SCATTER SEARCH COM ANINHAMENTO DE CONJUNTOS REFERÊNCIAS APLICADO AO PROBLEMA DE ESCALONAMENTO DE PROJETOS COM RESTRIÇÃO DE RECURSOS E MÚLTIPLOS MODOS DE PROCESSAMENTO

Leandro Geraldo Hoehene
FAESA – Unidade de Computação e Sistemas
Rua Anselmo Serrat, 199, Ilha de Monte Belo, Vitória – ES, CEP 29040-410
leandrofaesa@gmail.com

Luiz Carlos Chaves Küster
FAESA – Unidade de Computação e Sistemas
Rua Anselmo Serrat, 199, Ilha de Monte Belo, Vitória – ES, CEP 29040-410
lckuster@hotmail.com

Luciano Lessa Lorenzoni
FAESA – Unidade de Computação e Sistemas
Rua Anselmo Serrat, 199, Ilha de Monte Belo, Vitória – ES, CEP 29040-410
luciano@faesa.br

IFES – COMAT
Av. Vitória, 1729, Jucutuquara, Vitória – ES, CEP 29040-780
lllorenzoni@ifes.edu.br

RESUMO

Este artigo apresenta o problema de escalonamento de projetos com restrição de recursos e múltiplos modos de processamento. Devido a grande complexidade em se determinar a melhor solução para esse problema, aplicou-se a meta-heurística *Scatter Search*. Nos experimentos percebeu-se a necessidade de se modificar o algoritmo adicionando alguns métodos não contidos na lógica original do *Scatter Search*. Os resultados obtidos, de um modo geral, foram satisfatórios e encontram-se no mesmo nível dos melhores resultados apresentados pela literatura.

Palavras chave: Scatter Search; otimização combinatorial; problema de escalonamento com restrição de recursos e múltiplos modos de processamento.

ABSTRACT

This article presents the problem of scheduling projects with resource constraints and multiple modes of processing. Due to the complexity in determining the best solution to this problem, we applied meta-heuristics Scatter Search. In the experiments, revealed the need to modify the algorithm by adding some methods not contained in the original logic of Scatter Search. The results obtained, in general, were satisfactory and are on par with the best results reported in the literature.

Keywords: Scatter Search; combinatorial optimization; scheduling problem with resource constraints and multiple modes of processing.

1 Introdução

Os problemas de escalonamento estão associados à alocação de recursos escassos para um conjunto de tarefas em um determinado tempo. Segundo Blazewicz et al. (1996) escalonar significa designar recursos para a execução das tarefas até que todas tenham sido executadas sob as restrições impostas com o objetivo de minimizar o comprimento do escalonamento (tempo de processamento das tarefas) ou um outro critério estabelecido no contexto do problema.

A teoria do escalonamento é caracterizada por uma grande variedade de tipos de problemas que podem ser encontrados em diversas situações do mundo real e quando otimizadas permitem aumentar os lucros e diminuir os custos operacionais nas mais diversas organizações. Os problemas de escalonamento são classificados de acordo com os tipos de recursos existentes, das características das tarefas a serem executadas e do objetivo a ser alcançado. Uma descrição dos principais modelos pode ser encontrada em Artigues et al. (2008), Blazewicz et al. (1996), Brucker (1998) e Weglarz (1998). Especificamente, neste artigo, o foco é a classe de problemas de escalonamento com restrição de recursos e múltiplos modos de processamento que é reconhecidamente NP-Difícil e que tem atraído a atenção de forma crescente e desafiado vários pesquisadores ao longo dos anos.

Dentre os algoritmos exatos desenvolvidos para a resolução desse problema destacamos a abordagem *branch-and-bound* desenvolvida por Sprecher et al. (1997) e a proposta por Sprecher & Drexl (1998) baseada em árvore de precedência. Ressaltamos também os algoritmos heurísticos desenvolvidos por Drexl (1991) e Drexl & Grünewald (1993) que analisaram regras de prioridade baseadas em heurísticas multi-passo, Slowinski et al. (1994) e Bouleimen & Lecocq (2003) que implementaram uma versão da têmpera simulada, Hartmann (2001) que para o algoritmo genético e Verhoeven (1998) para a busca tabu. Já Kolisch & Drexl (1997) desenvolveram algoritmos de busca local próprios. Recentemente Jarboui et al. (2007) utilizaram a estratégia *particle swarm optimization* e Damak et al. (2009) a evolução diferencial.

As adaptações de meta-heurísticas para a resolução deste problema parecem ser as mais promissoras, de modo que elas continuam sendo investigadas e melhoradas. Neste artigo propomos a aplicação da meta-heurística *Scatter Search* (SS) para a resolução do problema. O restante deste artigo está organizado como segue. Na seção 2 caracterizamos o problema de escalonamento com restrição de recursos e múltiplos modos de processamento. O algoritmo baseado no scatter search é descrito na seção 3. Os resultados computacionais obtidos para instâncias padrões encontradas na literatura são apresentados e comparados com outros algoritmos na seção 4 e, finalmente na seção 5, as conclusões.

2 Descrição do Problema

No problema clássico de escalonamento com restrição de recursos e múltiplos modos de processamento, um conjunto de tarefas T tem que ser escalonado com o objetivo de minimizar, dentre todas as tarefas, o maior tempo de finalização de execução das tarefas, de tal forma que satisfaça as relações de precedências entre as tarefas e as restrições de capacidade de todos os recursos.

Um problema de escalonamento de projetos com restrições de recursos e múltiplos modos de processamento é formado por um conjunto de tarefas T, um conjunto de recursos $R = R^{ren} \cup R^{nren}$ onde R^{ren} contém os recursos renováveis e R^{nren} os não renováveis. A disponibilidade de cada recurso $r \in R$ é dado por c_r. Caso o recurso r seja renovável essa quantidade está disponível em qualquer momento e caso seja não renovável essa quantidade está disponível para toda a execução do escalonamento.

Além desses conjuntos comuns aos problemas de escalonamento com restrição de recursos, como cada tarefa pode ser processada de diversos modos, tem-se, para cada tarefa $j \in T$, um conjunto M_j com os possíveis modos de processamentos de j. O tempo de processamento da atividade j representado $p_{j,m}$ varia de acordo com o modo de execução selecionado e $q_{j,r,m}$ indica quantidade de recurso r utilizada para processar a tarefa j no modo $m \in M_j$.

Neste trabalho, assume-se o modelo não preemptivo, ou seja, uma vez iniciada o processamento da atividade, não haverá interrupção e também não poderá haver troca do modo de execução durante o processamento.

O objetivo do problema é encontrar o tempo de início t_j e o modo de processamento $m \in M_j$ para toda a tarefa $j \in T$ de modo a minimizar o maior tempo de finalização dentre todas as tarefas, conhecido na literatura como *makespan*, respeitando as relações de precedência entre as tarefas e satisfazendo as restrições de recursos.

3 Scatter Search Aplicado ao Problema de Escalonamento

3.1 Introdução

O Scatter Search é um método evolucionário proposto por Glover (1977) num estudo heurístico para a resolução de problemas de programação linear inteira. Aplicações desse método têm mostrado a sua eficiência na resolução de problemas em várias áreas. Destacam-se, entre os principais trabalhos desenvolvidos: Rochat & Taillard (1995) e Belfiore & Yoshizaki (2009) – Roteamento de Veículos, Cung et al. (1996) – Problema de Alocação Quadrática e Reeves & Yamada (1999) e Nowicki & Smutnicki (2006) – Flow Shop Scheduling.

Para Glover, Laguna e Martí (2000, p.653-684) em comum com outros métodos evolucionários, o Scatter Search opera sobre uma população de soluções, ao invés de uma única solução em cada iteração, e emprega procedimentos para combinar essas soluções com o intuito de gerar novas soluções. Por outro lado, diferentemente desses mesmos métodos, viola a premissa de que as abordagens evolucionárias devam ser baseadas em escolhas aleatórias, limitando o uso da randomização a procedimentos de diversificação. Esse método agrega princípios e estratégias que ainda não foram utilizados por outros métodos evolucionários e que têm se mostrado eficientes nas resoluções de uma grande variedade de problemas de otimização complexos.

O algoritmo *Scatter Search* pode ser estruturado, a partir do modelo proposto por Glover (1998), citado por Lorenzoni (2003, p. 71), com base nas seguintes rotinas:

1. Método Gerador de Soluções: tem por principal objetivo gerar uma população inicial (NP) com soluções de boa qualidade diferentes entre si.

2. Método de Melhoramento: método de busca local aplicado com o intuito de melhorar as soluções.

3. Método de Atualização do Conjunto Referência: cria e atualiza o conjunto referência, este é extraído a partir de soluções diversas.

4. Método de Geração de Subconjuntos: especifica a forma em que são selecionados os subconjuntos para aplicar o método de combinação.

5. Método de Combinação de Soluções: combina as soluções do conjunto referência. Para isso, consideram-se os subconjuntos formados pelo passo 4.

Na Figura 1 temos um fluxograma da meta-heurística *Scatter Search*.

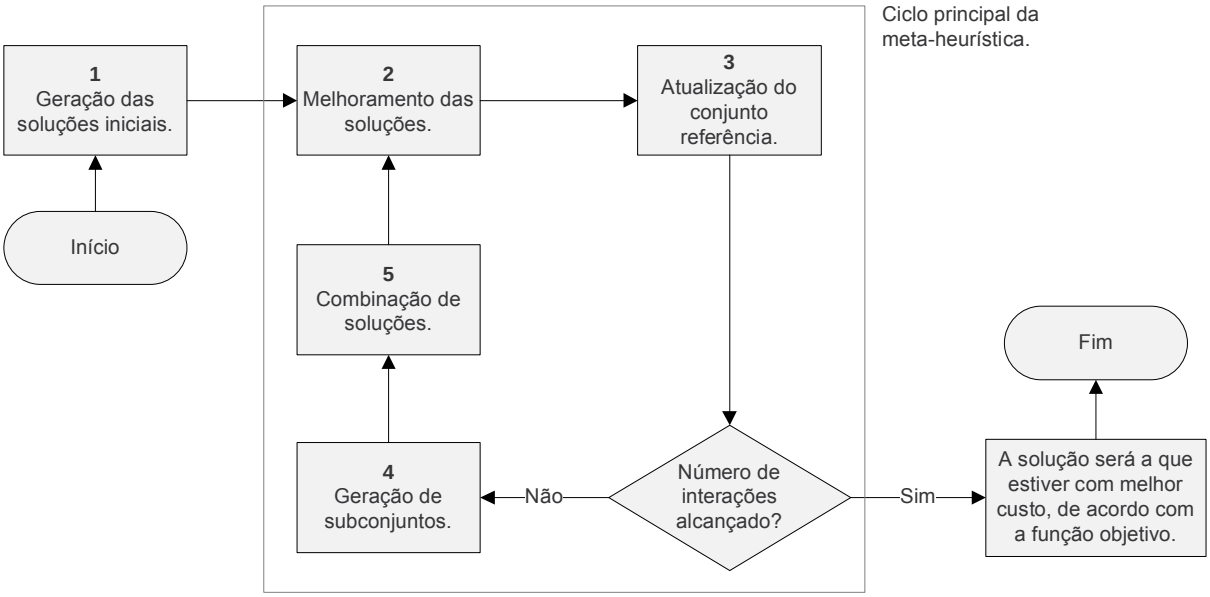

Figura 1 – Fluxograma da meta-heurística *Scatter Search*.

O algoritmo *Scatter Search* é bastante flexível de modo que cada um dos seus elementos pode ser implementado de diversas maneiras e em variados graus de sofisticação.

3.2 Implementação do Scatter Search

3.2.1 Método Gerador de Soluções

O método gerador soluções foi construído a partir de uma combinação de dois geradores:

O primeiro gerador é dividido em duas etapas. Na primeira etapa, escolhe-se em cada momento, aleatoriamente, uma tarefa dentre as elegíveis (aquelas cujos predecessores já foram todos escalonados), de modo a formar uma sequência de tarefas que satisfaça as relações de precedência. A segunda etapa consiste em definir o modo de processamento de cada uma das tarefas seguindo a sequência de tarefas estabelecidas na primeira etapa. A escolha do modo de processamento da tarefa também é feita aleatoriamente. Tal método pode gerar uma solução viável ou inviável.

No segundo gerador, para cada instante disponível, tenta-se alocar a maior quantidade possível de tarefas de forma aleatória, com o seu o respectivo modo de processamento também aleatório, respeitando as relações de precedência e os recursos, até que se conclua o escalonamento. Nesta implementação podemos levar em consideração ou não a limitação dos recursos não-renováveis.

No primeiro caso, a tentativa de geração do escalonamento pode não ser concluída pela extrapolação de recursos não-renováveis e, caso seja concluída, sempre gerará uma solução viável. No segundo caso, sempre será gerada uma solução, podendo ser viável ou inviável.

3.2.2 Método de Melhoramento

O método de melhoramento utilizado foi baseado na busca local *multi mode left shift*. Um *multi-mode left shift* de uma tarefa j é uma ação executada sobre o escalonamento que reduz o tempo de finalização da tarefa j sem modificar os modos de processamento, sem aumentar os tempos de finalizações das outras tarefas e sem violar as restrições de recursos. A tentativa de melhorar um escalonamento viável a partir de uma busca local é feita da seguinte forma: para cada tarefa do escalonamento, seguindo a ordem de processamento, o primeiro *multi-mode left shift*, se encontrado, é aplicado ao escalonamento. O resultado é um escalonamento com duração igual ou menor ao original.

Também foi desenvolvida uma adaptação da busca local *multi mode left shift*. Nessa adaptação verifica-se a possibilidade de troca do modo de cada tarefa por algum outro (com

qualquer duração) escolhido de forma randômica, e também, para cada instante possível, verifica-se a possibilidade de adiantamento das tarefas que ainda não foram processadas (randomicamente) em seu modo atual ou em outro modo, respeitando as relações de precedência, as restrições de recursos e sem aumentar os tempos de finalizações das demais tarefas. Esse método de melhoramento foi denominado por melhoramento modificado. Esse método quando aplicado o foi em todos os elementos gerados nas 2 últimas iterações e 5 vezes na melhor solução encontrada ao final da execução do algoritmo.

3.2.3 Método de Atualização do Conjunto Referência (RS)

Inicialmente, é gerada uma população com NP soluções (cada método gerador descrito em 3.2.1 gera $\frac{|NP|}{2}$ soluções). Após a calibração do algoritmo tem-se NP = 2000 e RS = 40. Para compor o RS escolhem-se as 10 melhores soluções e 10 soluções aleatórias obtidas com o primeiro gerador e as 20 melhores soluções obtidas com o segundo gerador. Dessa forma procurou-se estabelecer um conjunto com elementos de alta qualidade e com características diversas.

O conjunto referência é atualizado se a nova solução gerada pelo método de combinação de subconjuntos tiver melhor qualidade (de acordo com o *makespan*) que o pior elemento de RS e se não houver nenhuma outra solução igual no conjunto.

3.2.4 Método de Geração de Subconjuntos

Os subconjuntos foram gerados a partir da combinação dois a dois de todos os elementos do conjunto referência.

3.2.5 Método de Combinação de Soluções

Combina as soluções contidas no subconjunto para gerar novas soluções candidatas e a cada par de soluções combinadas são geradas duas novas soluções. Toda solução candidata gerada sempre respeitará as relações de precedência contudo pode ser viável ou inviável com relação aos recursos.

Na implementação é efetuado um sorteio entre os 3 métodos de *crossover*, todos com igual probabilidade de serem escolhidos, desenvolvidos a saber*:* único ponto, multiponto e uniforme. A seguir uma descrição dos operadores de *crossover* onde n corresponde ao número total de tarefas do escalonamento e *k* é a posição de uma determinada tarefa no escalonamento sendo *k=1* a posição da primeira tarefa não fictícia do escalonamento. Nos diagramas a primeira e a última tarefa são fictícias e são utilizadas para facilitar a representação da solução e para estar de acordo com a massa de testes encontrada na literatura.

Único Ponto: para executar a combinação o método seleciona aleatoriamente uma posição $0 < k < n$ para ser o ponto de *crossover*. Os primeiros *k* elementos da primeira solução (Pai 1) serão os primeiros *k* elementos do primeiro filho (Filho 1), os demais elementos serão obtidos da segunda solução (Pai 2), sendo inseridos na ordem em que estão dispostos. O segundo filho (Filho 2) é obtido de forma análoga conforme Figura 2.

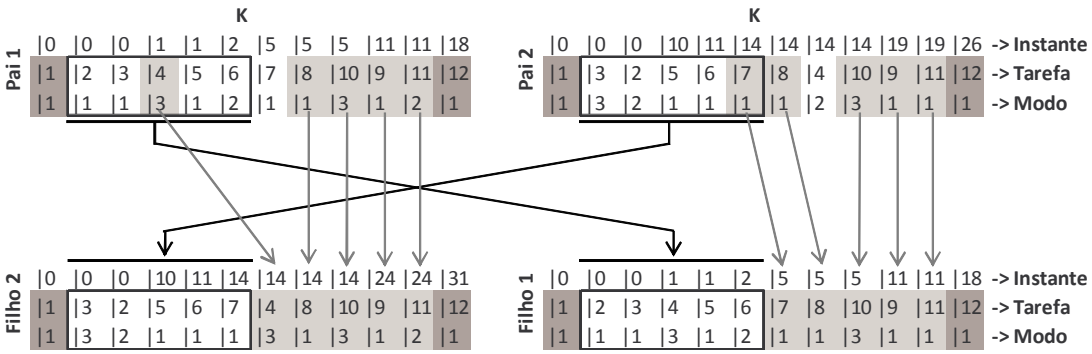

Figura 2 – Exemplo de combinação de um único ponto.

Multipontos: Para executar a combinação, o método seleciona aleatoriamente duas posições, $0 < k1 < (j-1)$ e $(k1+2) \leq k2 \leq j$ para serem os pontos de *crossover*. Os primeiros *k1* elementos da primeira solução (Pai 1) serão os primeiros elementos do primeiro filho (Filho 1) e os elementos a partir de *k2* serão seus elementos finais. Os *k1+1* até *k2-1* elementos serão obtidos da segunda solução (Pai 2), sendo inseridos na ordem em que estão dispostos. O segundo filho (Filho 2) é obtido de forma análoga conforme Figura 3.

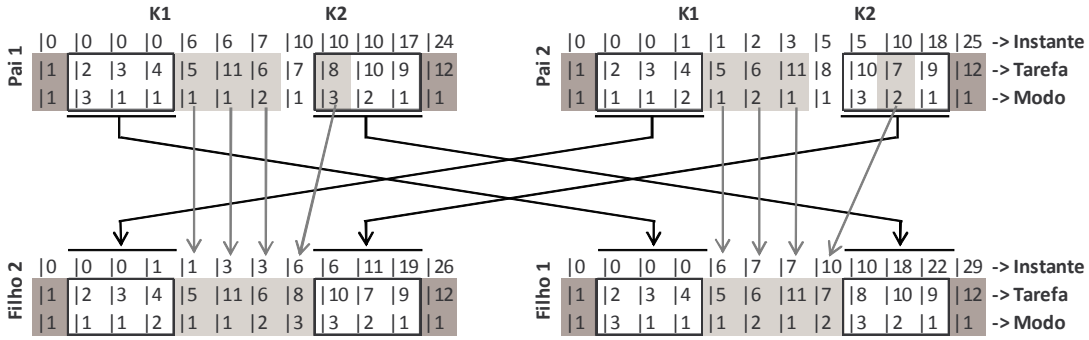

Figura 3 – Exemplo da combinação em multipontos.

Uniforme: não utiliza pontos de cruzamento, mas determina, por meio de uma máscara, quais elementos de cada escalonamento que cada filho herdará. A máscara é gerada através de um vetor de números randômicos variando entre 1 e 0, em que 1 significa que o primeiro filho (Filho 1) alocará a próxima tarefa disponível do primeiro pai (Pai 1) e 0 que ele alocará a próxima tarefa disponível do segundo pai (Pai 2). Para se obter o segundo filho (filho 2) basta inverter a atribuição entre os valores 0 e 1 conforme Figura 4.

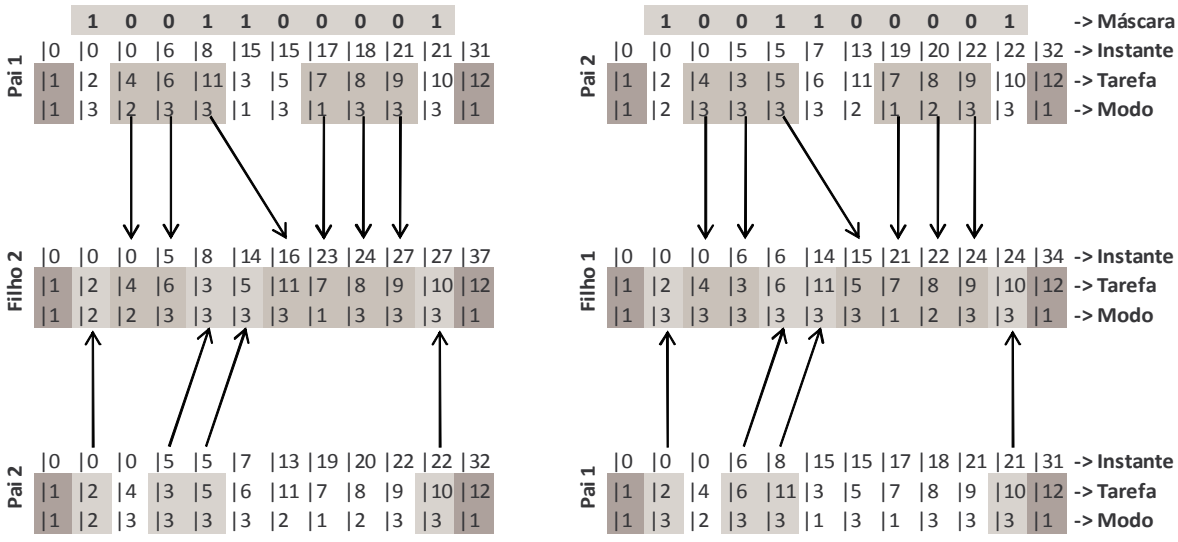

Figura 4 – Exemplo de combinação uniforme.

3.2.6 Mutação

Essa não é uma operação comumente utilizada nas implementações do *Scatter Search* mas nessa aplicação a operação tem por finalidade possibilitar a criação de novas sequências de tarefas que ainda não tenham sido produzidas ou introduzir um novo modo a uma tarefa que ainda não tenha sido testado. A mutação foi executada a cada nova combinação gerada

aplicando-se primeiro a *mutação 1* que seleciona aleatoriamente uma posição *k* da sequência de tarefas que compõe o indivíduo (sendo $0 < k < n$) e troca as tarefas *k* e *k+1* se permitido pela relação de precedência, e depois, a *mutação 2* que seleciona aleatoriamente uma posição *k* da sequência de tarefas que compõem o indivíduo e escolhe aleatoriamente um novo modo de processamento para a tarefa associada à posição *k*. Nas Figuras 5 e 6 têm-se uma representação dessas operações.

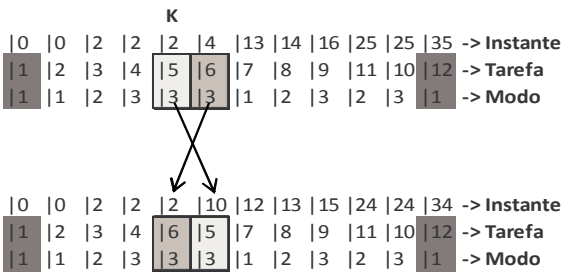

Figura 5 – Exemplo da Mutação 1

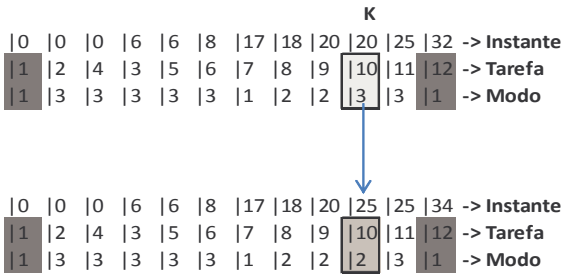

Figura 6 – Exemplo da Mutação 2.

4 Experimentos Computacionais

Todos os algoritmos foram implementados na linguagem de programação JAVA, utilizando a ferramenta Eclipse SDK (versão 3.5.1) e executado num computador pessoal INTEL Core 2 Duo (T5670) 1.80 GHz, com 3GB de memória DDR2 667 MHz, sobre a plataforma Microsoft Windows XP Professional SP3 de 32 bits.

Os testes foram realizados utilizando a classe de problemas J20 da biblioteca PSPLIB (*Project Scheduling Problem Library*). Esses problemas foram construídos pelo gerador ProGen (*Project Generator*) desenvolvido por Kolisch e Sprecher (1996). Todas as 554 instâncias da classe J20 possuem 20 tarefas e cada uma das tarefas 3 modos de processamento. Essa classe foi a escolhida para a realização dos testes por ser a de maior porte com soluções ótimas conhecidas e por também ter sido utilizada em outros trabalhos encontrados na literatura possibilitando a comparação do desempenho entre os mesmos. Neste trabalho para cada instância o algoritmo foi executado 4 vezes e a média dos resultados obtidos é considerada a solução do problema. O critério de parada foi o mesmo utilizado por Damak et al. (2009) que é o tempo de execução limitado a 150ms por tarefa, ou seja, 3000 ms. Os critérios de avaliação e de comparação dos algoritmos baseiam-se em desvio médio e percentagem de soluções ótimas encontradas. Por meio do desvio calculamos o quão distante a solução encontrada pelo algoritmo está da solução ótima e pode ser calculado como em (1) por:

$$\text{desvio} = \frac{(\text{solução encontrada} - \text{solução ótima})}{\text{solução ótima}} \quad (1)$$

Logo o desvio médio será obtido pela expressão (2) considerando que desvio_i representa a média de desvio para cada uma das 554 instâncias da classe J20.

$$\text{desvio médio} = \sum_{i=1}^{554} \frac{\text{desvio}_i}{554} \quad (2)$$

Já as soluções ótimas correspondem a percentagem dentre as 554 instâncias para as quais o algoritmo atingiu a solução ótima conhecida. Também foi calculado o desvio padrão médio (σ) para avaliar a robustez do algoritmo implementado.

Com a realização dos primeiros testes e avaliação dos resultados, as duas rotinas descritas a seguir foram incorporadas ao algoritmo com o intuito de melhorar o desempenho.

Pré-processamento

O pré-processamento reduz o espaço de busca diminuindo o número de soluções viáveis e inviáveis sem afetar o conjunto de soluções ótimas pela eliminação dos modos não executáveis,

ineficientes e dos recursos não-renováveis redundantes. Segundo Sprecher et al. (1997) um modo é dito não executável se a sua execução viola as restrições de recursos renováveis ou não renováveis e é considerado ineficiente se a sua duração não for menor e a sua requisição de recursos não for menos que os outros modos de processamento da mesma tarefa. Já um recurso não renovável é chamado redundante se o somatório da maior requisição desse recurso por tarefa não exceder a capacidade do recurso do projeto.

Aninhamentos de Conjuntos Referências

Esta dinâmica foi desenvolvida com o intuito de aumentar a quantidade de soluções de boa qualidade e ao mesmo tempo manter um grau de diversificação entre elas. Quando utiliza-se somente um conjunto referência, as soluções contidas neste tendem a ter em comum várias características, até que se acomodem em um ótimo local, que pode ser ou não um ótimo global, podendo não mais convergir para soluções ótimas globais apesar do aumento no número de iterações. Seguindo essa motivação criou-se os aninhamentos de conjuntos referências em níveis. Ao invés de se criar um único Conjunto Referência são criados vários conjuntos inicialmente. Particularmente, após a parametrização foram criados 9 conjuntos.

Cada nível acima possui um número maior de conjuntos. Estes após sofrerem as suas devidas iterações têm seus elementos cedidos para a geração dos próximos conjuntos. No aninhamento de 2 níveis, os conjuntos existentes no primeiro nível cedem seus elementos com melhores resultados para a construção de um outro conjunto. Já no aninhamento de 3 níveis, os conjuntos do primeiro nível são agrupados em partes iguais e cada grupo é responsável por ceder elementos para um conjunto do nível inferior.

Particularmente, após a parametrização, foram criados 9 conjuntos para o primeiro nível, 3 para o segundo e 1 para o terceiro. A Figura 7 ilustra o funcionamento dos aninhamentos.

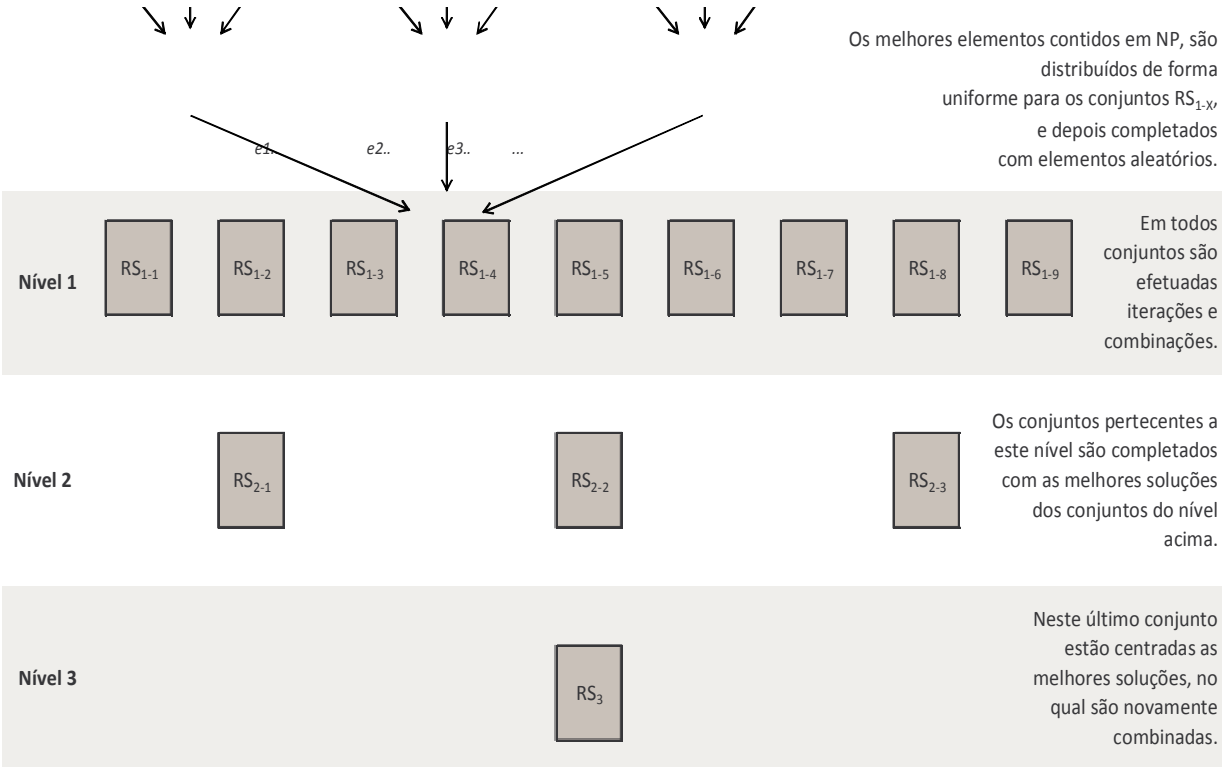

Figura 7 – Exemplo do funcionamento do aninhamento.

A notação $SS_{\{*,*,...,*\}}$ é usada para designar as várias versões do Scatter Search implementadas e os símbolos usados como subíndices na notação são: P – Pré-Processamento, LS - Multi-Mode Left Shift, An – Aninhamento de Conjuntos, Mu – Mutação e MM – Melhoramento Modificado. A Tabela 2 apresenta os resultados obtidos para as diversas versões

implementadas do algoritmo evidenciando a evolução do desempenho do algoritmo com a inclusão das novas rotinas. Observa-se que o desempenho do algoritmo melhora tanto com relação ao desvio médio quanto as soluções ótimas encontradas. Ressalta-se ainda que o desvio padrão médio, além de diminuir, é muito pequeno, o que caracteriza a robustez do algoritmo implementado.

Tabela 2 – Evolução do desempenho do SS com a inclusão das rotinas

Versão	Desvio Médio (%)	Soluções Ótimas (%)	Desvio Padrão Médio
$SS_{P,LS}$	2,57	57,43	0,0376
$SS_{P,LS,Mu}$	1,73	68,34	0,0304
$SS_{P,LS,Mu,An}$	1,55	72,24	0,0313
$SS_{P,LS,Mu,An,MM}$	0,93	81,06	0,0267

A Tabela 3 apresenta a comparação dos resultados da meta-heurística SS, implementada neste trabalho, com os melhores resultados existentes na literatura reportados em Damak et al. (2009).

Tabela 3 – Comparação entre meta-heurísticas - classe de problemas J20

Heurística	Desvio Médio (%)	Soluções Ótimas (%)
Differential Evolution Damak et al. (2009)	0,70	91,75
Particle Swarm Optimization Jarboui et al. (2008)	1,21	74,19
Simulated Annealing Bouleimen & Lecocq (2003)	2,10	66,90
$SS_{P,LS,Mu,An,MM}$	0,93	81,06

Os resultados exibidos na Tabela 3 mostram que o algoritmo $SS_{P,LS,Mu,An,MM}$ supera os algoritmos propostos por Bouleimen & Lecocq (2003) – *Simulated Annealing* e Jarboui et al. (2008) - *Particle Swarm Optimization* sendo inferior apenas ao proposto por Damak et al. (2009) - *Differential Evolution*.

5 Conclusões

Os problemas de escalonamentos têm sido aplicados em diversas situações do cotidiano, em especial o problema com restrição de recursos e múltiplos modos de processamento, foco desse artigo. Tais problemas por pertencerem a classe de problemas NP - Difícil têm uma alta complexidade de resolução o que justifica o desenvolvimento de meta-heurísticas para resolvê-los.

A estratégia Scatter Search, implementada nesse trabalho, mostrou-se flexível, possibilitando por meio da análise dos resultados obtidos, a inserção estratégica de novas rotinas, tal como o Aninhamento de Conjuntos Referências, melhorando ainda mais o desempenho do algoritmo. Vale ressaltar que o desvio padrão médio foi muito pequeno o que indica a robustez do algoritmo.

Na implementação proposta, os resultados obtidos, de um modo geral, foram muito satisfatórios e encontram-se no mesmo nível dos melhores resultados apresentados pela literatura, o que comprova a eficiência do algoritmo.

Como perspectivas futuras, além do aperfeiçoamento do algoritmo, com a inclusão de novos métodos de melhoramento e novas formas de aninhamento, prevê-se a aplicação dessa estratégia para a resolução de problemas de alocação de sondas de produção terrestre a poços de petróleo com o objetivo de minimizar a perda de vazão, problema este que pode ser modelado como um problema de escalonamento com restrição de recursos.

6 Referências

ARTIGUES, C.; DEMASSEY, S.; NÉRON, E. The Resource-Constrained Project Scheduling Problem. In: ____. (Eds.). Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications. 1. ed. United States of America: Wiley-ISTE, 2008. p. 21-35.

BELFIORE, P.; YOSHIZAKI, H. T. Y. Scatter Search for a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries in Brazil. European Journal of Operational Research, p. 750-758, 2009.

BLAZEWICZ, J.J., ECKER, K. H., PESCH, E., SCHMIDT, G. & WEGLARZ, J. Scheduling Computer and Manufacturing Processes. Springer Verlag, 1996. 500p.

BOULEIMEN, K. & LECOCQ, H. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research 149, 268-281, 2003.

BRUCKER, P. Scheduling Algorithms, Springer, Berlin. 1998. 377p.

CUNG, V-D.; MAUTOR, T.; MICHELON, P. & TAVARES, A. A Scatter Search based approach for the Quadratic Assignment problem. In T. Bäck, Z.; Michalewicz, X. Yao (eds.), Proceedings of ICEC´97, IEEE Press, p. 165-170, 1996.

DAMAK, N.; JARBOUI, B.; SIARRY, P.; LOUKIL, T. Differential evolution for solving multi-mode resource-constrained project scheduling problems. Computers & Operations Research, Elsevier, p. 2653-2659, 2009.

DREXL, A. Scheduling of project networks by job assignment, Management Science, 37, 1590-1602, 1991.

DREXL, A., GRÜNEWALD, J. Nonpreemptive multi-mode resource-constrained project scheduling, IIE Transactions, 25(5), 74-81, 1993.

Glover, F., Heuristics for integer programming using surrogate constraints. Decision Sciences, 8(1), p.156-166, 1977.

GLOVER, F. A template for scatter search and path relinking. In Hao, J.-K., Lutton, E., Ronald, E., Schoenauer, M. and Snyers D. (eds.), Artificial Evolution Lecture Notes in Computer Science, Springer, 1363, 13-54, 1998.

GLOVER, F.; LAGUNA, M.; MARTÍ,R. Fundamentals of Scatter Search and Path Relinking, Control and Cybernetics 29(3), p. 653-684, 2000.

HARTMANN, S. Project scheduling with multiple modes: A genetic algorithm. Annals of Operations Research 102, 111-135, 2001.

JARBOUI B, DAMAK N, SIARRY P, REBAI A. A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. Applied Mathematics and Computation, 195, p.299–308, 2008.

KOLISCH, R., DREXL, A. Local search for nonpreemptive multi-mode resource-constrained project scheduling, IIE Transactions, 29, 987-999, 1997.

KOLISCH, R.; SPRECHER, A. PSPLIB – A project scheduling problem library. European Journal of Operational Research, Germany, p. 205-216, 1996.

LORENZONI, L. L. Problema de escalonamento com restrição de recursos e múltiplos modos de processamento: novos métodos de resolução e uma aplicação no contexto portuário. 2003. 207 p. Tese (Doutorado em Engenharia Elétrica) – Universidade Federal do Espírito Santo, Vitória, 2003.

NOWICKI, E.; SMUTNICKI, C. Some aspects of scatter search in the flow-shop problem. European Journal of Operational Research, 169(2), p. 750-758, 2006.

REEVES, C.R.; YAMADA, T. Goal-oriented path tracing methods. In D. Corne, M. Dorigo, and Fred Glover, editors, New Ideas in Optimization, McGraw-Hill, 341-355, 1999.

ROCHAT, Y. & TAILLARD, E.D. Probabilistic diversification and intensification in Local Search for Vehicle Routing. Journal of Heuristics, 22, p. 158-166, 1995.

SLOWINSKI, R., SONIEWICKI, B., WEGLARZ, J. DSS for multi-objective project scheduling subject to multiple-category resource constraints, European Journal of Operational Research 79, 220-229, 1994.

SPRECHER, A., HARTMANN, S., DREXL A. An exact algorithm for project scheduling with multiples modes. OR Spektrum, 19, 195-203, 1997.

SPRECHER, A., DREXL A. Solving multi-mode resource constrained project scheduling problems by a simple, general and powerful sequencing algorithm. European Journal of Operational Research 107, 431-450, 1998.

VERHOEVEN, M. G. A. Tabu search for resource-constrained scheduling. European Journal of Operational Research, 106, 266-276, 1998.

WEGLARZ, J. Project Scheduling - Recent Models, Algorithms and Applications. 1. ed. United States of America: Springer, 1998, 552 p.