Revista Científica Fagoc Multidisciplinar
ISSN: 2525-488X

META-HEURÍSTICA MULTIOBJETIVO PARA SEQUENCIAMENTO DE MÁQUINAS PARALELAS NÃO RELACIONADAS COM TEMPOS DE PREPARAÇÃO DEPENDENTES DA SEQUÊNCIA

ABREU, Júnior César [1]
PEREIRA, Ana Amélia de Souza [2]

RESUMO
Neste trabalho é abordado o problema multiobjetivo de sequenciamento em máquinas paralelas com tempos de preparação dependentes da sequência e da máquina para minimização do tempo de conclusão total e o lateness máximo. Problemas de sequenciamento são extensivamente investigados pela literatura tanto pelo aspecto teórico como pelo prático e têm aplicações práticas em várias áreas, principalmente na indústria. Problemas dessa classe são frequentemente classificados como NP-difícil, não podendo ser resolvidos em tempo polinomial. Para resolução desse problema, será proposta uma adaptação à meta-heurística MOILS (Multiobjective Iterated Local Search) e ao ILSMulti (Multi-Objective Iterated Local Search), baseadas em busca local. A confiabilidade é verificada através de instância com resultados exatos e o desempenho é comparado com o NSGA-II (Non-dominated Sorting Genetic Algorithm II) através do Indicador de Hipervolume. Resultados indicam que as meta-heurísticas propostas ainda não superam o algoritmo da literatura.

PALAVRAS-CHAVE: Sequenciamento. Multiobjetivo. Máquinas paralelas não relacionadas. Busca local. Meta-heurística.

1 FAGOC. E-mail: juniorcesarabreu@live.com
2 FAGOC. E-mail: aamelia.mg@gmail.com

INTRODUÇÃO

Sequenciamento de produção é um importante processo de tomada de decisão em nível operacional usado em muitas indústrias de serviço e manufatura (NOGUEIRA et al., 2014). Essa classe de problemas pode ser encontrada em diferentes áreas como, por exemplo, no planejamento da produção, no gerenciamento de projetos, no sequenciamento de tarefas pelo processador de um computador, no controle de aterrissagens e decolagens em aeroportos. Problemas de sequenciamento lidam com a alocação de recursos para execução de um conjunto de tarefas em um dado intervalo de tempo e seu objetivo é otimizar um ou mais critérios (PINEDO, 2012).

Pertencente à categoria de problemas de produção, ele deve satisfazer as demandas sem atraso, respeitar a capacidade dos recursos disponíveis e minimizar os custos da produção (ARENALES et al., 2007). Esse tipo de problema tem sido bastante estudado e tem grande destaque na indústria (RIBEIRO, 2009).

Segundo Chen et al. (1998), problemas de sequenciamento de produção em máquinas paralelas se referem a problemas onde há um conjunto de máquinas que operam em paralelo e possuem as mesmas funções podendo ser classificados em três casos, de acordo com o ambiente de máquina: máquinas paralelas idênticas, máquinas paralelas uniformes e máquinas paralelas não relacionadas.

Em máquinas não relacionadas, o tempo de processamento depende tanto da tarefa como da máquina à qual aquela foi atribuída.

Esse ambiente representa o caso mais realístico e também uma generalização dos outros casos (VALLADA; RUIZ, 2011), pois, em situações do mundo real, é comum a necessidade de renovação do maquinário ou ampliação das instalações existentes, onde máquinas de diferentes fabricantes e modelos geralmente possuem capacidades de execução diferentes (ETCHEVERRY; ANZANELLO, 2013).

Este trabalho aborda o problema de sequenciamento de produção em máquinas paralelas não relacionadas, com n tarefas que devem ser processadas em m máquinas; cada tarefa deve ser processada apenas uma vez e em apenas uma das máquinas i. Cada tarefa j possui um tempo de processamento Pij, dependentes da máquina e da tarefa, sendo i e j, a máquina e a tarefa, respectivamente, uma data de entrega dj e um tempo de liberação (release time) Rj. As tarefas devem ser executadas exatamente uma vez, não podendo ser interrompidas durante o processamento. Além disso, as tarefas possuem um tempo de preparação (setup time) Sijk dependente da sequência e da máquina, ou seja, o tempo de preparação da tarefa j após a execução da tarefa k na máquina i é diferente do tempo de preparação da tarefa k após a execução da tarefa j na mesma máquina; além disso, o tempo de preparação entre as tarefas j e k na máquina i é diferente do tempo de preparação entre as mesma tarefas j e k mas na máquina i' (VALLADA; RUIZ, 2011).

Segundo Pinedo (2012), o problema de sequenciamento em máquinas paralelas está relacionado à determinação da melhor sequência de execução das tarefas, definindo os recursos mais adequados para a execução destas. Uma sequência pode ser definida como uma permutação de tarefas que serão executadas sobre determinadas máquinas, levando em consideração as restrições de tempo (PINEDO, 2012).

Os critérios de otimização serão a minimização da soma total do tempo de conclusão das tarefas (∑Cj) e do tempo de lateness máximo (Lmax). O tempo de conclusão total é utilizado com o objetivo de minimizar o custo total do sequenciamento, enquanto o lateness máximo é utilizado para minimizar o maior atraso na entrega (BRUCKER, 2007). Tarefas concluídas após o prazo podem gerar atrasos na cadeia distributiva e insatisfação por parte dos clientes (PEREIRA et al., 2014).

Revisão literátia

Alguns trabalhos abordam o problema multiobjetivo de sequenciamento em máquinas paralelas não relacionadas. Em Safaei (2008), são propostos dois algoritmos genéticos multiobjetivos (MOGA): o MOGAT, MOGA baseado no método TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) e o MOGAC, MOGA baseado no Non-dominated Sort e crowding distance para minimizar o tempo de conclusão máximo (makespan) e o número de tarefas atrasadas. A conclusão é que não houve estatisticamente diferença significativa entre as duas meta-heurísticas.

Em Tavakkoli-Moghaddam, Bazzazi e Taheri (2008) é proposto um algoritmo genético (GA) a partir de um modelo de programação inteira multiobjetivo em duas fases, para minimizar o número de tarefas atrasadas e o tempo de conclusão total de todas as tarefas. Os resultados mostram a eficácia do modelo para problemas de pequeno e médio porte.

Em Li et al. (2010), para resolução do problema, foram utilizadas duas meta-heurísticas evolucionárias: o NSGA-II (Non-Dominated Sorting Genetic Algorithm) e o SPEA-II (Strength Pareto Evolutionary Algorithm), e ambas utilizam o conceito de Pareto. Os objetivos considerados foram o tempo de conclusão máximo (makespan) e a soma total dos atrasos (total tardiness). Os testes computacionais demostraram que o NSGA-II foi mais vantajoso para resolução do problema abordado e, quando comparado ao algoritmo exato, foi capaz de obter soluções ótimas em todas as instâncias testadas.

Em Lin et al. (2010), é proposta uma heurística em quatro fases, o LP-ATC (Linear

Programming – Apparent Tardiness Cost rule), com o objetivo de minimizar o tempo de conclusão máximo e o atraso total ponderado. Os resultados computacionais mostram que a heurística proposta oferece qualidade de soluções razoáveis e eficiência computacional.

Afzalirad e Rezaeian (2017) consideram o problema com elegibilidade de máquina e restrição de precedência, propondo o algoritmo NSGA-II e o Multi-objective Ant Colony Optimization (MOACO) para minimização do tempo de fluxo ponderado médio (mean weighted flow time) e o atraso ponderado médio (mean weighted tardiness). Resultados indicam que o sugerido MOACO supera estatisticamente o NSGA-II.

Em Dcoutho e Moraga (2016), para minimização do atraso total ponderado (total weighted tardiness) e o tempo de conclusão total ponderado (total weighted completion time), é proposto o uso de Meta-heuristic for Randomized Priority Search (Meta-RaPS) em duas abordagens. A primeira utiliza a heuristica Apparent Tardiness Cost-bi (ATC-bi) na fase de construção para gerar soluções não dominadas. Na segunda, é incorporado o mecanismo de memória na fase de construção. Resultados mostram que a meta-heurística proposta é efetiva e flexível o suficiente para gerar fronteiras Pareto de modo a resolver o problema, sendo a abordagem com memória melhor.

Já Chang et al. (2010) propuseram um esquema de decodificação baseado em correspondência em duas fases, que é incorporado dentro de um Multi-Objective Simulated Annealing (MOSA). São apresentados dois MOSA para solucionar o problema: F-MOSA, probabilidade de aceitação baseada em objective fitness, e D-MOSA, baseado em regra de dominância. Além disso são propostos alguns métodos de codificação e decodificação, totalizando 8 MOSA testados, com o objetivo de maximizar a satisfação do makespan e atraso médio (average tardiness), em termos de medida fuzzy (fuzzy measure). Resultados experimentais indicam que o método proposto de decodificação em duas fases pode significativamente melhorar as soluções.

Para a solução do problema de sequenciamento em máquinas paralelas não relacionada com tempos de preparação dependentes da sequência das tarefas e da máquina, propõe-se neste trabalho a meta-heurística Multiobjective Iterated Local Search (MOILS), apresentada nos trabalhos de Assis et al. (2013) e Fonseca et al. (2012) para problemas de roteamento multiobjetivo, e a meta-heurística Multi-objective Iterated Local Search (ILSMulti), apresentada por Barros Junior e Arroyo (2010), para um problema de planejamento florestal multiobjetivo, ambas baseadas em busca local.

Para Lourenço et al. (2003), o ILS (Iterated Local Search) tem obtido bons resultados em problemas mono-objetivos. No entanto, considerável parte dos trabalhos na literatura utiliza outras abordagens, como algoritmos genéticos, abordagens baseadas em busca local pouco exploradas. Durante a revisão da literatura não foi encontrado qualquer trabalho que abordasse o uso de busca local para esse problema.

O restante deste trabalho está estruturado como segue. Na seção Referencial Teórico são introduzidos alguns conceitos relevantes para este estudo. A seção Métodos de Desenvolvimento apresenta a metodologia utilizada e os algoritmos desenvolvidos. Na seção Resultados Obtidos são apresentados os resultados computacionais. E, por fim, na Conclusão, são apresentadas as considerações finais deste trabalho, bem como as possíveis propostas de trabalhos futuros.

REFERENCIAL TEÓRICO

Problemas de sequenciamento

Esta classe de problemas pode ser encontrada em diversas áreas, por exemplo, no planejamento da produção, no gerenciamento de projetos, no sequenciamento de tarefas pela unidade central de processamento do computador, no controle de aterrisagens e

decolagens em aeroportos (PINEDO, 2012). Problemas de sequenciamento de produção consistem basicamente em alocar recursos para a execução de um conjunto de tarefas num dado intervalo de tempo de modo a otimizar um ou mais objetivos (PINEDO, 2012). Para Nogueira et al. (2014), problemas de sequenciamento constituem um importante processo para tomada de decisão em nível operacional que executa um papel crucial na área de serviços e indústria de manufatura.

Ainda segundo Nogueira et al. (2014), essa classe de problemas é extensivamente investigada na literatura devido a dois aspectos: o primeiro, relacionado a sua importância prática em várias industrias, como a indústria química, a metalúrgica e a têxti; e segundo, pela dificuldade em resolver a maioria desses problemas. Para Pinedo (2012), muitos problemas de sequenciamento podem ser classificados como NP-difícil, ou seja, não podem ser computados em tempo polinomial.

Problema multiobjetivo

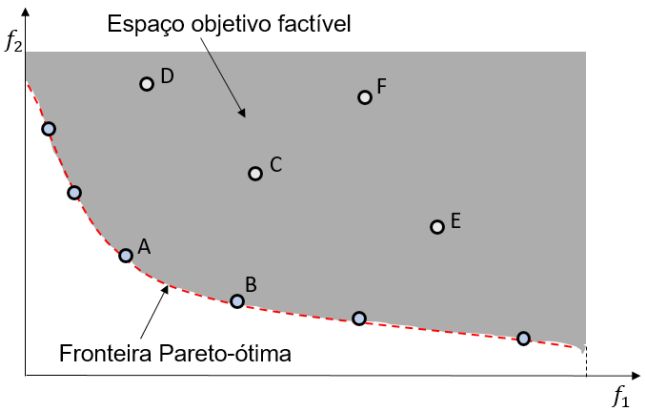
Para Grimme et al. (2013), a consideração de apenas um critério de otimização é insuficiente para se obter uma solução que assegure qualidade e satisfação do cliente na maioria dos cenários práticos de sequenciamentos. Portanto, um esquema de produção é geralmente julgado com respeito a vários critérios. Devido a essa complexidade em relação ao espaço de busca, algoritmos exatos geralmente se tornam inviáveis, portanto as heurísticas são uma melhor opção na tentativa de resolver problemas de grande porte (FONSECA et al., 2012).

Para resolver problemas multiobjectivos, as abordagens são geralmente divididas, segundo Yenisey e Yagmahan (2014), em três classes, de acordo com o papel do tomador de decisões no processo de solução: Abordagem prévia: as informações necessárias são disponibilizadas no início do processo; Abordagem posterior: é desenvolvido um conjunto de soluções eficientes (ou não dominadas ou Pareto-ótimo) e não apenas uma. Cabe, então, ao tomador de decisões, escolher uma solução que considere aceitável; e Abordagem iterativa: o tomador de decisões introduz suas preferências a cada passo durante o processo.

Os dois objetivos considerados neste trabalho são conflitantes, portanto não há uma solução ótima capaz de satisfazer ambos os objetivos simultaneamente, pois qualquer melhoria em um provocaria degradações no outro (ASSIS et al., 2013). Devido à dificuldade em definir se certa solução é melhor que outra, neste trabalho é proposto que se obtenha um conjunto de soluções factíveis e que a decisão sobre a melhor solução fique a cargo de um tomador de decisões. Assim, os seguintes conceitos são importantes, considerando um problema de minimização (YENISEY; YAGMAHAN, 2014):

- Dominância de Pareto: um vetor de soluções factíveis é dito dominar um outro vetor b (denotado a < b) se, e somente se, para todos os objetivos fi(a) <= fi(b) e, existe pelo menos um objetivo fi(a) < fi(b).
- Solução Pareto-ótima: uma solução a do conjunto de soluções é considerada Pareto-ótima se não há nenhuma outra solução b neste mesmo conjunto que domine a tal que fi(a) < fi(b), ou seja, a não é dominada por nenhuma outra solução.
- Fronteira Pareto-ótima: o conjunto de todas as soluções Pareto-ótimas é uma fronteira Pareto-ótima.

Figura 1: Exemplo de fronteira Pareto-ótima



Fonte: adaptado de Arroyo, 2002.

Na Figura 1, é representado o conceito de dominância. As soluções que se encontram na fronteira Pareto-ótima são soluções eficientes e dominam as outras soluções. Nesse caso, as soluções A e B pertencem à fronteira e dominam as outras soluções que estão fora, por exemplo, as soluções C, D e E.

Regras de prioridade

Regras de prioridade ou despacho são métodos bastante úteis, pois permitem obter soluções razoáveis ou até ótimas para problemas mais simples com apenas um objetivo; além disso, são fáceis de implementar (PINEDO, 2012).

Em problemas mais complexos, onde são considerados mais de um objetivo de otimização, essas regras são úteis também porque podem ser incorporadas em meta-heurísticas na tentativa de obter melhor desempenho, no entanto geralmente precisam ser combinadas para que produzam uma sequência factível (PINEDO, 2012).

Algumas regras que podem ser úteis considerando os objetivos de minimizar o lateness máximo e o tempo de conclusão total são:

- Regra EDD (Earliest Due Date first): ordena as tarefas em relação as suas datas de entrega em ordem crescente, ou seja, a tarefa mais próxima do vencimento é processada antes da tarefa com maior prazo (PINEDO, 2012).
- Regra SPT (Shortest Processing Time first): ordena as tarefas em relação ao seu tempo de processamento em ordem crescente, partindo da tarefa com menor tempo de processamento até a tarefa com maior tempo (PINEDO, 2012).
- Regra LPT (Longest Processing Time first): semelhante ao SPT, essa regra também ordena as tarefas em relação ao seu tempo de processamento, porém em ordem decrescente, ou seja, do maior tempo para o menor (PINEDO, 2012).

MÉTODO DE DESENVOLVIMENTO

Representação de uma solução

Uma solução para um problema de sequenciamento pode ser representada por uma permutação de tarefas em um vetor contendo a sequência de processamento das tarefas em determinada máquina.

Para exemplificar um caso de sequenciamento em máquinas paralelas, considere uma instância do problema com n = 5 tarefas e m = 2 máquinas; sendo que n representa o número de tarefas e m o número de máquinas, j = {1,...,n} a tarefa sendo processada e i = {1, 2} a máquina sobre a qual a tarefa j é executada.

Os tempos de processamento p1j = [21, 26, 16, 14, 9] e p2j = [17, 20, 20, 10, 7], os tempos de liberação rj = [3, 4, 10, 7, 0], as datas de entrega dj = [31, 45, 57, 33, 29] e os tempos de preparação dependentes da sequência e da máquina discriminados no Quadro 1.

Quadro 1: Tempos de preparação Sijk nas máquinas 1 e 2.

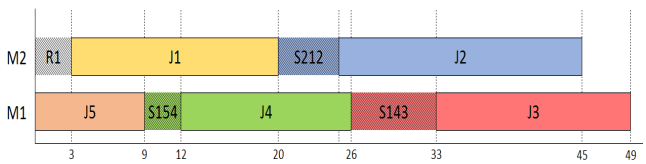
S^1_{jk}		1	2	3	4	5
	1	0	5	7	3	2
	2	4	0	1	1	2
	3	7	6	0	2	3
	4	8	3	7	0	3
	5	4	5	8	3	0

S^2_{jk}		1	2	3	4	5
	1	0	4	8	7	2
	2	7	0	3	3	6
	3	1	3	0	1	7
	4	4	5	5	0	7
	5	2	8	4	2	0

Fonte: dados da pesquisa.

Na Figura 2 é apresentado um exemplo de solução deste problema considerando a sequência de tarefas 5, 1, 4, 2 e 3, ordenada pela regra EDD, com os tempos de conclusão sendo 9, 20, 26, 45 e 49.

Figura 2: Exemplo de uma solução



Fonte: dados da pesquisa.

NSGA-II

Algoritmos genéticos são baseados no processo de seleção natural em que sobre uma população de indivíduos são aplicados operadores genéticos como mutação, cruzamento e seleção, na tentativa de obter soluções melhores (COELLO et al., 2007).

NSGA-II (Non-dominated Sorting Genetic Algorithm II) é um algoritmo genético elitista desenvolvido por Deb et al. (2002) para resolução de problemas de otimização multiobjetivo, que neste trabalho foi adaptado para solucionar o problema de sequenciamento multiobjetivo em máquinas paralelas.

Os passos do algoritmo representados no Algoritmo 1 seguem os princípios básicos do NSGA-II, propostos por Deb et al. (2002). O NSGA-II inicia criando uma população aleatória de indivíduos e em seguida a classifica em front de acordo com o nível de dominação. Os passos seguintes consistem em aplicar a seleção por torneio, fazer o cruzamento e a mutação (DEB et al., 2002). Para garantir o elitismo, é feita a atribuição de crowding distance para os indivíduos da população. Crowding distance é a abordagem utilizada para garantir a diversidade da população e consiste na densidade de soluções adjacentes a uma solução em particular (DEB et al., 2002).

Operador de cruzamento

Neste trabalho foi utilizado o operador de cruzamento SJOX (Similar Job Order) (RUIZ et al., 2008). NesSe operador os filhos herdam os pontos em comum nos pais; após essa etapa, um ponto de corte é definido aleatoriamente, e o filho 1 recebe as tarefas antes do ponto de corte do pai 1 ainda não alocadas e após o ponto de corte do pai 2. Enquanto o filho 2 recebe, antes do ponto de corte, as tarefas ainda não alocadas do pai 2 e, após, as tarefas do pai 1. A probabilidade utilizada para o operador de cruzamento, após testes realizados, foi pc = 100%.

Algoritmo 1 - NSGA-II

1 $R_t = P_t \cup Q_t$;
2 F = fastNonDominatedSorting(Rt);
3 $P_{t+1} = \emptyset$;
4 $i = 1$;
5 **enquanto** $P_{t+1} \cup F_i \leq N$ **faça**
6 atribuiCrowdingDistance(Fi);
7 $P_{t+1} = P_{t+1} \cup F_i$;
8 $i = i + 1$;
9 ordena($F_i, \prec_n$);
10 $P_{t+1} = P_{t+1} \cup F_i[1:(N - |P_{t+1}|)]$;
11 Q_{t+1} = geraNovaPopulação(P_{t+1});
12 $t = t + 1$;

Operador de mutação

O operador de mutação utilizado foi o Swap, que consiste em selecionar duas tarefas aleatórias da sequência e permutá-las. A probabilidade utilizada para o operador de mutação, após testes realizados, foi pm = 100%.

MOILS

O MOILS (Multiobjective Iterated Local Search) consiste em uma adaptação ao ILS (Iterated Local Search) com adição de mecanismos para resolução de problemas multiobjetivo e conceitos de dominância de Pareto, portanto segue as mesmas etapas do ILS tradicional (FONSECA et al., 2012).

O ILS é uma meta-heurística que faz uma busca por soluções melhores em um espaço reduzido (LOURENÇO et al., 2003). Seu mecanismo consiste de operações sucessivas de perturbação e busca local, constituídos por quatro etapas: geração da solução inicial, busca local, perturbação e critério de aceitação (FONSECA et al., 2012).

A estrutura básica do MOILS, apresentada no Algoritmo 2, é baseada nos trabalhos desenvolvidos por Assis et al. (2013) e Fonseca et al. (2012).

Algoritmo 2 - MOILS

```
1  Front = geraSoluçõesIniciaisNEH();
2  iter = 0;
3  enquanto não CritérioParada faça
4        s' = selecionaSolução(Front);
5        cont = 0
6        enquanto cont < maxCont faça
7              s'' = perturbação(s');
8              C = buscaLocal(s'');
9              inserido = atualiza(Front, C);
10             se inserido então
11                   cont = 0;
12                   s' = selecionaUltimaSoluçãoInserida(Front);
13             senão
14                   cont = cont + 1;
15 retorna Front;
```

O algoritmo inicia gerando um conjunto de soluções iniciais e inserindo-as no conjunto Front. Em seguida, inicia uma iteração limitada pelo critério de parada onde uma solução não dominada é selecionada do conjunto Front. A partir dessa solução é aplicada uma perturbação e, em seguida, uma busca local que retorna um conjunto de soluções obtidas. Em sequência, o conjunto Front é atualizado. Se alguma solução do conjunto obtido na busca local for inserida em Front, cont é reinicializado e a última solução no Front é selecionada e armazenada em s' para ser explorada na próxima iteração. Caso nenhuma solução seja inserida em Front, cont é incrementado em uma unidade e a iteração continua. maxCont indica o limite máximo de iteração em que uma solução é explorada até que uma nova solução seja escolhida do conjunto Front (ASSIS et al., 2013). O número de iterações definido para o MOILS, após testes realizados, foi maxCont = 5.

ILSMULTI

Além do MOILS, foi também usado o ILSMulti (Multi-Objective Iterated Local Search). Trata-se de um algoritmo proposto por Barros Junior e Arroyo (2010) para um problema de planejamento florestal multiobjetivo baseado no algoritmo MOIGS (Multi-Objective Iterated Greedy Search) proposto por Framinan e Leisten (2007) e se assemelha bastante ao MOILS, diferindo basicamente em relação ao loop interno, que no ILSMulti não está presente.

O Algoritmo 3 descreve seu funcionamento, que começa com a geração de um conjunto de soluções dominantes. Em seguida o algoritmo entra em uma iteração até que o critério de parada seja satisfeito. A cada iteração são executados os procedimentos de seleção de uma solução, perturbação, busca local e aceitação

Algoritmo 3 – ILSMulti

```
1  Front ← geraSoluçõesIniciaisNEH();
2  enquanto não CritérioParada faça
3        C ← Ø;
4        s' ← selecionaSolucao(Front);
5        s'' ← perturbação(s');
6        C ← buscaLocal(s'');
7        atualiza(Front, C);
8  fim – enquanto;
9  retorne Front;
10 fim ILSMulti;
```

Solução inicial no MOILS e ILSMulti

Começar com uma solução inicial de boa qualidade possibilita ao algoritmo evoluir mais rápido, uma vez que os algoritmos precisam encontrar os pontos situados entres esses dois extremos. Para o algoritmo MOILS e o ILSMulti, descrito a seguir, a solução inicial foi gerada a partir da heurística NEH de Nawaz, Enscore Jr. e Ham (1983) com as regras de prioridade EDD (Earliest Due Date) e SPT (Shortest Processing Time). O uso dessas regras de prioridade é justificado por elas obterem uma solução de melhor qualidade para cada objetivo, sendo que o EDD é conhecido por minimizar o lateness máximo; o SPT, por minimizar o tempo total de conclusão (GRIMME et al., 2013).

Seleção no MOILS e ILSMulti

O procedimento de seleção utilizado é o crowding distance do algoritmo NSGA-II. Esse procedimento escolhe soluções que estão mais afastadas das outras; dessa forma, permite uma melhor exploração do espaço de soluções, evitando a concentração da busca em uma área em especifico (ASSIS et al., 2013). Conforme proposto por Assis et al. (2013), o procedimento de seleção deste trabalho apresenta uma diferença em relação ao presente no NSGA-II, enquanto no NSGA-II a soluções nos extremos apresentam valores de crowding distance infinitos; e para o MOILS e o ILSMulti, as soluções extremas possuem o valor de crowding distance equivalente a duas vezes a sua distância para a solução mais próxima. Essa modificação é necessária para evitar que as soluções extremas sejam selecionadas demasiadamente, pois os algoritmos iniciam com poucas soluções iniciais.

Perturbação no MOILS e ILSMulti

Para a perturbação foi utilizado o Swap, equivalente a mutação utilizada neste trabalho para o NSGA-II.

Busca local no MOILS e ILSMulti

Para o MOILS e o ILSMulti, foi utilizado a busca local RVND, proposta por Souza et al. (2010), cujo trabalho mostra a eficiência do RVND em relação ao VND (HANSEN et al., 2008). No RVND, descrito pelo Algoritmo 4, diferentemente do que ocorre no VND, não há uma sequência fixa de procedimentos de busca local, ou seja, essa sequência é determinada aleatoriamente a cada chamada. Quando um procedimento de busca local melhora a solução, a iteração é reiniciada, e a solução encontrada é explorada a partir da primeira busca na sequência. Caso a solução não seja melhorada na busca local, a busca continua na próxima busca local e encerra quando todas as buscas locais forem encerradas (COTA; SOUZA, 2014). Neste trabalho, foram utilizadas 4 estruturas de vizinhança no RVND (CAMPOS et al., 2013):

- Inserção: esta vizinhança gera uma solução ao selecionar uma tarefa aleatória e inseri-la em outra posição, também aleatória, deslocando todas as tarefas para o lado.
- Troca entre duas tarefas: esta vizinhança gera uma solução trocando as posições de duas tarefas diferentes selecionadas aleatoriamente.
- Troca entre duas tarefas sucessivas: esta vizinhança é semelhante à anterior, porém são selecionadas duas tarefas consecutivas em vez de apenas uma.
- Troca xyz: esta vizinhança foi proposta por Campos et al. (2013) e consiste basicamente em gerar uma solução vizinha permutando três tarefas, sendo a posição intermediária fixa selecionada aleatoriamente.

Algoritmo 4 - RVND

```
1  v ← {1,2,3};
2  embaralhar(v);
3  k ← 1;
4  enquanto (k ≤ 4) faça
5      se k = v[1] então
6          s' ← busca1(s);
7      se k = v[2] então
8          s' ← busca2(s);
9      se k = v[3] então
10         s' ← busca3(s);
11     se k = v[4] então
12         s' ← busca4(s);
13
14     se f(s') < f(s) então
15         s ← s';
16         atualizaMelhor(s);
17         k ← 1;
18     senão
19         k++;
20 retorna s;
```

Também foi utilizado no algoritmo ILSMulti, uma busca local adaptada a partir do procedimento Vizinhança descrito no trabalho de Campos (2014). Neste trabalho, o procedimento Vizinhança é usado dentro de uma iteração

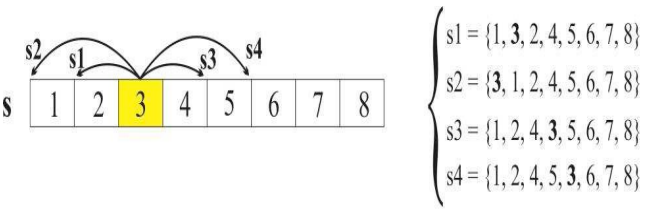
semelhante ao RVND, como representado pelo Algoritmo 5, em que a iteração continua enquanto achar soluções melhores, e termina quando não conseguir melhorar após um determinado número de iterações. Caso se encontre ao menos uma solução melhor no procedimento de Vizinhança, a iteração é reiniciada e a última solução encontrada passa a ser explorada. O número de iterações definido para o BuscaVizinhança, após testes realizados, foi iMax = 8. Esse algoritmo com a BuscaVizinhança será referenciado como ILSMulti1.

O método Vizinhança() gera vizinhos selecionando k tarefas aleatórias da solução e reinserindo-as em z posições adjacentes a esquerda e a direita como é possível visualizar na Figura 3 . Para este trabalho foi considerado o valor de k = 2 e z = 2.

Algoritmo 5 – BuscaVizinhança

```
1  B ← ∅;
2  i ← 1;
3  enquanto (i ≤ iMax) faça
4        B ← Vizinhanca(s);
5        se B atualizado então
6              s ← selecionaUltimaSolucaoInserida();
7              i ← 1;
8        senão
9              i = i + 1;
10 fim_enquanto
11 retorna B;
```

Figura 3: Vizinhança



Fonte: Campos (2014)

Também foi utilizado no ILSMulti uma combinação do RVND junto com a buscaVizinhança,q que consiste basicamente em obter uma lista de soluções em cada procedimento na mesma iteração do ILSMulti e juntá-las para serem avaliadas pelo critério de aceitação. Esse algoritmo ILSMulti, com a busca local RVND junto com a busca local BuscaVizinhança, será referenciado como ILSMulti2.

RESULTADOS OBTIDOS

Nesta seção são apresentados o ambiente de teste e os resultados. Todos os algoritmos foram implementados em C++ com o compilador MinGW, e todos os experimentos foram executados em um ambiente com processador AMD A8-5600K, 3.6Ghz, 8.0 GB de memória RAM e sistema operacional Windows 7 64-bits.

Instâncias de teste

Para a realização dos testes foram geradas 120 instâncias, as quais foram divididas em grupos de pequeno, médio e grande porte, as combinações de tarefa e máquina (n x m). Nas instâncias de pequeno porte, o número de tarefas é n E {15,20} e de máquinas, m E {3,5}. Nas instâncias de médio porte, o número de tarefas é n E {50, 60} e de máquinas, m E {10,20}. Já nas de grande porte, o conjunto é formado por n E {80,100} e m E {20,30}. Para cada combinação de tarefas e máquinas, foram geradas 10 instâncias do problema.

A instâncias são geradas conforme Pereira et al. (2014), em que o tempo de processamento Pij é um número inteiro gerado aleatoriamente, distribuído de forma uniforme no intervalo [50,100]. Os tempos de preparação são gerados uniformemente no intervalo [2/3ηp̄ e 4/3ηp̄], onde p̄ é a média dos tempos de processamento e η = 0.25. As datas de entrega dj das tarefas são geradas através de uma distribuição uniforme no intervalo [(1-R) d̄,d̄] com probabilidade τ, e uniformemente distribuído no intervalo [d̄,((Cmax-d̄)R)+d̄] com probabilidade (1-τ), onde d̄=Cmax (1-τ) é a mediana das datas de entrega e Cmax é calculado conforme a Eq. (1).

$$C_{max}=n/m(\bar{p}+\bar{S}(0.4+10m^2/n^2-n/7)) \quad (1)$$

Onde $\bar{S}$ representa a média dos tempos de preparação. Os parâmetros τ=0.3 e R = 0.25 são o fator de atraso e dispersão das datas de entrega. O instante de liberação é distribuído uniformemente dentro do intervalo [1,10].

Todos os algoritmos possuem o mesmo critério de parada. Para esse parâmetro foi adotada a Eq. (2) empregada por Pereira et al. (2014) como limite de tempo para a execução de cada algoritmo, sendo n o número de tarefa e m o número de máquinas.

$$TempoMax = n*(m/2)*90 \qquad (2)$$

Resultados e comparações

Para cada instância de teste, as meta-heurísticas MOILS, ILSMulti, ILSMulti1 e ILSMulti2 foram executadas 5 vezes para a métrica do Indicador de Hipervolume. O indicador de Hipervolume é uma métrica proposta por Zitzler e Thiele (1998), usada para a medida de qualidade das soluções não dominadas. Para um conjunto de pontos de uma região, esse indicador mensura a área formada entre cada ponto em relação a um ponto de referência R; em seguida, todas as áreas obtidas são somadas. O ponto R deve ser dominado por todos os outros pontos da região considerada. Um valor alto de Hipervolume indica que as soluções estão mais próximas da fronteira do Pareto e, portanto, houve um maior espalhamento e maior convergência das soluções (ZITZLER; THIELE, 1998).

O parâmetro considerado no MOILS foi o número de iterações maxCont. Foram testados os valores 5 e 15, sendo o valor 5 o que obteve melhores resultados no teste do hipervolume. Para o ILSMulti foi considerado o procedimento de busca local, sendo o ILSMulti com a busca local RVND, o ILSmulti1 com a busca local BuscaVizinhança e o ILSMulti2 com a busca local formada pela combinação do RVND com a BuscaVzinhança. O procedimento de perturbação escolhido foi o mesmo para o MOILS e o ILSMulti, o Swap. Para o NSGA-II, os parâmetros considerados foram a probabilidade de cruzamento, com valores 50% e 100%, sendo obtidos melhores resultados com o valor 100%, e a probabilidade de mutação, com valores 3%, 50% e 100%, também sendo obtidos melhores resultados com o valor 100%.

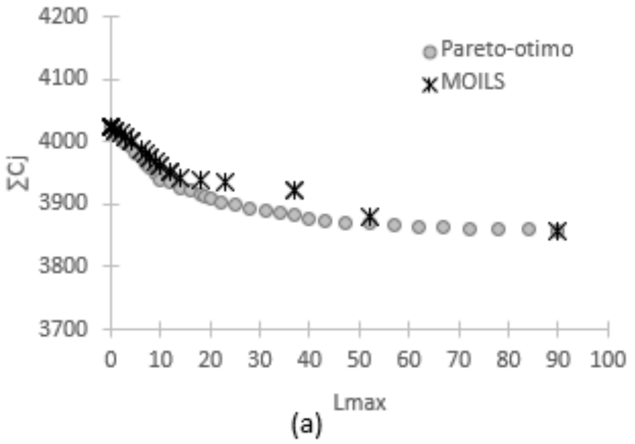
Tanto o MOILS quanto o ILSMulti possuem a mesma estratégia de geração de solução inicial. Como pode ser observado na Figura 4, as soluções obtidas pela regra de prioridade EDD e SPT obtiveram soluções de boa qualidade ao minimizar algum dos objetivos, justificando o uso de uma heurística construtiva como o NEH para ajudar a melhorar a qualidade da solução inicial.

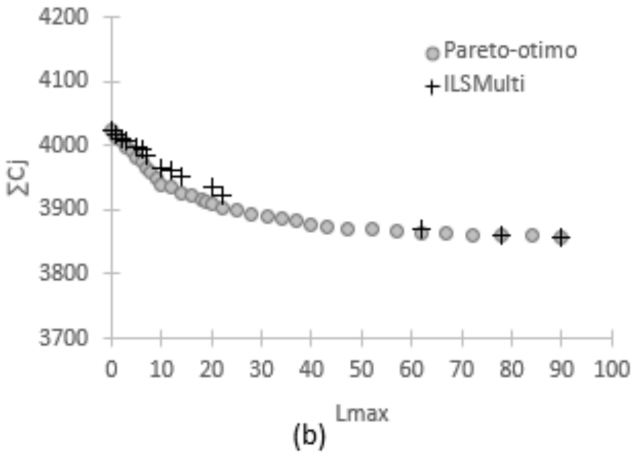
Figura 4: Soluções obtidas pela RP EDD e SPT e heurística NEH

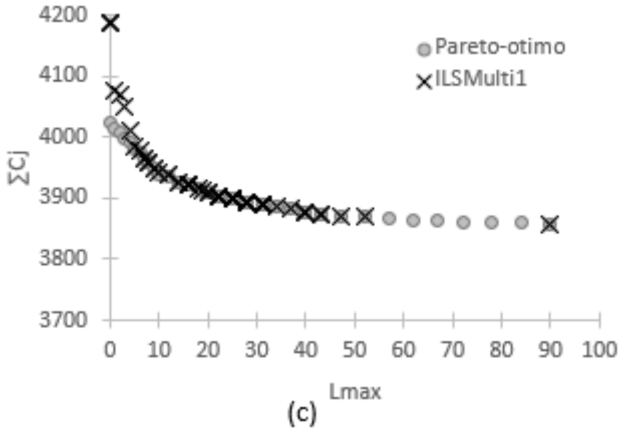
Fonte: dados da pesquisa.

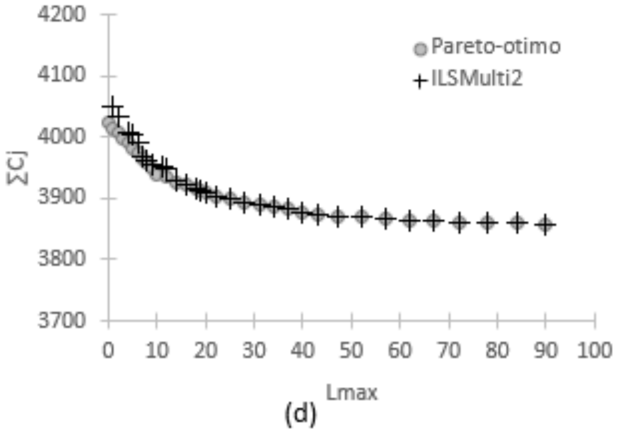
Na comparação com o conjunto de soluções exatas representado na Figura 5, o algoritmo ILSMulti2, usando a perturbação Swap e a busca local composta pelo RVND e BuscaVizinhança, obteve o melhor resultado, encontrando 19 das 34 soluções da fronteira Pareto-ótimo conhecidas para o problema, como pode ser observado em (d). Já em (a), (b) e (c), foram encontradas poucas soluções, sendo (b) a que obteve resultado menos satisfatório na comparação.

Figura 5: Exemplo das fronteiras obtidas pelos algoritmos









Fonte: dados da pesquisa.

Na Tabela 1, é exibido o valor de Hipervolume obtido para os algoritmos nas configurações que foram melhores no teste de Hipervolume. Observa-se que o MOILS obteve melhores resultados quando em relação a ILSMulti, ILSMulti1 e ILSMulti2. Ao comparar o MOILS, ILSMulti, ILSMulti1 e ILSMulti2 com NSGA-II, é possível observar que o último obteve melhor valor de Hipervolume. Além disso, é possível perceber que o ILSMulti obteve um melhor valor de Hipervolume para instâncias menores e que, à medida que aumenta o tamanho da instância, o valor tende a piorar.

CONSIDERAÇÕES FINAIS

Neste trabalho foi proposta uma adaptação à meta-heurística MOILS e ILSMulti. As meta-heurísticas foram utilizadas para a resolução de um problema de sequenciamento multiobjetivo em máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência das tarefas e da máquina. Diversos problemas reais de sequenciamento possuem natureza multiobjetivo, porém são ainda pouco explorados. Os objetivos de minimização são o tempo total de conclusão e o lateness máximo.

Os resultados obtidos com as instâncias geradas no Indicador de Hipervolume demonstram que o MOILS obteve melhor resultados quando

comparado ao ILSMulti, porém não conseguiu superar o NSGA-II. Quando comparado com resultados exatos de uma instância presente na literatura, o ILSMulti2 obteve 19 das 34 soluções ótimas para o problema.

Como um desenvolvimento futuro deste trabalho, sugere-se um estudo sobre o uso de outras estratégias de busca local, assim como modificar a forma de representação da solução para que possa ser possível aplicar estratégias de busca e perturbação que considerem a máquina no procedimento de exploração da solução.

Tabela 1: Resultados encontrados para a instânicas testes

n	m	ILSMulti	ILSMulti1	ILSMulti2	MOILS	NSGA-II
15	03	69.503,66	70.237,24	74.997,06	74.893,04	**78205,56**
15	05	25.769,60	26.509,28	28.681,26	28.382,52	**30068,3**
20	03	138.958,72	146.905,14	155.874,90	154.941,00	**168805,48**
20	05	61.428,36	61.853,12	65.733,26	66.064,40	**69618,44**
50	10	282.841,38	270.854,96	320.878,22	325.460,80	**359327,16**
50	20	33.533,26	33.347,26	34.513,20	34.580,20	**38668,38**
60	10	314.651,62	299.402,36	402.978,72	410.420,66	**487857,66**
60	20	78.543,38	79.367,84	87.197,66	87.786,10	**106112,54**
80	20	176.724,44	176.018,84	243.688,12	248.116,70	**305583,84**
80	30	37.042,74	38.036,38	44.659,56	45.012,34	**52601,14**
100	20	188.542,40	192.215,92	334.380,90	340.953,78	**507705,12**
100	30	78.972,82	80.968,90	113.069,64	114.843,90	**136261,44**
Média		123876,03	122976,44	158887,71	160954,62	**195067,92**
Mediana		78758,10	80168,37	100133,65	101315,00	**121186,99**
Maior		314651,62	299402,36	402978,72	410420,66	**507705,12**
Menor		25769,60	26509,28	28681,26	28382,52	**30068,30**
Desvio Padrão		97757,40	93377,19	132199,69	134909,98	**174798,73**

Fonte: dados da pesquisa.

REFERÊNCIAS

AFZALIRAD, Mojtaba; REZAEIAN, Javad. A realistic variant of bi-objective unrelated parallel machine scheduling problem: NSGA-II and MOACO approaches. Applied Soft Computing, v. 50, p. 109-123, 2017.

ARENALES, M. et al. Otimização discreta: Problemas de produção. In: _____. Pesquisa operacional para cursos de engenharia. Rio de Janeiro: Elsevier, 2007. p. 205–222.

ARROYO, J. E. C. Heurísticas e metaheurísticas para otimização combinatória multiobjetivo. 2002. 227 f. Tese (Doutorado) - Curso de Engenharia Elétrica, Unicamp, Campinas, 2002

ASSIS, L. P. de et al. Problema de Roteamento de Veículos Multiobjetivo com Coleta Seletiva. In: LOPES, Heitor Silvério; RODRIGUES, Luiz Carlos de Abreu; STEINER, Maria Teresinha Arns. Meta-heurísticas em pesquisa operacional. Curitiba: Omnipax, 2013. Cap. 12. p. 181-202.

BARROS JUNIOR, Antônio Almeida de; ARROYO, Elias Claudio. Aplicações de heurísticas em problemas de planejamento florestal multiobjetivo. 2010. 82 f. Dissertação (Mestrado) - Curso de Ciência da Computação, UFV, Viçosa, 2010.

BRUCKER, P. Scheduling algorithms. 5. ed. Berlin Heidelberg: Springer-Verlag, 2007. 380 p.

CAMPOS, Saulo Cunha; ARROYO, José Elias C.; GONÇALVES, Luciana Brugiolo. Uma heuristica grasp-vnd para o problema de sequenciamento de tarefas num ambiente assembly flowshop com três estágios e tempos de setup dependentes da sequência. In: XLVSBPO, 45., 2013, Natal. Anais SBPO. Natal: SBPO, 2013.

CAMPOS, Saulo Cunha. Aplicação de Metaheurísticas para o problema de programação da produção em um ambiente Assembly Flowshop com três estágios e tempos de preparação dependentes da sequência. 2014. 140 f. Dissertação (Mestrado) - Curso de Ciência da Computação, UFV, Viçosa, 2014.

CHEN, Bo; POTTS, Chris N.; WOEGINGERP, Gerhard J. A review of machine scheduling: complexity, algorithms and approximability. In: DU, Ding-zhu; PARDALOS, Panos M. Handbook of combinatorial optimization. [s.l.]: Springer Us, 1998. p. 21-169

CHYU, Chiuh-Cheng; CHANG, Wei-Shung; LI, Ruei-Chi. Archived simulated annealing with two-phase matching improvement for unrelated parallel machine scheduling to minimize fuzzy makespan and average tardiness. In: Second International Conference on Engineering Optimization, Lisboa, Portugal, Set. 2010, p. 6-9.

COELLO, C. A C.; LAMONT, G. B.; VELDHUIZEN, D. A VAN. Evolutionary algorithms for solving multi-objective problems. Boston, MA: Springer US, 2007.

COTA, Luciano Perdigão; SOUZA, Marcone Jamilson Freitas. Novos algoritmos para o problema de sequenciamento em máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência. 2014. 134 f. Dissertação (Mestrado) - Curso de Ciência da Computação, UFOP, Ouro Preto, 2014.

DCOUTHO, Nixon; MORAGA, Reinaldo. Meta-RaPS for a bi-objective unrelated parallel machine scheduling problem. In: Heuristics, metaheuristics and approximate methods in planning and scheduling. Springer International Publishing, 2016. p. 127-139.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, [s.l.],

v. 6, n. 2, p.182-197, abr. 2002.

ETCHEVERRY, Guilherme Vazquez; ANZANELLO, Michel. Sequenciamento em máquinas paralelas com tempos de setup dependentes da sequência. In: XXXIII Encontro Nacional de Engenharia de Produção, 23., 2013, Salvador. Anais ENEGEP. Salvador: ABEPRO, 2013.

FONSECA, O. P.; ASSIS, L.; VIVAS, A. Abordagem Multiobjetivo para o Problema de Roteamento de Veículos Aberto. In: CLAIO-SBPO, 16., 2012, Rio de Janeiro. Annals XVI CLAIO - XLIV SBPO. Rio de Janeiro: SOBRAPO, 2012.

FRAMINAN, J. M.; LEISTEN, R. A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria. [S.l.]: OR Spectrum, 2007.

GRIMME, C.; LEPPING, J.; SCHWIEGELSHOHN, U. Multi-criteria scheduling an agent-based approach for expert knowledge integration. Journal of Scheduling, [s.l.], v. 16, n. 4, p. 369-383, 5 out. 2011.

HANSEN, P.; MLADENOVIC, N. and PÉREZ, J. A. M. Variable neighborhood search: methods and applications. Quartely journal of the Belgian, French and Italian operations research societies 6, 319-360. 2008.

LI, X. et al. A Multiobjective optimization approach to solve a parallel machines scheduling problem. Advances in Artificial Intelligence, [s.l.], v. 2010, p. 1-10, 2010. Hindawi Publishing Corporation.

LIN, Y. K.; FOWLER, J. W.; PFUND, M. E. Bi-criteria heuristic for scheduling on unrelated parallel machines. In: International Conference on Applied Operational Research - ICAOR, 2., 2010, Turku. Lecture Notes in Management Science (2010) 2. Turku: Icaor, 2010. p. 266-274.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated Local Search. In: GLOVER, F.; KOCHENBERGER, G. A. Handbook of metaheuristics. Dordrecht: Kluwer Academic Publishers, 2003. Cap. 11. p. 334-366.

NAWAZ, M.; ENSCORE JR., E.E.; HAM, I. A heuristic algorithm for the m-machine n-job flow-shop sequencing problem. Omega, v.11, n. 1, p. 91-95. 1983.

NOGUEIRA, João Paulo de C. M. et al. Hybrid GRASP heuristics to solve an unrelated parallel machine scheduling problem with earliness and tardiness penalties. Electronic Notes in Theoretical Computer Science, [s.l.], v. 302, p.53-72, fev. 2014.

PEREIRA, A. A. DE S. et al. Metaheurística para o problema de máquinas paralelas não-relacionadas com penalidades por adiantamentos e atrasos. In: XXXV Iberian Latin American Congress on Computational Methods in Engineering - CILAMCE, 35. Fortaleza. Proceedings of the XXXIV Iberian Latin-American Congress on Computational Methods in Engineering. Fortaleza: ABMEC, 2014.

PINEDO, M. L. Scheduling: theory, algorithms, and systems. Boston, MA: Springer US, 2012. 694 p.

RIBEIRO, F. F. Um algoritmo genético adaptativo para a resolução do problema de sequenciamento em uma máquina com penalização por antecipação e atraso da produção. 2009. 123 f. Dissertação (Mestrado) - Curso de Modelagem Matemática e Computacional, CEFET-MG, Belo Horizonte, 2009.

RUIZ, Rubén; ŞERIFOĞLU, Funda Sivrikaya; URLINGS, Thijs. Modeling realistic hybrid flexible flowshop scheduling problems. Computers & Operations Research, v. 35, n. 4, p. 1151-1175. 2008.

SAFAEI, S.; NADERI, R.; SOHRABI, A. Scheduling of unrelated parallel machines using two multi objective genetic algorithm with sequence-dependent setup times and precedent constraints. Int J of Advanced Design and Manufacturing Technology, [s.l.], v. 2, n. 1, p.43-54, 2008.

SOUZA, M.; COELHO, I.; RIBAS, S.; SANTOS, H.; MERSCHMANN, L. A hybrid heuristic algorithm for the open-pit-mining operational planning problem, European Journal of Operational Research 207(2), 1041-1051, 2010.

TAVAKKOLI-MOGHADDAM, R.; BAZZAZI, F.; TAHERI, M. Multi-objective unrelated parallel machines scheduling with Sequence-dependent setup times and precedence constraints. International Journal of Engineering, Transactions A: Basics, [s.l.], v. 21, n. 3, p. 269–278, set. 2008.

VALLADA, E.; RUIZ, R. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. European Journal of Operational Research, [s.l.], v. 211, n. 3, p.612-622, jun. 2011.

YENISEY, M. M.; YAGMAHAN, B. Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. Omega, [s.l.], v. 45, p.119-135, jun. 2014.

ZITZLER, E.; THIELE, L. Multiobjective optimization using evolutionary algorithms: A comparative case study. Lecture Notes in Computer Science, [s.l.], p.292-301, 1998.