METAHEURÍSTICA GRASP COM PATH RELINKING APLICADA AO PROBLEMA DE PROGRAMAÇÃO DE EMBARCAÇÕES PLV

Maciel Manoel Queiroz
Universidade de São Paulo – EPUSP – Departamento de Engenharia Naval e Oceânica
Av. Prof. Mello Morais, 2231- Cidade Universitária, 05508-030 – São Paulo, SP
maciel.queiroz@usp.br

André Bergsten Mendes
Universidade de São Paulo – EPUSP – Departamento de Engenharia Naval e Oceânica
Av. Prof. Mello Morais, 2231- Cidade Universitária, 05508-030 – São Paulo, SP
andbergs@usp.br

RESUMO

O presente artigo aborda um problema de programação de embarcações que realizam o lançamento de linhas de produção e a interligação destes à infra-estrutura submarina, em uma operação de exploração de petróleo *offshore*. As tarefas são realizadas por embarcações PLVs (*pipe layer vessels*), e possuem como atributos: duração, em dias; lista de embarcações compatíveis; instante de liberação; penalidade relacionada ao atraso na execução da tarefa. Este problema é uma variação da classe de problemas de programação de máquinas paralelas não-relacionadas, em que o objetivo é minimizar a soma do atraso ponderado. Este trabalho empregará como método de solução a meta-heurística GRASP com *path relinking*. Testes foram feitos para comprovar o desempenho das heurísticas propostas, comparando-as com limitantes fornecidos pelo método geração de colunas.

PALAVARAS CHAVE. Programação de máquinas paralelas, Grasp, Path Relinking.

ABSTRACT

This paper addresses a fleet scheduling problem present in the offshore oil industry. Among the special purpose services one will find the pipe layer activities accomplished by the Pipe Layer Vessels (PLV). The jobs are characterized by a release date, which reflects the expected arrival date of the necessary material at the port. There are compatibility constraints between job and vessel, so that some vessels may not be able to perform a certain job. The duration of each job depends on the vessel and if a job is finished after its due date, a penalty is incurred. This is a variation of the unrelated parallel machine problem with total weighted tardiness objective function. In this paper it will be addressed a metaheuristic GRASP with Path Relinking, which have proved to be competitive and an effective solution strategy. Computational experiments were conducted, comparing solutions with bounds provided by linear column generation.

KEYWORDS. Scheduling Unrelated Parallel Machine, Grasp, Path Relinking.

1. Introdução

A área de programação, conhecida na literatura técnica como *scheduling*, está presente em diversos sistemas produtivos, e sempre envolve um número de atividades que estão atreladas à utilização de recursos por um determinado período de tempo (Morton e Pentico, 1993). A programação de tarefas e recursos possui um importante papel na maioria dos sistemas de manufatura, produção e ambientes de processamento de informação, como também nas áreas de transportes e em outros diversos tipos de indústrias (Pinedo, 2002).

Este artigo irá focar o problema de programação de uma frota de navios especializados no processo de lançamento de *risers* e interligação submarina, presente no desenvolvimento de campos de petróleo *offshore*. Conforme será discutido, o problema em questão é uma variação do problema de programação de máquinas paralelas não relacionadas, para o qual Liaw et al. (2003) demonstraram que a complexidade é *NP*-difícil. Foi utilizado um método heurístico para resolução do problema, combinando-se o GRASP com *path relinking*. Os limitantes inferiores foram estimados pela aplicação do método de geração de colunas, e indicaram bom desempenho do método proposto.

O artigo está organizado da seguinte forma: a seção 2 irá descrever o problema; a seção 3 revisa os principais trabalhos da literatura. Na seção 4, o modelo matemático representativo deste problema é apresentado, o qual, por meio da reformulação de *Dantzig-Wolfe*, foi utilizado para geração dos limitantes inferiores. A seção 5 descreve em detalhes a heurística GRASP com *path relinking*. Os resultados obtidos nas instâncias de testes são apresentados na seção 6. As principais conclusões são comentadas na seção 7 e, na seção 8, a bibliografia consultada.

2. Descrição do Problema

A exploração e produção de petróleo em alto mar, conhecida como exploração *offshore*, necessita de uma grande quantidade de atividades de apoio marítimo, requerendo a utilização dos mais variados tipos de embarcações especializadas. Uma etapa importante do desenvolvimento de um campo petrolífero é a de lançamento de *risers* ou linhas de produção.

As linhas ou dutos submarinos podem ser flexíveis ou rígidas, e são de fundamental importância para o transporte da produção de petróleo das plataformas marítimas (*offshore*) para refinarias ou tanques de armazenagem situados em terra (*onshore*). Cabe destacar também as linhas que coletam a produção dos poços e abastecem diretamente uma unidade estacionária de produção, ou o fazem por meio de um *manifold* submarino, o qual está instalado no leito marinho e recebe várias linhas dos poços em sua adjacência – desta caixa de conexão sai uma linha única para a plataforma.

As embarcações responsáveis pelo lançamento dos dutos e sua interligação são as PLVs – *pipe layer vessels*, que são recursos críticos que uma empresa petrolífera tem que administrar. As embarcações realizam estas operações basicamente de duas maneiras distintas: i) os dutos são soldados enquanto a embarcação está operando no próprio local de lançamento; ii) os dutos, previamente soldados em terra, são armazenados em grandes estruturas tipo carretéis, e são dispostos no leito marinho a uma determinada velocidade. A segunda opção é mais vantajosa do ponto de vista de desempenho (velocidade), e por não comprometer as operações de soldagem em condições climáticas adversas. A figura 1 mostra um certo navio, capaz de lançar entre 4 a 7 km de dutos por dia. Entre outros recursos, esta embarcação dispõe de um eficiente sistema de posicionamento dinâmico, permitindo a realização de serviços em áreas congestionadas.

Para a instalação de uma linha, seja ela flexível ou rígida, existem diversas atividades necessárias para que isto se concretize. No início da operação a embarcação necessita carregar as linhas e os equipamentos que serão utilizados na operação de lançamento. O próximo passo consiste em deslocar-se até ao local de instalação, dando início ao lançamento. Dependendo da profundidade e da extensão do lançamento, pode ser necessário retornar novamente ao

porto para carregar material adicional. Existem casos de maior complexidade, em que é necessária a requisição de embarcações de apoio como, por exemplo, um rebocador (AHTS) ou uma embarcação dotada de sonda de observação (ROV).

Figura 1. PLV Solitaire (Fonte: http://www.allseas.com/uk/20/equipment/solitaire.html)

Quando uma empresa petrolífera realiza o planejamento operacional da sua frota de PLVs, considera como uma macro-tarefa todas as atividades acima mencionadas. Em geral, o setor responsável pelo planejamento e execução desta operação avalia quais embarcações da frota estão aptas para realizar cada macro-tarefa (doravante, denominado simplesmente de tarefa), e elaboram uma matriz de compatibilidade entre cada par embarcação-tarefa, estimando o tempo necessário para que a tarefa seja realizada.

As diversas tarefas de interligação e lançamentos de linhas que compõem a demanda que a empresa deverá atender são decorrentes dos projetos de desenvolvimento de novos campos. Cada tarefa a ser realizada tem um potencial de produção associado, o qual reflete o volume diário de produção esperado ao longo da vida útil do poço que será interligado. Postergar uma tarefa, para realizar outra, implica em adiar o retorno sobre o investimento realizado.

Para programar a frota, algumas restrições operacionais devem ser observadas, como a compatibilidade embarcação-tarefa e a data mais cedo em que a tarefa pode ser iniciada. Esta data reflete o instante de disponibilização de material no porto para realizar aquela tarefa, além da obtenção de licença ambiental e da aprovação em instâncias internas à empresa. A extensão do horizonte de planejamento também deve ser considerada, já que a empresa trabalha com uma visão de médio prazo, que no caso da operação de lançamento de *risers* e interligação constitui um período de 3 a 6 meses. É importante ressaltar, contudo, que este horizonte, em princípio, não é conhecido com precisão. Os técnicos responsáveis pelo setor estimam um prazo capaz de atender a demanda e fixam este horizonte como meta.

É importante destacar que, quando uma tarefa é planejada, os tempos de navegação já são incluídos no tempo total de execução. Uma vez que uma operação inicia e termina em uma base operacional da empresa (um porto), o tempo de deslocamento entre duas tarefas consecutivas é zero. Objetiva-se achar a sequência de execução das tarefas, de forma a minimizar as perdas financeiras pelo atraso no início da execução das mesmas em relação aos respectivos instantes de liberação. Se o horizonte de planejamento for excedido, incidirá uma penalidade grande a esta violação.

Quanto à caracterização da demanda, adota-se como hipótese que a mesma é conhecida, bem como a duração da execução das tarefas. No caso da oferta, os recursos empregados são as embarcações, que se diferenciam uma das outras geralmente pela capacidade de armazenagem e tipo de lançamento das linhas. Admite-se conhecida a composição da frota, os instantes e os locais em que as embarcações estão disponíveis.

Este problema tem semelhanças com o problema de programação de máquinas paralelas não-relacionadas, com instante de liberação das tarefas, minimizando a soma dos atrasos ponderados das tarefas. Dado que algumas embarcações não estão habilitadas a realizar determinados serviços e, pelo fato das embarcações poderem realizar as tarefas em

tempos diferentes, fica caracterizado o aspecto das máquinas serem não-relacionadas. Conforme anteriormente dito, o horizonte de planejamento não é conhecido com total precisão. Contudo, o valor imposto pelo corpo técnico da empresa, passa a ser uma restrição rígida, já que a empresa estabelece metas a serem atingidas dentro do período de planejamento de suas operações.

3. Revisão da Literatura

A programação requer a alocação de tarefas em um ou mais intervalo de tempo em um ou mais recursos (Brucker, 2007). Panwalkar et al. (1993) propuseram a heurística PSK (sigla indicando as iniciais dos autores) com o objetivo de minimizar o atraso médio no caso de máquina única. Esta regra ordena as tarefas aplicando a regra SPT (*shortest processing time*), em que as tarefas com menor tempo de processamento são priorizadas. A heurística proposta mostrou-se eficiente para casos em que a data de entrega é apertada. Koulamas (1997) estudou o problema de máquinas idênticas com o objetivo de minimizar o atraso médio, utilizando as ideias da heurística PSK, dando origem a heurística KPM.

Liaw et al. (2003) abordaram um problema de máquinas paralelas não relacionadas com o objetivo de minimizar o atraso total ponderado, em que mostraram as propriedades de um programa ótimo e também propuseram um algoritmo *branch and bound*. Shim e Kim (2007) também exploraram as propriedades de um programa ótimo e sugeriram um algoritmo *branch and bound* aplicado ao caso de máquinas paralelas idênticas com o objetivo de minimizar o atraso total. Alidaee e Rosa (1997) destacam um caso de máquinas idênticas com objetivo de minimizar o atraso total ponderado por meio da heurística MDD.

Cao et. al. (2005) estudaram uma variação importante dos problemas de máquinas paralelas, ao minimizar simultaneamente o custo pelo atraso no processamento das tarefas e os custos que as máquinas carregam, ou seja, quando uma determinada máquina é selecionada para processar uma tarefa, é incorrido um certo custo. Para maiores detalhamentos sobre as variações de problemas de máquinas paralelas, Cheng e Sin (1990) fazem uma revisão do estado-da-arte em pesquisas de máquinas paralelas.

3.1. Metaheurística Grasp

Segundo Feo e Resende (1989, 1995), a meta-heurística GRASP (*Greedy Randomized Adaptive Search Procedure*) trabalha com múltiplas soluções iniciais e, em cada iteração, duas fases devem ser realizadas: fase de construção e busca local. Na primeira fase é construída uma solução viável, por meio de um algoritmo guloso aleatório; na segunda fase, a vizinhança da solução gerada é explorada, até que um ótimo local seja atingido.

Inicialmente, é requerido uma lista ordenada de tarefas de acordo com algum critério. Em seguida, constrói-se uma lista restrita de candidatos (RCL), que tem como objetivo armazenar os melhores elementos que possam fazer parte da solução parcialmente gerada, a qual será usada para sorteio da "próxima" tarefa que será incorporada à solução (parcial). Uma vez sorteado, a lista é refeita e o processo continua até que todas as tarefas tenham sido programadas.

Em relação à definição do melhor tamanho da RCL, alguns autores têm desenvolvido diferentes estratégias como, por exemplo, a de Prais e Ribeiro (2000), que elaboraram uma abordagem conhecida como GRASP Reativo, em que se objetiva encontrar o melhor tamanho para a lista restrita de candidatos (RCL), ao invés de trabalhar com o mesmo tamanho em toda iteração. Boudia et al. (2007) comprovaram a eficácia da abordagem reativa em relação à lista de tamanho fixa, em um problema acoplado de produção e distribuição.

3.2. Path Relinking

A heurística *path relinking* (reconexão por caminhos), descrita originalmente por Glover (1996), e também em Glover et al. (2000), consiste em explorar o caminho ou trajetória entre duas soluções, uma denominada de solução inicial e a outra de solução guia. Em cada iteração é realizado um movimento na solução inicial, com o objetivo de aproximá-la da solução guia. Isto implica em introduzir atributos encontrados na solução guia na solução inicial, até que ambas as soluções sejam iguais. Esta trajetória pode ser compreendida como um processo de intensificação e é importante por permitir explorar espaços de solução neste caminho de aproximação entre as duas soluções. É possível que, ao longo deste processo, um novo ótimo local seja alcançado, conforme ilustra a figura 2, onde a solução inicial é dada por x' e a solução guia x''. Para chegar de x'a x'' são produzidas novas soluções: $x' = x(1), x(2), \ldots, x(r) = x''$, indicada pela linha pontilhada.

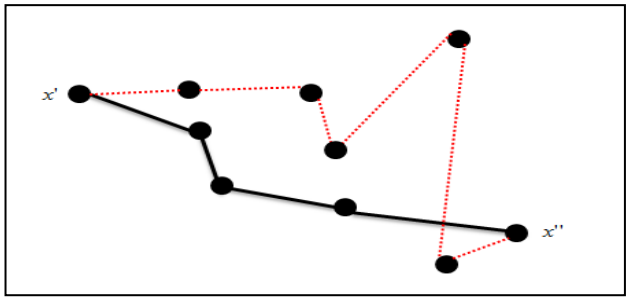

Figura 2. Exemplo de *path relinking*

O *path relinking* possui atributos ou formas distintas de implementação. Resende et al. (2010a) discutem a "atualização estática", em que apenas o conjunto de elite é preenchido e atualizado ao longo das iterações da aplicação do GRASP. Ao final, o *path relinking* é aplicado entre as soluções de elite, com o intuito de explorar o espaço que há entre cada par de soluções de elite. Na "atualização dinâmica", para cada solução gerada pelo GRASP escolhe-se, aleatoriamente, uma solução dentre o conjunto de elite, e o *path relinking* é aplicado. Em ambas as estratégias (estática ou dinâmica), a solução resultante é sempre submetida a um processo de busca local.

Resende e Werneck (2004) mesclaram estas duas estratégias. Ao longo das iterações, toda solução gerada foi submetida ao *path relinking*. E, ao final, a trajetória entre cada par de soluções do conjunto de elite foi explorada, usando uma estratégia evolucionária, denominada de EvPR (*evolutionary path relinking*): cada nova solução gerada pelo *path relinking*, após passar por determinados critérios de aceitação, era adicionada a uma nova população de soluções. As gerações foram sucessivamente criadas com a aplicação do *path relinking*, até que nenhum indivíduo gerado foi capaz de melhorar a solução global.

Resende e Ribeiro (2003) destacam outras variações importantes: *forward relinking:* a trajetória é construída partindo da solução inicial x', até alcançar a solução guia x''; *backward relinking:* esta estratégia é o inverso da anterior, ou seja, a solução inicial neste caso é x'' e a guia x'; *mixed relinking:* duas trajetórias são simultaneamente exploradas, a primeira iniciando em x' e a segunda em x'', até que se encontrem.

Em Resende et. al. (2010b) é discutido o *randomized path relinking (RPR)*. A versão usual do *path relinking* adota um critério guloso na escolha do melhor movimento em cada iteração, na construção da trajetória que ligará a solução inicial à guia. Isto faz com que, certamente, um único caminho será gerado, o que poderá impedir a obtenção de soluções melhores. O RPR introduz uma lista restrita de candidatos, na qual os melhores movimentos são inseridos e um deles, aleatoriamente, é escolhido. Isto garante que trajetórias diferentes serão geradas, aumentando a possibilidade de atingir novas soluções, eventualmente, de boa qualidade.

Ribeiro e Rosseti (2007), Ribeiro et al. (2009) trabalharam a ideia conhecida como *time-to-target solution* ou *value*, em que é determinada a probabilidade do algoritmo encontrar uma solução igual ou superior a uma dada solução (*target solution*) dentro de um determinado tempo. Os autores também abordam estratégias de implementação paralela em metaheurísticas com o objetivo de acelerar a busca e resolver problemas grandes. Duas abordagens foram apresentadas: a paralelização independente, em que os *threads* não trocam nenhuma informação e a paralelização cooperativa onde a informação é compartilhada e usada pelos outros *threads*.

4. Modelagem Matemática

Para o presente problema, foi formulado um modelo de programação linear inteira mista, usando as ideias apresentadas em van den Akker, et al. (2000), para um problema de máquina simples com função objetivo regular. Esta formulação considera que os parâmetros são conhecidos e inteiros, e utiliza uma escala de tempo discretizada, considerando que o período *t* inicia no instante de tempo *t-1* e acaba no instante *t*. É utilizada a notação proposta por Unlu e Masson (2010).

Conjuntos

J	Conjunto de tarefas	$j = 1, \dots, n$
M	Conjunto de embarcações	$i = 1, \dots, m$

Parâmetros

Horizonte de planejamento $= T$ (índice t)

r_j	Instante de liberação da tarefa j
p_j^i	Tempo de execução da tarefa j pela embarcação i
c_{jt}	Penalidade associada a tarefa j, quando é iniciada no instante t
a_{ij}	Parâmetro binário, que será igual a 1, se a tarefa j for compatível com a embarcação i

Variáveis de Decisão

$x_{ij}^t = 1$ Se a tarefa $j \in J$ é designada para a embarcação $i \in M$ no instante t; 0, em caso contrário.

Modelo

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{t=r_j}^{T-p_j^i+1} c_{jt}\, x_{ij}^t \qquad (1)$$

Sujeito a:

$$\sum_{i=1}^{m} \sum_{t=r_j}^{T-p_j^i+1} a_{ij}\, x_{ij}^t = 1 \qquad \forall\, j \qquad (2)$$

$$\sum_{j=1}^{n} \sum_{s=t-p_j^i+1}^{t} x_{ij}^s \leq 1 \qquad \forall\, t, i \qquad (3)$$

$$x_{ij}^t \in \{0,1\} \qquad \forall\, j, t, i \qquad (4)$$

A equação (1) é a função objetivo do problema e contempla a soma dos atrasos ponderados de todas as tarefas. A restrição (2) garante que toda a demanda seja atendida, por meio de embarcações compatíveis. A restrição (3) garante que, em cada instante, não seja violada a capacidade da embarcação. A equação (4) impõe que as variáveis sejam binárias.

Este modelo foi utilizado para gerar limitantes inferiores para os cenários de teste, por meio da técnica de geração de colunas. Utilizou-se a estrutura apresentada em van den Akker, et al. (2000), em que o problema foi reformulado utilizando-se a decomposição de *Dantzig-Wolfe*. O problema mestre restrito consistiu na seleção dos *pseudo-schedules* que cobrissem toda a demanda (conforme requerido pela restrição 2) a um mínimo custo. O subproblema consistiu em resolver um problema de caminho mínimo em uma rede acíclica

para cada embarcação, o qual tinha como produto um *pseudo-schedule*, onde as tarefas poderiam, eventualmente, ocorrer mais de uma vez em uma mesma rota, desde que não violasse a capacidade do recurso. As mudanças introduzidas, em relação ao trabalho de referência, foram a resolução de um subproblema por recurso (navio), a consideração do instante de liberação das tarefas, e a eventual incompatibilidade entre uma tarefa e uma embarcação.

5. Heurísticas Propostas

Para o presente trabalho foi realizada uma implementação da heurística GRASP e, posteriormente, adicionou-se o mecanismo de *path relinking*. A versão inicial segue a estrutura apresentada na figura 3.

```
Procedimento GRASP(maxIterações)
1     LerDados()
2     Para k=1,...,maxIterações faça
3           Atualiza configuração k
4           Para j=1,...,n faça
5              Atualiza lista restrita de candidatos de tamanho l(k)
6              Sorteia uma tarefa i em RCL(k)
7              Insere a tarefa i na melhor posição viável de inserção
8           Fim
9           BuscaLocal()
10          Atualiza MelhorSolução
11    Fim
12    Retorna MelhorSolução
Fim GRASP
```

Figura 3 – Estrutura básica do GRASP

Em cada iteração será atualizada em uma determinada "configuração" do algoritmo. Esta configuração consiste em três elementos: o tamanho da lista restrita de candidatos (l), a regra de ordenação que será aplicada às tarefas, e a atualização da semente do gerador de número aleatório. Quanto ao tamanho da lista restrita, os seguintes valores serão adotados: 2, 3, 4, 5, 6 e n. Isto permitirá trabalhar com regras aproximadamente gulosas (quando l=2), ou com busca totalmente aleatória (quando l=n).

Quanto à regra de ordenação, ao invés de usar uma regra dinâmica em que, após um determinado sorteio as tarefas remanescentes são reclassificadas, optou-se por utilizar uma regra fixa de ordenação. Desta forma, quando uma tarefa pertencente à lista restrita é escolhida, a próxima tarefa a compor a lista restrita já será conhecida. Seis regras de ordenação das tarefas foram testadas, sendo que cada regra possui dois critérios ordenação. Em caso de empate, após a aplicação do primeiro critério, o segundo critério servirá para desempatar. Se persistir o empate, então as tarefas serão ordenadas pelo seu índice. As regras desenvolvidas foram: (1) Menor Compatibilidade e Menor Instante de Liberação; (2) Menor Compatibilidade e Maior Penalidade; (3) Menor Instante de Liberação e Menor Compatibilidade; (4) Menor Instante de Liberação e Maior Penalidade; (5) Maior Penalidade e Menor Instante de Liberação; (6) Maior Penalidade e Menor Compatibilidade. A idéia de utilizar "menor compatibilidade" é para que tarefas com poucos navios compatíveis, isto é, aptos para realizarem o serviço, sejam priorizadas, de forma a facilitar a geração de soluções viáveis.

Quanto à semente, para cada combinação acima descrita (tamanho de lista restrita e regra de ordenação), serão testadas 100 diferentes sementes, perfazendo um total de 6 x 6 x 100 = 3.600 iterações. Após a leitura dos dados na linha 1, a configuração da iteração é atualizada, e a fase construtiva é aplicada entre as linhas 4 e 8, até que todas as tarefas tenham sido programadas. Cabe destacar que, após a atualização da lista restrita (linha 5) e o sorteio da próxima tarefa a ser programada (linha 6), realiza-se um procedimento, descrito na linha 7,

de identificação da melhor posição de inserção. Isto consiste em testar, para cada embarcação compatível com a tarefa sorteada, todas as possíveis posições de inserção, verificando aquela que gera o menor acréscimo de custo à solução parcialmente gerada.

Após alocar todas as tarefas, uma rotina de busca local é chamada na linha 9. Esta rotina aplica os operadores de inserção (*relocate*) e troca (*swap*). O primeiro operador remove a tarefa de sua posição atual e insere-a em todas as demais posições das embarcações compatíveis. O segundo operador realiza a troca entre cada par de tarefas, entre embarcações que sejam compatíveis. Em cada iteração da busca, realiza-se o movimento que proporciona a maior redução na função objetivo. Quando nenhuma melhoria for encontrada, uma solução ótima local terá sido encontrada e o algoritmo encerra.

Por último, caso a solução gerada seja melhor que a "MelhorSolução", esta será atualizada. O GRASP continuará até que o limite de iterações dado por "maxIterações" seja atingido. A versão implementada do *path relinking*, apresentada na figura 4, utiliza a estratégia de "atualização dinâmica", ou seja, cada solução gerada pelo GRASP é combinada com uma solução sorteada dentre o conjunto de soluções de elite e é submetida à rotina de *path relinking*. Este procedimento só será realizado após o conjunto de elite estar totalmente preenchido. Na presente implementação, adotou-se um conjunto de elite com tamanho igual a 10.

```
Procedimento GRASP_PathRelinking(maxIterações)
1     LerDados()
2     Para k=1,...,maxIterações faça
3           Atualiza configuração k
4           Para j=1,...,n faça
5              Atualiza lista restrita de candidatos de tamanho l(k)
6              Sorteia uma tarefa i em RCL(k)
7              Insere a tarefa i na melhor posição viável de inserção
8           Fim
9           BuscaLocal()
10          PathRelinking(sol_elite, sol_GRASP)
11          Atualiza ConjuntoElite
12          Atualiza MelhorSolução
13    Fim
14    Retorna MelhorSolução
Fim GRASP
```

Figura 4 – Estrutura básica do GRASP com *path relinking*

A rotina do Path Relinking inicia ajustando as soluções de referências (inicial e guia), dependendo da estratégia escolhida (*forward, backward, mixed*). Se for a estratégia *backward* ou *mixed*, a solução guia será o ponto de partida, isto é, fará o papel da solução inicial, a qual é progressivamente modificada, até que a outra solução seja atingida. A exceção ocorrerá se a solução inicial informada já for melhor que a solução de elite (guia). No caso da estratégia *forward*, a busca ocorre da solução inicial para a solução guia, e não há necessidade de inversão.

É importante destacar que o presente problema envolve diferentes embarcações. Esta característica torna imprescindível a resolução de um problema de pareamento (*matching*), semelhante do que foi proposto por Ho e Gendreau (2006). Isto consiste em identificar qual a rota da solução guia que é mais próxima de uma dada rota da solução inicial. Após todas as rotas da solução inicial serem avaliadas, aplica-se um critério guloso, estabelecendo o pareamento às rotas mais similares, até que todas as rotas recebam a indicação de uma rota equivalente.

A aplicação do *path relinking* consistirá em realizar os movimentos que inserem as tarefas que estão fora de suas rotas correspondentes (na solução guia), na solução inicial. Isto pode ser feito tanto por meio de um movimento de inserção (*relocate*), quanto por meio de um movimento de troca (*swap*) entre duas tarefas. Após todos os possíveis movimentos serem

avaliados, o melhor movimento será executado. É importante lembrar que o melhor movimento poderá piorar o valor da função objetivo.

Outro aspecto a ser comentado é que uma tarefa, ao ser transferida para uma nova rota, é posicionada na melhor posição de inserção. Contudo, após todas as tarefas terem sido enviadas às rotas correspondentes da solução guia, pode acontecer que algumas tarefas não estejam nas mesmas posições que em suas rotas de referência. Então, um procedimento de inserção/remoção intrínseco à rota é realizado, reposicionando as tarefas, uma a uma, até que as soluções inicial e guia sejam iguais.

Se a estratégia do *path relinking* for a *mixed*, então duas trajetórias estarão sendo exploradas, uma a partir de cada solução. Neste caso, uma forma eficiente de implementação é a inversão, a cada iteração, de quem é a solução inicial e quem é a guia, conforme sugerido por Resende et al. (2010b).

Por último, a versão *Randomized Path Relinking (RPR)* consiste em não necessariamente adotar o melhor movimento possível a partir de um dado estágio do *path relinking*. Antes, os custos associados aos movimentos são ordenados em ordem crescente do acréscimo causado na função objetivo. Seja z_{min} o custo do movimento que proporciona o menor acréscimo e z_{max} o custo associado ao movimento que proporciona o maior acréscimo na função objetivo. Seja também α um número aleatório sorteado entre 0 e 1, o qual estará fixo durante uma iteração do Path Relinking. Os movimentos que estarão presentes na lista restrita serão todos aqueles cuja variação z na função objetivo estejam na faixa dada por $[z_{min}, z_{min} + \alpha(z_{max} - z_{min})]$.

Após a execução da rotina do *path relinking* na linha 10, o conjunto de elite é atualizado. Uma solução só será aceita no conjunto de elite se o valor da função objetivo for inferior à solução de pior qualidade pertencente ao conjunto. Quanto à determinação de qual solução que será excluída, optou-se por substituir a solução com valor da função objetivo superior à solução candidata, e que possua a menor distância a esta solução. Isto fará com que a solução mais próxima ou similar será excluída, contribuindo para aumento da diversidade entre as soluções de elite. A medida de distância entre duas soluções é dada pelo número de vezes que uma tarefa possui uma tarefa sucessora diferente do que acontece na outra solução.

6. Testes Computacionais

Foi desenvolvido um gerador de cenários baseado em Crauwels et al. (1998), que fornece diferentes instâncias para o problema de máquina única (*single machine*), o qual foi devidamente adaptado para o problema em estudo. Em todos os testes realizados, foram verificados o *gap* das soluções obtidas, o qual é definido como a distância relativa da função objetivo gerada pelo método geração de colunas em relação à função objetivo da melhor solução obtida, calculado da seguinte forma: $GAP = 100(QM - GC)/QM$, onde $QM =$ Função objetivo do método empregado; GC = Função objetivo do método geração de colunas.

Os problemas testes foram gerados da seguinte forma: para cada tarefa j um tempo de processamento p_j inteiro foi gerado empregando-se uma distribuição uniforme entre $[1, 100]$ e, para a penalidade, utilizou-se a faixa [1,10]. O horizonte de planejamento é denotado por $\frac{\sum_{j=1}^{n} p_j}{v}$ em que v é o número de embarcações. A dificuldade do problema depende do fator RDD (*relative range of due dates*) e do TF (*tardiness factor*). Os valores de RDD e TF são {0,2; 0,4; 0,6; 0,8; 1,0}. O instante de liberação é calculado em duas etapas:

$$dp_j = \left[\sum_{j=1}^{n} p_j(1 - RDD/2), \sum_{j=1}^{n} p_j(1 + RDD/2)\right] \quad (5)$$

$$r_j = max\left(0, dp_j - (min_{k=1}^{n}\{dp_k\})\right) \quad (6)$$

Na equação (5) tem-se o cálculo do fator dp_j (instante de liberação preliminar) e, na equação (6) o instante de liberação efetivo. A data de finalização da tarefa (due date) é dada por $d_j = r_j + p_j$. Cada fator TF é combinado com todos os RDDs, o que totaliza 25

instâncias teste. Foram geradas 25 instâncias testes 4 embarcações e 30 tarefas ($4mx30n$), 5 embarcações e 40 tarefas ($5mx40n$) e 6 embarcações e 50 tarefas ($6mx50n$).

6.1. Análise dos Resultados

Os algoritmos foram processados em uma estação de trabalho com processador intel® Core™ i7 2.80GHz e 16374 MB de memória RAM DDR3 SDRAM para executar o programa codificado em linguagem C++. Os resultados estão apresentados na tabela 1, em que: a coluna 1 apresenta o número da instância; a coluna 2, o valor da relaxação linear da função objetivo, obtida pelo método de geração de colunas; das colunas 3 em diante, apresenta-se o gap, tal qual acima descrito, da aplicação dos seguintes algoritmos: GRASP; GRASP + Forward PR (PR Fw); GRASP + Backward PR (PR Bw); GRASP + Mixed PR (PR Mxd); GRASP + Randomized Forward PR (PR Fw Rnd); GRASP + Randomized Backward PR (PR Bw Rnd); GRASP + Randomized Mixed PR (PR Mxd Rnd).

Por último, realizou-se uma implementação *multi-threading* do *path relinking*, em que ao invés de selecionar uma única forma de exploração dos caminhos (PR Fw, PR Bw, Pr Mxd, etc.), as 6 formas de exploração foram testadas simultaneamente, sem penalizar o tempo de processamento. Cada solução gerada, por meio das 6 estratégias distintas foi submetida ao conjunto de elite.

A eficiência de combinar as 6 regras simultaneamente pode ser vista na figura 5, na qual esta versão modificada é comparada com a versão do GRASP puro, para a qual uma distribuição de probabilidade para que uma dada solução seja alcançada. De acordo com Resende e Ribeiro (2003), para medir o tempo que os algoritmos empregam para atingir uma solução (*time-to-target value*), é feito uma distribuição empírica em que é fixado uma solução para ser atingida (*target*) e em seguida executa-se cada algoritmo T independente vezes, gravando o tempo que o algoritmo levou para encontrar uma solução melhor ou igual a solução fixada. Para cada algoritmo é associado com o *i-ésimo* tempo ordenado t_i a probabilidade $p_i = \left(i - \frac{1}{2}\right)/T$ e plota-se os pontos $z_i = (t_i, p_i)$ para $i = 1, \ldots, T$. A tabela 2 compara os tempos médios de processamento das heurísticas e do método geração de colunas.

Tabela 1 – Resultado das instâncias de 30, 40 e 50 tarefas

	Gap % [30 Tarefas x 4 Navios]								Gap % [40 Tarefas x 5 Navios]								Gap % [50 Tarefas x 6 Navios]							
Instância	FOCG	GRASP	PR Fw	PR Bw	PR Mxd	PR Fw Rnd	PR Bw Rnd	PR Mxd Rnd	FOCG	GRASP	PR Fw	PR Bw	PR Mxd	PR Fw Rnd	PR Bw Rnd	PR Mxd Rnd	FOCG	GRASP	PR Fw	PR Bw	PR Mxd	PR Fw Rnd	PR Bw Rnd	PR Mxd Rnd
1	15.417	0,78%	0,73%	0,77%	0,77%	0,73%	0,76%	0,73%	12.354	0,87%	0,87%	0,87%	0,86%	0,86%	0,86%	0,87%	19.685	0,88%	0,87%	0,86%	0,87%	0,89%	0,92%	0,89%
2	9.059	1,53%	1,54%	1,52%	1,54%	1,55%	1,54%	1,54%	11.196	0,80%	0,74%	0,82%	0,75%	0,79%	0,77%	0,75%	11.768	1,43%	1,43%	1,37%	1,42%	1,47%	1,39%	1,40%
3	6.694	2,22%	2,15%	2,15%	2,15%	2,15%	2,15%	2,15%	6.130	2,76%	3,16%	2,76%	2,76%	3,08%	2,99%	2,98%	12.450	1,40%	1,34%	1,14%	1,11%	1,25%	1,22%	1,20%
4	2.107	3,13%	2,59%	2,59%	2,50%	3,61%	2,59%	2,50%	2.698	5,66%	5,33%	4,56%	4,36%	4,60%	5,43%	5,53%	7.817	2,80%	2,89%	2,49%	2,58%	2,64%	2,73%	2,53%
5	1.798	9,10%	9,10%	9,15%	9,10%	9,15%	9,15%	9,33%	789	6,18%	4,48%	4,48%	4,48%	6,07%	6,07%	3,66%	2.498	7,24%	6,83%	5,49%	6,23%	6,69%	6,83%	7,03%
6	6.326	0,28%	0,25%	0,13%	0,13%	0,13%	0,13%	0,14%	9.602	0,40%	0,45%	0,37%	0,37%	0,38%	0,38%	0,38%	22.599	0,69%	0,68%	0,69%	0,68%	0,71%	0,71%	0,71%
7	4.366	2,50%	2,50%	2,50%	2,50%	2,50%	2,50%	2,57%	5.918	1,63%	1,53%	1,20%	1,23%	1,30%	1,60%	1,27%	15.720	0,58%	0,63%	0,59%	0,59%	0,48%	0,48%	0,63%
8	2.309	3,31%	3,31%	3,31%	3,31%	3,31%	3,31%	3,31%	4.538	0,94%	0,94%	0,59%	0,59%	0,94%	0,59%	0,59%	11.165	2,16%	1,70%	1,98%	1,53%	1,68%	1,82%	1,66%
9	3.542	2,93%	2,93%	2,93%	2,93%	2,93%	2,93%	2,93%	3.370	3,69%	3,22%	2,97%	3,16%	3,24%	3,38%	2,99%	9.334	2,54%	2,62%	1,76%	2,16%	2,15%	2,98%	2,15%
10	1.711	3,82%	4,25%	3,82%	3,66%	3,82%	3,22%	4,57%	2.169	8,52%	7,62%	7,74%	6,55%	7,23%	9,21%	6,55%	2.802	3,74%	3,08%	3,58%	2,57%	3,91%	2,88%	2,84%
11	9.481	0,47%	0,47%	0,47%	0,47%	0,47%	0,47%	0,48%	16.600	0,47%	0,47%	0,47%	0,47%	0,49%	0,50%	0,47%	18.094	0,33%	0,32%	0,29%	0,34%	0,33%	0,29%	0,32%
12	8.908	0,69%	0,82%	0,70%	0,76%	0,76%	0,72%	0,60%	8.735	0,95%	1,08%	1,08%	1,01%	1,06%	0,94%	1,14%	15.734	0,87%	0,73%	0,64%	0,64%	0,84%	0,71%	0,76%
13	5.634	3,71%	3,84%	3,97%	3,71%	3,71%	3,84%	3,71%	6.746	2,36%	2,47%	1,89%	2,53%	2,37%	2,53%	2,58%	9.318	2,10%	2,05%	1,93%	2,10%	1,99%	1,90%	2,23%
14	4.388	4,84%	5,02%	5,04%	4,98%	5,06%	4,98%	5,06%	4.918	4,11%	3,64%	3,66%	3,63%	4,10%	3,93%	4,06%	14.299	2,37%	2,42%	2,52%	2,62%	2,62%	2,41%	2,10%
15	3.016	5,34%	5,22%	5,22%	5,10%	5,13%	5,04%	5,04%	5.939	3,67%	3,46%	3,07%	3,51%	3,51%	3,34%	3,46%	7.972	3,05%	2,96%	2,02%	3,28%	3,45%	2,72%	2,83%
16	8.652	1,13%	1,06%	1,06%	1,09%	1,07%	1,07%	1,07%	16.139	1,22%	1,19%	1,19%	1,18%	1,18%	1,19%	1,19%	24.191	0,83%	0,84%	0,83%	0,84%	0,82%	0,84%	0,84%
17	7.408	1,28%	1,25%	1,25%	1,25%	1,27%	1,25%	1,25%	10.847	1,65%	1,65%	1,61%	1,61%	1,65%	1,67%	1,62%	18.571	1,46%	1,52%	1,45%	1,46%	1,52%	1,48%	1,51%
18	7.484	2,36%	2,25%	2,18%	2,21%	2,35%	2,37%	2,25%	9.623	2,30%	2,27%	2,18%	2,18%	2,26%	2,15%	2,15%	17.025	1,29%	1,24%	1,24%	1,23%	1,26%	1,28%	1,32%
19	6.461	2,18%	1,99%	1,99%	2,00%	1,99%	1,99%	1,97%	6.392	2,01%	2,01%	2,01%	2,01%	2,01%	2,01%	2,01%	11.286	2,25%	2,32%	2,08%	2,34%	2,17%	2,44%	2,38%
20	1.601	4,07%	4,07%	4,07%	4,07%	4,07%	4,07%	4,07%	4.379	4,12%	4,16%	4,05%	4,37%	4,35%	4,45%	4,03%	11.832	2,20%	2,26%	2,02%	2,22%	2,29%	2,26%	2,27%
21	8.986	1,48%	1,47%	1,47%	1,47%	1,47%	1,47%	1,47%	10.728	1,69%	1,69%	1,69%	1,69%	1,69%	1,70%	1,69%	21.524	1,13%	1,13%	1,13%	1,13%	1,13%	1,15%	1,15%
22	11.008	1,62%	1,62%	1,62%	1,62%	1,62%	1,62%	1,62%	13.275	1,27%	1,28%	1,27%	1,28%	1,27%	1,27%	1,27%	18.114	1,33%	1,34%	1,32%	1,33%	1,36%	1,35%	1,32%
23	14.882	1,04%	1,04%	1,04%	1,04%	1,04%	1,04%	1,04%	13.592	1,19%	1,19%	1,18%	1,18%	1,19%	1,19%	1,20%	20.472	1,42%	1,41%	1,39%	1,41%	1,40%	1,41%	1,41%
24	8.002	1,47%	1,47%	1,42%	1,42%	1,42%	1,42%	1,42%	10.941	1,60%	1,61%	1,48%	1,65%	1,60%	1,62%	1,59%	20.639	1,14%	1,16%	1,13%	1,14%	1,16%	1,16%	1,15%
25	8.466	1,79%	1,78%	1,78%	1,78%	1,79%	1,78%	1,78%	10.350	2,55%	2,46%	2,46%	2,63%	2,50%	2,50%	2,50%	14.864	1,35%	1,41%	1,43%	1,28%	1,41%	1,43%	1,24%
	Média	**2,52%**	**2,51%**	**2,49%**	**2,46%**	**2,52%**	**2,46%**	**2,50%**	**Média**	**2,50%**	**2,36%**	**2,23%**	**2,24%**	**2,39%**	**2,49%**	**2,26%**	**Média**	**1,86%**	**1,81%**	**1,65%**	**1,72%**	**1,82%**	**1,79%**	**1,75%**

Tabela 2 – Comparação dos tempos de CPU dos métodos propostos

Tempo Médio de CPU em Segundos								
n x m	Geração de Colunas	GRASP	GRPR_Fw	GRPR_Bw	GRPR_Mixed	GRPR_FwRnd	GRPR_BwRnd	GRPR_MixedRnd
30 x 4	1,0000	174,8879	186,0704	186,6314	198,6712	186,6466	186,3094	195,8255
40 x 5	2,3200	488,6208	515,6409	515,6367	543,3603	518,0652	515,8389	542,7824
50 x 6	4,0000	1185,8061	1229,7670	1226,4200	1270,6215	1224,7536	1225,8483	1272,5372

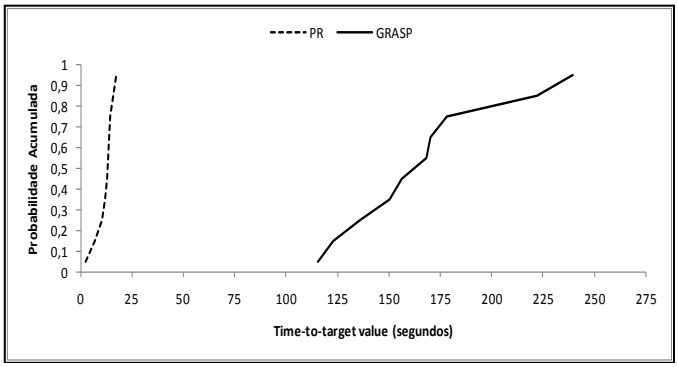

Figura 5 – Tempo para alcançar uma dada solução (GRASP x PR)

7. Conclusões e Futuros Trabalhos

A presente pesquisa estudou um problema real de programação de embarcações especializadas em operações de lançamento de *risers*, o qual foi modelado e resolvido como um problema de máquinas paralelas não-relacionadas. Foi adotada a abordagem heurística, sendo testado o GRASP com *path relinking*. Para avaliação dos resultados, foram utilizados limitantes inferiores dados pela versão linear do método de geração de colunas. Foi comprovado que as soluções fornecidas são de boa qualidade, tendo em vista que os *gaps* ficaram inferiores a 2,5%. A aplicação do *path relinking* ajudou a melhorar a qualidade das soluções. Em todas as instâncias, o GRASP com *path relinking*, com suas várias modalidades (*forward*, *backward* x *mixed* & *randomized*), em média, foi mais eficaz. As instâncias mais difíceis de resolver são as com $TF = 0{,}2$ e $0{,}4$, que nos mostra que quanto mais a data de entrega for apertada, a solução tende a não ser tão boa quanto as geradas pelos fatores $0{,}6$; $0{,}8$ e $1{,}0$.

Em relação a melhor regra de ordenação, as que obtiveram as melhores soluções foram respectivamente (1), (2) e (4). Isto mostra que é vantajoso utilizar várias regras de ordenação na fase de construção do GRASP ao invés de fixar apenas uma. Por meio da figura 5, pode-se enfatizar a eficiência do GRASP com *path relinking* frente ao GRASP puro por meio do *time-to- target value*, em que é exemplificada a probabilidade do PR encontrar uma solução em menos de 18 segundos sendo de 95%, ao passo que o GRASP puro, possui uma probabilidade de 5% para encontrar uma solução em menos de 115 segundos. Observou-se grande benefício com o processamento *multi-threading*, o qual poderá ser estendido para que uma solução gerada pelo GRASP seja combinada com mais de uma solução de elite, em cada iteração. Por último, a exploração da trajetória entre as soluções do conjunto de elite, após o término das iterações do GRASP, poderão gerar resultados promissores.

8. Referências

Alidaee, B. e Rosa, D. (1997), Scheduling parallel machines to minimize total weighted and unweighted tardiness, *Computers and Operations Research*, 24, 775-788.

Boudia, M.; Louly, M. A. O. e Prins, C. (2007), A reactive GRASP and path relinking for a combined production-distribution problem, *Computers & Operations Research*, 34, 3402-3419.

Brucker, P, *Scheduling Algorithms,* Springer, Berlin, 2007.

Cao, D.; Chen, M. e Wan, G. (2005), Parallel machine selection and job scheduling to minimize machine cost and job tardiness, *Computers & Operations Research*, 32, 1995-2012.
Cheng, T. C. E. e Sin, C. C. S. (1990), A state-of-the-art review of parallel-machine scheduling research, *European Journal of Operational Research*, 47, 271–292.
Crauwels, H. A. J, Potts, C. N. e Van wassenhove, L. N. (1998), Local search heuristics for the single total weighted tardiness scheduling problem, *Informs Journal on Computing*, 10, 341-350.
Feo, T. A. e Resende, M. G. C. (1989), A probabilistic heuristic for a computationally difficult set covering problem, *Operations Research Letter*, 8, 67-71.
Feo, T. A. e Resende, M. G. C. (1995), Greedy randomized adaptive search procedures, *Journal of Global Optimization*, 6, 109-133.
Glover, F., Tabu search and adaptive memory programming – advances, applications and challenges, em Barr, R. S.; Helgason, R.V. e Kennington, J. L. (Eds.), *Interfaces in Computer Science and Operations Research*, Kluwer Academic Publishers, Boston, 1-75, 1996.
Glover, F.; Laguna, M. e Martí, R. (2000), Fundamentals of scatter search and path relinking, *Control and Cybernetics*, 29, 653-684.
Ho, S. C. e Gendreau, M. (2006), Path relinking for the vehicle routing problem, *Journal of Heuristics*, 12, 55-72.
Koulamas, C. (1997), Decomposition and hybrid simulated annealing heuristics for the parallel-machine total tardiness problem, *Naval Research Logistics*, 44, 109-125.
Liaw, C-F., Lin, Y-K., Cheng, C-Y. e Chen, M. (2003), Scheduling unrelated parallel machines to minimize total weighted tardiness, *Computers & Operations Research*, 30, 1777-1789.
Morton, T. E. e Pentico, D. W, *Heuristic scheduling systems: with applications to production systems and project management*, John Wiley & Sons, New York, 1993.
Panwalkar, S. S.; Smith, M. L. e Koulamas, C. (1993), A heuristic for the single machine tardiness problem, *European Journal of Operational Research*, 70, 304-310.
Pinedo, M, *Scheduling: theory, algorithm, and systems*, Prentice-Hall, New Jersey, 2002.
Prais, M. e Ribeiro, C. C. (2000), Reactive GRASP: an application to a matrix decomposition problem in TDMA traffic assignment, *Informs Journal on Computing*, 12, 164-176.
Resende, M. G. C. e Ribeiro, C. C. (2003), Grasp with path relinking: recent advances and applications, *AT & T Labs Technical Report TD-5TU726*, 1-24.
Resende, M. G. C. e Werneck, R.F. (2004), A hybrid heuristic for the p-median problem, *Journal of Heuristics*, 10, 59-88.
Resende, M. G. C.; Martí, R.; Gallego, M. e Duarte, A. (2010a), Grasp and path relinking for the max-min diversity problem, *Computers & Operations Research*, 37, 498-508.
Resende, M. G. C.; Ribeiro, C. C.; Glover, F. e R. Martí, *Scatter search and path-relinking: Fundamentals, advances, and applications*, em Gendreau, M. e Potvin, J.-Y. (Eds.), *Handbook of Metaheuristics*, Springer, New York, 87-107, 2010b.
Ribeiro, C. C. e Rosseti, I. (2007), Efficient parallel cooperative implementations of GRASP heuristics, *Parallel Computing*, 33, 21-35.
Ribeiro, C. C.; Rosseti, I. e Vallejos, R., On the use of run time distributions to evaluate and compare stochastic local search algorithms em Stützle, T.; Birattari, M. e Hoos, H.H. (Eds.), *Lecture Notes in Computer Science*, Springer-Verlag, Berlin Heidelberg, 16-30, 2009.
Shim, S-O. e Kim, Y-D. (2007), Scheduling on parallel identical machines to minimize total tardiness, *European Journal of Operational Research*, 177, 135-146.
Unlu, Y. e Mason, S. J. (2010), Evaluation of mixed integer formulations for non-preemptive parallel machine scheduling problems, *Computers & Industrial Engineering*, 58, 785-800.
Van den Akker, J. M.; Hurkens, C. A. J. e Savelsbergh, M. W. P. (2000), Time-indexed formulations for machine scheduling problem: column generation, *INFORMS Journal on Computing*, 12, 111-124.