas diferenças entre as alturas dos posets P'_{i^+},P'_{i^-}, P'_{j^+} e P'_{j^-}, onde procuramos orientar primeiro aqueles pares em E que detêm as maiores diferenças. A idéia é que evitemos atribuir a $(i,j) \in E$ uma orientação que aumente muito a altura da ordem do poset original $\prec$, baseado na diferença das alturas dos posets já mencionados. Note que quanto maior esta diferença, maior seria o fator de crescimento da altura do poset original no caso de uma orientação mal designada.

Neste algoritmo, a idéia é que h_i^+ e h_i^- representem uma boa aproximação de $h(P^*_{i^+})$ e $h(P^*_{i^-})$, respectivamente, onde P^* é o poset ótimo. Sendo assim, caso uma outra aproximação melhor que $h(P'_{i^+})$

e $h(P'_{i^-})$ esteja disponível, ela poderá ser utilizada. No Capítulo 5 mostraremos como calcular limites inferiores para $h(P^*_{i^+})$ e $h(P^*_{i^+})$ usando programação por restrições.

Propomos a seguir quatro maneiras para definir o critério guloso utilizado na linha 4 do algoritmo descrito acima. Para cada maneira, enumeramos, logo em seguida, as opções para decidir a orientação da aresta compatíveis com cada critério escolhido. Observe que decidimos manter a notação utilizada para as variáveis do algoritmo durante a exposição dos critérios, ou seja, $h_i^+ = h(P'_{i^+})$ e $h_i^- = h(P'_{i^-})$.

As opções são as seguintes:

i) $\max_{(i,j)\in E} |h_i^- - h_j^-|$;

   (a) $\overrightarrow{ij}$ se $h_i^- \leq h_j^-$, $(i,j) \in E$;

   (b) $\overrightarrow{ji}$ se $h_j^- < h_i^-$, $(i,j) \in E$;

ii) $\max_{(i,j)\in E} |h_i^+ - h_j^+|$;

   (c) $\overrightarrow{ij}$ se $h_j^+ \leq h_i^+$, $(i,j) \in E$;

   (d) $\overrightarrow{ji}$ se $h_i^+ < h_j^+$, $(i,j) \in E$;

iii) $\max\{\max_{(i,j)\in E}\{|h_i^- - h_j^-|\}, \max_{(i,j)\in E}\{|h_j^+ - h_i^+|\}\}$;

   (e) se $|h_i^- - h_j^-| \geq |h_j^+ - h_i^+|$: (a) ou (b);

   (f) se $|h_j^+ - h_i^+| > |h_i^- - h_j^-|$: (c) ou (d);

iv) $\max_{(i,j)\in E} |h_i^- - h_j^- + h_j^+ - h_i^+|$;

   (g) se $|h_i^- - h_j^-| \geq |h_j^+ - h_i^+|$: (a) ou (b);

   (h) se $|h_j^+ - h_i^+| > |h_i^- - h_j^-|$: (c) ou (d).

É importante notar que para o cálculo de h_i^+ e h_i^-, para todo $i \in V$, é realizado um procedimento baseado em programação dinâmica, já que $h_i^- = \max\{h_j^- \mid j \vdash i\} + 1$. Na inicialização calculamos h_i^+ e h_i^- para todo $i \in V$ que participa de algum par $(i,j) \in E$. Entretanto, a cada iteração, os valores de h_i^+ e h_i^- só precisam ser atualizados para sucessores e antecessores do par escolhido, de acordo com a orientação dada. Note ainda que em *(i)* e *(ii)* apenas h_i^- ou h_i^+ precisam ser calculados, respectivamente.

Nos experimentos computacionais deste trabalho, as duas últimas opções foram escolhidas para implementação por contemplarem mais informações sobre o grafo, tanto no momento da escolha de um par a ser ordenado quanto na própria ordenação deste. Em ambos os casos, a cada vez que um par é ordenado, é necessário atualizar os valores das alturas de cada vértice que incidem sobre ele, se for o caso, através do mesmo procedimento de programação dinâmica utilizado para o cálculo dos seus valores iniciais. Logo, a complexidade total dos procedimentos utilizados é de $\mathcal{O}(|E|n^2)$, já que é necessário iterar sobre todos os pares em E ($\mathcal{O}(|E|)$) e atualizar as alturas relacionadas a cada vértice incidente em um processo recursivo ($\mathcal{O}(n^2)$).

Adotamos também, no caso do critério (*iii*), uma regra de desempate para a escolha do par a ser ordenado que funciona da seguinte maneira: para cada par (i,j) que fornece o máximo em (*iii*), consideremos o segundo maior valor entre $|h_i^- - h_j^-|$ e $|h_i^+ - h_j^+|$. Decidimos então pelo par com segundo maior valor. Em caso de novo empate, este será solucionado por uma escolha aleatória. As Figuras 4.1 e 4.2 ilustram a execução do algoritmo 1 utilizando o critério (*iii*).

4.2 Limites por coloração de grafos

Os demais métodos para o cálculo de limites superiores se baseiam no conceito de coloração em grafos, podendo ser divididos de acordo com a idéia central em que se baseiam: uma abordagem de coloração gulosa, com um modo especial de designação de cores, e outra de coloração *first-fit*.

Essencialmente, uma coloração de um grafo é uma partição dos seus vértices em conjuntos independentes, ou seja, conjuntos de vértices que não se relacionam por arestas. Esta característica também se manifesta no EIP, já que elementos de mesma altura induzem uma partição de conjuntos independentes.

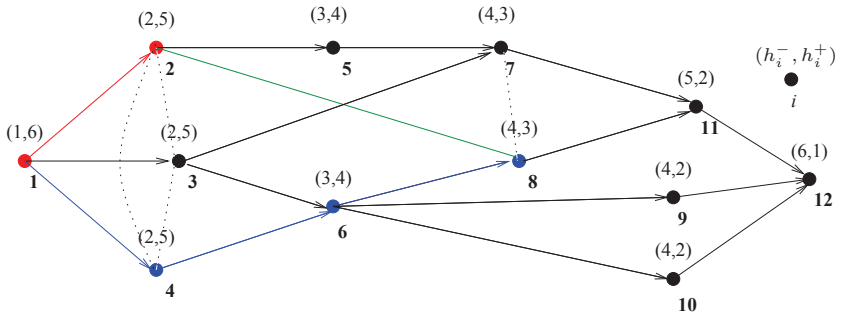


Figura 4.1: $(2, 8)$ é o primeiro par a ser escolhido, de valor 2.

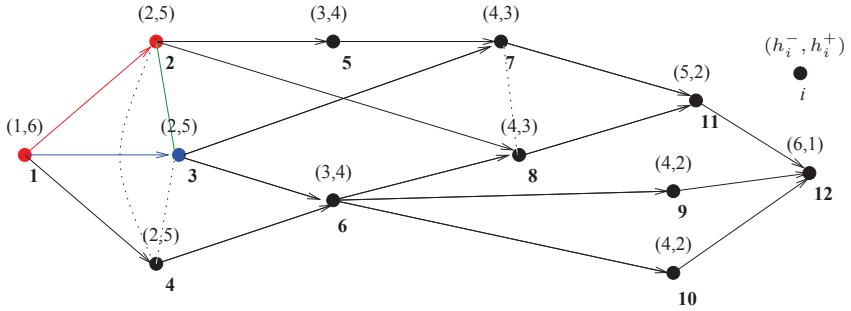


Figura 4.2: Segundo par escolhido $(2, 3)$, dada a orientação do par anterior $(2, 8)$.

Além disso, como visto na Seção 3.4, as orientações acíclicas do grafo são capazes de determinar seu número cromático. Deste modo, tentamos seguir uma linha oposta: tentar obter uma solução para o nosso problema, envolvendo orientações acíclicas do grafo, a partir das possíveis colorações para este grafo.

Essa relação entre coloração de G_E e solução viável para EIP, ou seja, a busca por uma coloração que induza uma orientação adequada para nossa nossas instâncias, é explorada nas heurísticas deste capítulo.

4.2.1 Coloração iterativa

Este método funciona de forma iterativa, concentrando-se a cada iteração no subgrafo de G_E induzido pelos vértices minimais presentes no subgrafo de G_P induzido pelos vértices ainda não coloridos. Cada subgrafo então é colorido de uma maneira especial que depende das cores atribuídas anteriormente aos demais vértices. Novamente, após a exposição do algoritmo faremos uma análise dos principais pontos que acreditamos estarem relacionados à obtenção de uma boa solução para o EIP.

A idéia básica do algoritmo é colorir, de forma iterativa, subgrafos induzidos por vértices de mesma altura em G_P. Para este fim, propomos um modo de coloração composto por duas fases e dependente da coloração realizada na iteração anterior (ver Algoritmo 2). Os subgrafos utilizados a cada iteração possuem como vértices aqueles que ainda não foram coloridos e cujos antecessores em $\prec$ já o foram. O conjunto destes vértices é denotado por V_{MIN}.

Na primeira parte procuramos realizar um procedimento de pré-coloração dos vértices em V_{MIN} usando cores já utilizadas, caso exista alguma. Os vértices de V_{MIN} são ordenados em uma lista L de forma crescente em relação à menor cor com a qual *podem* ser coloridos em G_E. Então, nesta ordem, cada vértice é considerado. Determina-se a menor cor c' a partir da qual o vértice pode ser colorido (o valor anterior c já não pode ser válido se algum vértice já tiver sido colorido durante esta iteração), considerando a coloração corrente de seus vizinhos em G_E. Por exemplo, suponha que até o momento foram utilizadas $ncolors$ cores e que v é o vértice de L a ser analisado. Caso seja possível colorir v com a cor $ncolors$, procuramos saber se uma coloração com a cor $ncolors - 1$ seria possível, e assim por diante, até encontrarmos a primeira cor $c' - 1$ com a qual não seja possível colorir v. Neste caso, atribuímos a v a cor c' e seguimos adiante com o próximo vértice em L. Caso contrário, v não é colorido nesse momento e

Algoritmo 2 *IterativeColoring*

1: $colored = \emptyset$
2: $ncolors = 0$
3: **enquanto** $colored \neq V$ **faça**
4: $V_{MIN} =$ vértices minimais em $G_P \setminus colored$
5: Ordene V_{MIN}
6: **para todo** $v \in V_{MIN}$ **faça**
7: Determine menor cor c tal que v pode receber qualquer cor $k \geq c$ em $G_E[v \cup colored]$
8: **se** $c \leq ncolors$ **então**
9: Colorir v com c
10: $colored = colored \cup \{v\}$
11: $V_{MIN} = V_{MIN} \setminus \{v\}$
12: **fim se**
13: **fim para**
14: Colorir grafo $G_E[V_{MIN}]$ com cores $ncolors + 1, \ldots, ncolors + k$, $k \geq 0$
15: $ncolors = ncolors + k$
16: $colored = colored \cup V_{MIN}$
17: **fim enquanto**
18: Ordene cada $(i, j) \in E$ do vértice de menor cor para o de maior cor, gerando $\prec'$
19: **retorne** $ncolors$ e $\prec'$

podemos seguir adiante com o próximo vértice em L. Após iterarmos sobre toda a lista L teremos parte dos vértices de V_{MIN} coloridos com cores já existentes e outra parte sem uma cor atribuída a eles. Aqui começa a segunda fase da coloração.

Nesta segunda fase, consideramos o grafo $G_E[V_{MIN}]$, que é formado pelos vértices em V_{MIN} ainda não coloridos e pelas arestas presentes em E cujas extremidades estejam em V_{MIN}. Observe que este grafo é não direcionado. Nesta fase, podemos colorir $G_E[V_{MIN}]$ através de qualquer método (heurístico) desejado, utilizando cores maiores que as $ncolors$ já utilizadas, ou seja, o método escolhido começa a colorir a partir da cor $ncolors + 1$.

Após a fase de coloração, os vértices coloridos nesta iteração juntam-se àqueles anteriormente coloridos para, na próxima iteração, o processo recomeçar, até que todos os vértices tenham sido coloridos. Cada cor na coloração obtida representa uma "altura" em $\prec'$, ou seja, a cor atribuída a i é a altura do *poset* P'_{i^-} induzido por $N^-_{\prec'}[i]$, $i \in V$, tomando a cor inicial como 1. A partir das cores (ou alturas) podemos orientar as arestas em E do elemento de menor altura para o de maior altura.

A idéia principal do método é tentar atribuir uma coloração própria ao grafo G_E respeitando a ordem de precedência, ou seja, cada vértice deve receber uma cor maior que a de seus antecessores. Procurando minimizar a altura, tentamos atribuir uma cor já usada ao maior número possível de vértices, permitido pela ordem de precedência. A primeira fase do processo de coloração reflete esta idéia. Somente após esta tentativa, se necessário, novas cores são atribuídas aos demais vértices, considerando somente as arestas em E.

Note que a primeira fase da coloração segue um procedimento definido por nós e bastante usual no sentido de ser basicamente uma coloração gulosa do subgrafo induzido, mas quanto à segunda há liberdade na escolha de que método utilizar para a coloração dos demais vértices. Isto gera uma certa flexibilidade no tratamento de instâncias muito densas ou esparsas em relação aos arcos de E de forma a avaliar vários critérios. No nosso caso, implementamos um método de coloração *first-fit* utilizando os seguintes critérios de ordenação para os vértices:

- *Grau dos vértices*: os vértices em V_{MIN} são ordenados em ordem decrescente de grau em $G_E[V \setminus colored]$. Com isso buscamos diminuir o impacto de colorir um vértice de grau elevado com uma cor que não seja a menor possível, o que vai levar, na iteração seguinte, a um número maior de vértices com cores mal atribuídas;

- *Altura*: os vértices são ordenados em ordem descrescente da altura $h(P_{i^+})$. Este critério tenta evitar a atribuição de cores grandes para vértices com um grande caminho até o vértice maximal, que têm maior chance de participar do caminho crítico no poset final.

Com a utilização da atribuição de cores especial descrita, é possível realizar a coloração do grafo G_E partindo da redução transitiva de $\prec$. Portanto, prodemos trabalhar com uma estrutura menor tanto para o grafo quanto para os subgrafos, embora ainda seja necessária, a cada iteração, a criação das visões dos grafos. Se considerarmos $|V_{MIN}| = K$, a coloração *first-fit* implementada por nós (linha 14 do Algoritmo 2) tem complexidade $\mathcal{O}(K^2)$ e a coloração que acontece nas linhas $5 - 13$ é da ordem de $\mathcal{O}(Kn)$. A determinação do conjunto V_{MIN} é da ordem de $\mathcal{O}(n)$ e junto com o *loop* mais externo da linha 3 ($\mathcal{O}(n/K)$) temos uma complexidade de $\mathcal{O}(n^2)$.

Para exemplificar o funcionamento do procedimento acima descrito, utilizando o grau dos vértices, considere o grafo já visto em seções anteriores e reproduzido nas figuras a seguir. Na primeira iteração, o vértice 1 é o único em V_{MIN} e logo em seguida recebe a primeira cor. Como não há mais nenhum vértice a se colorir em V_{MIN}, vamos para a segunda iteração. Desta vez, os elementos que não possuem predecessores em $G \setminus colored$ são $V_{MIN} = \{2, 3, 4\}$. Na primeira fase de coloração, nenhum deles pode ser colorido com a cor 1, devido às relações de precedência com 1. Na segunda fase, devido aos arcos de E que incidem sobre V_{MIN}, três novas cores são criadas e atribuídas a cada um deles (ver Figura 4.3).

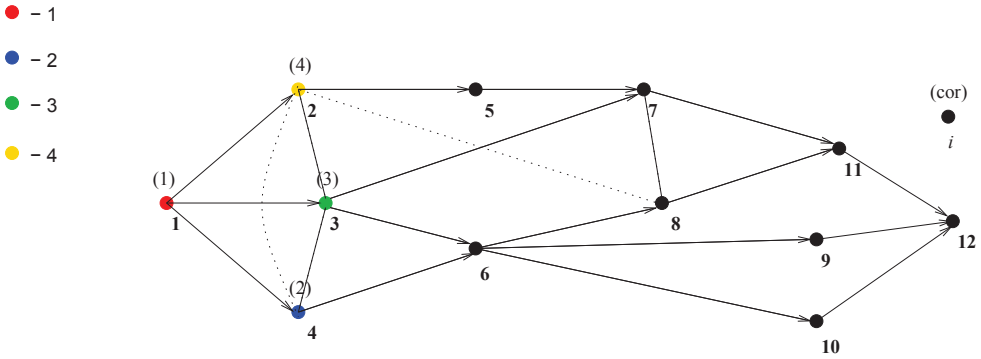


Figura 4.3: A coloração ao final da segunda iteração.

Na terceira iteração teremos $V_{MIN} = \{5, 6\}$. Na primeira parte da coloração desta iteração, concluímos que ao vértice 5 não pode ser atribuído uma cor já usada, pois ele é adjacente ao vértice 2, que recebeu a cor 4. Por outro lado, o vértice 6 pode receber a cor 4, e será assim colorido (ver Figura 4.4). Então, na segunda parte da coloração, resta colorir somente o vértice 5. Como nesta fase é proibido utilizar as cores anteriores, uma nova cor é atribuída ao vértice 5 (ver Figura 4.5). Continuando com esse processo, em mais 2 iterações chegamos à coloração da Figura 4.6, que fornece um limite superior igual a 8.

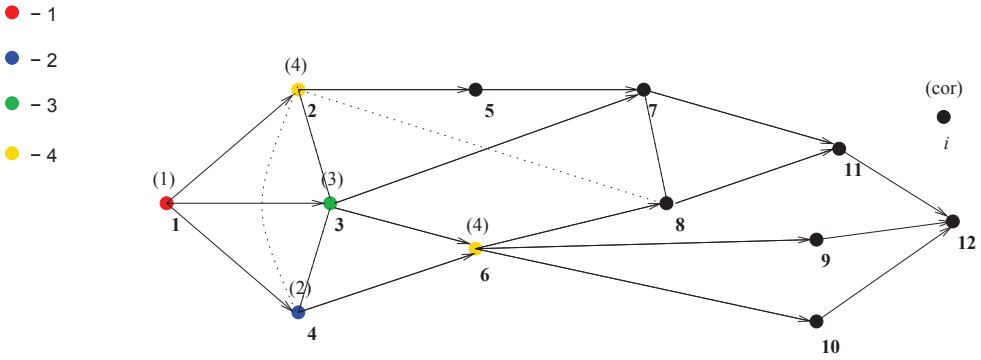


Figura 4.4: A coloração ao final da pré-coloração na terceira iteração.

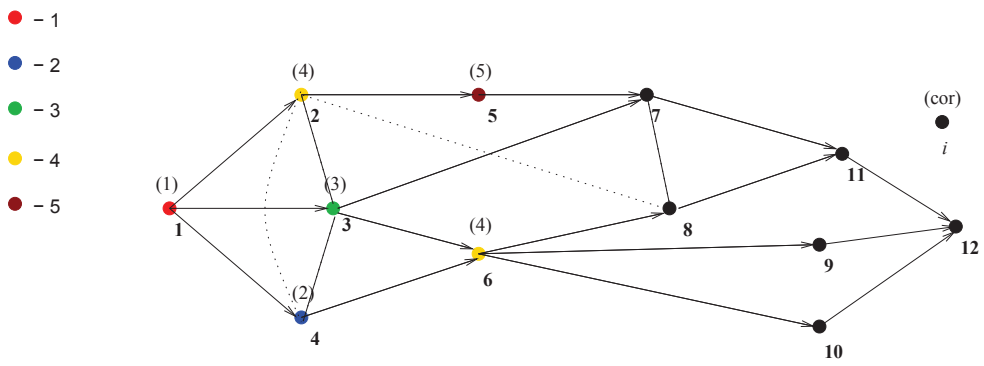


Figura 4.5: A coloração ao final da terceira iteração.

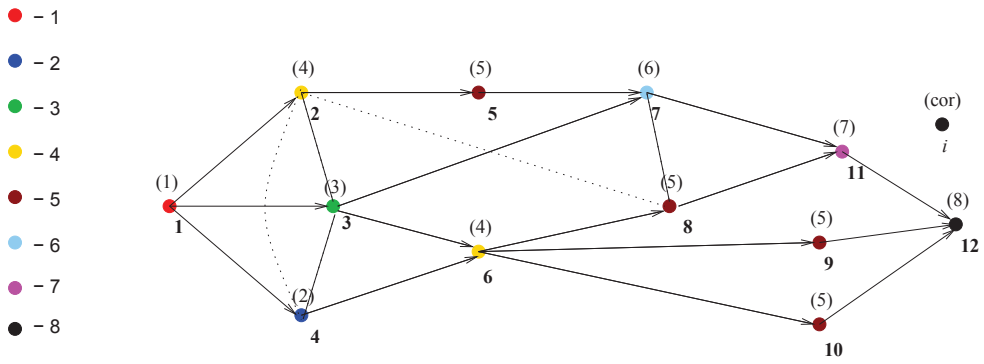


Figura 4.6: A coloração final.

4.2.2 Coloração *first-fit*

O próximo algoritmo combina idéias usadas nos dois anteriores, na medida que aplica uma coloração *first-fit*, com a ordenação dos vértices dada por uma função de valoração $f : V \to \mathbb{Z}$ baseada nas alturas dos elementos do poset. A função de valoração deve procurar medir, para cada vértice i, o quão próximo do vértice 1 e o quão distante do vértice n ele estará. Os vértices são escolhidos em uma ordem específica segundo f e recebem uma cor que deve ser diferente daquela dos seus antecessores, sucessores e demais vértices adjacentes a ele em G_E. O Algoritmo 3 exemplifica o esquema geral utilizado neste método, onde $c : V \to \mathbb{Z}^+$ é a função de atribuição de cores.

Algoritmo 3 *FirstFitPrimal*

1: $c(1) = 1, c(i) = -1, \forall i \in V \setminus \{1\}$
2: Determine $f(i)$, $\forall i \in V$
3: **enquanto** $\exists\, v, c(v) = -1$ **faça**
4: Escolha um vértice v tal que $f(v) \leq f(u)$, para todo $u \in V$
5: $c(v) = \max\{c(k) : k \vdash v \text{ ou } (k, v) \in E\} + 1$
6: Atualize $f(u)$ para todo $u \in V \setminus \{v\}$
7: **fim enquanto**
8: **retorne** $c(n)$

A maneira de colorir os vértices procura atribuir significado de altura dos vértices às cores, por isso é necessário somente contar com a redução transitiva da ordem parcial $\prec$ juntamente com as arestas induzidas por E. Ao longo do nosso trabalho foram utilizadas três definições para $f(i)$, baseadas nas alturas de P_{i^+} e P_{i^-} para cada elemento:

i) $f(i) = h(P_{i^-})$;

ii) $f(i) = h(P_{i^+})$;

iii) $f(i) = h(P_{i^-}) - h(P_{i^+})$.

Note que, do mesmo modo que podemos utilizar no primeiro método guloso valores mais próximos da altura final dos elementos para h_i^+ e h_i^-, também podemos aplicar a mesma idéia aqui e modificar a definição de f.

Similar ao Algoritmo 3, podemos definir um procedimento dual, que começa pelo vértice n e segue colorindo os vértices em ordem decrescente de $f(i)$, para cada uma das opções (i), (ii) e (iii).

Em nossos experimentos computacionais, quase todas as 6 possíveis versões foram implementadas e testadas, com execeção da definição (i), onde somente a ordenação primal foi testada, devido aos resultados mais promissores encontrados com a definição (iii), melhores do que a ordenação dual para a definição (ii).

A complexidade deste método é de $\mathcal{O}(n^3)$, já que é necessário iterar sobre cada vértice de V e a cada iteração uma atualização das alturas dos demais vértices é necessária, já que as definições para f que implementamos estão baseadas nestas mesmas alturas. Como já vimos anteriormente, tal atualização de alturas é da ordem de $\mathcal{O}(n^2)$.

A aplicação ao exemplo anterior do Algoritmo 3 com a definição (iii) pode ser visto na Figura 4.7, onde os rótulos dos vértices representam o valor obtido pela aplicação da função de valoração ao grafo inicial.

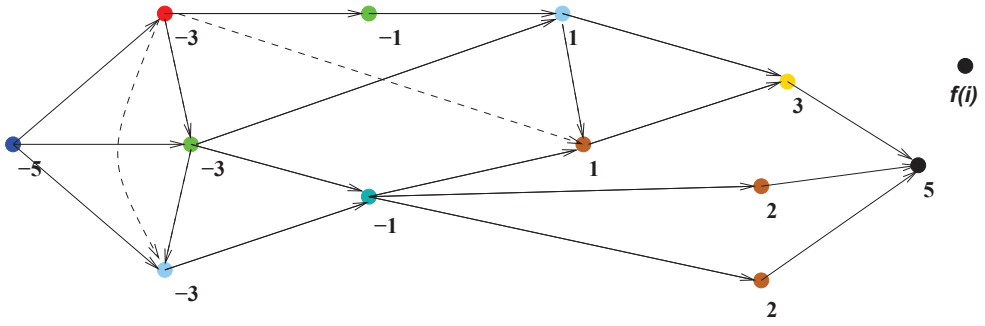


Figura 4.7: Uma solução viável representada por uma 8-coloração de G_E.

5

Limite Inferior Programação por Restrições

Este método se baseia em programação por restrições e é uma tentativa de utilizar procedimentos e técnicas que conseguiram bons resultados na resolução de problemas de escalonamento com restrições de recursos. Definimos o que vem a ser a programação por restrições, além de demais conceitos importantes, para em seguida descrever com mais detalhes em que consistiu nossa implementação.

Observamos que os procedimentos serão descritos para o grafo G_P, mas que podem ser aplicados a seus subgrafos induzidos, e portanto, podem fornecer limites inferiores para qualquer subproblema EIP induzido.

5.1 Programação por restrições

Nesta seção propomos a utilização da técnica de *programação por restrições* para obtenção de limites inferiores. A progamação por restrições é um paradigma de programação onde as relações entre as variáveis são estabelecidas através de restrições. Basicamente, consiste em descrever que critérios (restrições) a solução do problema deve obedecer, em vez de especificar uma sequência de passos para encontrá-la. Usando estas restrições iterativamente, procuramos reduzir o domínio inicial das variáveis, até que uma solução seja eventualmente encontrada.

No nosso trabalho, o emprego desta técnica consiste em aplicar ao problema, de forma iterativa, um conjunto de regras, que buscam determinar, para cada elemento no poset, o menor intervalo de variação de sua altura em uma solução ótima. Os resultados obtidos, mesmo que não determinem uma solução, podem ser utilizados para fortalecer as formulações de programação inteira que apresentamos no Capítulo 6, o que a torna bastante atrativa para ser utilizada também como técnica de pré-processamento do problema antes da resolução dos modelos.

Começamos apresentando o conceito de *matriz de distâncias de alturas*, que será utilizada durante o resto deste trabalho, além de ser a estrutura sobre a qual nossa implementação se apóia. Depois, apresentamos as regras utilizadas no nosso procedimento de programação por restrições e como elas são propagadas. Por último, abordamos duas técnicas utilizadas em conjunto com a programação por restrições que buscam fortalecer ainda mais seus resultados.

5.1.1 Matriz de distâncias de alturas

Durante o restante do texto fazemos uso do conceito de *matriz de distância de alturas* (chamada *SSD-matrix* em [17]). No nosso caso, uma matriz $B = (b_{ij})_{V \times V}$ é uma *SSD-matrix* se existe uma solução ótima $\prec^*$ para EIP tal que

$$h_j^* - h_i^* \geq b_{ij}, \text{ para todo } (i,j) \in V \times V, \tag{5.1}$$

onde h_i^* é a altura de i.

Assim como foi apontado em [17] para o problema de escalonamento de atividades com relações de precedência e restrições de recursos, decorre da condição (5.1) que o EIP pode ser visto como um problema de satisfação de restrições (CSP, em inglês) com variáveis $(h_j - h_i)$ e domínios $[b_{ij}, -b_{ji}]$. Consequentemente, os extremos deste intervalo também fornecem um limite inferior $b_{ij}+1$ e um limite superior $-b_{ji}+1$ para o EIP induzido pelos elementos que são simultaneamente sucessores de i e antecessores de j em $\prec$. De forma mais específica, note que $b_{1j}+1$ e $-b_{j1}+1$ são um limite inferior e superior, respectivamente, para a altura do elemento j em uma solução ótima, ou seja,

$$b_{1j} \leq h_j^* - 1 \leq -b_{j1}, \forall j \in V \tag{5.2}$$

Dado um limite superior T para o valor de uma solução ótima, uma possível inicialização para B é

$$b_{ij} = \begin{cases} 1 & i \vdash j, \\ 0 & i = j, \\ -T+1 & \text{caso contrário} \end{cases} \tag{5.3}$$

onde $\vdash$ é a redução transitiva de $\prec$. O limite superior T pode ser obtido por qualquer um dos procedimentos apresentados no Capítulo 4.

5.1.2 Regras

Apresentamos aqui 4 regras básicas que, combinadas, darão origem ao nosso algoritmo de programação por restrições. Tais regras procuram deduzir relações adicionais entre os elementos do poset, provocando ajustes nos domínios de suas alturas finais, ou melhor, aumentando algumas entradas de B. Durante este processo o valor de $b_{1n}+1$ é o limite inferior atual para o problema.

Consistência de caminhos Dados quaisquer $i, j \in V$, temos, trivialmente, por (5.1), que

$$h_j^* - h_i^* = (h_j^* - h_l^*) + (h_l^* - h_i^*) \geq b_{lj} + b_{il},\ \forall l \in V$$

Logo as entradas da matriz B podem ser atualizadas fazendo

$$b_{ij} = \max_{l \in V}\{b_{ij}, \max\{b_{il} + b_{lj})\}\},\ \forall i, j \in V$$

Tal atualização pode ser realizada pelo algoritmo de Floyd-Warshall (Algoritmo 4), cuja complexidade é $\mathcal{O}(n^3)$. Esse procedimento será aplicado a partir da inicialização descrita em (5.3), para calcular o fecho transitivo da matriz B, e sempre que várias entradas de B forem atualizadas por algum outro procedimento. No caso em que somente uma entrada b_{hl} é atualizada, podemos calcular a nova matriz em tempo $\mathcal{O}(n^2)$, atribuindo $b_{ij} = \max\{b_{ij}, b_{ih} + b_{hl} + b_{lj}\}$, para todo $(i, j) \in V \times V$. Chamamos este procedimento de *SimpleFloydWarshall*.

Algoritmo 4 *FloydWarshall*

```
1: para l = 1 até l = n faça
2:    para i = 1 até i = n faça
3:       para j = 1 até j = n faça
4:          b_ij = max{b_ij, b_il + b_lj}
5:       fim para
6:    fim para
7: fim para
```

Seleção imediata Caso $b_{ij} \geq 0$ e $(i,j) \in E$, então podemos acrescentar (i,j) em $\prec$. Esta simples regra é executada em $\mathcal{O}(|E|)$ Caso $b_{ij} = 0$ é necessário fazer, adicionalmente, $b_{ij} = 1$ e realizar uma chamada para o procedimento de *SimpleFloydWarshal* para uma entrada única, visando manter a consistência da matriz B.

Algoritmo 6 *ImmediateSelection*

```
1: para (i, j) ∈ E faça
2:    se b_ij >= 0 então
3:       E = E\{(i, j)}
4:       se b_ij == 0 então
5:          b_ij = 1
6:          SimpleFloydWarshall(b_ij)
7:       fim se
8:    senão
9:       se b_ji >= 0 então
10:         E = E\{(j, i)}
11:         se b_ji == 0 então
12:            b_ji = 1
13:            SimpleFloydWarshall(b_ji)
14:         fim se
15:      fim se
16:   fim se
17: fim para
```

Ordenação por cliques *Edge-finding* é o nome dado a técnicas particulares (de propagação de restrições) que estudam, em problemas de escalonamento com restrições de recursos, a ordem na qual várias atividades podem ser executadas em relação a um recurso específico ([15], [17], [3]). Consiste, por exemplo, em determinar se uma atividade deve, pode ou não pode ser executada antes (ou depois) de um conjunto de atividades que requerem o mesmo recurso. Podemos chegar a dois tipos de conclusões: novas relações de precedência e novos limites para a inicialização das atividades. Tais conclusões são derivadas a partir de *cliques de disjunções*, que são conjuntos onde qualquer par de elementos não pode ter a mesma altura.

No caso do EIP, esses conjuntos são cliques em G_E. Na primeira regra *edge-finding* que apresentamos a seguir o nosso objetivo é localizar, em uma clique C de G_E, que elemento deve ter a maior altura em relação aos demais. A propriedade abaixo fornece elementos para esse fim.

Proposição 6 *: Seja C uma clique em G_E e $j \in C$. Se*

$$\min_{i \in C} b_{1i} + |C| - 1 > \max_{i \in C \setminus \{j\}} \{-b_{i1}\} \tag{5.4}$$

então $h_j^* > h_i^*$, para todo $i \in C \setminus \{j\}$. Neste caso podemos atualizar

$$b_{1j} = \max\{b_{1j}, \max_{C' \subseteq C \setminus \{j\}} \{\min_{i \in C'} b_{1i} + |C'|\}\} \tag{5.5}$$

$$b_{i1} = \max\{b_{i1}, b_{j1} + 1\} \tag{5.6}$$

Algoritmo 5 *SimpleFloydWarshall*

```
1: para i = 1 até i = n faça
2:    para j = 1 até j = n faça
3:       b_ij = max{b_ij, b_ih + b_hl + b_lj}
4:    fim para
5: fim para
```

Prova: Como todos os elementos da clique C terão alturas diferentes, e usando (5.2), chegamos a:

$$\max_{i\in C} h_i^* \geq \min_{i\in C} h_i^* + |C| - 1 \geq \min_{i\in C} b_{1i} + |C|$$

Então, pela hipótese, e aplicando (5.2) novamente, obtemos

$$\max_{i\in C} h_i^* > \max_{i\in C\setminus\{j\}} \{-b_{i1}\} + 1 \geq \max_{i\in C\setminus\{j\}} h_i^*$$

Logo, $h_j^* > h_i^*$, $\forall i \in C \setminus \{j\}$, mostrando a primeira parte. Já as atualizações de b_{1j} e b_{i1}, $i \in C \setminus \{j\}$, seguem, respectivamente, das desigualdades

$$\begin{aligned} h_j^* - 1 &\geq \max_{i\in C\setminus\{j\}} h_i^* \geq \max_{C'\subseteq C\setminus\{j\}} \{\min_{i\in C'} b_{1i} + |C'|\} \\ h_i^* - 1 &\leq h_j^* - 2 \leq -b_{j1} - 1 \end{aligned}$$

□

Uma propriedade dual, que procura descobrir se um elemento deve ter a menor altura entre todos os elementos da clique, pode ser derivada de forma análoga como a seguir:

Proposição 7 *: Seja C uma clique em G_E e $j \in C$. Se $\max_{i\in C} -|C| + 1 < \min_{i\in C\setminus\{j\}} b_{1i}$ então $h_j^* < h_i^*$, $\forall i \in C \setminus \{j\}$. Neste caso*

$$b_{j1} = \max\{b_{j1}, \max_{C'\in C\setminus\{j\}} \{\min_{i\in C'} b_{i1} + |C|\}\} \tag{5.7}$$

A Proposição 6 justifica a definição do Algoritmo 7, uma adaptação direta daquele utilizado para o problema de escalonamento de tarefas com restrições de recursos em [17].

A complexidade de $\mathcal{O}(|C|^2)$ é mantida, onde $|C|$ é o tamanho daclique de entrada. Este procedimento não necessariamente localiza a subclique $C' \subseteq C \setminus \{j\}$ que fornece o máximo em (5.5), determinando apenas uma aproximação desse valor máximo. Na verdade, a cada iteração externa k, é avaliado um conjunto de subcliques $\mathcal{C}(k)$ de C tais que $\max_{i\in C'} -b_{i1} \leq -b_{ik}$, para todo $C' \in \mathcal{C}(k)$. Entre as linhas $6 - 14$ os valores $\min_{i\in C'} b_{1i} + |C'|$ são armazenados para cada $C' \in \mathcal{C}(k)$ pesquisado. O maior desses valores é guardado em $Cval$.

Então, para cada $j \in C$ que não pertence a qualquer clique em $\mathcal{C}(k)$, o teste (5.4) é realizado para duas cliques de $\mathcal{C}(k)$ apropriadas, adicionadas de j.

Algoritmo 7 *EdgeFindingPrimal*

Seja C uma clique em G_E
$v[i] = b_{1i}$, $\forall i \in C$
Ordene $C = \{v_1, v_2, \ldots, v_t\}$, $b_{1v_1} \leq b_{1v_2} \leq \ldots \leq b_{1v_t}$
para $k = v_1, v_2, \ldots, v_t$ **faça**
  $Cval = -\infty$, $P = 0$
  para $i = v_t, v_{t-1}, \ldots, v_1$ **faça**
    se $-b_{i1} \leq -b_{k1}$ **então**
      $P = P + 1$
      se $Cval < b_{1i} + P$ **então**
        $Cval = b_{1i} + P$
      fim se
    fim se
    $c[i] = Cval$
  fim para
  $H = -\infty$
  para $i = v_1, v_2, \ldots, v_t$ **faça**
    se $-b_{i1} \leq' b_{k1}$ **então**
      $H = \max\{H, b_{1i} + P\}$
      $P = P - 1$
    senão
      se $b_{1i} + P > -b_{k1}$ **então**
        $v[i] = c[i]$
      fim se
      se $H > b_{k1}$ **então**
        $v[i] = Cval$
      fim se
    fim se
  fim para
fim para
$b_{1i} = \max\{b_{1i}, v[i]\}$, $\forall i \in C$

As proposições 6 e 7 se concentram em verificar se um elemento j **deve** suceder (ou preceder) a clique $C' \subseteq C$. Como complemento natural desta abordagem podemos tentar determinar se um elemento **pode** suceder (ou preceder) C'. Isto leva às seguintes regras ([3]):

Proposição 8 *: Sejam C uma clique em G_E, $C' \subseteq C$ e $j \in C \setminus C'$. Se*

$$\max_{i \in C'}\{-b_{i1}\} - b_{1j} < |C'| \tag{5.8}$$

então $h_j^* > h_i^*$ para algum $i \in C'$, implicando em

$$b_{1j} = \max\{b_{1j}, \min_{i \in C'}\{b_{1i}\} + 1\}, \tag{5.9}$$

Analogamente, se

$$-b_{j1} - \min_{i \in C'} b_{1i} < |C'|, \tag{5.10}$$

então $h_j^* < h_i^*$, para algum $i \in C$, implicando

$$b_{j1} = \max\{b_{j1}, \min_{i \in C'}\{b_{i1}\} + 1\}. \tag{5.11}$$

Prova: Como C' é uma clique em G_E, temos por (5.2) que $\min_{i\in C'} h_i^* \leq \max_{i\in C'} h_i^* - |C'| + 1 \leq \max_{i\in C'}\{-b_{i1}+1\} - |C'| + 1 < b_{1j} + 2 \leq h_j^* + 1$.

Logo, $\min_{i\in C'} h_i^* \leq h_j^*$. Como $j \notin C'$ e $C' \cup \{j\}$ é clique,a desigualdade anterior vale estritamente, mostrando a primeira parte. De forma análoga, mostra-se a segunda. □

O problema de realizar todos os ajustes correspondentes às regras (5.8) e (5.9) é chamado de *not-first*, já que atualiza a menor altura possível de todo elemento j que não pode ter a menor altura dentre os elementos do conjunto $C' \cup \{j\}$. De forma similar, o problema correspondente às regras (5.10) e (5.11) é chamado de *not-last*, consistindo em atualizar a altura máxima de todo elemento j que não pode ter a maior altura dentre os elementos do conjunto $C' \cup \{j\}$. Como não há um algoritmo de baixo custo polinomial para a resolução dos dois problemas, as regras são aplicadas de maneira incompleta (somente parte do total de ajustes possíveis são feitos) utilizando um algoritmo de complexidade $\mathcal{O}(n^2)$ encontrado em [3].

Para descrever este algoritmo, seja C uma clique em G_E e considere que os elementos $1, 2, \ldots, t \in C$ estão em ordem não decrescente em relação a $-b_{l1}$, $l \in \{1, 2, \ldots, t\}$. Dados i e k, $C(ik)$ é o conjunto de índices $m \in \{1, \ldots, k\}$ tal que $b_{1i} \leq b_{1m}$ e $C(jik)$ denota $C(ik) - \{j\}$. Logo, se $j \notin C(ik)$ temos $C(jik) = C(ik)$. Defina $S_{ik} = |C(ik)|$ se $-b_{i1} \leq -b_{k1}$ e $S_{ik} = -\infty$ caso contrário. Por último, tome $\delta_{ik} = \min_{-b_{l1}\leq -b_{k1}}(-b_{l1} + 1 - S_{i,l})$.

O Algoritmo 8 é uma heurística para o problema *not-first*, que possui uma versão dual para *not-last*, baseado nos seguintes resultados obtidos em [3] para o problema de escalonamento de recursos com restrições de precedência e restrições de recursos e aqui reescrito para o EIP.

Lema 9 *: Para um dado i, os valores $\delta_{i,1}, \ldots, \delta_{i,t}$ podem ser computados em* $\mathrm{O}(n)$.

Lema 10 *: Se a regra* not-first *aplicada a um elemento j e ao conjunto C permitir a atualização de b_{1j} para $b_{1i} + 1$ então existe um elemento k, onde $-b_{i1} \leq -b_{k1}$, tal que a mesma regra aplicada ao elemento j e ao conjunto $c(jik)$ permite a atualização de b_{1j} para $b_{1i} + 1$.*

Lema 11 *: Sejam dois elementos i, j tais que $b_{1j} < b_{1i}$. Neste caso a regra* not-first *permite a atualização $b_{1j} = b_{1i} + 1$ se, e somente se, $b_{1j} + 1 > \delta_{it}$.*

Lema 12 *: Sejam dois elementos i, j tais que $b_{1j} \geq b_{1i}$. Neste caso a regra* not-first *permite a atualização $b_{1j} = b_{1i} + 1$ se, e somente se, $b_{1j} + 1 > \delta_{ij-1}$ ou $b_{1j} > \delta_{it}$.*

Algoritmo 8 *notFirst*

Seja $C = \{1, 2, \ldots, t\}$ uma clique em G_E com $-b_{i1} \geq -b_{i+1,1}$ para $i \in C \setminus \{t\}$
para $i = 1, 2, \ldots, t$ **faça**
  Calcule $\delta_{i1}, \ldots, \delta_{it}$
  para $j = 1, 2, \ldots, t$ **faça**
    se $b_{1j} < b_{1i}$ **então**
      se $b_{1j} + 1 > \delta_{it}$ **então**
        $b_{1j} = \max\{b_{1j}, b_{1i} + 1\}$
      fim se
    senão
      se $b_{1j} + 1 > \delta_{ij-1}$ ou $b_{1j} > \delta_{it}$ **então**
        $b_{1j} = \max\{b_{1j}, b_{1i} + 1\}$
      fim se
    fim se
  fim para
fim para

Alargamento por cliques Cliques em G_E também podem ser usadas para tentar aumentar o valor de b_{ij}, $\forall (i,j) \in V \times V$, especialmente o valor do limite inferior b_{1n} através da seguinte propriedade que derivamos:

Proposição 13 *: Dada uma clique C, podemos atualizar*

$$b_{ij} = \max\{b_{ij}, \min_{k \in C} b_{ik} + |C| - 1 + \min_{l \in C} b_{lj}\},\ \forall i,j \in V \tag{5.12}$$

Prova: Sejam $m, M \in C$ tal que $h^*_m = \min_{k \in C} h^*_k$ e $h^*_M = \max_{k \in C} h^*_k$. Dados $i, j \in V$, a condição (5.1) e o fato de que $h^*_M - h^*_m \geq |C| - 1$, levam a:

$$\begin{aligned} h^*_j - h^*_i &= (h^*_j - h^*_M) + (h^*_M - h^*_m) + (h^*_m - h^*_i) \\ &\geq b_{Mj} + |C| - 1 + b_{im} \\ &\geq \min_{k \in C} b_{kj} + |C| - 1 + \min_{k \in C} b_{ik}, \end{aligned}$$

mostrando o resultado desejado.□

Vale mencionar que a regra (5.12) generaliza uma restrição presente em [17] que, dada uma clique C, permite apenas a atualização de b_{1n}, e não pode ser propagada.

Com base na Proposição 13 desenvolvemos uma heurística que procura atualizar a matriz B (ver Algoritmo 9). Por iterar duas vezes sobre o conjunto de elementos ($\mathcal{O}(n^2)$) e ainda iterar sobre os elementos da clique escolhida ($\mathcal{O}(|C|)$) a complexidade total do procedimento é $\mathcal{O}(n^3)$.

Algoritmo 9 *CliqueEnlarge*

para $i = 1$ até $n - 1$ **faça**
  Ordene $C = \{v_1, v_2, \ldots, v_{|C|}\}$ tal que $b_{iv_1} \geq b_{iv_2} \geq \cdots \geq b_{iv_{|C|}}$
  para $j = 2$ até n **faça**
    $b_{lj} = b_{ik} = \infty$
    para $t = 1$ até $|C|$ **faça**
      $b_{lj} = \min\{b_{lj}, b_{v_t j}\}$, $b_{ik} = \min\{b_{ik}, b_{iv_t}\}$
      $b_{ij} = \max\{b_{ij}, b_{ik} + t - 1 + b_{lj}\}$
    fim para
  fim para
fim para
para $j = 2$ até n **faça**
  Ordene $C = \{v_1, v_2, \ldots, v_{|C|}\}$ tal que $b_{v_1 j} \geq b_{v_2 j} \geq \cdots \geq b_{v_{|C|} j}$
  para $i = 1$ até $n - 1$ **faça**
    $b_{ik} = b_{lj} = \infty$
    para $t = 1$ até $|C|$ **faça**
      $b_{ik} = \min\{b_{ik}, b_{iv_t}\}$, $b_{lj} = \min\{b_{lj}, b_{v_t j}\}$
      $b_{ij} = \max\{b_{ij}, b_{ik} + t - 1 + b_{lj}\}$
    fim para
  fim para
fim para

5.1.3 Propagação de restrições

Propagação de restrições é um processo que consiste na dedução de novas restrições a partir de outras já existentes. Em problemas de escalonamento com restrições de recursos, dados um conjunto de atividades e um conjunto de restrições (temporais, de recursos, específicos para o problema ou limites para diversos critérios de otimização), a propagação de restrições determina condições que um escalonamento das atividades deve possuir como propriedade para satisfazer a todas as demais restrições. Estas propriedades

podem ser utilizadas para encontrar uma solução ótima ou como guia para procedimentos heurísticos de busca em direção a "boas" soluções.

No caso do EIP, as restrições na Subseção 5.1.2 procuram limitar inferior e superiormente as diferenças $h_j^* - h_i^*$, para todos $i, j \in V$, e, particularmente, a altura de cada elemento do poset em uma solução ótima. Essas restrições podem ser combinadas e propagadas de várias formas.

Nossa implementação é baseada no método proposto em [17]. As regras são aplicadas iterativamente até que não haja mais ajustes a serem feitos nos domínios das alturas dos elementos. A estrutura geral do processo é apresentada no Algoritmo 10. Note que, na prática, este conjunto de regras é aplicado somente algumas vezes (em geral duas vezes, de acordo com nossos testes).

Algoritmo 10 *ConsProg*

```
1: Crie matriz B com limite superior T;
2: FloydWarshall(B);
3: repita
4:    ImmediateSelection(B);
5:    EdgeFinding();
6:    se B foi modificada então
7:       FloydWarshall(B);
8:    fim se
9: até não haver mais modificações em B
```

Nesse algoritmo o procedimento de *EdgeFinding* (linha 5) combina as regras baseadas em cliques, conforme descrito pelo Algoritmo 11.

Algoritmo 11 *EdgeFinding*

```
1: C = generateCoverCliques
2: para C ∈ C faça
3:    edgeFindingPrimal(C);
4:    edgeFindingDual(C);
5:    notFirst(C)
6:    notLast(C)
7: fim para
8: para C ∈ C faça
9:    CliqueEnlarge(C);
10: fim para
```

A computação de cliques (linha 1) é feita através do Algoritmo 12, que implementa uma heurística desenvolvida por nós, baseada naquela proposta por [5]. Em nosso caso enumeramos cliques maximais de G_E, de maneira a cobrir o mais rápido possível o conjunto E. O algoritmo trabalha sobre os elementos de E, iterando sobre cada um deles, e procura, de maneira construtiva, a maior clique que pode ser obtida a partir daquele par de E. Pares já utilizados em uma clique anterior não serão mais considerados para iniciar a busca, mas não são proibidos de aparecerem em outras cliques. Na nossa implementação, iteramos sobre vértices que incidem sobre pares em E na ordem decrescente do grau dos vértices em relação a E ($\delta_E(v)$), ou seja, o vértice v com maior grau terá todos os seus pares $\{(v,i)\text{ou}(i,v) \mid i \in V\} \in E$ visitados primeiro.

Devido à dupla iteração realizada no conjunto de vértices ($\mathcal{O}(n^2)$) e à verificação da validade de cada elemento para se tornar parte da clique em construção ($\mathcal{O}(|C|)$) a complexidade total é de $\mathcal{O}(|E|n^3)$. Embora de alta complexidade, na prática este procedimento pode ser mais rápido devido aos pares $(i,j) \in E$ que são incluídos em alguma clique antes de serem escolhidos como ponto de início para a procura por outras cliques.

Algoritmo 12 *generateCoverCliques*

1: $E' = E$
2: Ordene V em ordem decrescente no número de arestas de E incidentes em $v \in V$ ($\delta_{E'}(v_2) > \delta_{E'}(v_3) > \ldots > \delta_{E'}(v_{n-1})$);
3: **para** $i = 2$ até $i = n - 1$ **faça**
4: **enquanto** $\delta_{E'}(v_i) > 0$ **faça**
5: escolha um par $(v_i, k) \in E'$;
6: crie nova clique $C = \{v_i, k\}$;
7: Remova (v_i, k) de E';
8: **para** $j = 2$ até $j = n - 1$ **faça**
9: **se** $C \cup v_j$ é clique em G_E **então**
10: $C = C \cup \{v_j\}$
11: Remova (v_j, l) de E', $\forall\, l \in C$;
12: **fim se**
13: **fim para**
14: **fim enquanto**
15: **fim para**

5.1.4 *Shaving*

Com o objetivo de melhorar o processo de propagação de restrições, aplicamos uma adaptação da técnica de *shaving* implementada em [17]. O shaving consiste em adicionar uma restrição c temporariamente e executar o processo de propagação de restrições. Na ocorrência de uma inviabilidade então temos que a restrição oposta $\neg c$ é válida.

No nosso caso, estamos interessados na geração de restrições de precedência. Para cada par $(i, j) \in E$ testamos a validade de duas restrições: $i \prec' j$ e $j \prec' i$. Cada restrição é propagada separadamente através das 4 regras citadas anteriormente, gerando duas matrizes $B^{i \prec' j}$ e $B^{j \prec' i}$ relativas a uma mesma instância para o EIP. Se a matriz $B^{i \prec' j}$ é inconsistente, ou seja, $b_{ij}^{i \prec' j} > -b_{ji}^{i \prec' j}$ para algum par $(i, j) \in V \times V$, então a restrição de precedência $i \prec' j$ é refutada e deduções globais em B, a matriz original, podem ser feitas com $j \prec' i$. Do mesmo modo, caso $B^{j \prec' i}$ seja inconsistente, procedemos de maneira análoga.

No caso de nenhuma restrição ser refutada pelo procedimento, procedemos à atualização da matriz original B, componente a componente, da seguinte maneira:

$$B = \min\{B^{i \prec' j}, B^{j \prec' i}\}$$

Está claro que este processo demanda muito tempo computacional, já que há uma chamada do processo de propagação de restrições para cada par de elementos em E. Uma maneira de tentar reduzir o custo deste processo é restringir sua aplicação a somente uma porcentagem do número total de pares em E, como foi feito em nossa implementação. Uma vez que este conjunto $E' \subset E$ tenha sido criado, executamos o processo de propagação de restrições como descrito acima para E', de forma iterativa, até que novas modificações não sejam possíveis.

Para a escolha do conjunto de pares a sofrerem o processo de *shaving* utilizamos o seguinte critério. A quantidade máxima de pares a serem escolhidos é o menor valor entre uma constante numérica e $\frac{1}{4}|E|$. Nossos experimentos utilizaram como constante numérica o valor de 50. Uma maior quantidade de pares poderia acarretar em um procedimento mais demorado, embora a fixação de pares induza, por transitividade, a fixação de outros. Para ser escolhido, um elemento $(i, j) \in E$ deve atender à seguinte condição

$$|(b_{ij} + b_{ji})/2\text{UB}| \leq 0.3 \text{ ou } |(b_{ij} + b_{ji})/2\text{UB}| \geq 0.7 \tag{5.13}$$

onde UB é um limite superior para o problema. Caso os pares satisfazendo 5.13 forem maiores que o máximo permitido, faz-se uma escolha aleatória.

A condição acima procura escolher pares (i, j) dotados de grande diferença na distância entre os elementos que os compõem, na tentativa de que o procedimento de shaving seja capaz de fixar mais facilmente estes pares do que aqueles que não foram escolhidos.

5.1.5 Procedimento destrutivo

A segunda técnica que acoplamos à programação por restrições chama-se *procedimento destrutivo* ([24]), que consiste em aplicar uma espécie de *shaving*, onde a restrição c limita superiormente o valor ótimo do problema.

De forma geral, esta técnica funciona da seguinte maneira. Para um problema de otimização qualquer P, um valor L é utilizado para restringir o conjunto viável a soluções de valor máximo L. Se este conjunto restrito é vazio, o valor de qualquer solução para P deve exceder o valor L. Portanto $L+1$ é um novo limite inferior para P (dado que os valores possíveis para qualquer solução do problema são inteiros). Para descobrir se uma solução viável que não exceda o valor L existe ou não, dois passos são relizados.

Considere $P(L)$ como o problema restrito obtido a partir de P limitando o conjunto de soluções àquelas de valor máximo L, isto é, $P(L)$ é obtido adicionando a P a seguinte restrição:

$$f(x) \leq L$$

onde f é a função objetivo de P. O *primeiro* passo consiste em utilizar técnicas de redução para encontrar restrições adicionais ou modificar os parâmetros do problema para fortalecer os dados de que dispomos. Neste passo é possível que uma contradição ocorra e, portanto, a viabilidade de $P(L)$ é refutada. Caso contrário, no *segundo* passo, o problema $P(L)$ fortalecido é relaxado e resolvido. Se a relaxação não possui solução viável, $P(L)$ é inviável e, logo, L não é um limite válido para qualquer solução para P. Caso nenhum destes passos consiga mostrar a inviabilidade de $P(L)$, o processo pára. Caso contrário, L é adicionado de 1 e os dois passos anteriores são repetidos.

Nosso esquema destrutivo foi baseado no proposto em [17]. Começando de um limite superior UB para o problema, considere as variáveis $LB_{curr} = 0$ e $UB_{curr} = UB$. Nós realizamos uma busca dicotômica pelo maior valor T entre LB_{curr} e UB_{curr}, tal que a aplicação do processo de propagação de restrições, juntamente com a resolução de modelos relaxados para o EIP (que veremos mais adiante), prove que não há solução viável de valor menor ou igual a T. Neste caso o limite inferior destrutivo é $T+1$. O seguinte algoritmo caracteriza o método e como a combinação com a programação por restrições e relaxação linear foi feita:

Algoritmo 13 *DestructiveBound*

1: $UB_{curr} = UB$; $LB_{curr} = 0$;
2: $T = (UB_{curr} + LB_{curr})/2$
3: **enquanto** $LB_{curr} < UB_{curr}$ ou ($LB_{curr} \leq UB_{curr}$ e $UB_{curr} < UB$) **faça**
4: $ConsProg(T)$;
5: **se** $ConsProg$ é viável **então**
6: $LB_{curr} = \max\{LB_{curr}, b_{1n} + 1\}$
7: Resolva uma relaxação de EIP
8: **se** a relaxação é inviável **então**
9: $LB_{curr} = T + 1$
10: **senão**
11: $UB_{curr} = T - 1$
12: **fim se**
13: **senão**
14: $LB_{curr} = T + 1$
15: **fim se**
16: $T = (UB_{curr} + LB_{curr})/2$
17: **fim enquanto**

Neste processo, inviabilidade pode ser detectada em dois momentos. Primeiro, na fase de propagação de restrições (linha 5) quando durante a execução do algoritmo de *FloydWarshall* ocorre $b_{ij} > -b_{ji}$ para algum $i, j \in V$. Segundo, quando o conjunto de soluções viáveis do modelo relaxado é vazio (linha 8). Esses modelos relaxados serão apresentados no próximo capítulo. Vale adiantar que os modelos relaxados

são fortalecidos com base nos dados obtidos ao final da programação por restrições, que age na redução do domínio do problema.

6

Limite Inferior Programação Linear Inteira

A seguir, apresentamos duas formulações de programação linear inteira para o EIP. Ambas as formulações exploram o conceito de matriz de distância de alturas, incorporando elementos desta estrutura na sua descrição. Só para recordar, a matriz SSD $B = (b_{ij})_{V \times V}$ é tal que

$$h_j^* - h_i^* \geq b_{ij}, \text{ para todo } (i,j) \in V \times V,$$

onde h_i^* é a altura do elemento i em uma solução ótima $\prec^*$ de EIP. Além disso podemos assumir, sem perda de generalidade, que $b_{ij} \geq 1$ para qualquer $i, j \in V$ tal que $i \vdash j$.

A primeira formulação é baseada na formulação mais frequentemente encontrada na literatura para o problema de escalonamento de tarefas com restrições de recursos ([28, 16, 27]), que usa variáveis booleanas indexadas na altura do poset. Descreveremos as adaptações feitas na formulação original ao aplicá-la ao EIP, juntamente com alguns resultados acerca da sua corretude.

A segunda é baseada na formulação encontrada em [13] para o *multiprocessor scheduling problem* (MSP). Ao contrário da primeira, esta utiliza variáveis associadas às relações de precedência para cada elemento $i \in V$. Também apresentaremos resultados que comprovam a corretude desta segunda formulação.

Ambas as formulações se apóiam na seguinte idéia. Seja $P' = (V, \prec')$, onde $\prec' \supseteq \prec$ é uma extensão de $\prec$. Temos que $h(P'_{i^-})$ é o tamanho da maior cadeia em P' que tem i como elemento maximal. Alternativamente, do ponto de vista do grafo representativo de P', $h(P'_{i^-}) - 1$ é o tamanho do maior caminho entre 1 e i. Claramente, esse valor pode ser calculado recursivamente como:

$$h(P'_{1^-}) = 1 \tag{6.1}$$

$$h(P'_{j^-}) = \max_{i \vdash' j} \{h(P'_{i^-}) + 1\}, j \in V \tag{6.2}$$

onde $\vdash'$ é a redução transitiva de $\prec'$. Particularmente, a altura de P' será

$$h(P'_{n^-}) = \max_{i \vdash' n} \{h(P'_{i^-})\} + 1. \tag{6.3}$$

Na Seção 6.3 propomos algumas idéias para fortalecer as restrições das formulações, além de algumas desigualdades válidas baseadas nas estruturas que obtivemos do processo de programação por restrições do Capítulo 5.

6.1 Variáveis indexadas pela altura

A formulação descrita a seguir é baseada em variáveis booleanas y_{it} para cada elemento $i \in V$ e para cada possível altura sua $t \in I_i = \{b_{1i} + 1, \ldots, -b_{i1} + 1\}$, onde $y_{it} = 1$ se, e somente se, $h(P'_{i^-})$ é t, sendo P' uma solução viável.

Portanto, para cada $i \in V$, uma única variável y_{it} receberá valor 1, temos que $h(\prec'_{i^-}) = \sum_{t=b_{1i}+1}^{-b_{i1}+1} t y_{it}$. Assim, o EIP pode ser formulado da seguinte maneira:

$$\theta(P,E): \text{ minimizar } \sum_{t\in I_n} t y_{nt} \tag{6.4}$$

$$\text{s.a.} \quad \sum_{t\in I_i} y_{it} = 1 \qquad \forall i \in V, \tag{6.5}$$

$$\sum_{t\in I_j} t y_{jt} - \sum_{t\in I_i} t y_{it} \geq b_{ij} \qquad \forall i \vdash j, \tag{6.6}$$

$$y_{it} + y_{jt} \leq 1 \qquad \forall (i,j) \in E, \forall t \in I_{ij} \tag{6.7}$$

$$y_{it} \in \{0,1\} \qquad \forall i \in V, \forall t \in I_i. \tag{6.8}$$

onde $I_{ij} = I_i \cap I_j$. Na formulação, a função objetivo procura minimizar o valor de t tal que $y_{nt} = 1$, ou seja, minimizar a altura da extensão. Consequentemente, minimiza o valor do somatório $\sum_{t\in I_i} t y_{it}, \forall i \in V$, pelas restrições do grupo (6.6). Assim a função objetivo (6.4), juntamente com as restrições (6.5) e (6.6) asseguram a condição (6.2) para os pares (i,j) relacionados em $\vdash$. Já para os elementos (i,j) de E, orientados conforme a extensão, se faz necessário o uso da restrição (6.7), que vai garantir que i e j não possuam a mesma altura. Juntamente com a restrição (6.6), garantimos que a condição (6.2) seja satisfeita para estes elementos.

Agora mostramos a corretude desta formulação. Um poset $P' = (V, \prec')$ é obtido a partir de uma solução viável qualquer Y para $\theta(P,E)$, onde $Y = \{y_{it} \mid i \in V, t \in I_i\}$, da seguinte maneira:

i. Faça $\overrightarrow{E} = \emptyset$;

ii. para cada $(i,j) \in E$ faça $\overrightarrow{E} = \overrightarrow{E} \cup \{(i,j)\}$, se $\sum t y_{it} < \sum t y_{jt}$; ou $\overrightarrow{E} = \overrightarrow{E} \cup \{(j,i)\}$, caso contrário;

iii. $\prec'$ é o fecho transitivo da relação $\prec \cup \overrightarrow{E}$.

Teorema 14 *: Dada uma solução viável Y de $\theta(P,E)$, a relação $\prec'$ definida por (i)-(iii) é uma extensão de $\prec$ induzida por E.*

Prova: Seja Y uma solução viável para $\theta(P,E)$ e $\prec'$ obtida por (i)-(iii). Temos que $\prec'$ orienta todos os pares em E. Devemos então mostrar que $\prec'$ é uma ordem parcial. Claramente, $\prec'$ é irreflexiva e transitiva. Para mostrarmos que $\prec'$ é anti-simétrica é suficiente mostrar que o grafo direcionado $G_Y = (V, \prec')$ não contém ciclos direcionados.

Suponha, por absurdo, que G_Y contenha um ciclo direcionado $C = \{v_1,$
$v_2, \ldots, v_k, v_{k+1}\}$, onde $v_{k+1} = v_1$. Sejam $i \in \{1,2,\ldots,k\}$ e $T = \max_{i=1,\ldots,k}\{-b_{i1}+1\}$. Se $v_i \vdash v_{i+1}$, então as restrições (6.6) e $b_{i(i+1)}$ garantem que

$$\sum_{t=1}^{T} t y_{v_{i+1}t} > \sum_{t=1}^{T} t y_{v_i t}. \tag{6.9}$$

onde tomamos $y_{v_i t} = 0, \forall t \in T \setminus I_{v_i}$. A mesma relação é garantida por (ii), para $(v_i, v_{i+1}) \in \overrightarrow{E}$. Em qualquer outro caso, como (v_i, v_{i+1}) está no fecho de $\prec'$, a desigualdade (6.9) segue por transitividade.

Escrevendo cada desigualdade relacionada aos elementos de C teremos:

$$\begin{aligned}
\sum_{t=1}^{T} t(y_{v_2 t} - y_{v_1 t}) &\geq 1 \\
\sum_{t=1}^{T} t(y_{v_3 t} - y_{v_2 t}) &\geq 1 \\
&\vdots \\
\sum_{t=1}^{T} t(y_{v_k t} - y_{v_{k-1} t}) &\geq 1 \\
\sum_{t=1}^{T} t(y_{v_{k+1} t} - y_{v_k t}) &\geq 1
\end{aligned}$$

Somando os membros de cada desigualdade teremos:

$$\sum_{t=1}^{T} t \sum_{i=1}^{k} y_{v_i t} - \sum_{t=1}^{T} t \sum_{i=1}^{k} y_{v_i t} \geq k, \quad 0 \geq k,$$

um absurdo. Portanto, $\prec'$ é realmente uma extensão de $\prec$.□

6.2 Variáveis baseadas nas precedências

Para a segunda formulação, definimos as variáveis $x_i, \forall i \in V$, para indicar a altura de i, e a variável $w_{ij} \in \{0,1\}$, $\forall (i,j) \in E$, que assume valor 1 se (i,j) for orientado de i para j e valor 0 se for orientado de j para i. Note que, para cada par em E, uma única variável é definida.

Então, outra formulação de programação linear inteira para o EIP é a seguinte:

$$\begin{aligned}
\Pi(P,E): \text{ minimizar } & x_n && && (6.10)\\
\text{s.a. } & x_j - x_i \geq b_{ij} && \forall i \vdash j, && (6.11)\\
& x_j - x_i \geq (b_{ij} - 1)(1 - w_{ij}) + 1 && \forall (i,j) \in E, && (6.12)\\
& x_i - x_j \geq (b_{ji} - 1)w_{ij} + 1 && \forall (i,j) \in E, && (6.13)\\
& b_{1i} + 1 \leq x_i \leq 1 - b_{i1} && \forall i \in V && (6.14)\\
& w_{ij} \in \{0,1\} && \forall (i,j) \in E, && (6.15)
\end{aligned}$$

Na formulação, a função objetivo (6.10) procura minimizar os valores de x_i, conforme evidenciado pelas restrições do grupo (6.11), para $j = n$. Consequentemente, tais restrições asseguram a condição (6.3). Pelo mesmo motivo, apenas a desigualdade $x_j \geq \max_{i \vdash' j}\{x_i\} + 1$ precisa ser imposta para se obter a condição (6.2). Isto é garantido pelos grupos de restrições (6.11) - (6.13): enquanto o grupo (6.11) se refere aos pares da ordem original $\prec$, os grupos (6.12)- (6.13) englobam os pares criados em $\prec'$ devido às variáveis w_{ij}. Note que, das duas restrições em (6.12)-(6.13) correspondentes a cada par (i,j), uma será efetiva e outra redundante, conforme a ordem de precedência definida para (i,j).

Nesta formulação um poset $\prec'$ é obtido a partir de uma solução viável qualquer (X,W) para $\Pi(P,E)$, onde $W = \{w_{ij} \mid (i,j) \in E\}$ e $X = \{x_i \mid i \in V\}$, da seguinte maneira:

i. denote por $\overrightarrow{E}$ o conjunto de pares de E, ordenados segundo a orientação de cada par em E induzida por W;

ii. $\prec'$ é o fecho transitivo da relação $\prec \cup \overrightarrow{E}$.

Teorema 15 *: Dada uma solução viável (X,W) de $\Pi(P,E)$, a relação $\prec'$ definida por (i)-(ii) é uma extensão de $\prec$.*

Prova: A prova é similar à apresentada sobre a corretude da formulação anterior, utilizando as restrições (6.12),(6.13) ou (6.14) relacionadas à solução (X, W) e a um hipotético ciclo $C = \{v_1, v_2, \ldots, v_k, v_{k+1}\}$, onde $v_{k+1} = v_1$. Deste modo teremos

$$x_{v_{i+1}} \geq x_{v_i} + 1, \text{ para } i \in \{1, 2, \ldots, k\}. \tag{6.16}$$

Lembre que $b_{ij} \geq 1, \forall i \vdash j$.

Novamente a soma destas desigualdades levará ao absurdo $0 \geq k$. Portanto, $\prec'$ é realmente uma extensão de $\prec$.□

6.3 Fortalecimento das relaxações

Como foi visto anteriormente, o procedimento de propagação de restrições gera novos valores para os intervalos $[b_{ij}, -b_{ji}]$, para todo $i, j \in V$, pesquisando, por exemplo, subestruturas do poset como cliques de disjunções. Estes elementos podem ser usados para fortalecer os modelos relaxados de programação inteira apresentados nas seções anteriores (Seções 6.1 e 6.2).

A matriz de distâncias pode ter seus valores utilizados para fixar variáveis (limitando o domínio das variáveis de ambos os modelos, como visto anteriormente) e fortalecer as desigualdades lineares, enquanto que as cliques obtidas podem ser utilizadas para inferir novas desigualdades válidas sobre disjunções. Nesta seção nós explicamos como as estruturas da programação por restrições podem fortalecer cada formulação para o IEP.

Fixação de variáveis Podemos fixar variáveis em $\Pi(P, E)$ antes de sua resolução com a ajuda de B. Para $(i, j) \in E$, fazemos $w_{ij} = 1$ se $b_{ij} > 0$ ou $w_{ij} = 0$ se $b_{ji} > 0$. Como consequência, as duas desigualdades do grupo de (6.12)-(6.13) em $\Pi(P, E)$ são substituídas por

$$x_j \geq x_i + 1 \text{ ou } x_i \geq x_j + 1 \tag{6.17}$$

respectivamente.

Já em $\theta(P, E)$, como dito anterioremente, as variáveis y_{it} relacionadas a um elemento $i \in V$, indexadas no intervalo $I_i = [b_{1i} + 1, -b_{i1} + 1]$, podem vir a ter o seu domínio reduzido ainda mais pela diminuição do mesmo intervalo I_i.

Fortalecimento de restrições lineares Do mesmo modo que o domínio das variáveis nas duas formulações tinha limite definido pela matriz B, podemos fortalecer restrições lineares já existentes e criar outras para tentar eliminar soluções que não nos interessam. As novas restrições expressam a distância mínima entre as alturas de dois elementos quaisquer, não diretamente definida pela ordem original. É importante lembrar que as entradas de B podem fornecer valores mais próximos da altura final de cada elemento em uma solução ótima, devido à aplicação da técnica de programação por restrições.

Assim, na formulação $\Pi(P, E)$, podemos substituir o grupo de restrições (6.11) por :

$$x_j - x_i \geq b_{ij}, \text{ para todo } (i, j) \text{ tal que } b_{ij} > b_{1j} + b_{i1}$$

Note que, sendo $b_{ij} = b_{i1} + b_{1j}$, estas restrições são inicialmente satisfeitas tendo os limites inferiores impostos em (6.14). Já em $\theta(P, E)$, de forma análoga, substituímos (6.6) por

$$\sum_{t=b_{1j}+1}^{-b_{j1}+1} t y_{jt} - \sum_{t=b_{1i}+1}^{-b_{i1}+1} t y_{it} \quad \geq \quad b_{ij}, \text{ para todo } (i, j) \text{ tal que } b_{ij} > b_{1j} + b_{i1}$$

É fácil perceber que os dois grupos de restrições acima dominam as restrições ligadas às relações de precedência, além de fortalecer os intervalos entre elementos que antes não eram discriminados na formulação.

Desigualdades válidas via cliques As inequações apresentadas aqui se baseiam em cliques de disjunções C e indicam que para qualquer valor $t > 0$, pode haver no máximo um elemento $i \in C$ com

$h_i^* = t$. Neste sentido, dada uma clique C de G_E e um valor t, denotamos por C_t a subclique formada pelos elementos de C que podem ter altura t, ou seja,

$$C_t = \{i \in C \,:\, t \in I_i\}.$$

Definimos, também, $h_{min}(C) = \min\{b_{1i} + 1 \,:\, i \in C\}$ e $h_{max}(C) = \max\{-b_{i1} + 1 \,:\, i \in C\}$. Além disso, temos as seguintes desigualdades válidas para $\theta(P, E)$ que dominam (6.7):

$$\sum_{i \in C_t} y_{it} \leq 1, \forall C \text{ clique em } G_E, \forall t \in [h_{min}(C), h_{max}(C)] \tag{6.18}$$

Notamos ainda que, para $t \in [h_{max}(C) - |C| + 1, h_{min}(C) + |C| - 1]$, uma igualdade pode ser considerada em (6.18).

Para implementação destas inequações em $\theta(P, E)$ utilizamos cliques obtidas com uma heurística diferente daquela utilizada na programação por restrições. Este novo procedimento determina uma partição de cliques disjuntas que cubram o conjunto E. Nele utilizamos a mesma regra de prioridade sobre os elementos do poset para iniciar a busca por cliques e procuramos pela maior clique possível que não utilize pares $(i, j) \in E$ já utilizados na composição de outras cliques. Ao final deste procedimento teremos uma partição $\mathcal{C}$ destas cliques que cobrem todas os pares de E. Para cada clique $C \in \mathcal{C}$, geramos as inequações correspondentes de (6.18).

Relaxações lineares Para obtenção de limites inferiores usamos as relaxações lineares das formulações $\Pi(P, E)$ e $\theta(P, E)$, fortalecidas pelas restrições vistas acima. Especificamente, trabalhamos com as relaxações

$$\begin{aligned}
\overline{\theta}(P,E): \text{ minimizar } & \sum_{t \in I_n} t y_{nt} & & \\
\text{s.a.} \quad & \sum_{t \in I_i} y_{it} = 1 & & \forall i \in V, \\
& \sum_{t \in I_j} t y_{jt} - \sum_{t \in I_i} t y_{it} \geq b_{ij} & & \forall (i,j) \in V \times V \,:\, b_{ij} > b_{i1} + b_{1j}, \\
& \sum_{i \in C_t} y_{it} \leq 1 & & \forall C \in \mathcal{C}, \forall t \in [h_{min}(C), h_{max}(C)], \\
& y_{it} \geq 0 & & \forall i \in V, \forall t \in I_i,
\end{aligned}$$

onde $\mathcal{C}$ é uma coleção de cliques disjuntas cobrindo E.

$$\begin{aligned}
\overline{\Pi}(P,E): \text{ minimizar } & x_n & & , \\
\text{s.a.} \quad & x_j - x_i \geq b_{ij} & & \forall (i,j) \in V \times V \,:\, b_{ij} > b_{i1} + b_{1j}, \\
& x_j - x_i \geq (b_{ij} - 1)(1 - w_{ij}) + 1 & & \forall (i,j) \in \overline{E}, \\
& x_i - x_j \geq (b_{ji} - 1) w_{ij} + 1 & & \forall (i,j) \in \overline{E}, \\
& b_{1i} + 1 \leq x_i \leq 1 - b_{i1} & & \forall i \in V, \\
& 0 \leq w_{ij} \leq 1 & & \forall (i,j) \in \overline{E},
\end{aligned}$$

onde $\overline{E} = \{(i, j) \in E \,:\, b_{ij} < 0, b_{ji} < 0\}$

7

Resultados Computacionais

Para avaliar a qualidade dos nossos limites, implementamos os vários algoritmos propostos e os aplicamos a vários problemas de teste criados por nós, devido à inexistência de instâncias de *benchmark* específicas para o nosso problema. Os nossos experimentos foram realizados em um computador IBM PC Pentium IV 2.8 GHz, com 1Gb de memória e executando Linux *Fedora Core* 4. Nossos procedimentos foram todos implementados em Java 5.0 para permitir a utilização da biblioteca *pargoworks*.

A biblioteca *pargoworks* está sendo desenvolvida por membros do grupo de pesquisa ParGO (*Paralelismo, Grafos e Otimização*), do Departamento de Computação da Universidade Federal do Ceará (UFC), com a finalidade de manipular estruturas matemáticas na área de teoria dos grafos e afins, e apoiar a resolução de modelos de programação matemática. Isto é obtido oferecendo uma mesma interface para trabalhar com quaisquer *solvers* que desejemos (a interface é o ponto de comunicação entre nós, desenvolvedores, e o *solver* de nossa escolha, e deve ser implementada antes de ser utilizada) e *frameworks* para métodos de programação inteira específicos, que incluem métodos de planos de corte e *branch-and-bound*. A utilização desta biblioteca também faz parte dos esforços do grupo em manter os projetos desenvolvidos sobre uma mesma base e facilitar a reutilização das técnicas desenvolvidas em trabalhos anteriores.

Devido à inexistência de instâncias de *benchmark* para nosso problema tivemos que gerá-las, adaptando instâncias ou geradores já existentes para problemas relacionados. Implementamos, então, dois processos de geração de instâncias.

No primeiro processo, usamos GDAs com estruturas especiais e aleatórias oriundos de instâncias criadas por uma ferramenta chamada *ANDES-Synth* [23]. Para cada GDA, as porcentagens de 25%, 50% e 75% foram utilizadas para limitar o tamanho dos conjuntos E criados, tendo como base todos os pares de incomparáveis encontrados nestas instâncias. Adicionalmente, para cada porcentagem, três conjuntos E diferentes foram gerados para uma maior diversidade.

Além deste conjunto de instâncias, decidimos utilizar um segundo processo de criação baseado no gerador de instâncias para problemas de escalonamento com restrições de recursos criado por [25], *ProGen*, já que nosso problema pode ser considerado um caso particular desta classe e muitas instâncias hoje utilizadas para *benchmark* foram criadas utilizando tal gerador. A geração das instâncias é feita com base em vários parâmetros passados através de um arquivo texto para o programa. Com este processo, foram criadas 60 instâncias, divididas em seis famílias, de acordo com os parâmetros utilizados na sua criação. Cada instância é nomeada por `pro`xy, onde $x \in \{a, b, c, d, e, f\}$ e $y \in \{1, 2, \ldots, 10\}$, identificando a família e o número do problema, respectivamente. Os parâmetros de criação e seus efeitos na resolução das instâncias geradas são descritos no Apêndice A.2 deste trabalho.

Um resumo das características das instâncias utilizadas por nós pode ser visto nas tabelas 7.1 e 7.2, referentes às instâncias criadas a partir dos GDAs e àquelas criadas com o auxílio do gerador *ProGen*, respectivamente. Ao longo deste capítulo de resultados referimo-nos às instâncias criadas por nós a partir de grafos já existentes por "instâncias ANDES", enquanto as instâncias criadas com o auxílio do gerador *ProGen* são chamadas de "instâncias ProGen".

Nas próximas seções apresentamos resumidamente os resultados computacionais obtidos com os algoritmos propostos neste trabalho, quando aplicados às instâncias descritas acima. Primeiro testamos as

Instância	dimensão $\|V\|$	$\|\vdash\|$	$\|i\|\|j\|$	$\|E\|$ 75%	50%	25%
bin5	31	30	367	276	184	92
bin6	63	62	1695	1272	848	424
bin7	127	126	7359	5520	3680	1840
di25	25	40	100	75	50	25
di36	36	60	225	169	113	57
di64	64	112	784	588	192	196
ran2	80	190	2255	1692	1128	564
ran9	124	104	7395	5547	3698	1849
ran15	152	419	8793	6595	4397	2199
gauss19	18	23	72	54	36	18
gauss34	33	47	261	196	131	66
irr41	41	69	393	295	197	99
molecular	40	66	393	295	197	99

Tabela 7.1: Descrição das instâncias ANDES

nossas heurísticas para limites superiores, avaliando a qualidade das soluções obtidas e os tempos gastos. (Seção 7.1). De posse dos melhores limites superiores obtidos prosseguimos com a avaliação das técnicas de limites inferiores propostas neste trabalho: a coloração fracionária e a programação por restrições, em conjunto com a relaxação linear dos modelos e sozinhas, o método de shaving e, depois, a programação por restrições e relaxação linear em um procedimento destrutivo. Finalmente utilizamos o *solver* CPLEX, com os limites inferiores e superiores encontrados, procuramos resolver exatamente cada instância.

Aqui apresentamos os dados que levam às principais conclusões. Tabelas detalhadas podem ser vistas no Apêndice B.

7.1 Limites superiores

As três heurísticas para cálculo de limites superiores (gulosa, coloração iterativa e coloração *first-fit*), nas várias diferentes variações descritas, foram complementadas e testadas com todas as instâncias. Os experimentos computacionais mostraram que as heurísticas *first-fit* que utilizam como função de valoração $f(i) = h(P_i^-) - h(P_i^+)$ se saíram melhor. Para chegar a esta conclusão foram analisados os melhores limites obtidos para cada instância e o tempo médio de obtenção dos limites.

As duas heurísticas citadas conseguiram obter o melhor limite superior na maioria das instâncias (142 de 177). As heurísticas gulosas, baseadas nas alturas das subordens induzidas por sucessores e antecessores dos elementos, conseguiram também bons resultados quando as instâncias em questão são de pequeno porte. Entretanto, à medida que aumentam de tamanho, a qualidade dos limites obtidos diminue. Já os limites superiores gerados pelas heurísticas de coloração iterativa foram sempre piores ou equivalentes àqueles das outras instâncias.

Quanto ao tempo computacional, as heurísticas baseadas em coloração iterativa foram as mais rápidas, já que suas estruturas são mais simples (apesar da criação de diferentes visões de subgrafos a cada iteração) e não requerem atualizações de valores durante sua execução, como acontece no caso das outras heurísticas. As heurísticas gulosas foram, geralmente, as mais lentas devido às frequentes atualizações das alturas das subordens a cada iteração, embora possuam um bom tempo de execução entre instâncias de pequeno e médio porte, o que não ocorre nas demais instâncias. Seu custo se torna tanto pior do que o dos demais métodos quanto maior o tamanho da instância e, portanto, maior a cardinalidade de E.

Embora não tenham sido as mais rápidas, as heurísticas *first-fit* não foram de modo algum lentas. A necessidade de atualização da altura de alguns elementos, mesmo que em menor número quando comparado às heurística gulosas, não permitiu que as versões *first-fit* fossem mais rápidas que a coloração iterativa. Mesmo assim, conseguiram manter baixas médias de tempo de execução até entre as instâncias maiores.

Instância	$\|V\|$	$\|\vdash\|$	$\|E\|$	Instância	$\|V\|$	$\|\vdash\|$	$\|E\|$
proa1	30	43	15	prob1	30	43	60
proa2	30	43	27	prob2	30	44	167
proa3	30	43	32	prob3	30	43	128
proa4	30	43	24	prob4	30	42	113
proa5	30	43	19	prob5	30	43	172
proa6	30	42	15	prob6	30	43	172
proa7	30	43	14	prob7	30	43	81
proa8	30	43	15	prob8	30	42	102
proa9	30	43	11	prob9	30	43	127
proa10	30	42	36	prob10	30	44	58
proc1	60	88	343	prod1	60	87	1063
proc2	60	86	119	prod2	60	84	1217
proc3	60	82	149	prod3	60	87	1009
proc4	60	87	190	prod4	60	86	967
proc5	60	87	123	prod5	60	87	764
proc6	60	88	196	prod6	60	86	959
proc7	60	83	242	prod7	60	89	1066
proc8	60	83	170	prod8	60	87	1169
proc9	60	85	343	prod9	60	87	963
proc10	60	84	212	prod10	60	84	837
proe1	90	187	919	prof1	90	183	2107
proe2	90	178	753	prof2	90	183	2737
proe3	90	183	650	prof3	90	188	2308
proe4	90	185	885	prof4	90	181	1995
proe5	90	184	458	prof5	90	184	1777
proe6	90	186	597	prof6	90	184	2768
proe7	90	180	1002	prof7	90	184	2384
proe8	90	184	627	prof8	90	189	1777
proe9	90	185	802	prof9	90	178	2316
proe10	90	186	653	prof10	90	186	2617

Tabela 7.2: Descrição das instâncias - *ProGen*

Outro fator que as impediram de se destacar ainda mais é a construção de relações que se dá entre o vértice que acaba de ser colorido e todos aqueles que, juntamente com ele, incidem sobre algum par em E. A heurística *first-fit* foi elaborada com a idéia de misturar o melhor dos métodos de coloração iterativa e guloso através da união das suas principais idéias: a observação do estado atual do poset e das alturas de subordens induzidas e a utilização de um procedimento rápido de coloração.

As tabelas 7.3 e 7.4 apresentam um resumo dos limites superiores obtidos para as instâncias ANDES e ProGen, respectivamente. Mostramos apenas os dados que consideramos de maior relevância e que condizem, de maneira fiel, conclusões de que as heurísticas *first-fit* apresentaram o melhor custo benefício entre desempenho e qualidade. Os resultados completos podem ser encontrados no ApêndiceB deste trabalho.

O significado das colunas é o seguinte:

- Pior/Melhor: o pior e melhor limite superior encontrado, considerando todas as heurísticas;
- CHsAsc/CHsDes: heurística *first-fit* utilizando $f(i) = h(P_i^-) - h(P_i^+)$ em ordem não decrescente (versão primal) e não crescente (versão dual), respectivamente.

Na Tabela 7.3, cada uma destas colunas, por sua vez, guarda três valores que representam o limite

superior obtido para cada instância criada para o poset em questão, para uma mesma porcentagem de elementos em E.

Instância	Pior	Melhor	CHsAsc	CHsDes
Instâncias 25%				
bin5	13/12/14	10/09/09	10/09/09	10/09/09
bin6	26/26/27	15/14/14	15/14/14	15/14/15
bin7	51/49/48	20/21/21	20/22/21	21/21/22
di25	14/13/15	12/12/12	13/13/12	12/12/14
di36	19/17/18	15/14/15	16/15/15	16/14/15
di64	32/32/31	23/23/23	26/25/26	24/23/25
ran2	29/31/31	17/18/17	17/18/19	19/18/18
ran9	40/48/47	17/17/18	17/19/18	18/19/18
ran15	56/58/56	27/28/27	28/28/28	28/28/27
gauss_19	10/10/10	08/08/08	08/09/08	08/08/09
gauss_34	16/18/18	12/14/12	14/15/14	13/15/12
irr41	21/18/19	16/15/15	16/17/15	16/15/16
molecular	18/18/19	13/13/15	13/14/16	14/15/16
Instâncias 50%				
bin5	16/19/21	12/13/13	12/15/13	12/13/14
bin6	36/35/38	21/20/20	21/20/20	21/22/23
bin7	72/73/80	32/32/33	32/32/33	33/32/37
di25	17/17/15	13/14/14	14/16/15	14/14/15
di36	29/26/25	20/20/19	21/21/21	20/20/19
di64	41/40/39	30/29/28	33/29/28	30/29/30
ran2	51/52/46	27/29/26	27/32/26	28/29/29
ran9	73/78/69	29/30/31	31/34/32	29/30/31
ran15	90/94/97	44/42/44	44/46/45	44/43/44
gauss_19	13/14/13	10/10/10	10/11/11	10/11/10
gauss_34	21/23/24	16/17/16	16/19/18	18/18/16
irr41	28/26/28	18/20/21	21/22/22	20/20/21
molecular	27/24/26	18/18/19	18/20/21	18/18/19
Instâncias 75%				
bin5	25/28/24	16/16/15	16/16/15	17/16/18
bin6	46/49/50	29/26/30	31/26/30	30/32/32
bin7	101/100/92	51/49/49	51/53/49	51/49/49
di25	22/21/22	16/18/19	17/19/19	16/19/20
di36	29/31/31	23/23/25	24/23/26	24/24/25
di64	56/50/51	39/40/41	39/40/41	39/40/42
ran2	65/65/62	41/40/37	41/40/38	40/40/37
ran9	98/101/98	46/47/45	48/47/47	46/47/45
ran15	125/113/117	67/65/64	67/65/64	67/67/68
gauss_19	17/16/15	12/12/12	15/13/13	12/12/13
gauss_34	27/28/29	22/22/20	27/23/24	22/24/20
irr41	33/37/32	24/25/25	26/26/26	24/25/25
molecular	33/28/33	26/22/24	27/25/25	26/25/24

Tabela 7.3: Resumo de limites superiores - Instâncias ANDES

Instância	Pior	Melhor	CHsAsc	CHsDes	Instância	Pior	Melhor	CHsAsc	CHsDes
proa1	10	8	8	8	prob1	15	12	13	12
proa2	12	11	11	11	prob2	21	16	19	16
proa3	14	11	11	12	prob3	20	13	13	14
proa4	14	12	13	12	prob4	18	15	16	15
proa5	11	9	9	9	prob5	22	16	17	17
proa6	11	10	10	11	prob6	23	16	17	17
proa7	10	9	9	9	prob7	15	12	13	14
proa8	9	8	8	8	prob8	16	12	13	12
proa9	10	8	8	8	prob9	19	16	17	17
proa10	9	8	8	8	prob10	13	12	13	12
proc1	26	20	22	20	prod1	49	34	34	35
proc2	20	16	16	17	prod2	52	41	41	41
proc3	18	14	15	16	prod3	50	36	36	38
proc4	19	15	16	15	prod4	48	34	35	34
proc5	16	11	12	12	prod5	42	29	30	29
proc6	20	16	16	16	prod6	51	37	39	37
proc7	22	13	13	13	prod7	48	34	34	37
proc8	16	12	14	12	prod8	56	39	40	39
proc9	28	17	17	19	prod9	50	30	32	30
proc10	19	14	14	14	prod10	45	29	31	29
proe1	39	25	27	25	prof1	71	53	54	54
proe2	40	23	23	24	prof2	82	65	68	65
proe3	34	21	26	24	prof3	77	57	57	63
proe4	45	28	31	28	prof4	68	49	50	50
proe5	27	21	22	23	prof5	68	41	44	41
proe6	30	24	25	24	prof6	86	65	65	69
proe7	40	26	28	26	prof7	73	50	50	53
proe8	34	21	21	23	prof8	64	45	46	48
proe9	42	26	28	27	prof9	76	55	55	56
proe10	33	21	23	25	prof10	80	62	63	62

Tabela 7.4: Resumos de limites superiores - Instâncias ProGen

Apresentamos também tabelas resumidas com a média do tempo de processamento para instâncias ANDES mais difíceis - densidade 75% (Tabela 7.5) e para os grupos de instâncias criadas com o gerador ProGen (Tabela 7.6). Do mesmo modo que ocorre com os limites, no Apêndice B são apresentadas tabelas completas, com a média dos tempos de execução para cada grupo de instâncias. Cada coluna apresenta as seguintes informações relativa a uma instância:

- Pior/Melhor: a pior e melhor média encontrada, considerando todas as heurísticas, respectivamente;
- CHsAsc/CHsDes: heurística *first-fit* utilizando $f(i) = h(P_i^-) - h(P_i^+)$ em ordem não decrescente (versão primal) e não crescente (versão dual), respectivamente.

Instância	Pior	Melhor	CHsAsc	CHsDesc
bin5	0,19	0,09	0,174	0,176
bin6	0,862	0,162	0,042	0,444
bin7	10,996	0,406	1,908	2,078
di25	0,148	0,08	0,134	0,148
di36	0,24	0,09	0,188	0,19
di64	0,432	0,15	0,338	0,374
ran2	1,316	0,172	0,724	0,762
ran9	10,002	0,364	2,158	2,288
ran15	14,936	0,546	3,05	3,484
gauss19	0,122	0,06	0,122	0,118
gauss34	0,206	0,086	0,206	0,204
irr41	0,274	0,102	0,274	0,264
molecular	0,26	0,098	0,26	0,26

Tabela 7.5: Médias de tempo - Instâncias ANDES 75%

Instância	Pior	Melhor	CHsAsc	CHsDesc
proa	0,28	0,07	0,24	0,28
prob	0,36	0,09	0,36	0,27
proc	0,53	0,11	0,37	0,53
prod	0,84	0,14	0,7	0,77
proe	0,93	0,17	0,93	0,9
prof	3,41	0,22	1,36	1,48

Tabela 7.6: Médias de tempo - Instâncias ProGen.

7.2 Limites inferiores

Apresentamos aqui os resultados acerca dos limites inferiores proporcionados pela coloração fracionária, pela programação por restrições e pelas relaxações dos modelos propostos, juntamente com a análise de como estes métodos se comportoram. Em seguida o método de shaving e o procedimento destrutivo são comparados com a programação por restrições.

A Tabela 7.7 está organizada da seguinte maneira. Há três grandes blocos de linhas, cada um eles associado a instâncias com um mesmo percentual de incomparáveis do poset original colocados em E. Cada linha refere-se a um mesmo GDA base, apresentando três blocos de colunas relativas às três instãncias geradas aleatoriamente. Para cada instância, temos os seguintes resultados armazenados nas colunas:

- C: valor (arredondado) da aproximação do número cromático dado pelo algoritmo proposto em [12];
- PR: limite inferior obtido pelo processo de programação por restrições sozinho (Algoritmo 10);
- PL: melhor valor obtido pelos modelos relaxados fortalecidos pela programação por restrições ($\overline{\Pi}(P, E)$ e $\overline{\theta}(P, E)$).

A Tabela 7.8, relativa às instâncias ProGen, possui organização semelhante.

Instância	CF	PR	PL	CF	PR	PL	CF	PR	PL
Instâncias 25%									
bin5	7	8	8	7	7	7	7	8	8
bin6	8	10	10	8	9	9	8	9	9
bin7	9	11	11	9	11	11	10	12	12
di25	11	11	11	11	11	12	11	12	12
di36	13	14	14	13	13	13	14	15	15
di64	20	19	20	20	19	20	19	19	19
ran2	12	12	13	12	12	13	12	13	13
ran9	7	7	7	7	8	8	8	7	7
ran15	13	13	14	14	14	15	13	13	14
gauss19	8	8	8	8	8	8	8	8	8
gauss34	12	12	12	13	13	14	12	12	12
irr41	12	12	13	13	12	12	12	13	13
molecular	11	12	12	11	12	12	12	12	12
Instâncias 50%									
bin5	9	9	9	9	8	8	9	11	11
bin6	13	13	13	12	12	12	13	12	12
bin7	15	14	14	15	15	15	15	15	15
di25	12	13	13	14	14	14	13	12	12
di36	16	16	16	17	17	17	16	16	16
di64	24	24	24	23	23	23	23	21	22
ran2	16	14	14	18	15	15	18	14	14
ran9	13	10	10	13	10	10	13	10	10
ran15	20	16	16	20	15	15	20	17	17
gauss19	9	9	9	10	10	10	9	9	10
gauss34	14	14	14	15	15	15	14	14	15
irr41	15	14	14	15	16	16	15	15	15
molecular	14	14	14	14	15	15	14	14	14
Instâncias 75%									
bin5	13	13	13	13	12	12	13	12	12
bin6	20	18	18	19	16	16	20	18	18
bin7	28	22	22	28	22	22	29	22	22
di25	14	15	15	16	16	16	17	17	17
di36	19	19	19	19	19	19	20	19	19
di64	29	27	27	30	26	26	30	25	25
ran2	26	19	19	25	19	19	25	19	19
ran9	26	18	18	26	18	18	27	19	19
ran15	36	24	24	35	22	22	36	23	23
gauss19	11	11	11	11	11	11	10	10	10
gauss34	17	15	15	18	16	17	18	17	17
irr41	18	17	17	20	18	18	19	19	19
molecular	21	20	20	18	17	17	19	18	18

Tabela 7.7: Limites inferiores - Instâncias ANDES

Nome	CF	PR	PL	Nome	CF	PR	PL
proa1	8	8	8	prob1	11	11	12
proa2	10	10	10	prob2	14	15	15
proa3	11	11	11	prob3	11	11	11
proa4	12	12	12	prob4	12	14	14
proa5	9	9	9	prob5	12	13	13
proa6	9	9	9	prob6	14	14	14
proa7	9	9	9	prob7	11	12	12
proa8	8	8	8	prob8	10	12	12
proa9	8	8	8	prob9	13	13	13
proa10	7	7	8	prob10	11	12	12
proc1	14	16	16	prod1	26	26	26
proc2	14	15	15	prod2	29	29	29
proc3	11	14	14	prod3	27	25	25
proc4	13	13	13	prod4	23	24	24
proc5	9	10	10	prod5	22	22	22
proc6	13	15	15	prod6	27	26	26
proc7	9	10	10	prod7	25	25	25
proc8	10	11	11	prod8	29	28	28
proc9	11	12	12	prod9	22	22	22
proc10	11	12	12	prod10	21	20	20
proe1	17	19	19	prof1	38	36	36
proe2	18	18	18	prof2	48	46	46
proe3	16	15	16	prof3	43	42	42
proe4	18	18	18	prof4	34	33	33
proe5	19	20	20	prof5	32	33	33
proe6	18	17	17	prof6	51	51	51
proe7	15	15	15	prof7	37	36	36
proe8	15	15	15	prof8	36	36	36
proe9	18	18	18	prof9	40	37	37
proe10	16	17	17	prof10	44	43	43

Tabela 7.8: Limites inferiores - Instâncias ProGen.

Para as instâncias ANDES, a programação por restrições (PR) obteve desempenho parecido à coloração fracionária (CF) entre aquelas com menor tamanho de E (25%). No entanto, à medida que $|E|$ cresce, CF se torna a melhor opção para avaliar limites inferiores, chegando a derrotar PR nas instâncias de maior complexidade (75%).

As instâncias do grupo `ran`x foram aquelas que evidenciaram uma superioridade de CF sobre PR de maneira mais mais clara em relação à diferença nos valores dos limites inferiores obtidos, que é maior em todas as instâncias do grupo se comparada com a diferença que encontramos em outros, a partir das intâncias de médio porte (50%). Essa diferença propavelmente deve-se ao fato de que o algoritmo de planos-de-corte usado para CF, explora, além de cliques, outras estruturas como buracos e anti-buracos, que também afetam a altura da extensão induzida.

Para as instâncias ProGen, o desempenho de CF e PR foi comparável, com leve superioridade de uma e de outra em alguns casos. Quanto às relaxações lineares, que já incorporam os limites de PR, conseguiram melhorá-los de uma unidade em 13 das 117 instâncias ANDES e em apenas 1 entre as 60 instâncias ProGen.

Além da análise dos limites encontrados, realizamos uma análise da relevância das regras de propagação de restrições utilizadas por nós, observando a quantidade de mudanças que elas induziam sobre a matriz B e, a partir deste "critério", supor que regra teria mais impacto na solução obtida.

De acordo com este critério, as regras *edge-finding*, como implementadas por nós, mostraram-se ineficazes para o EIP, ao contrário do que é relatado na literatura para o RCPSP, quando este grupo de regras mostrou-se poderosa. Uma possível explicação é que no EIP não lidamos com o conceito de "duração" (ou peso) nos nós da rede induzida por nossos posets, fato que é explorado nesta regra. Sabemos que

Instância	$\overline{\Pi}(P,E)$	$\overline{\theta}(P,E)$	$\overline{\theta}(P,E)$+Cliq
bin5	0,181	0,437	0,22
bin6	0,496	2,491	6,349
bin7	1,384	19,506	522,117
di25	0,107	0,167	0,164
di36	0,179	0,531	0,309
di64	0,376	1,464	7,886
ran2	0,605	3,929	29,154
ran9	1,423	17,54	279,274
ran15	1,817	28,957	2133,004
gauss19	0,087	0,139	0,116
gauss34	0,17	0,386	0,311
irr41	0,227	0,563	0,537
molecular	0,223	0,51	0,478

Tabela 7.9: Relaxação linear - Média de tempo instâncias ANDES 75%.

Instância	$\overline{\Pi}(P,E)$	$\overline{\theta}(P,E)$	$\overline{\theta}(P,E)$+Cliq
proa	0,100	0,098	0,094
prob	0,150	0,201	0,162
proc	0,223	0,342	0,305
prod	0,505	2,235	1,788
proe	0,501	1,207	5,056
prof	0,937	7,734	83,681

Tabela 7.10: Relaxação linear - Média de tempo instâncias ProGen.

existe na literatura outras regras deste tipo para problemas de escalonamento ditos *disjuntivos*, aqueles onde a execução de uma atividade que necessita de um determinado recurso impossibilita a execução de qualquer outra atividade que necessite do mesmo recurso ([3]). Talvez estas outra variações possam ter maior eficiência.

Já a regra de alargamento por cliques se mostrou de grande eficiência nas instâncias. Seu foco é verificar como as cliques de disjunções atuam na diferença das alturas entre os elementos do poset, particularmente na diferença entre os elementos artificiais 1 e n, o valor que mais nos interessa. No entanto, ela não fixa diretamente relações de precedência, como faz o *edge-finding*. Neste sentido, ela é complementada pela regra de seleção imediata. Esta destina-se a apoiar o processo como um todo, retirando de E aqueles pares que, devido a mudanças realizadas até o momento, já se encontram ordenados ou já podem ser ordenados, onde neste caso são realizadas todas as devidas atualizações.

Ainda com respeito às duas relaxações usadas, vale ressaltar que não houve diferença entre os limites encontrados na grande maioria das instâncias. Somente 17 instâncias, de um total de 177, tiveram diferença (e de somente uma unidade) entre os valores. Os modelos com variáveis indexadas na altura foram os responsáveis pela diferença positiva nestas 17 instâncias, mas, por outro lado, estes mesmos modelos precisaram de mais tempo para retornar uma solução. A Tabela 7.9, referente às maiores instâncias, nos dá uma idéia de quanto custa a resolução de cada modelo, em segundos, e a Tabela 7.10 fornece as médias de cada grupo de instâncias ProGen. Nestas tabelas, apresentamos o tempo médio, com relação às três instâncias de 75%, necessário para resolver o modelo relaxado $\overline{\Pi}(P,E)$ e as relaxações de $\theta(P,E)$ apenas com restrições de aresta ($\overline{\theta}(P,E)$) e com as restrições de cliques ($\overline{\theta}(P,E)$+Cliq).

As tabelas 7.11 e 7.12 fornecem os resultados obtidos com o *shaving* e o procedimento destrutivo. Os limites gerados com a programação por restrições também são novamente apresentados para comparação. A estrutura destas tabelas é similar àquela das tabelas 7.7 e 7.8, mas agora com as seguintes colunas:

- PR: o limite inferior encontrado pela programação por restrições (algoritmo 10);
- SH: o limite inferior encontrado pelo *shaving*;
- MD: o limite inferior encontrado pelo método destrutivo (algoritmo 13).

A execução dos procedimentos foi limitada a um tempo máximo de 60 minutos. Instâncias que extrapolaram este limite apresentam um "-" no local onde o resultado estaria. Outras instâncias onde nenhum par em E foi escolhido para *shaving* estão sinalizadas por "ND", já que deste modo apenas o procedimento normal de programação por restrições é executado.

Em 97 instâncias o método destrutivo conseguiu aumentar o limite inferior obtido somente com a programação por restrições. Já o processo de *shaving* trouxe uma melhora no limite inferior para somente 35 instâncias. Em ambos os caso as melhoras foram tímidas, consistindo de somente uma unidade, na maioria das instâncias, e em raras ocasiões a duas unidades, observadas no método destrutivo. O método destrutivo obteve, em 47 instâncias, resultados superiores ao *shaving*.

Como era de se esperar, o *shaving* é a técnica mais cara em termos computacionais, da maneira como foi implementada por nós (escolha de no máximo 50 pares a partir de um critério guloso), de acordo com as tabelas 7.13 e 7.14. Na maioria dos casos, somente instâncias de grande porte (do tipo `ran`x, `bin`6, `bin`7 e das famílias `prod`x e `prof`x) selecionaram, pelo critério guloso (Equação 5.13), um número de pares a sofrerem *shaving* bem maior que 50.

Instância	PR	SH	MD	PR	SH	MD	PR	SH	MD
Instância 25%									
bin5	8	8	8	7	7	8	8	8	8
bin6	10	ND	11	9	ND	10	9	ND	10
bin7	11	ND	12	11	ND	12	12	ND	12
di25	11	11	11	11	12	12	12	12	12
di36	14	15	15	13	13	13	15	15	15
di64	19	20	21	19	20	21	19	19	20
ran2	12	13	14	12	12	13	13	13	14
ran9	7	7	8	8	8	9	7	8	9
ran15	13	13	14	14	14	16	13	-	15
gauss19	8	8	8	8	8	8	8	8	8
gauss34	12	12	12	13	14	14	12	12	12
irr41	12	13	13	12	12	13	13	13	14
molecular	12	12	13	12	12	13	12	13	13
Instância 50%									
bin5	9	ND	9	8	ND	10	11	ND	11
bin6	13	ND	13	12	ND	13	12	ND	13
bin7	14	-	15	15	-	15	15	-	15
di25	13	13	13	14	14	14	12	13	13
di36	16	17	17	17	18	18	16	17	17
di64	24	24	26	23	23	25	21	22	23
ran2	14	14	16	15	16	17	14	15	17
ran9	10	-	11	10	-	11	10	-	11
ran15	16	-	18	15	-	17	17	-	18
gauss19	9	9	9	10	10	10	9	10	10
gauss34	14	14	15	15	15	15	14	14	15
irr41	14	15	16	16	17	18	15	ND	17
molecular	14	15	16	15	15	16	14	ND	15
Instância 75%									
bin5	13	ND	13	12	ND	12	12	ND	13
bin6	18	18	18	16	17	17	18	18	18
bin7	22	-	23	22	-	23	22	-	22
di25	15	15	15	16	17	17	17	18	18
di36	19	19	21	19	20	20	19	20	21
di64	27	ND	29	26	ND	29	25	ND	29
ran2	19	20	20	19	19	19	19	20	20
ran9	18	-	18	18	-	18	19	-	19
ran15	24	-	25	22	-	22	23	-	23
gauss19	11	11	11	11	11	11	10	11	11
gauss34	15	15	18	16	17	19	17	17	17
irr41	17	ND	19	18	ND	20	19	19	21
molecular	20	20	22	17	17	19	18	18	20

Tabela 7.11: Programação por restrições, *shaving* e método destrutivo - instâncias ANDES.

Instância	PR	SH	MD	Instância	PR	SH	MD
proa1	8	8	8	prob1	11	12	12
proa2	10	11	11	prob2	15	15	15
proa3	11	11	11	prob3	11	11	11
proa4	12	12	12	prob4	14	ND	14
proa5	9	9	9	prob5	13	ND	13
proa6	9	9	10	prob6	14	ND	14
proa7	9	9	9	prob7	12	12	12
proa8	8	8	8	prob8	12	12	12
proa9	8	ND	8	prob9	13	14	14
proa10	7	8	8	prob10	12	12	12
proc1	16	16	16	prod1	26	26	27
proc2	15	15	16	prod2	29	29	29
proc3	14	14	14	prod3	25	26	26
proc4	13	13	13	prod4	24	24	24
proc5	10	10	10	prod5	22	22	22
proc6	15	15	15	prod6	26	26	26
proc7	10	10	11	prod7	25	25	26
proc8	11	12	12	prod8	28	28	28
proc9	12	ND	13	prod9	22	23	22
proc10	12	ND	12	prod10	20	20	21
proe1	19	19	20	prof1	36	36	37
proe2	18	18	18	prof2	46	-	46
proe3	15	15	16	prof3	42	-	42
proe4	18	18	19	prof4	33	33	33
proe5	20	20	20	prof5	33	33	33
proe6	17	ND	18	prof6	51	-	51
proe7	15	15	16	prof7	36	-	36
proe8	15	ND	16	prof8	36	36	36
proe9	18	18	19	prof9	37	-	38
proe10	17	17	17	prof10	43	-	43

Tabela 7.12: Programação por restrições, *shaving* e método destrutivo - instâncias ProGen.

Instância	PR	SH	MD
bin5	0,206	-	0,909
bin6	2,277	513,752	13,338
bin7	56,553	-	222,641
di25	0,134	4,257	0,624
di36	0,196	19,373	1,174
di64	1,417	-	10,909
ran2	4,825	1141,865	36,119
ran9	50,175	-	247,151
ran15	111,143	-	1.256,894
gauss19	0,091	0,552	0,587
gauss34	0,195	4,829	1,316
irr41	0,398	1,179	2,390
molecular	0,391	2,06	1,930

Tabela 7.13: Programação por restrições, *shaving* e método destrutivo - Tempo médio instâncias ANDES 75% (em segundos).

Instância	PR	SH	MD
proa	0,059	0,201	0,038
prob	0,125	1,394	0,499
proc	0,260	13,055	0,951
prod	1,490	538,524	9,469
proe	1,768	121,344	9,660
prof	11,092	1921,157	93,932

Tabela 7.14: Programação por restrições, *shaving* e método destrutivo - Tempo médio instâncias ProGen (em segundos).

7.3 Soluções ótimas

Para encerrar este capítulo, as tabelas 7.15 e 7.16 apresentam um resumo geral dos melhores limites obtidos, comparando-os ao valor da solução ótima para cada instância, quando disponível. Os limites consistem no melhor valor encontrado como limite inferior (LI), como limite superior (LS) e, no caso de existir, o valor ótimo de qualquer solução (Opt.). A obtenção destes valores ótimos foi obtida através da resolução do modelo $\Pi(P, E)$ não relaxado. Instâncias que extrapolaram o tempo máximo de processamento de 60 minutos não contam com seu valor ótimo para exibição.

Para as instâncias ANDES, poucas vezes LI é igual a LS. Quando a diferença é de até 3 (e às vezes 4!) o CPLEX consegue encontrar a solução ótima. Quando o *gap* é maior, isto não acontece. As instâncias ProGen tiveram o *gap* fechado em 17 das 60 instâncias. Para aquelas com *gap* maior que 3 (as famílias `prod`x, `proe`x e `prof`x) o mesmo cenário visto com as instâncias ANDES se repete: o CPLEX não consegue encontrar, no tempo máximo permitido, a solução exata.

A grande diferença entre os limites inferior e superior para algumas instâncias nos levam a crer que os procedimentos implementados não são capazes de explorar todas as estruturas do problema que afetam a altura da extensão ótima.

Instâncias	LI	LS	Opt.	LI	LS	Opt.	LI	LS	Opt.
Instâncias 25%									
bin5	8	10	9	8	9	8	8	9	8
bin6	11	15	-	10	14	-	10	14	-
bin7	12	20	-	12	21	-	12	21	-
di25	11	12	11	12	12	12	12	12	12
di36	15	15	15	13	14	13	15	15	15
di64	21	23	22	21	23	22	20	23	22
ran2	14	17	-	13	18	-	14	17	-
ran9	8	17	-	9	17	-	9	18	-
ran15	14	27	-	16	28	-	15	27	-
gauss19	8	8	8	8	8	8	8	8	8
gauss34	12	12	12	14	14	14	12	12	12
irr41	13	16	14	13	15	13	14	15	14
molecular	13	13	13	13	13	13	13	15	13
Instâncias 50%									
bin5	9	12	10	10	13	11	11	13	11
bin6	13	21	-	13	20	-	13	20	-
bin7	15	32	-	15	32	-	15	33	-
di25	13	13	13	14	14	14	13	14	13
di36	17	20	18	18	20	19	17	19	18
di64	26	30	-	25	29	-	23	28	-
ran2	16	27	-	18	29	-	18	26	-
ran9	13	29	-	13	30	-	13	31	-
ran15	20	44	-	20	42	-	20	44	-
gauss19	9	10	9	10	10	10	10	10	10
gauss34	15	16	15	15	17	15	15	16	15
irr41	16	18	18	18	20	19	17	21	17
molecular	16	18	16	16	18	16	15	19	17
Instâncias 75%									
bin5	13	16	-	13	16	-	13	15	-
bin6	20	29	-	19	26	-	20	30	-
bin7	28	51	-	28	49	-	29	49	-
di25	15	16	15	17	18	17	18	19	18
di36	21	23	21	20	23	21	21	25	21
di64	29	39	-	30	40	-	30	41	-
ran2	26	41	-	25	40	-	25	37	-
ran9	26	46	-	26	47	-	27	45	-
ran15	36	67	-	35	65	-	36	64	-
gauss19	11	12	11	11	12	12	11	12	12
gauss34	18	22	19	19	22	19	18	20	18
irr41	19	24	-	20	25	-	21	25	-
molecular	22	26	-	19	22	-	20	24	-

Tabela 7.15: Resultados finais

Instâncias	LI	LS	Opt.	Instâncias	LI	LS	Opt.
proa1	8	8	8	prob1	12	12	12
proa2	11	11	11	prob2	15	16	15
proa3	11	11	11	prob3	11	13	12
proa4	12	12	12	prob4	14	15	14
proa5	9	9	9	prob5	13	16	14
proa6	10	10	10	prob6	14	16	14
proa7	9	9	9	prob7	12	12	12
proa8	8	8	8	prob8	12	12	12
proa9	8	8	8	prob9	14	16	14
proa10	8	8	8	prob10	12	12	12
proc1	16	20	-	prod1	27	34	-
proc2	16	16	16	prod2	29	41	-
proc3	14	14	14	prod3	27	36	-
proc4	13	15	13	prod4	24	34	-
proc5	10	11	10	prod5	22	29	-
proc6	15	16	15	prod6	27	37	-
proc7	11	13	11	prod7	26	34	-
proc8	12	12	12	prod8	29	39	-
proc9	13	17	14	prod9	23	30	-
proc10	12	14	12	prod10	21	29	-
proe1	20	25	-	prof1	38	53	-
proe2	18	23	-	prof2	48	65	-
proe3	16	21	-	prof3	43	57	-
proe4	19	28	-	prof4	34	49	-
proe5	20	21	20	prof5	33	41	-
proe6	18	24	-	prof6	51	65	-
proe7	16	26	-	prof7	37	50	-
proe8	16	21	-	prof8	36	45	-
proe9	19	26	-	prof9	40	55	-
proe10	17	21	-	prof10	44	62	-

Tabela 7.16: Resultados finais - Instâncias ProGen

8

Conclusões

Durante a elaboração desta dissertação, definimos o problema de extensões induzidas de altura mínima de conjuntos parcialmente ordenados e propomos duas formulações de programação inteira. Mostramos que o problema pertence à classe de problemas $\mathcal{NP}$-completo, através de uma redução do problema de coloração de vértices de um grafo. Também experimentamos variações de três heurísticas desenvolvidas ou adaptadas por nós, para geração de limites superiores, e uma adaptação da técnica de programação por restrições para problemas de escalonamento de atividades, para fornecer limites inferiores, com o objetivo de limitar o domínio de valores que as soluções para uma instância do problema pode ter.

Com respeito às heurísticas propostas, verificamos que aquela baseada em um procedimento de coloração gulosa, orientada pelas alturas dos elementos no poset corrente, mostrou-se capaz de gerar limites superiores menores que as demais heurísticas em tempo hábil, que não chega a ser o menor, mas suficiente para obter o melhor custo/benefício.

Adaptamos a técnica de programação por restrições para problemas de escalonamento de tarefas, utilizando regras de propagação de restrições e *edge-finding* já existentes para tal problema. Uma nova regra baseada em cliques para limitar inferiormente as diferenças de altura entre os elementos do poset, mesmo aqueles que não estão na clique, foi criada. Vimos que tal regra foi fundamental para a obtenção do limite inferior das instâncias. Por outro lado, a regra *edge-finding*, que se esperava mais poderosa, dadas as experiências reportadas para o RCPSP na literatura, não conseguiu os resultados esperados.

O limite inferior decorrente da resolução do problema de coloração fracionária por planos-de-cortes superou aquele fornecido pela programação por restrições em várias instâncias, principalmente aquelas consideradas de grande porte, o que atesta o poder do primeiro método. A resolução dos modelos relaxados fortalecidos não foi capaz de melhorar significamente os resultados obtidos com a programação por restrições. Na verdade, a capacidade destes modelos está fortemente ligada ao procedimento de programação por restrições, que é usado para fortalecer o domínio das variáveis. As experiências com as relaxações destes modelos visavam verificar o efeito das inequações que propomos na qualidade dos limites. O modelo com variáveis contínuas, apesar de não ter sido acrescido de nenhuma nova desigualdade, foi o mais rápido e eficiente.

A utilização da técnica de *shaving* e do método destrutivo trouxeram pouca melhora aos limites já obtidos, além de tomarem mais tempo de processamento. A pouca melhora se deve à sua dependência em relação ao próprio método de programação por restrições, que está embutido no centro destas duas técnicas. Pode ser possível melhorar o tempo de processamento de ambas as técnicas, considerando a linguagem de programação utilizada e aperfeiçoando não somente as regras utilizadas na propagação de restrições, mas também o gerenciamento das estruturas implementadas para estas técnicas, que precisaram ser criadas e destruídas muitas vezes. Isto demanda intenso uso de memória, fazendo com que o coletor de lixo Java gaste uma parte considerável do seu tempo removendo objetos não mais utilizados. Uma política de reaproveitamento destas estruturas que estão sendo destruídas e criadas frequentemente durante o processo pode vir a melhorar o desempenho dos algoritmos implementados.

Uma direção para trabalhos futuros seria fortalecer as regras de propagação de restrições para gerar limites inferiores maiores e explorar estruturas que não foram abordadas em nossos experimentos. A

perspectiva de derivar uma regra *edge-finding* mais adequada ao problema poderia gerar grandes avanços. Também é possível aprofundar os estudos na relação entre coloração e o EIP, adaptando resultados teóricos existentes ou criando outros específicos ao problema. Isto incluiria também a utilização de técnicas modificadas de resolução de problemas de coloração. Indo além dos métodos pertencentes à fase de pré-processamento do problema, a geração de novas inequações a partir das alterações no processo de programação por restrições para ambas as formulações também seria interessante, para fortalecer os modelos propostos. Finalmente, o desenvolvimento de um algoritmo *branch-and-bound* específico, que explore as propriedades do problema, ajudado pela melhora dos limites, pode levar à resolução exata de instâncias maiores.

A

Criação de Instâncias

Este apêndice trata da descrição do processo de criação das instâncias utilizadas neste trabalho. Através do conhecimento detalhado deste processo é possível analisar de forma mais crítica os resultados apresentados anteriormente. Além disso, podemos verificar como os vários procedimentos se comportaram em relação a certos tipos de estruturas ou características presentes nestas instâncias.

A.1 Instâncias ANDES

No primeiro processo, utilizamos GDAs com estruturas especiais e aleatórias, oriundos de instâncias criadas por uma ferramenta chamada *ANDES-Synth* ([23]). Esta ferramenta gera grafos direcionados de tarefas que capturam as principais características de programas paralelos bem conhecidos.

Para cada GDA obtivemos o fecho transitivo, o que gera a estrutura de um poset: um conjunto de elementos (os vértices) em uma relação não reflexiva, anti-simétrica e transitiva (os arcos orientados). Em seguida, enumeramos todos os pares de incomparáveis do poset. De forma aleatória, uma porcentagem destes pares foram escolhidos para compor o conjunto E, descrito na definição do EIP, Seção 3.1. Para cada GDA, as porcentagens de 25%, 50% e 75% foram utilizadas. Adicionalmente, para cada porcentagem, três conjuntos E diferentes foram gerados para uma maior diversidade.

Os grafos que serviram de base para a criação das instâncias são:

`bin`h**,** `di`n **e** `irr`41 Estes grafos são árvores binárias completas de altura h, grafos diamante com n vértices e um grafo irregular com 41 vértices, respectivamente;

`gauss`x **e** `molecular` Estes grafos representam programas de eliminação gaussiana com x vértices e física molecular, respectivamente;

`ran`x Estes são dags gerados aleatoriamente da seguinte forma. Primeiro, escolhe-se a quantidade de elementos. Depois, para cada elemento i, escolhe-se aleatoriamente o número de sucessores imediatos, que são então eleitos dentre a lista de elementos candidatos, usando uma distribuição uniforme. A referência x apenas diferencia os posets.

A Tabela 7.1, apresentada no Capítulo 7, possui a descrição completa destas instâncias.

A.2 Instâncias *ProGen*

O programa *ProGen*, desenvolvid por [25], gera instâncias para problemas de escalonamento com restrições de precedência e compartilhamento de recursos, através de um processo controlado por vários parâmetros. As instâncias criadas com o gerador *ProGen* foram uma alternativa à criação aleatória de instâncias a partir de grafos direcionados acíclicos usadas em [12]. Sua utilização teve como objetivo gerar instâncias onde pudéssemos saber, de antemão, o grau de dificuldade, de acordo com os parâmetros utilizados para

sua geração, mas sem necessariamente implicar em instâncias de tamanho maior do que aquelas do tipo `ran`x.

Nesta seção descrevemos como se deu o processo de criação, explicando de que forma os pararâmetros utilizados por nós afetam as instâncias criadas. Somente os parâmetros de maior impacto na dificuldade de resolução das instâncias, de acordo com [25], e de utilidade prática para nós serão abordados. O processo de criação é descrito aqui de maneira simplificada. Para maior rigor e detalhes é necessário consultar [25]. O resumo dos parâmetros aplicados a cada família de instâncias gerada com o *ProGen* será descrita no final deste capítulo.

Como o *ProGen* é um gerador de instâncias para problemas de escalonamento de atividades com restrições de recursos, os seguintes parâmetros devem ser definidos, além da rede de relações de precedência entre atividades: o número de recursos, que recursos cada atividade demanda e em que quantidade e a disponibilidade de cada recurso. Uma vez definidos todos estes valores, uma instância do EIP é criada tendo como poset inicial a própria rede de atividades. O conjunto E é construído a partir de cada par de atividades que demandam um mesmo recurso e onde tal demanda exceda a disponibilidade daquele recurso.

Lembrando que um poset $Z = (A, R)$ pode ser representado por um GDA, o primeiro passo na criação de uma instância é a criação da rede de precedência que define o poset inicial. Para isso, é definido o número mínimo e máximo de elementos presentes no poset, J^{min} e J^{max} respectivamente, onde $a = 1$ e $a = J$ são o único elemento "fonte" e "sumidouro", respectivamente. Logo, para todo elemento $a \in A \setminus \{1, J\}$ há um caminho de 1 a a e de a até J. Para cada $a \in A \setminus \{1, J\}$ é designado um predecessor e um sucessor, e só depois outros arcos são adicionados, respeitando o número máximo de sucessores (S_a^{max}) e predecessores (P_a^{max}). O processo pára quando a *complexidade da rede*, C, é atingida. A complexidade da rede C é a quantidade média de arcos não redundantes por elemento, onde um arco (h, j) em um dag $D = (V, P)$ é chamado de *arco redundante* caso existam arcos $(i_0, i_1), \ldots, (i_{s-1}, i_s) \in P$ onde $i_0 = h$, $i_s = j$ e $s \geq 2$. De acordo com [25], quanto maior a complexidade da rede, mais facilmente o problema pode ser resolvido, já que redes mais densas limitam o conjunto de soluções.

A quantidade de recursos disponíveis $|\tau|$ é escolhida aleatoriamente do intervalo $[|\tau|^{min}, |\tau|^{max}]$. Para criar a demanda por recursos, o conceito de *fator de recursos RF* é utilizado. O fator de recursos $RF \in [0, 1]$ é a razão média de recursos requisitados por elemento. Um $RF = 1$ indica que todo elemento demanda todos os recursos e $RF = 0$ indica que nenhum elemento demanda recursos. Dependendo da demanda de um elemento por cada recurso, os elementos que demandam um mesmo recurso podem vir a estar na mesma clique de disjunções do poset. A geração de demanda é feita escolhendo-se aleatoriamente pares (a, r), onde $a \in A$ e $r \in \tau$, que indicam se o elemento a requer o recurso r, até que o RF desejado seja atingido e respeitando a demanda máxima permitida por recursos, Q^{max}. Segundo [25], quanto maior a quantidade média de recursos necessitados por atividade, mais difícil o problema se torna. No nosso caso, o efeito deste parâmetro esta relacionado ao número de cliques que podem vir a ser encontradas na instância para o EIP e ao tamanho destas cliques.

Quando uma dupla (a, r), $a \in A$ e $r \in \tau$, é escolhida, é necessário saber que quantidade do recurso r o elemento a precisa. A quantidade é escolhida aleatorimante do intervalo $[\mathcal{U}_r^{min}, \mathcal{U}_r^{max}]$.

A disponibilidade de recursos é gerada automaticamente com a ajuda do parâmetro denominado *poder do recurso*, RS, que expressa a relação entre demanda e disponibilidade de recursos. A disponibilidade do recurso r é dada por $K_r = K_r^{min} + RS(K_r^{max} - K_r^{min})$, onde K_r^{min} é a menor quantidade requisitada por um elemento $a \in A$ do recurso r e K_r^{max} é a maior demanda por período do recurso r em um escalonamento onde cada elemento da rede é escalonado o mais cedo possível, respeitando a ordem de precedência. Segundo estudos em [25], a dificuldade de resolução das instâncias é inversamente proporcional a RS.

Através do estudo deste processo de criação, fornecemos certos valores ao gerador para criação de instâncias que pudessem ser utilizadas em nosso problema e das quais tivéssemos uma noção da sua real dificuldade. As famílias das instâncias geradas se encontram descritas abaixo. Para cada família foram criadas 10 instâncias.

Alguns dos parâmetros mencionados anteriormente não estão na tabela por terem sido utilizados com valores fixo. São eles:

- $\mathcal{U}_r^{min}$ e $\mathcal{U}_r^{max}$, de valor 1;
- RS, de valor 0.

Utilizando os valores constantes $\mathcal{U}_r^{min} = \mathcal{U}_r^{max} = 1$, qualquer outro valor para RS não iria gerar instâncias que pudessem ser utilizadas efetivamente, de acordo com nosso processo de transformação para instâncias

Família	J^{min}	J^{max}	P_a^{max}	S_a^{max}	$\|\tau\|^{min}$	$\|\tau\|^{max}$	Q^{max}	RF	C
proa	30	30	3	3	4	13	6	0,10	1,5
prob	30	30	3	3	4	13	6	0,25	1,5
proc	60	60	6	6	9	27	13	0,10	1,5
prod	60	60	6	6	9	27	13	0,25	1,5
proe	90	90	9	9	13	40	20	0,10	2,1
prof	90	90	9	9	13	40	20	0,25	2,1

Tabela A.1: Famílias de instâncias ProGen e suas características

do EIP. Problema similar foi provocado quando da geração utilizando $\mathcal{U}_r^{min} = 1$, $\mathcal{U}_r^{max} = 10$ e $RS > 0,25$, por exemplo.

Os valores de $|\tau|^{min}$ e $|\tau|^{max}$ correspondem a 15% e 45% do valor de J^{min}, respectivamente. Já Q^{max} é igual à metade do número máximo de recursos permitidos, $|\tau|^{max}$.

Os aparentemente baixos valores para RF se justificam pela relação entre quantidade de recursos e cliques que são geradas por elementos que demandam o mesmo recurso. Tivemos o cuidado de não extrapolar muito este número, o que implicaria em conjuntos E muito densos para nosso problema, já que não estávamos interessados em somente instâncias difíceis, mas em gerar diversidade quanto à complexidade de resolução. A Tabela 7.2, também apresentada no Capítulo 7, possui a descrição completa das instâncias ProGen criadas.

B

Tabelas Detalhadas

Nas tabelas seguintes estão os resultados computacionais detalhados obtidos para as nossas instâncias para os vários métodos implementados por nós.

Nas tabelas B.1 a B.3 apresentamos os resultados detalhados sobre as heurísticas de limite superior para as instâncias ANDES. Cada coluna principal associa-se a uma heurística. Cada uma destas colunas, por sua vez, guarda três valores, que representam o limite superior obtido para cada conjunto E criado para o poset em questão, relativo à porcentagem a qual a tabela se refere. As colunas principais indicam os limites superiores obtidos a partir de:

- GIII: procedimento guloso baseado nas alturas das subordens induzidas (algoritmo 1) que utiliza o critério (iii);
- GIV: o mesmo procedimento guloso utilizando o critério (iv);
- ICDeg: coloração iterativa (algoritmo 2) utilizando como regra de prioridade o grau do vértice v;
- ICHeight: o mesmo procedimento de coloração iterativa utilizando como regra de prioridade a altura dos elementos;
- CHsAsc: coloração *first-fit* utilizando $f(i) = h(P_i^-) - h(P_i^+)$ em ordem não decrescente;
- CHsDes: coloração *first-fit* utilizando $f(i) = h(P_i^-) - h(P_i^+)$ em ordem não crescente;
- CH-Asc: coloração *first-fit* utilizando $f(i) = h(P_i^-)$ em ordem não decrescente;
- CH+Des: coloração *first-fit* utilizando $f(i) = h(P_i^+)$ em ordem não crescente;
- CH+Asc: coloração *first-fit* utilizando $f(i) = h(P_i^+)$ em ordem não decrescente.

A Tabela B.4, relativa às instâncias geradas utilizando o gerador *ProGen*, têm como única diferença a apresentação de um único valor. Para cada heurística, temos associado o limite superior de cada instância.

A média de tempo necessário para inicialização das estruturas envolvidas em cada método e o tempo médio destinado à execução do método em si, em segundos, encontram-se nas tabelas B.5, B.6 e B.7. Novamente, cada coluna representa um dos métodos implementados, como descrito anteriormente. Cada coluna relativa a uma instância apresenta a média do tempo de inicialização seguida pela média do tempo de execução do procedimento em si, separadas por um espaço. A Tabela B.8, relacionada às instâncias ProGen, tem uma organização semelhante, mas as médias estão agrupadas segundo o tipo das instâncias (`proa`, `prob`, ...).

Nas tabelas B.9 e B.10 apresentamos os resultados originais (não arrendondados) obtidos com o algoritmo de planos-de-cortes para resolução do problema de coloração fracionária (CF), além dos resultados obtidos pela programação por restrições (PR) e melhor valor obtido pela resolução dos modelos relaxados (RL).

As tabelas B.11 e B.12 apresentam o tempo médio gasto, em segundos, na resolução da relaxação dos modelos de programação inteira e suas variantes e nos processos de programação por restrições, *shaving* e método destrutivo, respectivamente, para todas as famílias de instâncias.

Instância	GIII	GIV	ICDeg	ICHeight	CH-Asc	CH+Des	CH+Asc	CHsAsc	CHsDes
bin5	10/09/10	10/09/10	13/11/14	13/11/14	12/12/09	13/11/12	10/10/10	10/09/09	10/09/09
bin6	16/16/16	16/16/16	24/26/27	24/26/27	16/17/16	26/23/30	17/16/15	15/14/14	15/14/15
bin7	28/26/26	28/26/26	41/42/42	41/42/42	27/28/23	51/49/48	22/26/25	20/22/21	21/21/22
di25	14/12/13	14/12/13	13/13/12	13/12/13	13/13/13	13/13/15	12/12/14	13/13/12	12/12/14
di36	17/16/16	17/16/16	17/16/16	17/15/16	19/16/16	18/17/18	15/14/17	16/15/15	16/14/15
di64	25/27/23	25/27/23	23/26/27	27/27/31	25/29/28	32/31/28	27/30/26	26/25/26	24/23/25
ran2	21/19/19	19/19/17	27/30/29	27/28/31	22/22/20	29/31/27	20/24/23	17/18/19	19/18/18
ran9	23/23/21	22/22/21	36/43/38	40/42/39	20/17/19	37/48/47	19/20/19	17/19/18	18/19/18
ran15	27/31/28	32/32/32	43/49/47	48/44/47	32/36/31	56/58/53	32/35/31	28/28/28	28/28/27
gauss_19	08/08/08	08/08/08	08/10/09	08/10/09	09/09/08	09/10/09	10/08/10	08/09/08	08/08/09
gauss_34	12/16/12	12/14/12	15/18/13	15/18/13	15/16/13	15/17/18	16/15/14	14/15/14	13/15/12
irr41	17/16/16	17/16/16	17/16/16	17/18/19	16/17/17	21/18/19	16/15/16	16/17/15	16/15/16
molecular	13/15/17	13/15/17	15/17/19	18/18/18	16/13/17	17/16/17	15/15/15	13/14/16	14/15/16

Tabela B.1: Limites superiores - Instâncias ANDES 25%

Instância	GIII	GIV	ICDeg	ICHeight	CH-Asc	CH+Des	CH+Asc	CHsAsc	CHsDes
bin5	14/16/17	14/14/16	15/19/21	15/19/21	13/14/14	16/24/19	13/14/14	12/15/13	12/13/14
bin6	26/28/25	26/24/25	34/33/36	34/33/36	22/24/24	36/35/38	24/22/22	21/20/20	21/22/23
bin7	44/46/47	47/45/48	72/73/79	72/73/79	41/37/39	74/67/80	34/35/35	32/32/33	33/32/37
di25	14/15/14	14/16/15	15/16/15	13/16/15	17/16/15	17/17/16	15/15/14	14/16/15	14/14/15
di36	21/22/20	20/22/20	23/21/22	23/23/21	21/21/21	29/26/25	21/21/20	21/21/21	20/20/19
di64	33/32/30	31/30/29	41/40/35	37/39/36	38/32/33	41/38/36	39/37/39	33/29/28	30/29/30
ran2	29/31/31	31/32/34	51/46/46	49/45/45	30/34/31	43/52/41	33/37/31	27/32/26	28/29/29
ran9	41/41/41	41/44/43	64/66/59	65/64/62	33/30/31	73/78/69	30/31/31	31/34/32	29/30/31
ran15	58/58/55	51/54/55	81/80/84	84/77/80	48/52/45	90/94/97	50/42/46	44/46/45	44/43/44
gauss_19	10/10/10	11/10/10	13/14/13	13/14/13	10/11/12	11/12/13	11/11/11	10/11/11	10/11/10
gauss_34	18/17/18	18/17/17	20/22/23	20/22/23	19/20/19	21/23/24	19/18/16	16/19/18	18/18/16
irr41	18/24/23	18/24/23	24/25/23	21/26/28	23/22/23	28/25/27	21/21/23	21/22/22	20/20/21
molecular	20/19/21	20/19/21	20/20/26	23/20/25	22/19/24	27/24/25	22/20/22	18/20/21	18/18/19

Tabela B.2: Limites superiores - Instâncias ANDES 50%

Instância	GIII	GIV	ICDeg	ICHeight	CH-Asc	CH+Des	CH+Asc	CHsAsc	CHsDes
bin5	17/18/20	19/19/19	22/25/24	22/25/24	19/17/18	25/28/24	16/19/18	16/16/15	17/16/18
bin6	38/38/38	37/37/40	46/49/50	46/49/50	32/34/33	46/48/49	29/30/32	31/26/30	30/32/32
bin7	73/72/75	68/75/72	101/99/92	101/99/92	54/56/57	101/100/90	54/50/55	51/53/49	51/49/49
di25	18/18/20	19/18/20	19/19/22	19/18/22	18/20/21	22/21/22	18/21/22	17/19/19	16/19/20
di36	24/27/26	23/26/25	25/27/28	27/31/28	29/29/26	25/29/31	26/31/28	24/23/26	24/24/25
di64	42/41/43	41/40/42	49/49/51	48/49/51	48/45/46	56/49/51	42/50/49	39/40/41	39/40/42
ran2	50/49/43	50/50/48	61/59/59	65/65/54	43/41/40	64/65/62	45/43/48	41/40/38	40/40/37
ran9	78/70/75	68/69/70	94/92/89	98/93/87	46/49/47	95/101/98	48/47/45	48/47/47	46/47/45
ran15	90/87/88	82/85/89	104/110/117	114/111/117	81/72/69	125/113/115	71/70/78	67/65/64	67/67/68
gauss_19	14/12/14	14/12/14	14/13/14	14/13/14	16/14/13	17/16/15	12/14/12	15/13/13	12/12/13
gauss_34	22/22/24	22/22/23	25/28/29	25/28/29	22/23/24	30/27/26	24/23/24	27/23/24	22/24/20
irr41	29/28/27	29/28/27	33/31/32	29/32/30	27/29/28	30/37/32	30/28/27	26/26/26	24/25/25
molecular	26/26/28	26/26/28	33/28/30	32/27/33	29/26/28	31/26/31	29/22/27	27/25/25	26/25/24

Tabela B.3: Limites superiores - Instâncias ANDES 75%

Instância	GIII	GIV	ICDeg	ICHeight	CH-Asc	CH+Des	CH+Asc	CHsAsc	CHsDes
proa1	8	8	8	8	10	8	9	8	8
proa2	11	11	11	11	11	12	11	11	11
proa3	11	11	12	12	11	14	14	11	12
proa4	12	12	13	13	13	14	14	13	12
proa5	9	9	9	9	11	11	11	9	9
proa6	10	10	11	11	10	10	10	10	11
proa7	9	9	10	9	10	10	10	9	9
proa8	8	8	8	8	8	8	9	8	8
proa9	8	8	8	8	8	9	10	8	8
proa10	8	8	8	8	9	9	9	8	8
prob1	12	12	13	13	13	15	14	13	12
prob2	17	17	21	22	21	24	19	19	16
prob3	14	14	16	15	17	20	15	13	14
prob4	15	15	17	17	17	18	15	16	15
prob5	17	17	20	20	17	22	16	17	17
prob6	16	17	21	21	17	23	17	17	17
prob7	13	12	15	15	14	14	13	13	14
prob8	14	13	15	15	15	16	13	13	12
prob9	17	16	18	18	18	19	18	17	17
prob10	12	12	13	13	13	13	13	13	12
proc1	21	20	24	24	24	26	21	22	20
proc2	16	16	20	20	19	19	18	16	17
proc3	15	14	17	17	16	18	18	15	16
proc4	16	15	19	19	19	21	18	16	15
proc5	12	11	16	15	16	13	15	12	12
proc6	16	16	18	18	17	20	18	16	16
proc7	13	14	16	16	16	22	15	13	13
proc8	14	12	16	16	14	16	14	14	12
proc9	19	19	23	23	21	28	19	17	19
proc10	15	14	18	19	16	19	14	14	14
prod1	38	39	45	45	35	49	36	34	35
prod2	46	47	52	53	42	49	47	41	41
prod3	37	37	45	50	41	49	40	36	38
prod4	37	37	42	45	37	48	39	35	34
prod5	32	33	38	38	34	42	31	30	29
prod6	38	37	46	46	39	51	39	39	37
prod7	38	38	44	44	37	48	41	34	37
prod8	43	43	50	50	42	56	45	40	39
prod9	35	35	42	45	34	50	33	32	30
prod10	31	33	38	36	30	45	31	31	29
proe1	30	31	39	38	29	39	32	27	25
proe2	27	24	36	35	26	40	27	23	24
proe3	21	24	33	34	26	33	24	26	24
proe4	30	30	39	41	30	45	33	31	28
proe5	21	24	26	27	24	28	25	22	23
proe6	24	24	28	29	29	30	25	25	24
proe7	26	26	35	35	35	40	31	28	26
proe8	26	23	33	34	25	30	27	21	23
proe9	29	30	39	39	26	42	27	28	27
proe10	23	21	33	33	30	32	25	23	25
prof1	60	61	67	70	57	71	53	54	54
prof2	74	76	80	81	71	82	67	68	65
prof3	65	64	76	75	66	77	63	57	63
prof4	54	53	62	67	55	68	49	50	50
prof5	52	52	57	59	46	68	45	44	41
prof6	74	74	84	82	72	86	72	65	69
prof7	56	59	70	73	53	73	57	50	53
prof8	48	47	64	63	54	64	45	46	48
prof9	60	63	73	74	62	76	64	55	56
prof10	68	69	75	77	63	80	63	63	62

Tabela B.4: Limites superiores - Instâncias ProGen

Instância	GIII	GIV	ICDeg	ICHeight	CH-Asc	CH+Des	CH+Asc	CHsAsc	CHsDes
bin5	0,101 0,055	0,085 0,052	0,079 0,056	0,015 0,070	0,094 0,048	0,071 0,026	0,074 0,064	0,096 0,063	0,095 0,061
bin6	0,126 0,210	0,122 0,227	0,103 0,097	0,028 0,113	0,142 0,152	0,098 0,052	0,117 0,181	0,141 0,225	0,140 0,220
bin7	0,166 1,654	0,196 1,674	0,137 0,300	0,035 0,300	0,211 0,797	0,136 0,174	0,177 0,891	0,210 0,961	0,210 0,945
di25	0,091 0,016	0,088 0,015	0,084 0,046	0,014 0,061	0,106 0,020	0,079 0,023	0,091 0,031	0,107 0,027	0,107 0,027
di36	0,109 0,033	0,108 0,031	0,098 0,059	0,017 0,072	0,135 0,034	0,098 0,032	0,113 0,054	0,136 0,042	0,138 0,047
di64	0,268 0,106	0,155 0,113	0,133 0,102	0,022 0,113	0,194 0,102	0,134 0,069	0,165 0,145	0,192 0,130	0,192 0,142
ran2	0,164 0,277	0,173 0,334	0,145 0,104	0,029 0,118	0,215 0,206	0,144 0,081	0,187 0,271	0,215 0,303	0,212 0,294
ran9	0,133 1,829	0,133 1,571	0,118 0,202	0,034 0,214	0,178 0,802	0,113 0,142	0,143 0,812	0,178 0,980	0,177 0,867
ran15	0,323 2,620	0,301 2,392	0,268 0,301	0,035 0,321	0,434 1,219	0,287 0,340	0,382 1,612	0,433 1,454	0,432 1,497
gauss19_1	0,073 0,007	0,076 0,007	0,070 0,033	0,013 0,046	0,082 0,022	0,066 0,019	0,071 0,028	0,082 0,027	0,081 0,027
gauss34_1	0,094 0,040	0,126 0,046	0,085 0,052	0,016 0,068	0,118 0,034	0,088 0,031	0,099 0,052	0,119 0,047	0,119 0,044
irr41	0,118 0,053	0,162 0,073	0,107 0,061	0,017 0,076	0,144 0,053	0,100 0,040	0,121 0,074	0,144 0,064	0,143 0,065
molecular_1	0,126 0,056	0,124 0,053	0,100 0,058	0,017 0,074	0,137 0,050	0,107 0,043	0,115 0,072	0,139 0,063	0,140 0,064

Tabela B.5: Médias de tempos - Instâncias ANDES 25%

Instância	GIII	GIV	ICDeg	ICHeight	CH-Asc	CH+Des	CH+Asc	CHsAsc	CHsDes
bin5	0,111 0,080	0,090 0,077	0,112 0,071	0,019 0,068	0,103 0,063	0,076 0,026	0,082 0,077	0,094 0,064	0,096 0,068
bin6	0,105 0,437	0,110 0,428	0,113 0,113	0,031 0,118	0,151 0,231	0,096 0,063	0,110 0,266	0,137 0,229	0,135 0,249
bin7	0,153 5,543	0,163 5,301	0,266 0,614	0,047 0,389	0,239 1,230	0,134 0,213	0,182 1,448	0,219 1,320	0,212 1,436
di25	0,082 0,024	0,083 0,021	0,084 0,043	0,015 0,059	0,115 0,025	0,078 0,022	0,089 0,039	0,107 0,024	0,116 0,027
di36	0,105 0,059	0,102 0,055	0,103 0,055	0,019 0,071	0,145 0,052	0,096 0,030	0,110 0,065	0,130 0,051	0,128 0,053
di64	0,148 0,180	0,149 0,177	0,139 0,131	0,029 0,113	0,207 0,128	0,131 0,063	0,165 0,177	0,193 0,124	0,186 0,155
ran2	0,166 0,666	0,159 0,622	0,146 0,119	0,036 0,127	0,217 0,309	0,144 0,091	0,187 0,402	0,215 0,423	0,214 0,439
ran9	0,145 4,843	0,133 4,563	0,128 0,278	0,043 0,285	0,188 1,334	0,121 0,204	0,149 1,439	0,185 1,515	0,185 1,515
ran15	0,293 7,360	0,292 6,953	0,270 0,401	0,045 0,449	0,436 1,825	0,263 0,391	0,380 2,219	0,437 2,114	0,434 2,323
gauss19_1	0,073 0,017	0,074 0,015	0,071 0,034	0,014 0,115	0,082 0,024	0,066 0,019	0,072 0,033	0,082 0,034	0,134 0,030
gauss34_1	0,094 0,062	0,094 0,058	0,087 0,055	0,018 0,068	0,120 0,044	0,082 0,033	0,099 0,066	0,120 0,068	0,123 0,064
irr41	0,118 0,085	0,117 0,081	0,108 0,067	0,020 0,077	0,146 0,068	0,103 0,041	0,123 0,095	0,147 0,093	0,146 0,091
molecular_1	0,112 0,085	0,110 0,081	0,101 0,063	0,020 0,074	0,141 0,068	0,097 0,040	0,116 0,094	0,140 0,092	0,141 0,093

Tabela B.6: Médias de tempos - Instâncias ANDES 50%

Instância	GIII	GIV	ICDeg	ICHeight	CH-Asc	CH+Des	CH+Asc	CHsAsc	CHsDes
bin5	0,078 0,110	0,082 0,108	0,101 0,071	0,020 0,069	0,103 0,076	0,070 0,027	0,081 0,093	0,099 0,074	0,097 0,079
bin6	0,108 0,753	0,126 0,726	0,270 0,133	0,033 0,128	0,156 0,281	0,098 0,063	0,119 0,325	0,142 0,277	0,138 0,305
bin7	0,154 10,842	0,160 10,297	0,155 0,618	0,060 0,437	0,245 1,501	0,144 0,262	0,189 1,833	0,222 1,686	0,217 1,860
di25	0,082 0,040	0,082 0,037	0,095 0,052	0,016 0,063	0,116 0,031	0,075 0,021	0,093 0,042	0,106 0,028	0,115 0,032
di36	0,103 0,078	0,110 0,079	0,117 0,064	0,018 0,072	0,139 0,100	0,093 0,030	0,110 0,074	0,130 0,057	0,129 0,061
di64	0,148 0,284	0,148 0,280	0,147 0,106	0,031 0,119	0,210 0,147	0,135 0,065	0,166 0,201	0,197 0,140	0,188 0,186
ran2	0,153 1,163	0,153 1,127	0,143 0,130	0,034 0,138	0,212 0,392	0,140 0,095	0,182 0,476	0,212 0,512	0,213 0,548
ran9	0,130 9,872	0,130 9,294	0,128 0,318	0,047 0,327	0,187 1,800	0,117 0,246	0,152 1,881	0,188 1,969	0,185 2,102
ran15	0,293 14,643	0,292 14,417	0,280 0,521	0,053 0,492	0,483 2,180	0,270 0,490	0,395 2,664	0,484 2,565	0,558 2,925
gauss19_1	0,073 0,029	0,072 0,025	0,074 0,034	0,013 0,047	0,084 0,029	0,065 0,019	0,071 0,039	0,082 0,039	0,082 0,035
gauss34_1	0,093 0,089	0,096 0,084	0,088 0,054	0,018 0,068	0,120 0,051	0,086 0,035	0,102 0,078	0,120 0,086	0,120 0,083
irr41	0,116 0,114	0,117 0,112	0,112 0,063	0,022 0,079	0,148 0,085	0,104 0,041	0,124 0,108	0,149 0,124	0,148 0,116
molecular_1	0,110 0,118	0,112 0,113	0,105 0,061	0,022 0,075	0,140 0,080	0,101 0,038	0,120 0,109	0,141 0,119	0,140 0,119

Tabela B.7: Médias de tempos - Instâncias ANDES 75%

Instância	GIII		GIV		ICDeg		ICHeight		CH-Asc		CH+Des		CH+Asc		CHsAsc		CHsDes	
proa	0,1	0,02	0,09	0,02	0,08	0,05	0,01	0,06	0,25	0,03	0,12	0,07	0,16	0,03	0,16	0,08	0,25	0,03
prob	0,09	0,07	0,09	0,07	0,08	0,05	0,02	0,07	0,2	0,05	0,11	0,03	0,08	0,1	0,24	0,12	0,2	0,07
proc	0,12	0,13	0,12	0,12	0,11	0,08	0,02	0,09	0,38	0,15	0,1	0,05	0,11	0,11	0,15	0,22	0,25	0,28
prod	0,12	0,72	0,12	0,67	0,11	0,1	0,03	0,11	0,21	0,42	0,11	0,11	0,2	0,38	0,25	0,45	0,31	0,46
proe	0,18	0,58	0,17	0,56	0,16	0,14	0,03	0,14	0,32	0,38	0,3	0,09	0,21	0,51	0,27	0,66	0,31	0,59
prof	0,18	3,23	0,17	2,97	0,16	0,17	0,04	0,18	0,36	0,76	0,29	0,11	0,3	0,69	0,32	1,04	0,33	1,15

Tabela B.8: Médias de tempos - Instâncias ProGen.

Instância	CF	PR	RL	CF	PR	RL	CF	PR	RL
Instâncias 25%									
bin5	6,4343	8	8	7	7	7	6,4285	8	8
bin6	8	10	10	7,9999	9	9	7,9999	9	9
bin7	8,9119	11	11	8,9547	11	11	9,0026	12	12
di25	11	11	11	11	11	12	11	12	12
di36	13	14	14	13	13	13	14	15	15
di64	19,7142	19	20	20	19	20	19	19	19
ran2	12	12	13	12	12	13	12	13	13
ran9	6,9999	7	7	6,9999	8	8	7,1631	7	7
ran15	12,0314	13	14	13,7999	14	15	12,6827	13	14
gauss19	8	8	8	8	8	8	8	8	8
gauss34	12	12	12	13	13	14	12	12	12
irr41	11,6153	12	13	12,0099	12	12	11,8823	13	13
molecular	11	12	12	11	12	12	11,3333	12	12
Instâncias 50%									
bin5	8,9615	9	9	8,5789	8	8	8,5546	11	11
bin6	12,3591	13	13	11,9049	12	12	11,8916	12	12
bin7	14,6374	14	14	14,641	15	15	14,6407	15	15
di25	12	13	13	14	14	14	12,5	12	12
di36	16	16	16	17	17	17	16	16	16
di64	24	24	24	22,8202	23	23	22,5294	21	22
ran2	15,887	14	14	17,0086	15	15	17,1293	14	14
ran9	12,6002	10	10	12,5476	10	10	12,7312	10	10
ran15	19,4449	16	16	19,2691	15	15	19,6681	17	17
gauss19	9	9	9	10	10	10	9	9	10
gauss34	14	14	14	15	15	15	14	14	15
irr41	14,5	14	14	15	16	16	15	15	15
molecular	14	14	14	14	15	15	13,8999	14	14
Instâncias 75%									
bin5	13	13	13	12,5	12	12	12,5	12	12
bin6	19,0979	18	18	18,6839	16	16	19,1337	18	18
bin7	27,9387	22	22	27,641	22	22	28,0985	22	22
di25	14	15	15	16	16	16	17	17	17
di36	19	19	19	19	19	19	20	19	19
di64	29	27	27	30	26	26	30	25	25
ran2	25,1308	19	19	24,0812	19	19	24,4161	19	19
ran9	25,5704	18	18	25,7943	18	18	26,2891	19	19
ran15	35,071	24	24	34,9965	22	22	35,0959	23	23
gauss19	11	11	11	11	11	11	10	10	10
gauss34	17	15	15	18	16	17	18	17	17
irr41	18	17	17	19,3333	18	18	19	19	19
molecular	21	20	20	18	17	17	19	18	18

Tabela B.9: Limites inferiores - Instâncias ANDES.

Nome	CF	PR	RL	Nome	CF	PR	RL
proa1	8	8	8	prob1	11	11	11,076
proa2	10	10	10	prob2	14	15	15
proa3	11	11	11	prob3	11	11	11
proa4	12	12	12	prob4	12	14	14
proa5	9	9	9	prob5	12	13	13
proa6	9	9	9	prob6	14	14	14
proa7	9	9	9	prob7	11	12	12
proa8	8	8	8	prob8	10	12	12
proa9	8	8	8	prob9	13	13	13
proa10	7	7	7,33	prob10	11	12	12
proc1	14	16	16	prod1	26	26	26
proc2	14	15	15	prod2	29	29	29
proc3	11	14	14	prod3	26,15	25	25
proc4	13	13	13	prod4	23	24	24
proc5	9	10	10	prod5	21,5	22	22
proc6	13	15	15	prod6	27	26	26
proc7	8,5	10	10	prod7	25	25	25
proc8	10	11	11	prod8	28,33	28	28
proc9	11	12	12	prod9	21,4516	22	22
proc10	11	12	12	prod10	20,1261	20	20
proe1	17	19	19	prof1	38	36	36
proe2	18	18	18	prof2	48	46	46
proe3	16	15	15,417	prof3	43	42	42
proe4	17,4122	18	18	prof4	33,5303	33	33
proe5	19	20	20	prof5	32	33	33
proe6	18	17	17	prof6	51	51	51
proe7	15	15	15	prof7	36,0909	36	36
proe8	15	15	15	prof8	36	36	36
proe9	18	18	18	prof9	39,2499	37	37
proe10	16	17	17	prof10	44	43	43

Tabela B.10: Limites inferiores - Instâncias ProGen.

Instância	$\overline{\Pi}(P,E)$	$\overline{\theta}(P,E)$	$\overline{\theta}(P,E)$+Cliq
Instâncias 25%			
bin5	0,12	0,322	0,151
bin6	0,302	0,826	0,615
bin7	0,766	2,737	17,282
di25	0,085	0,122	0,107
di36	0,125	0,182	0,16
di64	0,335	0,553	1,278
ran2	0,436	1,154	2,018
ran9	0,735	2,636	12,136
ran15	1,057	5,612	105,628
gauss19	0,065	0,084	0,073
gauss34	0,093	0,127	0,114
irr41	0,169	0,299	0,265
molecular	0,163	0,345	0,219
Instâncias 50%			
bin5	0,162	0,277	0,188
bin6	0,407	1,236	1,302
bin7	1,056	7,597	70,769
di25	0,081	0,1	0,1
di36	0,162	0,227	0,212
di64	0,364	0,861	3,241
ran2	0,541	2,256	8,416
ran9	1,132	7,635	56,943
ran15	1,409	13,995	426,222
gauss19	0,076	0,093	0,093
gauss34	0,151	0,204	0,184
irr41	0,192	0,372	0,339
molecular	0,185	0,327	0,28
Instâncias 75%			
bin5	0,181	0,437	0,22
bin6	0,496	2,491	6,349
bin7	1,384	19,506	522,117
di25	0,107	0,167	0,164
di36	0,179	0,531	0,309
di64	0,376	1,464	7,886
ran2	0,605	3,929	29,154
ran9	1,423	17,54	279,274
ran15	1,817	28,957	2133,004
gauss19	0,087	0,139	0,116
gauss34	0,17	0,386	0,311
irr41	0,227	0,563	0,537
molecular	0,223	0,51	0,478

Tabela B.11: Médias de tempos - Relaxações lineares.

	Instância 25%			Instância 50%			Instância 75%		
Instância	PR	SH	MD	PR	SH	MD	PR	SH	MD
bin5	0,147	0,249	0,695	0,164	-	0,776	0,206	-	0,909
bin6	0,671	-	2,305	1,224	-	5,774	2,277	513,752	13,338
bin7	10,947	-	61,785	21,444	-	65,078	56,553	-	222,641
di25	0,111	0,191	0,246	0,123	0,583	0,241	0,134	4,257	0,624
di36	0,159	1,289	0,285	0,215	28,795	0,910	0,196	19,373	1,174
di64	0,929	93,491	3,086	1,445	310,886	7,228	1,417	-	10,909
ran2	1,736	67,185	7,485	2,777	619,374	21,661	4,825	1141,865	36,119
ran9	7,792	1166,82	46,455	20,812	-	92,355	50,175	-	247,151
ran15	25,149	3455,535	176,330	49,136	-	266,357	111,143	-	1.256,894
gauss19	0,035	0,057	0,028	0,070	0,245	0,186	0,091	0,552	0,587
gauss34	0,137	1,336	0,066	0,193	15,346	0,703	0,195	4,829	1,316
irr41	0,232	17,037	0,920	0,318	26,4	1,469	0,398	1,179	2,390
molecular	0,235	13,055	0,385	0,322	45,112	1,308	0,391	2,06	1,930

Tabela B.12: Médias de tempos - Programação por restrições, *shaving* e método destrutivo.

Referências Bibliográficas

[1] D. Applegate, W. Cook. "A computational study of job-shop scheduling." *ORSA Journal of Computing* 3, pp. 149-156, 1991.

[2] E. Balas. "Machine sequencing via disjunctive graphs: An implicit enumeration algorithm." *Operations Research* 17, pp. 941-957, 1969.

[3] P. Baptiste, C. Le Pape. "Edge-finding Constraint Propagation Algorithms for Disjunctive and Cumulative Scheduling." In: *Proceedings of the fifteenth Workshop of the U.K. Planning Special Interest Group*, 1996.

[4] P. Baptiste, C. Le Pape. "Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems." *Constraints* 5, pp. 119-139, 2000.

[5] P. Brucker, S. Knust, A. Schoo, O. Thiele. "A branch-and-bound algorithm for the resource-constrained project scheduling problem." *European Journal of Operational Research* 107, pp. 272-288, 1998.

[6] P. Brucker, S. Knust. "A linear programming and constraint propagation-based lower bound for the RCPSP." *European Journal of Operational Research* 127, pp. 355-362, 2000.

[7] J.-C. Bermond, R. Corrêa, J. Yu. "Gathering algorithms on paths under interference constraints." In: *6th Conference on Algorithms and Complexity - CIAC*, Roma, 2006.

[8] J.-C. Bermond, J. Galtier, R. Klasing, N. Morales, S. Perennes. "Gathering in specific radio networks." In: *8èmes rencontres francophones sur les aspects algorithmiques de télécommunications - ALGOTEL*, Trégastel, 2006.

[9] J.-C. Bermond, J. Galtier, R. Klasing, N. Morales, S. Perennes. "Hardness and approximation of gathering in static radio networks." Relatório Técnico. *Institut National de Recherche en Informatique et en Automatique - INRIA*, França, 2005.

[10] S. Busygin. "A New Trust Region Technique for Maximum Weight Clique Problem." *Discrete Applied Mathematics* 154, pp. 2080–2096, 2006.

[11] M. Campêlo, V. Campos, R. Correa. "On the asymmetric representatives formulation for the vertex coloring problem." *Discrete Applied Mathematics*, 2007. A aparecer.

[12] M. Campêlo, V. Campos, R. Correa. "Um Algoritmo de Planos-de-Corte para o Número Cromático Fracionário de um Grafo." In: *Congresso Nacional de Matemática Aplicada e Computacional - CNMAC*, Florianópolis, 2007.

[13] M. Campêlo, R. Corrêa, N. Maculan, F. Protti. "ILP formulations for scheduling ordered tasks on a bounded number of processors." *Eletronic Notes in Discrete Mathematics* 7, pp. 166–169, 2001.

[14] M. Campêlo, R. Corrêa, N. Maculan, F. Protti. "Improved Lower Bounds for Scheduling Ordered Tasks on a Bounded Number of Processors." In: *XI Congresso Latino-Americano de Investigación Operativa - CLAIO*, Concepcion. Ata de Trabalhos, 10 pag, 2002.

[15] J. Carlier, E. Pinson. "A practical use of Jackson's preemptive schedule for solving the job-shop problem." *Annals of Operations Research* 26, pp. 269-287, 1990.

[16] N. Christofides, R. Alvarez-Valdés, J. M. Tamarit. "Project scheduling with resource constraints: A branch and bound approach." *European Journal of Operational Research* 29, pp. 262-273, 1987.

[17] S. Demassey, C. Artigues, P. Michelon. "Constraint-Propagation-Based Cutting Planes: An application to the Resource-Constrained Project Scheduling Problem." *INFORMS Journal on Computing* 1, vol. 17, pp. 52-65, 2005.

[18] E. Demeulemeester, W. Herroelen. "New benchmark results for the resource-constrained project scheduling problem." *Management Science* 43, pp. 1485-1492, 1997.

[19] R. W. Deming. "Acyclic orientations of a graph and chromatic and independence numbers." *Journal of Combinatory Theory*, Series B 26, pp. 101-110, 1979.

[20] R. M. V. Figueiredo, V. C. Barbosa, N. Maculan. "New 0-1 integer formulations of the graph coloring problem." In: *XI Congresso Latino Iberoamericano de Investigación de Operaciones - CLAIO*, 10 pag, 2002.

[21] Y. A. M. Frota. *"Estudo Computacional de Algoritmos Exatos para Coloração de Grafos."* Dissertação de Mestrado. Universidade Federal do Ceará, Fortaleza, 2003.

[22] M.R. Garey, D.S. Johnson. *"Computers and Intractability - A Guide to the Theory of NP-Completeness."* W.H. Freeman and Company, 1979.

[23] J. Kitajima, B. Plateau. "Modelling parallel program behaviour in ALPES." *Information and Software Technology* 36(7), pp. 457–464, 1994.

[24] R. Klein, A. Scholl. "Computing lower bound by destructive improvement: An application to resource-constrained project scheduling." *European Journal of Operational Research* 112, pp. 322-346, 1999.

[25] R. Kolisch, A. Sprecher, A. Drexl. "Characterization and generation of a general class of resource-constrained project scheduling problems." In: *Management Science* 41, pp. 1693-1703, 1995.

[26] I. S. Lima, M. Campelo, R. Correa. "Limites para a altura mínima de extensões induzidas de posets." In: *XXXIX Simpósio Brasileiro de Pesquisa Operacional - SBPO*, Fortaleza, 2007.

[27] R. H. Mohring, A. Schultz, F. Stork, M. Uetz. "Solving project scheduling problems by minimum cut computations." *Management Science* 49, pp. 330-350, 2003.

[28] A. Pritsker, L. Watters, P. Wolfe. "Multi-project scheduling with limited resources: A zero-one programming approach." In: *Management Science* 16, pp. 93-108, 1969.

[29] S. Skiena, "Partial Orders." *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Reading, MA: Addison-Wesley, pp. 203–209, 1990.

[30] A. Sprecher, R. Kolisch, A. Drexl. "Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem." *European Journal of Operational Research* 80, pp. 94-102, 1995.

[31] V. Valls, F. Ballestin, S. Quintanilla. "Justification and RCPSP: A technique that pays." *European Journal of Operational Research* 165, pp.375-386, 2005.