Um método híbrido para um problema de escalonamento de projetos

André Renato Villela da Silva

Departamento de Computação
Instituto de Ciência e Tecnologia - PURO
Universidade Federal Fluminense
R. Recife s/n, Jardim Bela Vista - Rio das Ostras/RJ CEP 28890-000
avillela@ic.uff.br

Resumo

Este trabalho trata de um problema de escalonamento de projetos onde as tarefas consomem recursos para serem ativadas, porém passam a produzí-los após este momento. Este problema é conhecido como Problema de Escalonamento de Projetos com Restrição de Recursos Dinâmicos (PEPRRD). Três métodos foram propostos para dividir o problema em partes menores e resolvê-las separadamente. Cada solução parcial é obtida pelo otimizador CPLEX e é utilizada para geração de soluções parciais mais completas. Os resultados obtidos mostram que este método híbrido funciona muito bem.

Palavras-chave: Problema de Escalonamento de Projetos, Otimização Combinatória, Métodos híbridos, Heurísticas

Área principal: Otimização Combinatória

Abstract

This work deals with a project scheduling problem where the tasks consume resources to be activated, but start to produce them after that. This problem is known as Dynamic Resource-Constrained Project Scheduling Problem (DRCPSP). Three methods were proposed to divide the problem into smaller parts and solve them separately. Each partial solution is obtained by CPLEX optimizer and is used to generate more complete partial solutions. The obtained results show that this hybrid method works very well.

Keywords: Project Scheduling Problem, Combinatorial Optimization, Hybrid Methods, Heuristics

Main area: Combinatorial Optimization

1 Introdução

Nos Problemas de Escalonamento de Projetos (PEP), dois elementos são de fundamental importância: as tarefas, que constituem cada uma das etapas do projeto a ser executado, e os recursos, que são os insumos necessários para que uma tarefa seja executada. As tarefas estão conectadas entre si através de relações de precedência que determinam a ordem em que as tarefas podem ou não ser executadas. Normalmente, estas relações são do tipo *finish-to-start*, ou seja, é preciso que uma tarefa predecessora seja completamente executada antes de uma tarefa sucessora começar a ser. Também é muito comum que uma tarefa possa ter mais de uma predecessora. Neste caso, todas as predecessoras precisam ter sido executadas antes da tarefa em questão começar. O objetivo mais comum é fazer com que todas as tarefas do projeto sejam executadas o mais rapidamente possível, respeitando as restrições de precedência e de utilização dos recursos.

Os recursos que são necessários para a execução das tarefas são o outro elemento a ser administrado no PEP. Uma classificação bastante tradicional os divide em dois grupos: recursos renováveis e recursos não-renováveis. Os recursos renováveis são aqueles que, após ser utilizados na execução de uma tarefa do projeto, ficam novamente disponíveis para ser utilizados em outra tarefa ainda não executada. Alguns exemplos de recursos desta classe são as máquinas (escavadeiras, tratores, computadores) e os profissionais (engenheiros, programadores, assistentes). Estes recursos podem ser reutilizados ao final de uma etapa de projeto. Os recursos são classificados como não-renováveis se eles estiverem disponíveis uma única vez durante todo horizonte de tempo no qual deve ser tratado o problema. Uma vez que eles são utilizados (consumidos) não é mais possível contar com eles até o fim do problema. Exemplos mais comuns são combustíveis e dinheiro, entre outros.

Tradicionalmente, os PEPs não supõem a geração e sim o consumo dos recursos que são dados de entrada com valores pré-definidos uma vez que ou estes têm caráter não-renovável ou têm sua taxa de renovação bem definida pelo problema, como visto em [Valls et al., 2008, Nonobe e Ibaraki, 2002]. Estes cenários, no entanto, não são capazes de modelar certas situações onde, a partir do término da execução de uma etapa do projeto, esta passa a gerar recursos adicionais.

Como forma de ilustrar estas situações, suponha que o projeto em questão seja a expansão comercial de uma companhia. Após a construção de uma nova filial, é razoável supor que ela possa contribuir financeiramente com a matriz por meio do lucro que se espera dela. O recurso que se apresenta mais importante neste caso, sem dúvida, é o retorno financeiro. A quantia, uma vez investida na construção da filial, não fica novamente disponível ao final desta etapa, ao contrário de uma máquina ou um trabalhador. O que ocorre é que após a finalização deste investimento, o mesmo passa a retornar novos recursos financeiros que poderão ser aplicados na execução de outras etapas: abertura de novas filiais, contratação de pessoal, compra de equipamentos, entre outros.

No modelo que será alvo de estudo, o dinheiro é um recurso renovável, mas não como nos modelos tradicionais, onde há uma quantidade máxima disponível a cada instante de tempo. No modelo proposto, um recurso é renovável por ser possível haver diminuição e aumento de sua quantidade disponível ao longo do tempo. A produção de recursos (dinheiro, no caso) permanece desde o término da construção da filial até o fim do horizonte de planejamento em questão.

Este trabalho abordará o PEP onde as tarefas consomem recursos ao serem ativadas e, a partir de então, passam a gerar recursos até o final do horizonte de planejamento cujo tamanho é dado por um parâmetro de entrada. O modelo pressupõe ainda uma quantidade inicial de recursos que poderão ser gastos nas primeiras ativações. O objetivo deste modelo é chegar ao final do horizonte de planejamento com a maior quantidade possível de recursos. O recurso abordado neste caso não pode ter uma quantidade máxima definida, já que o objetivo citado perderia o sentido. Este recurso que é consumido e, posteriormente, produzido apresenta variações de quantidade ao longo do horizonte de planejamento que são difíceis de prever. Por isto este tipo de recurso é chamado de Recurso Dinâmico, sendo o problema de escalonamento chamado de Problema de Escalonamento de Projetos com Restrição de Recursos Dinâmicos- PEPRRD.

O PEPRRD encontra aplicações potenciais em expansões comerciais ou industriais, como no citado exemplo da abertura de filiais que produzem lucro após sua abertura. Também é possível utilizar o PEPRRD em problemas de prestação de serviço ou provimento de infra-estrutura em regiões mais afastadas. Suponha, por exemplo, que se deseja expandir uma rede de fibra ótica por várias cidades do interior do estado ou do país. Não é possível, a princípio, fornecer um ponto de acesso em qualquer localidade por questões físicas, logísticas ou mesmo financeiras. A ampliação da rede deve ocorrer atendendo primeiramente as regiões onde o custo de implantação de um ponto de acesso é pequeno ou onde é possível conseguir maior lucratividade com a prestação de algum tipo de serviço (internet, telefonia, entre outros),

desde que esta implantação seja tecnicamente viável, obviamente. Cria-se então uma situação de precedência entre as localidades que diferencia este problema de outros como os das *p*-medianas tratados por [Mladenović et al., 2007, Rosing e Hodgson, 2002], por exemplo.

O restante deste trabalho está organizado da seguinte forma: a seção 2 traz uma breve revisão da literatura sobre o problema. A seção 3 trará a definição formal do problema. Métodos híbridos - que mesclam características dos algoritmos exatos e heurísticos - serão o assunto da seção 4. Na seção 5 serão apresentados os resultados computacionais. Por fim, as conclusões serão apresentadas na seção 6.

2 Revisão da literatura

Um dos primeiros trabalhos sobre o PEPRRD foi apresentado por [Silva e Ochi, 2007a]. O objetivo do trabalho foi propor uma solução para o principal ponto fraco de boa parte dos algoritmos evolutivos: a convergência prematura. Após algumas gerações, os evolutivos existentes anteriormente tinham dificuldade em continuar melhorando a qualidade dos indivíduos por meio do operador de cruzamento.

A proposta do artigo foi utilizar outra meta-heurística onde fosse possível descartar soluções ruins ou que não pudessem ser melhoradas de maneira eficiente. A meta-heurística escolhida para substituir os algoritmos evolutivos foi o GRASP [Feo e Resende, 1995].

No GRASP, a cada iteração, uma nova solução é gerada por um método construtivo e depois aprimorada por alguma busca local. Além do construtivo ADDR, dos critérios de escolha das tarefas e das buscas locais, foi utilizada uma nova busca local proposta naquele artigo. A busca local chamada de LS3 aproveita parte de um escalonamento já construído e o completa utilizando um critério menos guloso do que aquele utilizado pelo construtivo ADDR. Para que a busca local pudesse funcionar bem, era necessário que a parte do escalonamento a ser mantida não fosse pequena, pois pouca informação seria aproveitada da solução original, nem fosse muito grande, já que desta forma poucas alterações seriam possíveis em relação à solução original. Os resultados computacionais mostrados naquele artigo indicam que a busca LS3 tem capacidade de melhorar uma solução bem maior que as buscas locais propostas anteriormente. As duas versões de GRASP testadas no artigo foram melhores do que os evolutivos propostos anteriormente.

A busca local LS3 foi empregada novamente em [Silva e Ochi, 2007b], como parte de um novo algoritmo evolutivo. Desta vez, além dos operadores evolutivos básicos, foram propostos outros mecanismos para tentar evitar a convergência prematura. Os mecanismos de intensificação e diversificação mais específicos começam a agir quando a população passa algumas gerações sem aprimorar a melhor solução encontrada. Num primeiro lugar, a melhor solução encontrada servirá de semente para a geração de indivíduos bastante semelhantes. Usando a busca LS3, este melhor indivíduo dá origem uma população inteira de novos outros. Esta fase funciona como uma intensificação ao redor desta solução, o que pode fazer com que melhores soluções sejam mais facilmente encontradas.

Se porém isto não ocorrer e mais algumas gerações se sucederem sem aprimoramentos na melhor solução, toda a população será descartada e uma nova população será criada a partir do algoritmo construtivo ADDR, como foi feito na população inicial do algoritmo evolutivo. Esta fase tem o objetivo de diversificar o foco do algoritmo evolutivo para outras soluções diferentes daquelas nas quais ele está trabalhando no momento.

Também neste artigo é proposto um método híbrido capaz de combinar um escalonamento parcial gerado por um método exato com o novo evolutivo proposto. Este escalonamento parcial é obtido pelo software CPLEX, de acordo com a formulação matemática proposta em [Silva e Ochi, 2007a] e limitando-se a ativação das tarefas até um determinado instante de tempo. O objetivo é gerar a população inicial e as populações produzidas pela diversificação a partir do escalonamento parcial e da aplicação da busca local LS3. Com a primeira parte do escalonamento a cargo do método exato, espera-se passar para a busca local uma solução parcial de qualidade superior em relação àquelas geradas de forma heurística.

O método híbrido se mostrou particularmente eficiente nas instâncias que tinham um horizonte de planejamento não muito longo (até 25 unidades de tempo aproximadamente), mesmo que houvesse grande número de tarefas. Com o horizonte de planejamento mais extenso, o tempo computacional gasto pela primeira parte do escalonamento não se refletia na qualidade da solução gerada, o que fez com que este método fosse considerado inviável para estas instâncias.

Estudos mais aprofundados como [Lin et al., 2009, Poorzahedy e Rouhani, 2007] mostram que métodos que agreguem características provenientes de mais de um tipo de heurística ou que misturem componentes heurísticos e exatos podem fornecer resultados muito interessantes e que dificilmente poderiam ser alcançados por métodos heurísticos tradicionais.

3 Definição do problema

Antes de apresentar a definição do problema, é importante deixar claro alguns conceitos utilizados no funcionamento do modelo. Alguns deles são provenientes do clássico Problema de Escalonamento de Projetos, outros foram definidos especificamente para este modelo.

- *Tarefa*: é cada uma das etapas de um projeto que precisa ou deve ser executada. Seu dado mais importante é o instante de tempo em que ela deve ser executada.
- *Solução*: é um vetor de n números inteiros não-negativos, onde n é o número de tarefas do projeto. Cada elemento v_i do vetor indica o momento em que a tarefa i deve ser executada.
- *Recurso*: é qualquer insumo (material, equipamento, profissional ou qualquer outro item) necessário para executar cada uma das tarefas do projeto. Embora possível em vários outros modelos, o PEPRRD admite apenas um único tipo de recurso que pode acumular quantidade não-limitada de unidades ao longo da execução do projeto.
- *Recursos Disponíveis*: são os recursos que podem ser aplicados num determinado instante de tempo t para a execução de tarefas. Denota-se por Q_t.
- *Custo de uma tarefa*: é a quantidade total (positiva) de recursos necessários para que uma tarefa possa ser executada. O custo de uma tarefa i será denotado por c_i.
- *Lucro de uma tarefa*: é a quantidade não-negativa de recursos que serão produzidos pela tarefa a partir do instante de tempo seguinte a sua ativação. Será denotado por p_i para cada tarefa i do problema.
- *Lucro acumulado*: é a soma do lucro das tarefas ativadas até um instante de tempo t. A notação é feita por P_t.
- *Ativação de uma tarefa*: é a indicação de um instante de tempo t no qual a tarefa deve ser executada. Neste instante de tempo, deve haver recursos disponíveis em quantidade igual ou maior do que o custo da tarefa em questão. Também é necessário que todas as tarefas predecessoras da tarefa em questão já estejam ativadas até o instante de tempo anterior $(t-1)$.
- *Duração da tarefa*: é o tempo necessário para que uma tarefa possa ser inteiramente processada, ou seja, ter a sua execução completada. No PEPRRD, já no final do instante da sua ativação, a tarefa é considerada completada e passa, então, a produzir recursos até o final do horizonte de planejamento.
- *Horizonte de planejamento*: é o conjunto de instantes de tempo nos quais as tarefas podem ser executadas. No PEPRRD, o tempo é discretizado em unidades (instantes) de 1 até H, sendo este um dado de entrada do problema.

O PEPRRD é composto de um grafo acíclico direcionado $G = (V, A)$, onde V é um conjunto de vértices e A é o conjunto de arcos que unem os vértices. A cada tarefa $i \in V$, está associado um custo c_i e um lucro p_i, inteiros não-negativos. Inicialmente, existe uma quantidade de recursos disponíveis $Q_0 > 0$ e um lucro acumulado $P_0 = 0$. O escalonamento deverá ser realizado durante um horizonte de planejamento composto por H unidades de tempo. O objetivo do problema é maximizar a quantidade de recursos (recursos disponíveis e lucro acumulado) ao final do horizonte de planejamento.

A Figura 1 e a Tabela 1 apresentam um exemplo deste modelo de escalonamento sendo resolvido por um algoritmo arbitrário. No exemplo, o horizonte de planejamento é composto por quatro unidades ($H = 4$) e a quantidade inicial de recursos disponíveis $Q_0 = 4$. Por questões de simplificação, a quantidade de recursos disponíveis Q_t e o lucro acumulado P_t serão indicados por Q e P, respectivamente. As tarefas já ativadas estão em branco, as que estão disponíveis para ativação aparecem em cinza e as que ainda não podem ser ativadas são mostradas em preto.

No instante de tempo $t = 1$, Figura 1(a), as tarefas 1 e 2 estão disponíveis para ativação. Suponha que o algoritmo escolha a tarefa 2 para ser ativada. É necessário consumir 3 unidades de recursos, já que $c_2 = 3$. O lucro acumulado que inicialmente era zero passa a conter 2 unidades ($p_2 = 2$). Com o recurso disponível restante não é possível ativar a tarefa 1, logo, o tempo $t = 1$ deve ser encerrado como na Figura 1(b). Ao se passar para o instante seguinte ($t = 2$, Figura 1(c)) este lucro acumulado torna-se recurso disponível, uma vez que ele representa os recursos que estão sendo produzidos pela tarefa já ativada. Com a ativação da tarefa 2 realizada, a tarefa 4 fica apta a ser ativada.

Tarefa	Custo	Lucro
1	2	1
2	3	2
3	4	4
4	1	2
5	2	3
6	4	5

Tabela 1: Custos e lucros utilizados no exemplo

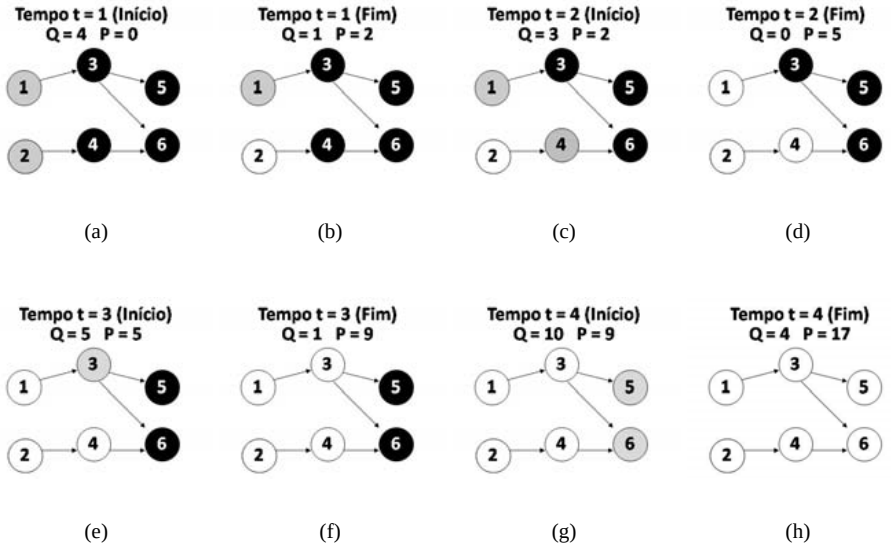

Figura 1: Etapas do exemplo de escalonamento

A quantidade de recursos disponíveis, agora, permite que ambas as tarefas 1 e 4 sejam ativadas. Esta decisão faz com que sejam subtraídas 3 unidades de recursos ($c_1 + c_4 = 3$) e sejam incorporadas mais 3 unidades ($p_1 + p_4 = 3$) ao lucro acumulado, como na Figura 1(d). Como não há mais tarefas para serem ativadas neste momento, é necessário iniciar a transição para o instante de tempo seguinte: o lucro acumulado é somado aos recursos disponíveis e o estado das tarefas é atualizado.

No início do tempo $t = 3$ (Figura 1(e)), a tarefa 3 pode ser ativada já que há recursos suficientes. Atualizados os recursos disponíveis, o lucro acumulado e o estado das tarefas, chega-se ao último instante de tempo ($t = 4$, Figura 1(g)), quando as tarefas 5 e 6 serão ativadas. Ao final deste instante (Figura 1(h)), restaram quatro recursos disponíveis e o lucro acumulado é de dezessete unidades. O resultado deste escalonamento, portanto, é igual a vinte e uma unidades de recurso.

4 Métodos propostos

O algoritmo híbrido CPLEX+EA3 proposto em [Silva e Ochi, 2007b] introduziu uma idéia bem interessante: dividir o horizonte de planejamento em duas metades e realizar o escalonamento destas partes em sequência, pelo método exato CPLEX e pelo evolutivo EA3, respectivamente. O algoritmo apresentou resultados significativos em instâncias de médio porte ou que possuam um curto horizonte de planejamento.

Para instâncias maiores, principalmente as detentoras de um extenso horizonte de planejamento, o algoritmo consumia muito tempo para produzir o escalonamento parcial da primeira metade (sob responsabilidade do CPLEX) e, consequentemente, não conseguia terminar em tempo aceitável.

A proposta agora é utilizar a melhor formulação conhecida para o PEPRRD, originalmente discutida em [Silva e Ochi, 2010], e dividir o horizonte em partes menores para que um resultado parcial seja produzido mais rapidamente. Esta solução é, então, utilizada como base para que outra solução parcial (que contemple mais unidades de tempo) seja também produzida até que todo o horizonte de planejamento seja computado.

A limitação de ativação dentro do horizonte de planejamento é feita por meio da inclusão de restrições que impeçam a ativação após a unidade de tempo desejada. Sempre que uma solução parcial é produzida, as tarefas ativadas têm suas variáveis fixadas na formulação de acordo com o tempo de ativação da tarefa. Isto deve ser feito para que o problema não se torne cada vez mais difícil com a possibilidade de realizar o escalonamento em intervalos cada vez maiores. A divisão do horizonte pode ser feita de diversas maneiras. Foram propostas três formas de divisão, utilizando dois critérios distintos, que serão explicados a seguir. Ambos os critérios, por não tratar o horizonte de planejamento todo uma de vez, são considerados métodos heurísticos e podem não fornecer a solução ótima para o problema. Cada partição gerada, no entanto, será resolvida por um método exato, permitindo classificar os métodos como híbridos.

- Intervalos de tamanho fixo: nesta forma de divisão, o horizonte de planejamento será seccionado em intervalos de K unidades de tempo. Por exemplo, se $H = 7$ para uma instância qualquer e $K = 2$, serão executados quatro intervalos: os três primeiros com duas unidades e o último com apenas a unidade de tempo restante.

- Intervalos de tamanhos variáveis: outra forma de realizar a divisão é definir a quantidade de intervalos mas permitir que estes tenham tamanhos distintos. Embora o ideal seja que os intervalos tenham o mesmo tamanho, o que aumenta ou diminui a dificuldade de execução é, principalmente, a quantidade de tarefas (e por consequência de variáveis) que devem ser tratadas em cada intervalo. Assim, intervalos cujos tamanhos tenham uma pequena diferença são perfeitamente aceitáveis.

  Entretanto, testes preliminares mostraram que levar em consideração apenas a quantidade de variáveis binárias não apresenta resultados tão bons, pois os primeiros instantes de tempo terão menos variáveis binárias devido às restrições de menor tempo. Isto acarreta em intervalos que precisam reunir mais unidades de tempo, demorando mais para serem executados. De fato, a contagem que vai determinar o tamanho de cada intervalo será baseada na quantidade de tarefas que têm o menor tempo de ativação dentro do intervalo em questão.

  O pré-processamento computará, em cada tempo t, a quantidade de tarefas i que têm $menor(i) = t$, esta quantidade será denotada por $w(t)$. O valor de n (número total de tarefas) será dividido pela quantidade de intervalos v para saber o tamanho médio m dos intervalos. Os intervalos agregarão consecutivas unidades de tempo até que a soma dos valores $w(t)$ seja o mais próximo possível de m. Se não for possível atingir exatamente o valor de m, a quantidade que sobrar ou que faltar será levada em consideração no cálculo do próximo intervalo. É importante mencionar que a quantidade de intervalos v é um dado passado pelo usuário do algoritmo; só assim será possível realizar os cálculos necessários para esta forma de divisão do horizonte de planejamento. As Figuras 2 e 3 mostram como particionar um grafo em 4 ou 6 intervalos, respectivamente, no qual as tarefas podem ser iniciadas até o tempo $t = 8$. Não foram mostrados os arcos para facilitar a visualização das figuras, as tarefas são representadas pelos círculos cinzas.

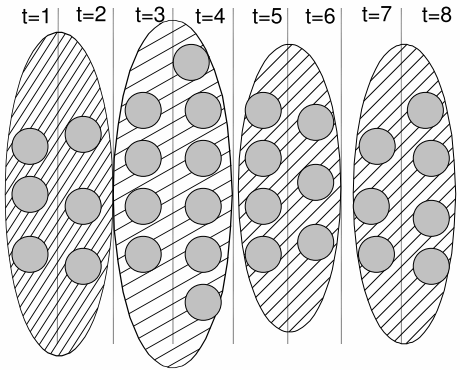

Figura 2: Exemplo de divisão do horizonte de planejamento em 4 partições

Como o grafo contém 30 tarefas, a divisão em 4 intervalos na Figura 2 nós dá uma média de 7.5 tarefas por intervalo. A primeira unidade de tempo possui apenas 3 tarefas, o que fica muito longe do nosso objetivo (7.5). Incluindo o tempo 2, ficamos com 6 tarefas. Se incluirmos o tempo 3, teríamos 10

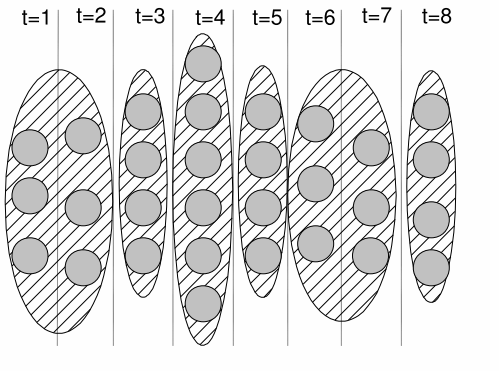

Figura 3: Exemplo de divisão do horizonte de planejamento em 6 partições

tarefas - mais distante de 7.5 do que o valor 6. Portanto, nosso primeiro intervalo termina em $t = 2$, faltando "uma tarefa e meia" para o objetivo. O segundo intervalo vai levar isto em consideração e buscar por um grupo que chegue perto de 7.5 + 1.5 = 9 tarefas. A melhor escolha é agrupar os tempos 3 e 4 em outro intervalo, agora com uma tarefa de sobra. O algoritmo prossegue até chegar a $t = 8$, neste caso.

Para a divisão em 6 intervalos, temos um valor médio de 5 tarefas para cada partição. Na Figura 3, pode-se ver como ficariam estas partições utilizando o mesmo raciocínio empregado no exemplo anterior.

- Múltiplos intervalos de tamanhos variáveis: a idéia é bem semelhante à anterior, mas ao final de cada execução completa, ou seja, após realizar o escalonamento em todo o horizonte de planejamento, os intervalos já definidos serão modificados de forma a ficarem uma unidade de tempo mais longo ou mais curto. Após cada modificação, o escalonamento será realizado levando em consideração a nova configuração. Portanto, este mecanismo executa diversos escalonamentos semelhantes ao item anterior, porém com pequenas alterações no que diz respeito ao tamanho dos intervalos. A Figura 4 mostra um exemplo com 5 intervalos ($v = 5$) e $H = 24$. Cada unidade de tempo é representada por um quadradinho e a divisão dos intervalos por uma linha vertical tracejada. Quadradinhos que apresentam o mesmo estilo de fundo estão no mesmo intervalo. As execuções seguem a ordem estabelecida pela figura, de cima para baixo. O particionamento original, no caso com intervalos de tamanho 4, 5, 3, 5 e 7, é definido como no critério anterior (intervalos de tamanho variável).

Estas três formas de seccionamento também fazem com que a solução ótima possa não ser encontrada mais, pois a solução ótima de um escalonamento parcial pode não fazer parte da solução ótima do escalonamento por completo. Assim, estas versões híbridas não podem ser consideradas algoritmos exatos para o PEPRRD.

5 Resultados computacionais

Os experimentos com os métodos híbridos, onde o horizonte de planejamento é particionado em intervalos, consistem em executar cada um deles até que todo o escalonamento tenha sido realizado. Embora, a divisão do horizonte de planejamento abrevie bastante o tempo necessário para concluir a otimização em cada intervalo, não é possível prever um tempo máximo para que isto ocorra. Desta forma, será definido um limite de 3 horas (10800 segundos) para que a otimização de cada parte seja concluída. Se o limite for atingido, a solução incumbente será considerada como o resultado deste intervalo e servirá de base para outras eventuais etapas. É importante dizer que estes algoritmos são determinísticos, uma vez que não há variáveis aleatórias envolvidas.

O primeiro esquema testado realiza a divisão em intervalos de tamanho K. A Tabela 2 mostra os resultados finais para diversos valores de K (de 2 a 10). A última coluna apresenta a média dos diversos resultados. As instâncias são as mesmas que foram analisadas por [Silva e Ochi, 2007a].

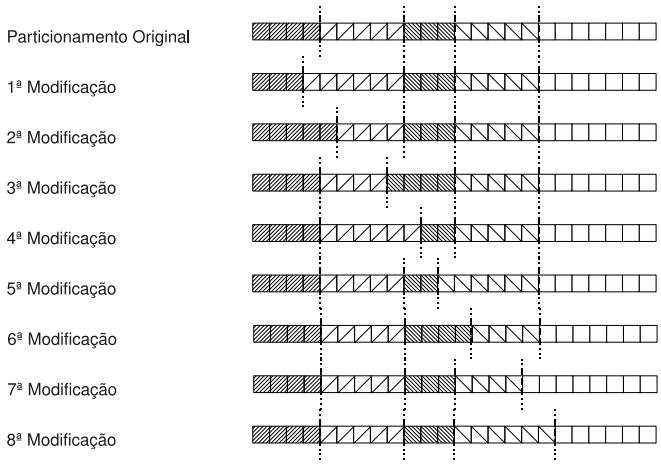

Figura 4: Esquema de múltiplos intervalos

	Tamanho do Intervalo(K)									
Instância	k=2	3	4	5	6	7	8	9	10	Média
100a	303	304	303	**304**	**304**	**304**	**304**	**304**	**304**	303.8
200a	632	628	632	**636**	632	635	632	635	**636**	632.5
300a	1898	1927	1965	1946	2020	2041	**2043**	2025	1950	1959.2
400a	5618	6143	6641	6169	**6741**	6732	6656	6460	6158	6320.9
500a	12689	13239	13006	13433	13397	13408	**13941**	13846	13510	13272.0
600a	8890	9554	9424	9321	9809	9526	9417	9864	**10096**	9407.9
700a	25454	28769	28398	28599	28772	29882	**30587**	29919	28824	28209.5
800a	32889	35127	35614	34118	35683	35300	37007	**37363**	37145	35263.0
900a	31981	33635	35070	33594	36046	34442	34619	36730	**37529**	34372.9
1000a	64744	65840	65207	67699	67885	66024	67867	69338	**69717**	66473.3

Tabela 2: Resultados para divisão do horizonte de planejamento em intervalos de tamanho fixo

Duas subdivisões com valores de K semelhantes apresentam resultados também semelhantes. Era esperado que um valor de K maior produzisse resultados melhores por abranger uma parte maior do escalonamento. Isto não se confirmou porque a visão de apenas parte do escalonamento pode influenciar na percepção dos algoritmos sobre a importância de uma tarefa para o escalonamento como um todo. Exemplo: uma tarefa pode ser considerada muito ruim se olhar só para ela e não considerar a possibilidade de ativação de suas sucessoras; porém, se existir tempo suficiente para que também as sucessoras sejam ativadas, a tarefa inicialmente ruim pode ser considerada muito importante. De forma contrária, uma tarefa inicialmente considerada boa, pode ser preterida por outras que venham ser consideradas melhores.

Obviamente que uma diferença muito grande nos valores de K tende a produzir resultados igualmente diferentes. Outra constatação é que o tempo consumido também perde seu caráter crescente conforme aumenta o número de tarefas da instância ou dos intervalos de tempo. Na verdade, cada subdivisão de cada instância vai corresponder uma subproblema novo de dificuldade muito complicada de ser prevista. A Tabela 3 mostra os tempos totais (em segundos).

5.1 Intervalos de tamanho variável

Neste experimento, a divisão do horizonte de planejamento se dará por meio do número de tarefas que podem começar a ser ativadas em cada intervalo. Com isso, pode-se ter algumas subdivisões contemplando poucas unidades de tempo e outras com muitas unidades, desde que o número de tarefas relacionadas seja o mais semelhante possível. A Tabela 4 mostra o resultado final para escalonamentos particionados de 2 a 10 vezes de acordo com este critério.

	Tamanho do Intervalo(K)									
Instância	K = 2	3	4	5	6	7	8	9	10	Média
100a	0.0	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.1	0.1
200a	0.0	0.2	0.2	0.3	0.3	0.3	0.4	0.7	0.9	0.3
300a	0.9	0.6	0.7	1.0	1.4	2.6	0.9	1.4	1.8	1.3
400a	1.4	0.9	0.9	2.4	1.8	2.0	3.8	14.1	43.1	7.3
500a	2.5	2.9	5.6	2.6	12.1	9.0	10.5	41.5	2043.7	213.5
600a	3.8	2.9	3.4	11.3	12.7	7.5	514.3	870.5	50.0	148.3
700a	5.3	5.8	5.2	8.5	11.7	31.3	23.0	19.6	22.3	14.4
800a	7.5	7.4	16.0	9.2	203.7	164.1	1346.3	2081.9	555.1	440.6
900a	9.3	7.7	7.8	22.4	59.9	10805.9	10803.9	7788.6	1729.8	3125.2
1000a	13.0	9.3	12.3	9.7	1158.3	102.1	694.1	22.2	1742.6	378.7

Tabela 3: Tempos computacionais para divisão do horizonte de planejamento em intervalos de tamanho fixo (segs)

	Quantidade de Intervalos(v)									
Instância	v = 10	9	8	7	6	5	4	3	2	Média
100a	**304**	**304**	**304**	**304**	**304**	**304**	303	**304**	303	303.8
200a	627	628	628	628	628	628	629	631	**635**	629.1
300a	1784	1833	1869	1839	1885	1841	2026	2030	**2042**	1905.4
400a	5374	5999	5570	5503	5503	5647	**6641**	6612	6460	5923.2
500a	12994	13113	13376	13288	13230	13501	13846	13938	**14225**	13501.2
600a	9183	9110	9230	9441	9414	9814	9830	**10090**	10035	9571.9
700a	25106	25237	25135	25291	27415	29311	27445	28563	**31329**	27203.6
800a	32686	33542	34555	35287	36362	35601	37508	37089	**38511**	35682.3
900a	32447	32004	31705	34266	34345	34431	36475	37295	**38684**	34628.0
1000a	63840	63796	64821	62294	64253	65332	65779	68297	**70519**	65436.8

Tabela 4: Resultados para divisão do horizonte de planejamento em intervalos de tamanho variável

Semelhantemente à divisão em intervalos de tempo fixo, os resultados desta segunda estratégia não apresentam um comportamento bem definido do resultado em relação à quantidade de intervalos. Em teoria, quanto menor esta quantidade, maiores serão os intervalos e melhores tendem a ser as soluções. No entanto, estipular um mapeamento exato entre as Tabelas 2 e 4 é muito complicado porque mesmo em uma instância onde um valor de K corresponda a v intervalos, estes podem agrupar tarefas completamente diferentes. Por exemplo: a instância 400a tem o tamanho do horizonte $H = 20$. A primeira divisão com $K = 10$ produzirá dois escalonamentos: $H_1 = 1..10$ e $H_2 = 11..20$. Já realizando a divisão pelo segundo critério com $v = 2$, pode-se ter dois escalonamentos $H_1 = 1..10$ e $H_2 = 11..20$, ou $H_1 = 1..9$ e $H_2 = 10..20$, ou $H_1 = 1..11$ e $H_2 = 12..20$, ou $H_1 = 1..8$ e $H_2 = 9..20$ ou outro qualquer dependendo da quantidade de tarefas que estão inseridas em cada intervalo. Logo, não é possível fazer uma comparação caso a caso, mas se forem tomados os resultados de um e de outro critério, juntamente com os tempos computacionais mostrados nas Tabelas 3 e 5, pode-se verificar que o segundo critério é um pouco mais eficiente.

Tomemos por comparação o caso onde $v = 2$. Esta é subdivisão mais árdua, em teoria, pois tem-se apenas dois intervalos. Mais difícil do que isto só se for considerado o horizonte inteiro de uma única vez. As instâncias com mais de 400 tarefas, terão mais de 20 unidades de tempo, já que $H = \sqrt{n}$ nas instâncias testadas. Se $K = 10$, teremos para o primeiro critério pelo menos três subdivisões, sendo a última geralmente menor do que as demais. O tempo computacional gasto, para estas instâncias, pelo primeiro critério foi de 29184.3 segundos; pelo segundo, apenas 25442.0. Os resultados do segundo critério são 4.5% melhores do que os do primeiros critério, sendo que em apenas uma instância (600a) o primeiro critério teve resultado melhor. Se levarmos em conta também a instância 400a, que no caso seria subdividida em duas partes pelos dois critérios, o segundo critério teria desempenho ainda ligeiramente melhor. Assim, o segundo critério com apenas dois intervalos consegue se mostrar mais eficiente do que o primeiro critério com três ou mais intervalos.

A subdivisão por intervalos variáveis tem como principal vantagem, nas instâncias testadas, o fato de que os primeiros intervalos contemplam um número maior de tarefas do que na divisão por intervalos de tamanho

fixo. No "início" do grafo, poucas tarefas estão livres para serem ativadas. Assim, para conseguir agrupar o número de tarefas desejado, é preciso incorporar mais unidades de tempo a esses intervalos. Com mais unidades de tempo sendo consideradas inicialmente, melhor o escalonamento parcial naqueles intervalos. Como os demais escalonamentos são baseados nos resultados dos escalonamentos anteriores, melhor tende a ser o resultado final.

Tudo isto, que em teoria faz sentido, provou ser bom também na parte prática. A questão do menor consumo de tempo pode ser explicada de forma semelhante: os escalonamentos iniciais não costumam demorar muito; se forem bem feitos, tendem a oferecer limites primais melhores para os escalonamentos seguintes, que tenderão a terminar mais rapidamente.

	Quantidade de Intervalos(v)									
Instância	v = 10	9	8	7	6	5	4	3	2	Média
100a	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0
200a	0.0	0.0	0.0	0.0	0.1	0.4	0.2	0.2	0.5	0.2
300a	1.0	1.0	0.8	0.8	0.6	0.6	1.0	1.5	5.7	1.4
400a	1.2	1.0	1.2	1.1	1.0	0.8	0.9	1.3	16.9	2.8
500a	3.2	2.8	2.5	3.8	2.2	3.2	3.3	11.0	342.5	41.6
600a	3.5	3.2	5.2	3.5	3.9	8.8	20.3	172.4	9910.8	1125.7
700a	4.4	4.2	3.7	3.3	3.3	3.2	3.4	7.1	164.7	21.9
800a	6.0	5.6	5.5	5.3	18.2	5.6	29.6	87.4	6616.0	753.2
900a	8.6	6.4	6.3	5.9	7.0	5.4	28.1	81.5	4072.4	469.1
1000a	8.5	8.3	7.7	6.9	7.1	6.9	7.8	31.8	488.1	63.7

Tabela 5: Tempos computacionais para divisão do horizonte de planejamento em intervalos de tamanho variável (segs)

5.2 Múltiplos intervalos

Como mencionado, esta divisão utiliza o mesmo critério da anterior, porém ao terminar de tratar todo o horizonte de planejamento, os intervalos são alongados ou encurtados e um novo escalonamento é realizado. Este critério, portanto, é uma extensão do anterior, já que se trata de repetidas execuções que abordam a "vizinhança" dos intervalos. De fato, o número execuções é igual a $2v-1$, onde v é a quantidade desejada de intervalos, pois o último deles termina obrigatoriamente em $t = H$, mas há uma primeira execução idêntica à execução do segundo critério. A Tabela 6 mostra o resultado para diversos valores de v.

	Quantidade de Intervalos(v)									
Instância	v = 10	9	8	7	6	5	4	3	2	Média
100a	304	304	304	304	304	304	304	304	304	304.0
200a	**636**	632	632	632	632	632	635	**636**	635	633.6
300a	1830	1890	1901	1929	2003	1993	2026	2053	**2061**	1965.1
400a	6311	6349	6156	6349	6349	6510	6716	**6741**	6656	6459.7
500a	13441	13273	13490	13501	13369	13792	14014	14127	**14239**	13694.0
600a	9429	9429	9342	9616	9702	9969	9921	10123	**10167**	9744.2
700a	27321	28272	27297	28844	28830	29310	28409	29810	**31567**	28851.1
800a	35877	35432	35878	36140	37456	35663	37692	38096	**38579**	36757.0
900a	33144	33678	35034	34982	35953	36404	37537	38287	**38793**	35979.1
1000a	64284	65532	65295	66010	66933	67589	67551	68301	**71419**	66990.4

Tabela 6: Resultados para múltiplos intervalos de tamanho variável

Como esperado, os resultados desta estratégia de particionamento são um pouco melhores do que a anterior - cerca de 2.8%. O tempo computacional, no entanto, é muito maior, não só porque são executadas várias iterações, mas também porque ao se alterar o tamanho dos intervalos, é possível deixá-los mais difíceis de serem resolvidos. A Tabela 7 mostra os tempos de execução. Pelos valores apresentados, pode-se dizer que este método tem uma relação custo-benefício pior do que o anterior, já que demora muito mais, mesmo quando há grande número de intervalos. Portanto, das três estratégias de particionamento, a segunda - com intervalos de tamanho variável - apresenta os melhores resultados em um espaço de tempo não muito longo.

	Quantidade de Intervalos(v)									
Instância	v = 10	9	8	7	6	5	4	3	2	Média
100a	2.0	1.6	1.4	0.3	0.0	0.0	0.0	0.1	0.1	0.6
200a	8.4	7.0	6.5	0.7	1.4	1.6	1.5	1.0	1.1	3.2
300a	21.0	17.7	13.5	10.8	7.3	5.3	7.6	8.9	12.2	11.6
400a	32.3	27.2	22.0	13.7	9.1	12.2	8.9	8.4	98.5	25.8
500a	76.3	53.3	46.2	36.4	24.0	33.9	27.6	358.0	1340.6	221.8
600a	73.2	59.4	68.5	58.5	51.1	110.0	205.9	495.3	21043.5	2462.8
700a	84.5	75.3	59.1	51.0	44.0	40.9	264.7	71.7	1069.3	195.6
800a	151.8	117.6	97.2	88.7	195.0	88.0	174.5	5548.7	10840.0	1922.4
900a	133.9	111.6	108.9	96.2	104.9	84.5	278.6	1485.6	13762.2	1796.3
1000a	156.8	132.2	115.2	108.4	97.1	106.0	91.1	1283.5	4404.4	721.6

Tabela 7: Tempos computacionais para múltiplos intervalos de tamanho variável (segs)

	Resultado		Tempo(seg)	
Inst.	Partic.	CPLEX	Partic.	CPLEX
100a	304	304	0.0	0.1
200a	635	**636**	0.5	6.5
300a	2042	**2073**	5.7	196.3
400a	6641	**7271**	0.9	467.8
500a	14225	**14336**	342.5	6688.7
600a	10090	**10157**	172.4	50000.0
700a	31329	**31820**	164.7	50000.0
800a	**38511**	38351	6616.0	50000.0
900a	38684	**38733**	4072.4	50000.0
1000a	70519	**71864**	488.1	50000.0

Tabela 8: Comparação entre o melhor particionamento e o melhor limite primal conhecido

Por fim, a Tabela 8 traz uma interessante comparação entre os resultado obtido pelo melhor particionamento de cada instância com o valor do limite primal obtido pelo otimizador CPLEX, considerando um limite máximo de 50000.0 segundos, como explicado em [Silva e Ochi, 2010]. Como já foi dito, o particionamento do horizonte de planejamento acaba por não tratar o problema como um todo. Isto implica em não obrigatoriamente alcançar o valor do ótimo global. Desta forma, embora o algoritmo de resolução por particionamento possa ser considerado híbrido por incorporar escolhas heurísticas e métodos exatos, seu resultado final apresenta uma característica heurística mais marcante. O tempo computacional gasto pelo melhor particionamento é bem inferior ao tempo gasto pela otimização completa das instâncias e o resultado computacional é relativamente bem próximo, o que valida o algoritmo proposto neste artigo. Vale muito destacar o resultado obtido para a instância 800a. O método de particionamento conseguiu resultado melhor que o limite primal obtido pela execução da formulação matemática. Isto se deve porque o tempo de 50000.0 segundos foi atingido sem haver terminado a otimização da instância. De acordo com [Silva e Ochi, 2010], o limite dual para esta instância ficou em 38729.8. Ou seja, é bem provável que o resultado obtido pelo particionamento não seja ótimo, mas o método proposto neste artigo conseguiu superar o desempenho do CPLEX, considerando o tempo computacional gasto pelo otimizador.

6 Conclusão

Os métodos híbridos propostos utilizam apenas o otimizador CPLEX, mas trabalhando com partições (intervalos) do horizonte de planejamento. Quando um intervalo é executado, as tarefas ativadas até o final do intervalo são fixadas e o intervalo seguinte é executado até que seja completada a análise de todo o horizonte de planejamento.

Para realizar as divisões do horizonte, foram propostas três estratégias. A primeira trabalha com intervalos de tamanho fixo, ou seja, todos os intervalos (à exceção do último) têm obrigatoriamente a mesma quantidade de unidades de tempo. A segunda estratégia observa a quantidade de tarefas que podem iniciar sua ativação em cada unidade de tempo. De acordo com o número de intervalos que o usuário deseja, é calcu-

lada uma quantidade média de tarefas em cada intervalo. As consecutivas unidades de tempo serão agrupadas de forma a conter aproximadamente a quantidade média, independentemente do tamanho que venha a ter este intervalo.

A terceira estratégia é uma extensão da segunda. Um primeiro particionamento é definido e executado conforme a estratégia anterior. Logo após, o primeiro intervalo é encurtado e procede-se a uma nova execução. Em seguida, o intervalo é alongado em relação ao tamanho original e outra execução acontece. Passa-se, então, para o segundo intervalo realizando os mesmos dois procedimentos até que o penúltimo intervalo seja considerado. Trata-se, portanto, de repetidas execuções da segunda estratégia, com pequenas diferenças no tamanho dos intervalos.

Os resultados mostraram que a segunda estratégia apresenta melhores resultados médios do que a primeira e em menor tempo. De fato, o desempenho da execução do CPLEX em cada intervalo vai depender muito mais da quantidade de tarefas (e consequentemente de variáveis) a serem analisadas do que de unidades de tempo.

Os resultados médios da terceira estratégia são ainda um pouco melhores do que os da segunda, porém os tempos computacionais crescem significativamente. Fazendo uma comparação da relação custo-benefício da segunda e terceira estratégias, foi escolhida a segunda como a mais vantajosa estratégia de particionamento do horizonte de tempo.

A comparação do melhor particionamento com o resultado da literatura para a otimização completa das instâncias testadas mostrou que o particionamento é uma estratégia válida e que apresenta resultados de caracter heurístico bastante satisfatórios.

Referências

[Feo e Resende, 1995] Feo, T. e Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.

[Lin et al., 2009] Lin, S.-W., Lee, Z.-J., Ying, K.-C., e Lee, C.-Y. (2009). Applying hybrid meta-heuristics for capacitated vehicle routing problem. *Expert Systems with Applications*, 36(2):1505–1512.

[Mladenović et al., 2007] Mladenović, N., Brimberg, J., Hansen, P., e Pï¿½rez, J. A. M. (2007). The p-median problem: A survey of metaheuristic approaches. *European J Operational Research*, 179(3):927–939.

[Nonobe e Ibaraki, 2002] Nonobe, K. e Ibaraki, T. (2002). Formulation and tabu search algorithm for the resource constrained project scheduling problem. In Ribeiro, C. e Hansen, P., editors, *Essays and Surveys in Metaheuristics*, pp. 557–588.

[Poorzahedy e Rouhani, 2007] Poorzahedy, H. e Rouhani, O. M. (2007). Hybrid meta-heuristic algorithms for solving network design problem. *European J, of Operational Research*, 182(2):578–596.

[Rosing e Hodgson, 2002] Rosing, K. E. e Hodgson, M. J. (2002). Heuristic concentration for the p-median: an example demonstrating how and why it works. *Computers & Operations Research*, 29(10):1317–1330.

[Silva e Ochi, 2007a] Silva, A. R. V. e Ochi, L. S. (2007a). Effective grasp for the dynamic resource-constrained task scheduling problem. In *Proc. of International Network Optimization Conference (INOC)*, Spa (Bélgica).

[Silva e Ochi, 2007b] Silva, A. R. V. e Ochi, L. S. (2007b). A hybrid evolutionary algorithm for the dynamic resource constrained task scheduling problem. In *Proc. of the International Workshop on Nature Inspired Distributed Computing (NIDISC'07)*, LongBeach (EUA).

[Silva e Ochi, 2010] Silva, A. R. V. e Ochi, L. S. (2010). Hybrid heuristics for dynamic resource-constrained project scheduling problem. In *Proc. of the 7th International Workshop on Hybrid Metaheuristics*, Viena (Áustria).

[Valls et al., 2008] Valls, V., Ballestín, F., e Quintanilla, S. (2008). A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 185:495–508.