Um algoritmo *branch-and-bound* para o problema de programação de projetos com custo de disponibilidade de recursos e múltiplos modos

Denise Sato Yamashita
Reinaldo Morabito

Resumo

Em um estudo anterior (YAMASHITA; MORABITO, 2007a), foi proposto um algoritmo exato para resolver o problema de programação de projetos com custo de disponibilidade de recursos, considerando múltiplos modos de execução para as atividades. O algoritmo é uma adaptação de outro algoritmo exato da literatura para o caso particular do problema em que só existe um modo de executar as atividades. No presente estudo, é proposto um novo algoritmo exato baseado no método branch-and-bound para tratar do problema com múltiplos modos de execução. Como o problema é NP-difícil, o algoritmo é viável computacionalmente apenas para resolver exemplares de tamanho moderado. Diversos testes computacionais utilizando o programa gerador Progen da literatura foram realizados para comparar o desempenho do algoritmo proposto com o algoritmo anterior e também com o aplicativo CPLEX. Os resultados obtidos mostram que a versão atual do algoritmo é competitiva com os outros métodos, e estimulam a pesquisa e o desenvolvimento de versões mais elaboradas deste algoritmo.

Palavras-chave: *Programação de projetos. Múltiplos modos de execução. Custo de disponibilidade de recursos. Algoritmo branch-and-bound.*

1 *Introdução*

Problemas de programação de projetos (*project scheduling*) envolvem programar ao longo do tempo atividades que necessitam de recursos para serem concluídas. Esta programação também deve levar em conta restrições de precedência entre as atividades, isto é, certas atividades só podem ser iniciadas após outras terem sido terminadas. Exemplos de problemas de programação de projetos aparecem em diversas áreas (MODER et al., 1983; POLLACK-JOHNSON; LIBERATORE, 1998), tais como construção civil (TAKAMOTO et al., 1995), manufatura (NEUMANN et al., 2002), telecomunicações, desenvolvimento de *software*/computação, manutenção de aeronaves (GEMMILL; EDWARDS, 1999), entre outras. Por exemplo, na construção de uma ponte, a aquisição de material, a contratação de mão-de-obra e as diversas etapas que compõem a construção, como preparação do terreno, fundação, estruturas de pilares, vigas e lajes, pavimentação, são algumas das atividades do projeto. Essas atividades têm uma duração prevista em função dos recursos alocados e, para serem executadas, também estão sujeitas a relações de precedência. A execução das atividades consome diferentes recursos, como máquinas, mão-de-obra, dinheiro, etc.

Alguns problemas de programação de projetos permitem que as atividades sejam executadas de modos alternativos (múltiplos modos de execução). Por exemplo, uma atividade que requer 10 períodos de tempo para ser executada por 1 trabalhador, poderia alternativamente ser realizada em 4 períodos de tempo utilizando 2 trabalhadores. O objetivo da programação de projetos é determinar um programa (*schedule*) das atividades de forma a otimizar um ou mais critérios (e.g., minimizar custos e data de entrega do projeto), obedecendo às relações de precedência entre as atividades e a disponibilidade de recursos. Por exemplo, o problema de programação de projetos com recursos limitados e múltiplos modos de execução (PPPRLMM – *multi-mode resource constrained project scheduling problem*) consiste em minimizar o instante de término da última atividade do projeto (*makespan*) considerando múltiplos modos de execução e respeitando as relações de precedência entre as atividades e a disponibilidade de recursos do projeto. Note-se que a consideração de múltiplos modos de execução das atividades torna o

problema de programação de projetos mais realista e abrangente, mas resulta num aumento de variáveis de decisão do problema.

O presente estudo trata de um problema determinístico de programação de projetos com custo de disponibilidade de recursos e com múltiplos modos de execução para as atividades. Este problema é conhecido na literatura como problema de custo de disponibilidade de recursos com múltiplos modos de execução (PCDRMM – *multi-mode resource availability cost problem*). No PCDRMM, os recursos disponíveis para o projeto são variáveis de decisão, sendo que existe um custo associado à disponibilidade de cada recurso. O objetivo é definir um programa de execução das atividades, de modo a minimizar o custo total de alocação dos recursos que estarão disponíveis durante o projeto, respeitando as restrições de precedência entre as atividades e o prazo de entrega pré-estabelecido do projeto.

Do ponto de vista da teoria de complexidade, o PCDRMM é um problema difícil de ser resolvido otimamente, dado que o seu caso particular, em que todas as atividades têm apenas um modo de execução, denominado problema de custo de disponibilidade de recursos (PCDR - *resource availability cost problem*) é NP-difícil (MÖHRING, 1984). Poucos trabalhos foram encontrados na literatura estudando o PCDRMM. Em particular, não temos conhecimento de outros trabalhos apresentando algoritmos exatos para resolver o PCDRMM. Uma exceção é o método proposto no estudo anterior (YAMASHITA; MORABITO, 2007a; YAMASHITA; MORABITO, 2007b), aqui chamado de algoritmo DA (DEMEULEMEESTER, 1995 adaptado), que é uma adaptação do algoritmo exato desenvolvido por Demeulemeester (1995) para resolver o PCDR, baseado na solução de uma seqüência de PPPRLMMs.

Resumidamente, o primeiro passo do algoritmo DA consiste em determinar um limitante inferior para a quantidade de recursos disponíveis. Uma vez fixada a disponibilidade de recursos, um PPPRLMM é resolvido, com o objetivo de descobrir se existe um programa que termine dentro da data de entrega e respeite as restrições de disponibilidade de recursos e precedência das atividades. Se tal programa existir, esta disponibilidade de recursos é a solução ótima do PCDRMM. Caso não exista um programa que termine antes da data de entrega do projeto, então a disponibilidade de um dos tipos de recursos é aumentada em uma unidade, e o PPPRLMM relativo a esta disponibilidade de recursos é resolvido. Este processo de aumento gradual de recursos e subseqüente solução de um PPPRLMM (análise marginal) é repetido enquanto não for possível encontrar um programa que respeite a data de entrega. O método foi apresentado para gerar curvas de *tradeoff* entre o custo e o prazo de um projeto, e é viável computacionalmente para resolver apenas problemas de tamanho moderado. Estas curvas de *tradeoff* auxiliam o tomador de decisão na difícil tarefa de balancear os custos e datas de entrega de um projeto.

Além do algoritmo proposto por Demeulemeester (1995), são poucos os trabalhos na literatura que abordam o PCDR. Möhring (1984) propôs um algoritmo exato baseado em teoria de grafos e motivado por projetos de construção de pontes. Experimentos computacionais realizados em problemas testes destes projetos indicaram que o algoritmo de Demeulemeester (1995) é mais rápido do que o de Möhring (1984). Rangaswamy (1998) também propôs um algoritmo exato para o PCDR baseado no método *branch-and-bound*, com resultados computacionais competitivos em relação aos anteriores nos mesmos problemas testes. A comparação destes resultados não é conclusiva devido a diferentes plataformas computacionais utilizadas nos experimentos. Limitantes inferiores e superiores para o valor da solução do PCDR, baseados em relaxação Lagrangiana e métodos de geração de colunas, foram propostos em Drexl e Kimms (2001). Recentemente, um método heurístico para resolver o PCDR foi proposto em Yamashita et al. (2006).

No presente trabalho, é apresentado um novo algoritmo exato baseado no método *branch-and-bound* para tratar do PCDRMM. A versão atual do algoritmo é um ponto de partida para o desenvolvimento de métodos exatos mais efetivos para resolver o PCDRMM. Diversos testes computacionais utilizando o programa gerador Progen da literatura (KOLISCH; SPRECHER, 1996) foram realizados para comparar o desempenho do algoritmo com o algoritmo DA e também com o aplicativo GAMS/CPLEX (BROOKE et al., 1997) para resolver uma formulação inteira mista do problema. Os resultados obtidos mostram que o algoritmo é competitivo com os outros métodos. Este artigo está organizado da seguinte forma: na seção 2, o PCDRMM é brevemente descrito e formulado matematicamente; na seção 3, o algoritmo proposto é apresentado; na seção 4, os resultados computacionais obtidos com a aplicação dos métodos são comparados e analisados; e, na seção 5, são discutidas as considerações finais deste trabalho e perspectivas para pesquisa futura.

2 *Definição e modelagem do PCDRMM*

O material desta seção está baseado em Yamashita e Morabito (2007a) e é aqui apresentado para que o texto do artigo fique autocontido, facilitando a compreensão do leitor. Considere-se um projeto com n atividades em que as atividades 1 e n são atividades artificiais, indicando o início e o fim do projeto, respectivamente. As relações de precedência entre as atividades são definidas pelo conjunto H, que consiste de pares (h, j), tal que se (h, j)

$\in$ H, então a atividade h precede a atividade j. Cada atividade j pode ser executada em um de M_j possíveis modos. Quando executada no modo i, a atividade j tem duração d_{ji} e requer r_{jik} unidades do recurso do tipo k (k = 1,...,m) durante a sua execução. Estes recursos são chamados renováveis porque, após serem utilizados, ficam disponíveis novamente para uso no período seguinte. Exemplos de tais recursos são trabalhadores e máquinas; contra exemplos são matérias-primas e dinheiro. O tempo do projeto é discretizado em instantes de tempo t = 0, 1, 2,..., D, em que D é a data de entrega do projeto. Admite-se que d_{ji} e r_{jik} sejam números inteiros e define-se $C_k(a_k)$ como uma função de custo não decrescente associada à disponibilidade a_k do recurso do tipo k.

Nos trabalhos da literatura que abordam tanto o PCDR quanto o PCDRMM, os experimentos computacionais são realizados considerando $C_k(a_k)$ como uma função linear, pelo fato de em geral utilizarem métodos de programação linear inteira. Neste artigo, o mesmo se aplica; entretanto, é importante observar que o algoritmo aqui proposto também pode ser utilizado (sem modificações) para tratar de problemas em que a função $C_k(a_k)$ é não-linear. As variáveis do problema são as variáveis inteiras relacionadas com a disponibilidade de recurso a_k e com a determinação dos instantes de término e modo de execução das atividades. Em certas aplicações, a disponibilidade de recurso a_k pode variar com o instante de tempo t, nestes casos, denotada por a_{kt}. Entretanto, neste trabalho, admite-se que a variável a_k é constante em relação ao tempo. Um modelo matemático para o PCDRMM, baseado no modelo proposto por Talbot (1982) para o PPPRLMM, é formulado a seguir. Neste modelo, as variáveis de decisão são dadas por:

a) $x_{jit}\begin{cases}1, \text{ se a atividade } j = 1,...,n \text{ é executada}\\ \text{no modo } i = 1,...,M_j \text{ e termina no instante}\\ t = 1,...,D.\\ 0, \text{ caso contrário}\end{cases}$;e

b) a_k = quantidade de recurso do tipo k disponível ao longo do projeto, k = 1,..., m (admitida como um número inteiro).

Os dados de entrada adicionais do problema são:

a) LF_j = instante de término mais tarde que a atividade j pode ser completada, sem violar a data de entrega do projeto, obedecendo às relações de precedência entre as atividades, sem considerar as restrições de recursos. Este valor pode ser obtido ao fazer a programação das atividades de forma regressiva, começando da última atividade artificial (n), que é programada no instante D. As demais atividades são então programadas iterativamente, sempre que todos os seus sucessores já tenham sido programados; e

b) EF_j = instante de término mais cedo que a atividade j pode ser completada, obedecendo às restrições de precedência entre as atividades, sem considerar as restrições de recursos. Este valor pode ser obtido ao fazer a programação das atividades de forma progressiva, começando da primeira atividade artificial (1), que é programada no instante 0. As demais atividades são então programadas o mais cedo possível, após todos os seus predecessores terem sido programados.

O modelo é definido por:

$$\text{Minimizar } \sum_{k=1}^{m} C_k(a_k) \quad (1)$$

$$\sum_{i=1}^{M_j}\sum_{t=EF_j}^{LF_j} x_{jit} = 1,\ j = 1,...,n \quad (2)$$

$$\sum_{i=1}^{M_h}\sum_{t=EF_h}^{LF_h} t \cdot x_{hit} \le \sum_{i=1}^{M_j}\sum_{t=EF_j}^{LF_j} (t - d_{ji}) \cdot x_{jit} \quad (3)$$

j = 1,...,n, e para todo h, tal que (h, j) $\in$ H

$$\sum_{j=1}^{n}\sum_{i=1}^{M_j}\sum_{b=t}^{t+d_{ji}-1} r_{jik}\, x_{jib} \le a_k \quad k = 1,...,m \quad t = 0,...,D \quad (4)$$

$$x_{jit} \in \{0,1\}\ j = 1,...,n,\ t = 1,...,D,\ i = 1,...,M_j \quad (5)$$

$$a_k \ge 0,\ a_k \in Z,\ k = 1,...,m \quad (6)$$

Nas formulações 1-6 acima, a formulação 1 refere-se à função objetivo do PCDRMM que consiste em minimizar o custo total de alocação de recursos do projeto. A restrição 2 assegura que cada atividade é executada exatamente uma vez, e em apenas um modo de execução. A restrição 3 garante que as relações de precedência entre as atividades são obedecidas, e a restrição 4 assegura que a soma da quantidade de recursos utilizados pelas atividades num determinado instante de tempo t não ultrapasse a quantidade de recursos disponíveis para o projeto. A restrição 5 define as variáveis de decisão binárias e a restrição 6 garante que a quantidade de recursos alocada seja não-negativa e inteira. Observe-se que a restrição que assegura que o projeto termine antes da data de entrega D é considerada implicitamente no modelo, uma vez que nele as variáveis binárias x_{jit} sempre têm t $\le$ D.

A título de ilustração, no Apêndice 1, apresentamos um simples exemplo do PCDRMM descrito em Yamashita e Morabito (2007a).

3 Algoritmo *branch-and-bound* para o PCDRMM

O procedimento aqui descrito para gerar a árvore de busca (enumeração) foi inspirado em trabalhos que propuseram algoritmos *branch-and-bound* para resolver

o PPPRLMM (PATTERSON et al., 1989; TALBOT, 1982; SPRECHER, 1994; SPRECHER; DREXL, 1998; BRUCKER et al., 1999; HARTMANN, 2000; HARTMANN; KOLISCH, 2000; KOLISCH; PADMAN, 2001). A idéia do procedimento é programar uma atividade em cada nível da árvore e, associada a esta atividade, determinar seu instante de início e seu modo de execução. Desta forma, a árvore de busca terá no máximo n níveis, resultando num programa completo ao atingir o n-ésimo nível.

No algoritmo proposto, aqui chamado simplesmente de algoritmo BB (*branch-and-bound*), a árvore de busca é percorrida em profundidade, existindo basicamente duas regras para interromper a exploração de um ramo da árvore: a primeira regra é baseada no cálculo do custo do projeto e a segunda, no instante de seu término.

Na primeira regra, à medida que as atividades vão sendo programadas em cada nível da árvore, calcula-se a estimativa de custo do projeto, baseado na utilização de recursos das atividades que já foram programadas até então. Se este custo parcial exceder o melhor custo já obtido, então não é preciso continuar examinando este ramo da árvore. Na segunda regra, a cada nível da árvore, calcula-se o tempo de término do projeto completando o programa parcial obtido neste nível com as atividades que ainda não foram programadas. Estas atividades que ainda não pertencem ao programa parcial são programadas o mais cedo possível, utilizando o modo de execução que tem a menor duração, e obedecendo às relações de precedência entre as atividades. Se este programa exceder a data de entrega do projeto, então não é preciso continuar a explorar este ramo da árvore.

Um pseudocódigo do algoritmo é descrito na Figura 1 e um fluxograma resumido do algoritmo é apresentado no Apêndice 2. No Passo 1, o parâmetro g refere-se ao nível da árvore de busca; act_g refere-se à atividade candidata a ser programada no nível g; s_{act_g} é o instante de início da atividade candidata a ser programada no nível g; $modo_{act_g}$ é o modo de execução da atividade act_g no nível g; LS é custo da melhor solução conhecida para o problema; e PP_g é o programa parcial no nível g, que consiste das atividades que já foram programadas em níveis anteriores a g. O algoritmo começa programando a atividade artificial 1 no instante 0 e a inclui no programa parcial.

No Passo 2, são calculadas as atividades que são viáveis com respeito às relações de precedência, isto é, todas as atividades que a precedem já se encontram em PP_g. Estas atividades são inseridas no conjunto das atividades elegíveis CAE_g. Se a atividade n pertencer a CAE_g, significa que o programa está completo e, portanto, deve-se fazer

Passo 1: Atribua valores iniciais
g = 1; act_1 = 1 $modo_{act1}$ = 1; s_{act_1} = 0; LS = infinito; PP_1 = {1};

Passo 2: Calcule atividades elegíveis
g = g + 1; $PP_g = PP_{g-1} \cup \{act_{g-1}\}$; CAE_g = conjunto das atividades que não pertencem a PP_g, mas que possuem todos os seus predecessores em PP_g. Se n $\in$ CAE_g, então guarde a solução corrente e vá para o Passo 6.

Passo 3: Selecione a próxima atividade
Se todas as atividades de CAE_g já foram testadas, então vá para o Passo 6; caso contrário, selecione uma atividade não testada, $act_g \in CAE_g$.

Passo 4: Selecione um modo de execução
Se todos os modos da atividade act_g já foram testados, então marque act_g como testada:
Se para todos os modos de act_g não foi possível encontrar ao menos um valor s_{act_g} viável, então vá para o Passo 6; caso contrário, vá para o Passo 3.
Caso contrário, selecione o próximo modo $modo_{act_g} \in M_{act_g}$ ainda não testado.
Calcule o instante de tempo mais cedo, viável em termos de relação de precedência e tal que $s_{act_g} \geq s_{act_{g-1}}$. Vá para o Passo 5.

Passo 5: Calcule os instantes de início
Calcule o instante de término mais cedo do projeto a partir de PP_g, considerando que act_g é programada no instante s_{act_g} e as demais atividades que não pertencem a PP_g são programadas num instante de tempo maior ou igual a s_{act_g}. Se essa estimativa ultrapassa a data de entrega, então marque o modo corrente como "testado" e retorne ao Passo 4.
Calcule o custo do projeto ao programar act_g em s_{act_g}($custo_g$). Se $custo_g$ for superior a LS, então $s_{act_g} = \min\{f_j, j \in PP_g \mid f_j > s_{act_g}$ e $custo_g < LS\}$, e repita o Passo 5.
Programe act_g em s_{act_g}, e vá para o Passo 2.

Passo 6: *Backtracking*
g = g – 1.
Se g = 0, então pare.
Se g > 0, então $s_{act_g} = s_{act_g} + 1$, e vá para o Passo 5.

Figura 1. Algoritmo BB.

um *backtrack* no Passo 6. Caso contrário, o algoritmo prossegue para o Passo 3.

No Passo 3, seleciona-se uma atividade act_g pertencente a CAE_g, que ainda não foi testada. Se todas as atividades elegíveis foram testadas, então é feito um *backtrack* (Passo 6), caso contrário, prossegue-se para o Passo 4.

No Passo 4, inicialmente verifica-se se todos os modos de execução já foram testados. Se este for o caso, então a atividade act_g é marcada como testada, e verifica-se se foi possível encontrar ao menos uma combinação de modo de execução e instante de início para a atividade act_g. Se act_g não puder ser programada em PP_g em nenhum modo ou instante de início sem tornar o programa parcial inviável, então não é possível prosseguir com este programa e um *backtrack* pode ser realizado. Se existir ao menos um modo e um instante s_{act_g} para programar act_g, então o programa retorna ao Passo 3. Se nem todos os modos de execução do Passo 4 tiverem sido testados, seleciona-se um modo de execução ainda não testado para a atividade act_g, e calcula-se o instante de início mais cedo (EST_{act_g}) que a atividade pode ser programada em PP_g, respeitando-se as relações de precedência, e ignorando-se as restrições de limitação de recursos. Note-se que, para evitar redundâncias, $s_{act_g} \geq \max\{s_{act_{g-1}}, EST_{act_g}\}$, pois o programa em que $s_{act_g} < s_{act_{g-1}}$ já foi (ou será) explorado em outro ramo da árvore de busca.

Definido o modo de execução e um possível instante de início para a atividade, no Passo 5, verifica-se se é possível completar o programa parcial PP_g programando-se a atividade act_g no instante s_{act_g}, e as demais atividades que ainda não estão em PP_g, o mais cedo possível a partir de s_{act_g}, considerando-se somente as relações de precedência entre as atividades. Desta forma, obtém-se um limitante inferior para o instante de término do projeto para PP_g. Se este limitante inferior ultrapassar a data de entrega, então é preciso escolher outro modo de execução para a atividade; caso contrário, verifica-se se existe um aumento de custo ao programar act_g. Seja $custo_g$ a estimativa de custo do projeto, baseado na utilização de recursos das atividades no programa parcial PP_g após incluir a atividade act_g em PP_g no instante s_{act_g}. Se o custo do projeto for superior a LS, então o instante de início de act_g é aumentado para $s_{act_g} = \min\{f_j, j \in PP_g \mid f_j > s_{act_g}$ e $custo_g < LS\}$. Dessa forma, avançamos no tempo até que uma ou mais atividades de PP_g, que estão sendo executadas no instante s_g, terminem de ser processadas, e liberem recursos para que act_g possa ser programada sem exceder LS. Retorna-se, então, ao início deste passo. Se for possível encontrar um instante de tempo em que act_g é programada sem exceder LS e D, então a atividade é programada e prossegue-se para outro nível da árvore.

No Passo 6, retrocede-se a busca para o nível anterior da árvore. Se o nível 0 tiver sido atingido, então o algoritmo pára, uma vez que a busca está completa. Caso contrário, aumenta-se o instante de início da atividade act_g (atividade examinada neste nó) em uma unidade e prossegue-se para o Passo 5.

A seguir, o desempenho do algoritmo BB é comparado com o algoritmo exato DA e com o aplicativo GAMS/CPLEX para resolver o modelo 1-6 da seção 2. Conforme mencionado antes, o algoritmo DA é uma adaptação do algoritmo exato desenvolvido por Demeulemeester (1995) para resolver o PCDR, baseado na solução de uma seqüência de PPPRLMMs. O algoritmo utilizado dentro do algoritmo DA para resolver os PPPRLMMs envolvidos foi proposto por Sprecher e Drexl (1998). Não dispomos do código original de Sprecher e Drexl (1998); portanto, o desempenho do código original pode ser melhor do que o desempenho da nossa implementação. Além disso, a implementação do algoritmo DA poderia ser melhorada utilizando estruturas de dados mais eficientes como, por exemplo, listas dinâmicas, o que não foi feito neste trabalho.

4 *Resultados computacionais*

Para analisar o desempenho do algoritmo BB em relação ao algoritmo DA e o aplicativo GAMS/CPLEX, diversos experimentos computacionais foram realizados utilizando um microcomputador PC Pentium 4, 2,8 GHz, 1,00 GB RAM. Todos os códigos foram implementados na linguagem C++. Os problemas testes para o PCDRMM foram gerados pelo programa Progen (KOLISCH; SPRECHER, 1996), que é um gerador amplamente utilizado na literatura para problemas de programação de projetos com recursos limitados.

Dois importantes parâmetros do Progen são a complexidade de rede (*network complexity* – NC) e o fator de recursos (*resource factor* – RF). A complexidade da rede NC reflete o número médio de sucessores imediatos de uma atividade. O fator de recursos varia entre [0,1] e reflete a densidade dos diferentes tipos de recursos que uma atividade requer para ser processada. Por exemplo, se RF = 1, então cada atividade requer todos os m tipos de recursos, enquanto se RF = 0, as atividades não precisam de qualquer tipo de recurso.

É também necessário determinar uma data de entrega para o projeto. Drexl e Kimms (2001) sugeriram calcular a data de entrega como uma função do caminho crítico do projeto: $D = DF \max\{EF_n,\}$, no qual DF é o fator de data de entrega do projeto e EF_n é o instante de término mais cedo do projeto. Para cada problema teste, os custos c_k são gerados por uma distribuição uniforme U(0, 10]. Os valores de parâmetros utilizados para gerar os problemas testes foram:

a) n = 10, m = 4, e M_j = 3 (j = 1,..,n);
b) RF: 0,25, 0,5, 0,75 e 1,0;
c) NC: 1,5, 1,8 e 2,1; e
d) DF: 1,0, 1,2, e 1,4.

Para cada combinação de RF, NC e DF, foi gerado um problema teste, totalizando 4 x 3 x 3 = 36 problemas testes. Inicialmente foi realizado um breve experimento computacional com estes exemplos para avaliar o impacto dos parâmetros do aplicativo GAMS (versão 2.0.10.0) e CPLEX (versão 7) no desempenho do algoritmo de solução usado por este pacote. Para realizar o experimento, foram utilizadas algumas sugestões contidas em ILOG (2007), documento disponibilizado pela própria empresa que desenvolve o CPLEX. Foram comparadas basicamente 4 configurações:

a) **configuração 1:** utiliza as configurações padrões (*default*) do CPLEX;
b) **configuração 2:** antes de ramificar a árvore de busca, é possível realizar uma exploração (*probing*) que, em geral, consiste em fixar o valor de uma variável em seu limitante superior ou inferior, e analisar a conseqüência desta fixação. Este tipo de exploração pode ser poderosa, entretanto, pode ter um alto custo computacional. Esta configuração faz uma exploração completa. Note-se que nas outras configurações aqui estudadas, o valor do parâmetro de *probing* utilizado é o valor padrão do CPLEX, no qual o próprio aplicativo decide a quantidade de exploração automaticamente;
c) **configuração 3:** nesta configuração, todos os parâmetros estão no valor padrão do CPLEX, com exceção do parâmetro *mipemphasis*, ao qual foi atribuído o valor "1". Nesta configuração, o aplicativo tende a dar mais ênfase na busca por uma solução viável, ao invés de buscar a solução ótima. Normalmente, se este valor não for especificado, tende-se a dar maior ênfase ao encontro da solução ótima.
d) **configuração 4:** nesta configuração, procura-se escolher valores para os parâmetros que tendem a gerar cortes mais agressivos. No valor *default*, a geração destes cortes só é efetuada se estiver ajudando o algoritmo. Na configuração 4, os parâmetros que definem a quantidade de cortes de arredondamento inteiro misto, cortes de clique, cortes de cobertura, cortes disjuntivos, cortes de Gomory fracionais e cortes de cobertura de limitantes superiores generalizados, foram escolhidos para gerar cortes de forma agressiva.

Em geral, não se notou uma diferença muito grande entre as configurações testadas. O tempo médio gasto para resolver os 36 problemas testes utilizando a configuração padrão foi de 18,61 segundos. A configuração que resultou no menor tempo computacional médio foi a configuração 3, com tempo médio de 17,55 segundos, e a configuração com maior tempo computacional foi a configuração 2 que levou 18,73 segundos. Apesar da configuração 3 ter apresentado, em média, um tempo computacional um pouco menor do que as demais, ela foi melhor do que a configuração padrão em apenas 17 dos 36 problemas testados. As configurações 2 e 4 apresentaram tempo computacional menor do que a configuração padrão em 19 e 21 problemas testes, respectivamente. Uma vez que as configurações do aplicativo GAMS/CPLEX não apresentaram grande diferença de desempenho na solução do modelo para este conjunto de exemplares, optamos por utilizar nos testes a seguir os resultados obtidos pela configuração padrão.

Os algoritmos foram executados até atingir um limite de tempo computacional fixado arbitrariamente em 600 segundos. A Tabela 1 compara os tempos computacionais (em segundos) dos três métodos de solução, de acordo com o fator de data de entrega. É importante observar que nos problemas testes, nos quais DF = 1,4, houve dois exemplos em que o algoritmo DA excedeu o limite de tempo. Nos demais problemas testados, os três métodos terminaram antes do limite de tempo. Note-se que, para DF = 1,2 e DF = 1,4, em que a data de entrega é mais folgada, o tempo computacional total consumido pelo aplicativo GAMS/CPLEX foi menor do que os tempos consumidos pelos algoritmos BB e DA, enquanto nos problemas em que a data de entrega é mais apertada (DF = 1,0), o algoritmo BB demandou o menor tempo total.

O algoritmo BB apresentou um tempo computacional menor do que GAMS/CPLEX em 28 dos 36 problemas testes, e foi mais rápido do que o algoritmo DA em 27 dos 36 problemas testes. Em geral, o algoritmo BB teve um desempenho pior do que os outros métodos em problemas com RF = 0,25, em que cada atividade requer poucos tipos de recursos. A Tabela 2 mostra os tempos computacionais médios consumidos pelo algoritmo BB e pelo aplicativo GAMS/CPLEX para resolver os 36 problemas testes, de acordo com DF e RF. Com exceção dos problemas em que RF = 0,25, em geral, o algoritmo BB tem um desempenho melhor do que GAMS/CPLEX, especialmente para RF = 1,0, em que cada atividade requer todos os tipos de recursos. É interessante notar que os tempos computacionais dos três métodos aumentam à medida que o fator de data de entrega cresce. Isso se deve, provavelmente, ao

Tabela 1. Tempos computacionais médios (em segundos) em função de DF.

DF	GAMS/CPLEX	BB	DA
1,0	0,07	0,05	0,40
1,2	0,30	0,56	0,81
1,4	1,48	2,94	9,57

Tabela 2. Tempos computacionais médios (em segundos) em função de RF e DF.

	DF = 1,0		DF = 1,2		DF=1,4	
RF	**GAMS/CPLEX**	**BB**	**GAMS/CPLEX**	**BB**	**GAMS/CPLEX**	**BB**
0,25	0,60	2,05	1,09	23,24	2,38	125,55
0,5	0,58	0,12	2,97	0,42	8,12	3,65
0,75	0,96	0,26	2,41	2,63	12,31	10,05
1,00	1,06	0,06	7,90	0,52	48,20	1,91

aumento no número de variáveis dos problemas testados, com o aumento de D.

A Tabela 3 classifica problemas testes em termos do número médio de sucessores imediatos da rede (NC) e mostra os tempos computacionais médios do algoritmo BB e do aplicativo GAMS/CPLEX. Nos dois métodos, os tempos computacionais diminuem à medida que a complexidade da rede aumenta, mas no caso do algoritmo BB a redução é mais significativa. A título de ilustração, a Tabela 4 apresenta alguns resultados adicionais gerados pelo algoritmo BB ao resolver um dos 36 problemas testes. Note-se que o problema foi resolvido para três datas de entrega distintas, e que o custo do projeto diminui à medida que as datas de entrega ficam mais folgadas. Como já observado na Tabela 2, os tempos computacionais e o número de nós gerados pelo algoritmo BB aumentam à medida que o fator de data de entrega cresce.

Tabela 3. Tempos computacionais médios (em segundos) em função de NC.

NC	GAMS/CPLEX	BB
1,5	10,42	28,51
1,8	7,89	12,70
2,1	3,82	1,41

Tabela 4. Resultados de um dos 36 problemas testes utilizando três fatores de data de entrega distintos.

	DF = 1,0	DF = 1,2	DF = 1,4
Valor do custo	403,05	321,72	294,06
Data de entrega	11	13	15
Número de nós	25677	147063	590457
Programas completos	9	9	17
Tempo (s)	0,41	2,34	9,37

5 *Considerações finais*

Neste trabalho foi apresentado um algoritmo exato baseado no método *branch-and-bound* (algoritmo BB) para resolver o problema PCDRMM. O algoritmo é relativamente simples e pode ser visto como um ponto de partida para o desenvolvimento de métodos exatos mais efetivos para resolver o PCDRMM. O desempenho do algoritmo foi comparado com o algoritmo exato apresentado em Yamashita e Morabito (2007a) (algoritmo DA) e com o aplicativo GAMS/CPLEX para resolver a formulação 1-6 do PCDRMM. Os resultados computacionais mostraram que o algoritmo BB é computacionalmente viável para resolver problemas de tamanho moderado. Em particular, este algoritmo é tratável apenas em situações em que é razoável considerar a disponibilidade de cada tipo de recurso em unidades inteiras razoavelmente pequenas.

Foram utilizados 36 problemas testes com diferentes características, gerados aleatoriamente pelo programa Progen da literatura, para avaliar o desempenho dos métodos. O algoritmo BB resolveu otimamente todos os problemas testes, demandando tempos computacionais menores do que GAMS/CPLEX em 28 dos 36 problemas, e foi mais rápido do que o algoritmo DA em 27 dos 36 problemas. Em particular, o algoritmo BB teve um desempenho superior aos demais em problemas em que as atividades requerem mais tipos de recursos (e.g. RF = 1,0), as datas de entrega do projeto são mais apertadas (e.g. DF = 1,0) e a complexidade da rede é maior (e.g. NC = 2,1). Entretanto, nos problemas com RF = 0,25 (cada atividade requer poucos tipos de recursos), o algoritmo BB teve um desempenho bastante inferior em relação aos outros dois métodos testados. Apesar disso, acreditamos que o algoritmo proposto é um bom ponto de partida para o desenvolvimento de algoritmos exatos mais eficazes, além de ser uma alternativa à utilização de aplicativos comerciais de otimização de última geração.

Como pesquisa futura, pretende-se desenvolver versões mais elaboradas do algoritmo BB, por exemplo, incorporando procedimentos para resolução mais eficiente de problemas de programação linear em cada nó da árvore de busca para obter limitantes inferiores. Outras perspectivas interessantes para pesquisa futura são: estudar a aplicação do algoritmo em problemas com funções de custo não lineares, estender o algoritmo para tratar de problemas com recursos não renováveis, e desenvolver heurísticas e metaheurísticas para o PCDRMM para resolver problemas de maior porte.

A branch-and-bound algorithm for the resource constrained project scheduling problem with resource availability cost and multiple modes

Abstract

In a recent study (YAMASHITA; MORABITO, 2007a), it was proposed an exact algorithm to solve problems of resource-constrained project scheduling with resource availability costs under multiple modes of execution. That algorithm is an adaptation of another exact algorithm recorded in the literature for the particular case where there is only a single mode for executing the tasks. In the present study, we propose a new exact algorithm based on the branch and bound method to deal with multiple performing modes problem. Since the problem is NP-hard, the algorithm is computationally viable only for problems of moderate size. Numerous computational tests using the generator ProGen were run to compare the performance of the proposed algorithm with the former algorithm and with the CPLEX software. The results show that the proposed version of the algorithm is competitive with the other methods and encourage further research for the development of more elaborate versions of this algorithm.

Keywords: *Project scheduling. Multiple modes of execution. Resource availability cost. Branch and bound algorithm.*

Referências bibliográficas

BROOKE, A.; KENDRIK, D.; MEERAUS, A. **GAMS, Sistema Geral de Modelagem Algébrica**. Edgard Blucher Ltda, 1997.

BRUCKER, P.; DREXL, A.; MÖHRING, R.; NEUMANN, K.; PESCH, E. Resource constrained project scheduling: Notation, classification, models, and methods. **European Journal of Operational Research**, v. 112, n. 1, p. 3-41, 1999.

DEMEULEMEESTER, E. Minimizing resource availability costs in time-limited project networks. **Management Science**, v. 41, n. 10, p. 1590-1598, 1995.

DREXL, A.; KIMMS, A. Optimization guided lower and upper bounds for the resource investment problem. **Journal of the Operational Research Society**, v. 52, n. 3, p. 340-351, 2001.

GEMMILL, D. D.; EDWARDS, M. L. Improving resource-constrained project scheduling with look-ahead techniques. **Project Management Journal**, v. 30, n. 3, p. 44-55, 1999.

HARTMANN, S. **Project scheduling under limited resources: models, methods, and applications**. Lecture Notes in Economics and Mathematical Systems 478, Springer-Verlag, 2000.

HARTMANN, S.; KOLISCH, R. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. **European Journal of Operational Research**, v. 127, n. 2, p. 394-407, 2000.

ILOG. Disponível em: <https://support.ilog.com/public/products/faq.cfm?FAQ=106&Product=CPLEX>. Acesso em: julho 2007.

KOLISCH, R.; PADMAN, R. An integrated survey of deterministic project scheduling. **Omega:** The International Journal of Management Science, v. 29, n. 3, p. 249-272, 2001.

KOLISCH, R.; SPRECHER, A. PSPLIB- A project scheduling library. **European Journal of Operational Research**, v. 96, n. 1, p. 205-216, 1996.

MODER, J.J.; PHILLIPS, C.R; DAVIS, E.W. **Project Management with CPM, PERT and Precedence Diagramming**, 3. ed. New York: Van Nostrand, 1983.

MÖHRING, R. F. Minimizing costs of resource requirements in project networks subject to a fixed completion time. **Operations Research,** v. 32, n. 1, p. 89-120, 1984.

NEUMANN, K.; SCHWINDT, C.; ZIMMERMANN, J. **Project scheduling with time windows and scarce resources**. Berlin: Springer-Verlag, 2002.

PATTERSON, J. H.; SLOWINSKI, R.; TALBOT, F. B.; WEGLARZ, J. An algorithm for a general class of precedence and resource-constrained scheduling problems. **Advances in Project Scheduling**. ELSEVIER; SLOWINSKI, R.; WEGLARZ, J. 1989, p. 3-28, Amsterdan.

POLLACK-JOHNSON, B.; LIBERATORE, J. M. Project management software usage patterns and suggested research directions for future developments. **Project Management Journal**, v. 29, n. 2, p. 19-28, 1998.

RANGASWAMY, B. **Multiple Resource Planning and Allocation in Resource-Constrained Project Networks**. (Ph.D. thesis) - Graduate School of Business, University of Colorado, 1998.

SPRECHER, A. Resource-constrained project scheduling: Exact methods for the multi-mode case. **Lecture Notes in Economics and Mathematical Systems**. Springer, v. 409, 1994, Berlin.

SPRECHER, A.; DREXL, A. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. **European Journal of Operational Research**, v. 107, n. 2, p. 431-450, 1998.

TALBOT, F. B. Resource-constrained project scheduling problem with time-resource tradeoffs: The nonpreemptive case. **Management Science**, v. 28, n. 10, p. 1197-1210, 1982.

TAKAMOTO, M.; YAMADA, N.; KOBAYASHI, Y.; NONAKA, H. Zero-one quadratic programming algorithm for resource levelling of manufacturing process schedules. **Systems and Computers in Japan**, v. 26, n. 10, p. 2075-2082, 1995.

YAMASHITA, D. S.; ARMENTANO, V. A.; LAGUNA, M. Scatter search for project scheduling with resource availability cost. Special Issue on Scatter Search. **European Journal of Operational Research**, v. 169, n. 2, p. 623-637, 2006.

YAMASHITA, D. S; MORABITO, R. Um algoritmo exato para o problema de programação de projetos com custo de disponibilidade de recursos e múltiplos modos. **Pesquisa Operacional**, v. 27, n. 1, p. 27-49, 2007a.

YAMASHITA, D. S.; MORABITO, R. A note on time/cost tradeoff curve generation for project scheduling with multi-mode resource availability costs. 2007b. Aceito para publicação no **International Journal of Operations Research**.

APÊNDICE 1: Exemplo ilustrativo do PCDRMM (exemplo 1)

Considere um projeto com n = 7 atividades e m = 3 tipos de recursos, com as restrições de precedência da Figura 1, e com as durações d_{ji} e as quantidades de recursos r_{jik} utilizados por cada atividade j quando executada no modo i conforme apresentados na Tabela 1 (note que a atividade 4 tem dois modos de execução para as atividades, isto é, $M_4 = 2$). O custo do projeto é dado por: $a_1 + 5a_2 + 2a_3$, e a data de entrega do projeto é D = 20. Considere que: $a_1 = 2$, $a_2 = 4$ e $a_3 = 6$. A Figura 2 mostra um programa infactível para este projeto com relação à sua data de entrega, porém factível com relação à quantidade máxima de recursos utilizados. No eixo horizontal da figura, estão os instantes inicial e final de cada atividade e, no eixo vertical, a quantidade de recursos do tipo k = 1, 2, 3, disponíveis para o projeto (a atividade 4 está sendo executada no modo 2).

Tabela 1. Dados de entrada do exemplo 1.

Atividade J		Modo de execução i	Duração d_{ji}	Recurso 1 r_{ji1}	Recurso 2 r_{ji2}	Recurso 3 r_{ji3}
1	**$M_1 = 1$**	1	0	0	0	0
2	**$M_2 = 1$**	1	5	1	2	3
3	**$M_3 = 1$**	1	5	1	2	3
4	**$M_4 = 2$**	1	5	3	1	1
		2	10	1	1	1
5	**$M_5 = 1$**	1	5	1	3	2
6	**$M_6 = 1$**	1	5	1	3	2
7	**$M_7 = 1$**	1	0	0	0	0

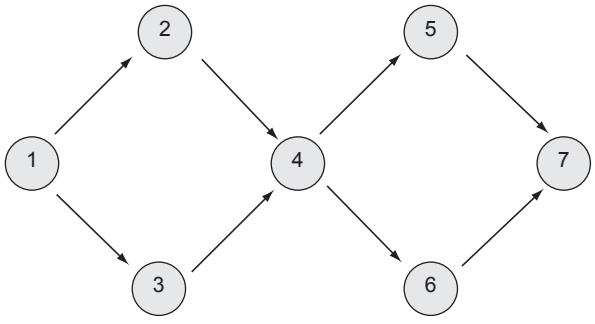


Figura 1. Relações de precedência do projeto do exemplo 1.

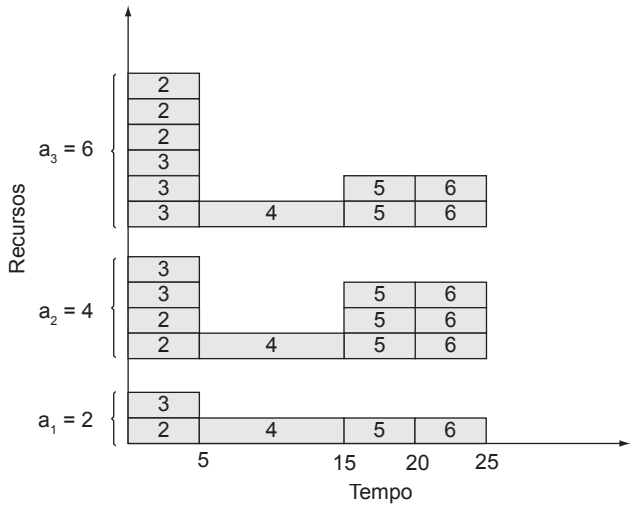
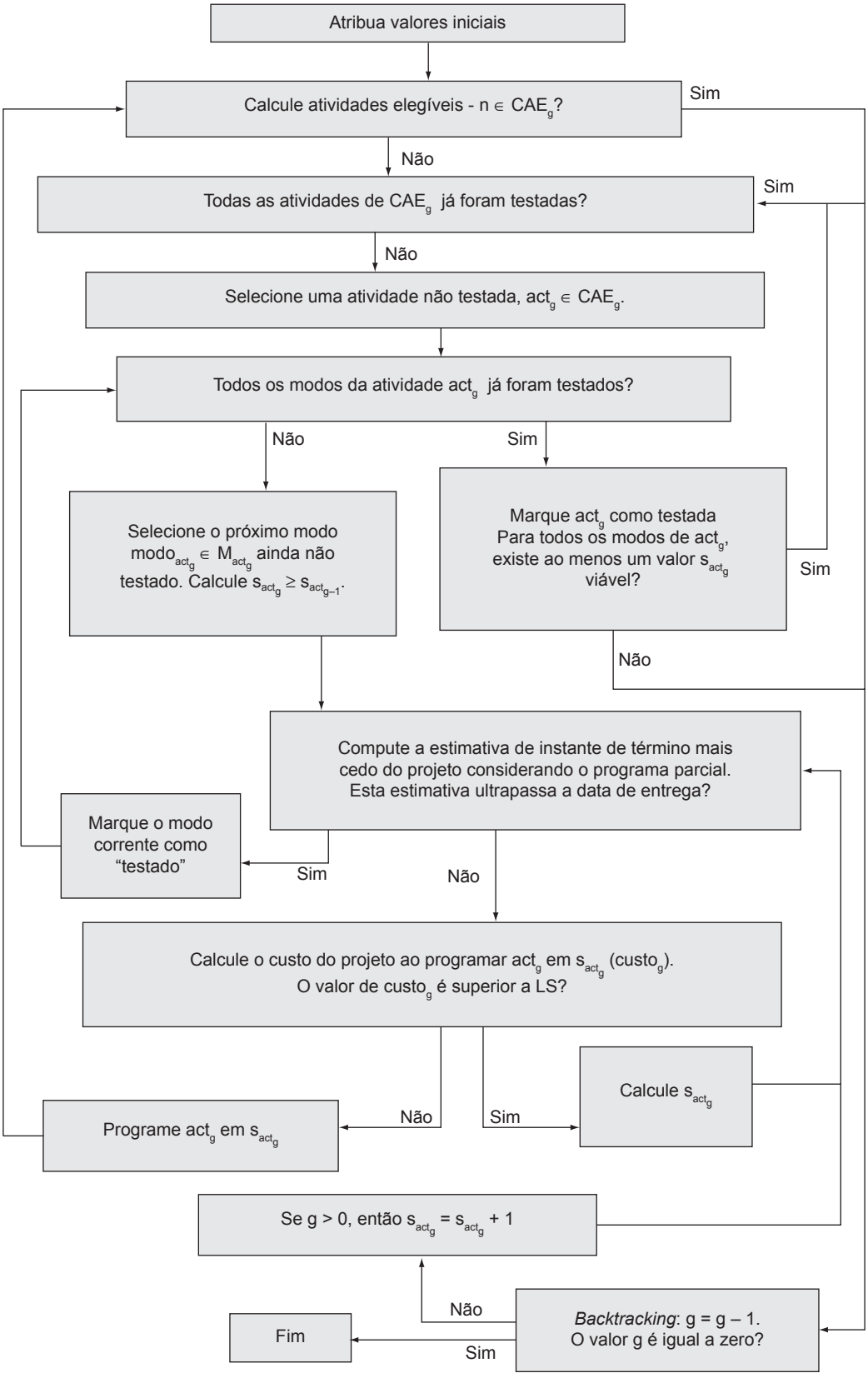


Figura 2. Diagrama de Gantt para o projeto do exemplo 1.

APÊNDICE 2: Fluxograma do algoritmo BB



Sobre os autores

Denise Sato Yamashita

Reinaldo Morabito

Departamento de Engenharia de Produção, Universidade Federal de São Carlos,
CEP 13565-905, São Carlos, SP, Brasil,
e-mails: denisesy@dep.ufscar.br; morabito@ufscar.br

Agradecimentos: Os autores agradecem aos dois revisores anônimos pelos úteis comentários e sugestões. Esta pesquisa contou com o apoio do CNPq (processos 150337/2004-3 e 522973/95-4) e da FAPESP (processos 07/00209-6 e 2006/03496-3).

Recebido em 22/6/2007
Aceito em 06/11/2007