Minimização do atraso e avanço em máquinas paralelas não relacionadas: Análise de uma abordagem VND integrada à metaheurística GRASP

Rodney Oliveira Marinho Diana
Centro Federal de Educação Tecnológica de Minas Gerais - CEFETMG
Av. Amazonas, 7675 - Nova Gameleira- CEP 30510-000 - Belo Horizonte, MG, Brasil
rodneyoliveira@dppg.cefetmg.br

Sérgio Ricardo de Souza
Centro Federal de Educação Tecnológica de Minas Gerais - CEFETMG
sergio@dppg.cefetmg.br

RESUMO

Este trabalho aborda o problema de sequenciamento de tarefas em máquinas paralelas não relacionadas para a minimização da soma ponderada dos atrasos e avanços. Este problema é dividido em duas etapas: na primeira etapa deve-se obter uma sequencia de tarefas nas máquinas disponíveis continuamente. Em seguida, é necessária a resolução de um subproblema de otimização, que consiste na inserção de tempos ociosos (*idle times*) entre o processamento das tarefas. Para a resolução do problema, foi proposta uma implementação da metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*), utilizando, como busca local, uma abordagem *Variable Neigborhood Descent* (VND). Foram avaliadas seis estruturas de vizinhança, sendo mostrado, através da metodologia de calibração robusta de parâmetros proposta por Taguchi, que todas as estruturas contribuem na convergência do algoritmo. Ainda foram realizados experimentos com um conjunto de instâncias de *benchmark* disponível na literatura, o método proposto apresentando desempenho superior a abordagem *state-of-art*.

PALAVRAS CHAVE. Sequenciamento, Minimização de atrasos e avanços, Máquinas paralelas não relacionadas, Variable Neighborhood Descent, GRASP.

Área Principal: Metaheurísticas (MH). Otimização Combinatória (OC).

ABSTRACT

This work is addressed to the job scheduling problem for the minimization of the total weighted tardiness and earliness in unrelated parallel machines. This problem is divided into two main steps. In the first step, the method must obtain a sequence of jobs in the machines continuously available. Then, an optimization subproblem is solved, consisting of the insertion of idle times between the processing of consecutive jobs. An implementation of the GRASP (Greedy Randomized Adaptive Search Procedure) metaheuristic is proposed for solving the problem, using, as a local search, the Variable Neighborhood Descent (VND) approach. Six neighborhood structures were evaluated, being shown, by the methodology of robust parameter calibration proposed by Taguchi, that all the structures contribute in the convergence of the algorithm. The proposed approach is evaluated by a set of instances from literature for the problem and the GRASP+VND was able to find better solutions than the state-of-art approach.

KEYWORDS. Scheduling, Total weighted tardiness and earliness, Unrelated parallel machines, Variable Neighborhood Descent, GRASP.

Main Area: Metaheuristics (MH). Combinatorial Optimization (OC).

1. Introdução

O processo da globalização econômica trouxe para as manufaturas a necessidade de continuamente incrementarem sua eficiência e, ao mesmo tempo, garantir um nível de satisfação dos clientes, já que a insatisfação dos mesmos pode levar estes a terceirização de sua produção em mercados com menor custo de produção. A eficiência de uma manufatura depende da sua natureza, nos ambientes em que os contratos são pre-estabelecidos com datas de entregas acordadas com clientes, o custo de atraso na produção pode acarretar um grande prejuízo devido a multas ao fornecedor. Ao mesmo tempo, a produção antecipada de produtos pode acarretar em custos relacionados a estocagem, tanto para o produtor, quanto para o próprio cliente. O critério de otimização utilizado na literatura neste cenário é denominado minimização da soma ponderada dos atrasos e avanços.

Problemas de otimização dos atrasos e avanços apresentam uma característica peculiar que os fazem ser divididos em duas etapas. A primeira etapa, compartilhada por diversos critérios de otimização, consiste na definição da sequencia de tarefas a serem processadas. Já a segunda etapa consiste na resolução de um subproblema de otimização, que define a inserção de *idle times* entre o processamento de duas tarefas consecutivas. *Idle times* consistem no intervalo de tempo ocioso que uma máquina deve permanecer entre o processamento de duas tarefas consecutivas. Na minimização ponderada dos atrasos e avanços, os *idle times* devem ser inseridos, e a sua não inserção acarreta na minimização dos atrasos, mas ao mesmo tempo, maximiza os custos relacionados aos avanços. O processamento de uma tarefa, assim que possível, maximiza o período da mesma em estoque.

As configurações e características dos ambientes de máquinas apresentam grandes variações e determinam a forma em que as tarefas devem ser executadas no sistema. Ambientes de máquinas paralelas (PMSP) são muito estudados na literatura, por serem muito presentes em ambientes reais de manufatura, como a industria têxtil (Liao et al., 2016), manufatura de *displays* de cristal líquido (Chen, 2009), industria cerâmica (Ruiz e Andrés-Romano, 2011), dentre outros. Em geral, estes ambientes possuem um conjunto de máquinas paralelas que podem ser classificadas em 3 tipos: (i) Máquinas idênticas; (ii) Máquinas uniformes; (iii) Máquinas não relacionadas (UPMSP). UPMSP consiste em ambientes nos quais o tempo de processamento de uma tarefa varia de acordo com a máquina na qual a tarefa foi atribuída. Este ambiente vem ganhando espaço na literatura recente afim de aproximar trabalhos acadêmicos de ambientes reais de manufaturas.

Apesar da grande relevância do objetivo de minimização da soma ponderada dos atrasos e avanços, poucos trabalhos são encontrados na literatura relativo a este critério em ambientes UPMSP. Zhu e Heady (2000) propuseram um *Mixed-Integer Programming* (MIP) para a minimização da soma do atrasos e avanços em máquinas paralelas. Randall e Kurz (2007) realizaram um estudo de operadores de *crossover* em algoritmos genéticos. Sen e Bülbül (2015) utilizou o algoritmo de decomposição de Benders para a resolução de MIPs com relaxações, sendo que nestes modelos não foram considerados tempos de preparação. Já Nogueira et al. (2014) propuseram um MIP para o problema, considerando tempos de preparação dependentes da sequência e da máquina. Para a resolução de problemas de grande dimensão, os mesmos autores propuseram uma metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*) integrada a uma abordagem *Iterated Local Search* (ILS) e *Path Relinking* (PR).

Neste trabalho é proposto o estudo de minimização da soma ponderada dos atrasos e avanços em um ambiente UPMSP, considerando tempos de preparação dependentes da sequência e das máquinas. Para isto, é realizado um estudo da metaheurística GRASP integrada a uma busca local VND (*Variable Neighborhood Descent*). Foi proposta a utilização de seis estruturas de vizinhança, todas considerando características do problema para a redução do espaço de busca, característica necessária devido ao alto custo computacional acarretado pela inserção dos *idle times*. Foi utilizado um método de inserção dos *idle times*, proposto na literatura para avaliação em uma única máquina, e no presente artigo adaptado para a avaliação de máquinas paralelas. Os resultados obtidos pela abordagem são comparados com os alcançados pela metaheurística GRASP+ILS+PR

proposta por Nogueira et al. (2014).

O restante deste trabalho segue a seguinte organização. Na Seção 2 é apresentado uma definição formal do problema estudado. Já na Seção 3 são apresentados os detalhes da metaheurística GRASP utilizada, o algoritmo de inserção dos *idle times* e as seis estruturas de vizinhança utilizadas pela busca local VND. Na Seção 4 é apresentado o ambiente e o conjunto de instâncias utilizados, além da metodologia experimental, uma análise dos parâmetros e estruturas de vizinhança utilizadas, além dos resultados obtidos pela abordagem. Por fim, não Seção 5 são apresentados as conclusões e perspectivas para trabalhos futuros.

2. Definição do Problema

O problema estudado neste trabalho consiste no sequenciamento de $N = \{1, 2, ..., n\}$ tarefas em um conjunto $M = \{1, 2, ..., m\}$ de máquinas disponíveis continuamente e não relacionadas. Cada tarefa $j \in N$ possui um tempo de processamento p_{ij} dependente da máquina i na qual esta é processada. Duas tarefas não podem ser processadas na mesma máquina ao mesmo tempo e, uma vez iniciado o processamento de uma tarefa j, este processamento não pode ser interrompido até sua conclusão. Além dos tempos de processamento, cada tarefa j possui um tempo de preparação s_{ijk}, o qual é dependente da sequencia e da máquina, ou seja, depende da tarefa k anterior à tarefa j na sequencia da máquina i. Cada tarefa j também possui uma data de entrega (*due date*) d_j, a qual representa a data de entrega desejada da tarefa.

Considerando este ambiente, pode-se definir o instante de término de uma tarefa j como: $C_{ij} = S_{ij} + s_{ijk} + p_{ij} : S_{ij} \geq C_{ik}$, sendo S_{ij} o instante inicial de processamento da tarefa j; já k representa a tarefa que precede j na sequencia da máquina i. O objetivo estudado nesta trabalho é a soma ponderada dos atrasos e avanços, representado por:

$$minimize \;\; f(x) = \sum_{j=1}^{N} \beta_j T_j + \alpha_j E_j \tag{1}$$

O atraso é representado por $T_j = \max(C_{ij} - d_j, 0)$; já o avanço é definido como $E_j = \max(d_j - C_{ij}, 0)$, sendo que β_j e α_j representam, respectivamente, os fatores de ponderação de atraso e avanço.

Para a avaliação do atraso e avanço ponderado é necessária a definição dos instantes iniciais S_{ij} de processamento de cada tarefa j atribuída a uma máquina i. Esta definição requer a resolução de um subproblema de otimização para a inserção dos *idle times* entre a execução de tarefas consecutivas. *Idle times* são necessários para a determinação dos instantes de início da execução de cada tarefa, já que, se esta for iniciada no primeiro instante de tempo que estiver disponível para processamento, o custo relacionado ao atraso será minimizado, mas, ao mesmo tempo, o custo relacionado ao avanço será maximizado. Assim, é necessário encontrar as datas de início de cada tarefa que minimizem o custo total de atraso e avanço.

3. *Greedy Randomized Adaptive Search Procedure* - GRASP

GRASP é uma metaheurística de busca iterativa (Feo e Resende, 1995) que realiza, a cada iteração, dois passos principais. Como apresentado no Algoritmo 1, o primeiro passo consiste na construção de uma solução de partida s, seguida do refinamento desta solução por um método de busca local.

A fase construtiva da metaheurística GRASP apresentada no Algoritmo 2, consiste em gerar uma solução $\bar{s}$, iterativamente, por seleção aleatória de elementos ordenados por uma função gulosa. Feo e Resende (1995) propõem que, a cada iteração da fase construtiva, os candidatos a entrarem em $\bar{s}$ sejam avaliados segundo uma função $g(\bar{s})$, gulosa e adaptativa, formando uma lista restrita de candidatos LRC e selecionando aleatoriamente um candidato desta lista para compor $\bar{s}$. Para o problema estudado, a construção de uma solução consiste primeiramente na criação de uma solução parcial $\bar{s}$, sem nenhuma tarefa alocada (Linha: 2) e criação de um conjunto de tarefas

Algoritmo 1 GRASP

1: **procedure** GRASP(α_{grasp}, N, M)
2: $s^* \leftarrow NULL$;
3: **while** (criterio de parada não satisfeito) **do**
4: $s \leftarrow ConstrucaoGRASP(\alpha_{grasp}, N, M)$;
5: $s \leftarrow VND(s)$;
6: **if** $f(s) < f(s^*)$ **then**
7: $s^* \leftarrow s$;
8: **end if**
9: **end while**
10: **end procedure**

não alocadas, $\bar{N}$ (Linha: 3). Em seguida é realizado o processo incremental para geração de uma solução. A cada iteração, todas as tarefas $j \in \bar{N}$, são inseridas em $\bar{s}$, uma a uma, ao final da sequencia de cada máquina $m \in M$, gerando $|\bar{N}| * |M|$ soluções parciais $\bar{s}'$, inserindo estas em uma lista de soluções candidatas LC (Linhas 5 à 12). Após esta fase, é gerado uma lista restrita de candidatos LRC, contendo as $\max(1, \alpha_{grasp} \times |LC|)$ soluções parciais $\bar{s}' \in LC$, que apresentarem as melhores avaliações da função objetivo $f(\bar{s}')$ (Linhas 13 à 15). Uma nova solução parcial é selecionada aleatoriamente com probabilidade uniforme em LRC, substituindo a solução parcial $\bar{s}$ (Linha: 16). Por fim, o conjunto de tarefas não alocadas $\bar{N}$ é atualizado, retirando-se a tarefa recém-alocada na solução parcial $\bar{s}$ (Linha: 17). Este processo é repetido até que o conjunto $\bar{N}$ fique vazio.

Algoritmo 2 Fase construtiva GRASP

1: **procedure** CONSTRUCAOGRASP(α_{grasp}, N, M)
2: $\bar{s} \leftarrow \{\}$;
3: $\bar{N} \leftarrow \{j \in N\}$;
4: **while** ($\bar{N} \neq \{\}$) **do**
5: **for** $i = 1, ..., |M|$ **do**
6: **for** $j \in \bar{N}$ **do**
7: $\bar{s}' \leftarrow$ nova solução a partir de $\bar{s}$, inserindo a tarefa j, ao fim da sequencia da
8: máquina i;
9: $\bar{s}' \leftarrow$ inserir *idle times* na solução $\bar{s}'$;
10: $LC \leftarrow LC \cup \bar{s}'$
11: **end for**
12: **end for**
13: $LC \leftarrow$ ordena LC em ordem crescente de $f(\bar{s}')$;
14: $size \leftarrow \max(1, \alpha_{grasp} \times |LC|)$;
15: $LRC \leftarrow \{x \in LC : x_1, ..., x_{size}\}$;
16: $\bar{s} \leftarrow$ solução aleatória com probabilidade uniforme de LRC;
17: $\bar{N} \leftarrow \bar{N} - jobs(\bar{s})$;
18: **end while**
19: **return** $\bar{s}$;
20: **end procedure**

3.1. Inserção dos *idle times* para avaliação da função objetivo

Neste trabalho é utilizado, para a inserção do *idle times*, o algoritmo proposto por Rosa et al. (2017). Este algoritmo foi inicialmente proposto para sequenciamento em uma única máquina, mas pode ser estendido para máquinas paralelas, já que a inserção dos *idle times* de cada máquina

é um processo independente. O método de inserção dos *idle times* é descrito no Algoritmo 3. Neste algoritmo, primeiramente cada tarefa j, atribuída a uma máquina i, tem seu *start time* S_{ij} e *completion time* C_{ij} avaliados sem a inserção dos *idle times*, ou seja, $S_{ij} \leftarrow C_{ik}$ e $C_{ij} \leftarrow C_{ik} + s_{ijk} + p_{ij}$, sendo k a tarefa que precede j na máquina i (Linhas 4 à 8). Em seguida, da última para a primeira tarefa presente na sequencia da máquina i (Linha: 9), é verificado se a inserção de *idle times* acarretará em benefício para a redução da função objetivo. Assim, primeiramente é verificado se a última tarefa apresenta avanço; caso não apresente, não será necessária a inserção de *idle times* para esta tarefa; caso contrário, é criado um bloco de tarefas B com a última tarefa da sequencia da máquinas i. Neste ponto é verificado, se a redução no custo de avanço pelo deslocamento à direita das tarefas presentes no bloco B, compensa o incremento no atraso das mesmas tarefas. Para isto é calculado o valor:

$$MC(B) = \sum_{j \in B: C_j \geq T_j} \beta_j - \sum_{j \in B: C_j < E_j} \alpha_j \tag{2}$$

Duas situações podem ocorrer, dependendo do resultado de $MC(B)$:

(i) se $MC(B) \geq 0$: A redução dos custos de avanço, por um possível deslocamento à direita das tarefas do bloco B, não compensarão o atraso acarretado pelo mesmo deslocamento;

(ii) se $MC(B) < 0$: Neste caso, o deslocamento do início de processamento do bloco B, à direita, em φ unidades de tempo, irá acarretar em decremento da soma ponderado dos avanços e atrasos da máquina i. Neste caso, deve-se definir o tamanho do passo φ , sendo $\varphi = \min\{m_1, m_2, m_3\}$. A variável m_1, representa a diferença entre o instante de término da última tarefa $last(B)$, presente no bloco B, e o instante de tempo inicial da tarefa $last(B)+1$, a qual é a primeira tarefa, após a tarefa $last(B)$, na sequencia da máquina i. Logo, $m_1 = C_{i,last(B)} - S_{i,last(B)+1}$. Já $m_2 = \min_{j \in B: E_j \leq C_{ij} < T_j}\{T_j - C_{ij}\}$ e $m_3 = \min_{j \in B: C_{ij} < E_j}\{E_j - C_{ij}\}$.

Definido o grau de deslocamento à direita, dado por φ, o instante de processamento inicial e de término de todas as tarefas pertencentes ao bloco B são incrementados em φ unidades (Linhas 14 à 17). Caso φ seja definido por m_1, ou seja, o instante de término da última tarefa do bloco B, tenha encontrado com o instante inicial da primeira tarefa posterior ao bloco, $last(B) + 1$, esta tarefa ($last(B) + 1$) deve ser incorporada ao bloco (Linhas 18 à 20). Em seguida, $MC(B)$ é recalculado e o procedimento é repetido (Linha: 12), até encontrar um valor de $MC(B) \geq 0$, ou seja, a redução no custo de avanço devido ao deslocamento à direita do bloco de tarefas B, não compensa o incremento no custo de atraso das tarefas do mesmo bloco.

A partir do momento em que $MC(B) \geq 0$, deve-se criar um novo bloco B, com a tarefa que precede na máquina i, a primeira tarefa presente no último bloco B (Linha: 11). Assim, na primeira iteração, apenas a última tarefa sequenciada na máquina i estará em B; Na segunda iteração, o novo bloco B, deve ser criado com a penúltima tarefa sequenciada na máquina i, e o procedimento de deslocamento à direita (inserção do *idle times*) das tarefas presentes no bloco B se repete; Na terceira iteração, o bloco inicia com a tarefa que precede a penúltima tarefa escalonada na máquina i, e assim por diante. O algoritmo termina, após todas as tarefas presentes na sequência da máquinas i, tenham sido exploradas. Deve-se observar que todo bloco inicialmente apresenta apenas uma tarefa. Este bloco pode incorporar novas tarefas, à medida em que o deslocamento à direita do bloco, acarretar no encontro do instante final do bloco B, com o instante inicial da tarefa que suceder este.

3.2. *Variable Neighborhood Descent*

Para a fase de refinamento do GRASP, é proposta a utilização da metaheurística VND introduzida por Mladenovic e Hansen (1997). Este é um método de refinamento, que realiza a busca no espaço de soluções alternando, de modo cíclico e sistemático, as estruturas de vizinhança a serem consideradas.

Algoritmo 3 Inserção de Idle Times

for $i = 1, ..., m$ **do**
  $C_{i0} \leftarrow 0$;
  $k \leftarrow 0$;
  for $j = 1, ..., n_i$ **do**
    $S_{ij} \leftarrow C_{ik}$;
    $C_{ij} \leftarrow C_{ik} + s_{ijk} + p_{ij}$;
    $k \leftarrow k + 1$;
  end for
  for $j = n_i, n_i - 1, n_i - 2, ..., 1$ **do**
    if $C_{ij} < E_j$ **then**
      $B \leftarrow \{j\}$
      while $(MC(B) < 0)$ **do**
        $\varphi = \min(m_1, m_2, m_3)$;
        for $j' \in B$ **do**
          $C_{ij'} \leftarrow C_{ij'} + \varphi$;
          $S_{ij'} \leftarrow S_{ij'} + \varphi$;
        end for
        if $C_{i,last(B)} = S_{i,last(B)+1}$ **then**
          $B \leftarrow B \cup last(B) + 1$;
        end if
      end while
    end if
  end for
end for

Considere um conjunto $N_s = \{N_1, N_2, ..., N_{kmax}\}$ composto por diferentes estruturas de vizinhança. Como descrito no Algoritmo 4, o procedimento começa pela exploração da primeira estrutura de vizinhança $N_1(s)$, alcançado um ótimo local (no caso da exploração *Best Improvement* – BI ou *First Improvement* – FI), passa-se à exploração da segunda vizinhança $N_2(s)$. Se, ao final da exploração de $N_2(s)$, é observada melhoria, a exploração da primeira estrutura de vizinhança, $N_1(s)$, é retomada. Caso contrário, passa-se a explorar a próxima vizinhança, $N_3(s)$. Este processo de alternância cíclica e sistemática de vizinhanças continua, até que a última vizinhança N_{kmax}, pertencente a N_s, seja explorada sem melhora na solução corrente.

3.3. Estruturas de vizinhança

3.3.1. Inserção Interna - N_{ii}

Em um movimento de Inserção Interna, uma tarefa j, que ocupa a posição p na sequência de tarefas atribuídas à máquina i, é retirada de sua posição original p e inserida em uma outra posição $p' \neq p$, na mesma máquina i. Para explorar esta estrutura, uma máquina i é selecionada aleatoriamente, de modo que $i \in M_t = \{i \in M : E_i + T_i > 0\}$, e aplicada uma estratégia *Best improvement* (BI) ou *First Improvement* (FI) na máquina i selecionada.

3.3.2. Troca Interna - N_{ti}

Em um movimento de troca interna, duas tarefas k e j, atribuídas a uma mesma máquina i, têm suas posições trocadas entre si. Esta estrutura é explorada da mesma forma da estrutura N_{ii} (Seção 3.3.1).

3.3.3. Inserção Externa - N_{ie}

Uma tarefa k é retirada da máquina i, à qual estava atribuída, e posteriormente inserida em uma máquina w, em que $i \neq w$. Esta estrutura é explorada de forma reduzida, sendo primeiramente

Algoritmo 4 Variable Neighborhood Descent

```
1:  procedure VND(s)
2:      N_s ← {N_1, N_2, ..., N_kmax};
3:      s* ← s;
4:      k ← 1;
5:      while (k < kmax) do
6:          s' ← argmin N_s[k](s*);
7:          if f(s') < f(s*) then
8:              s* ← s';
9:              k ← 0;
10:         else
11:             k ← k + 1;
12:         end if
13:     end while
14:     return s*;
15: end procedure
```

selecionada aleatoriamente uma máquina i de M_t, sendo $M_t = \{i \in M : \forall\ E_i + T_i > 0\}$; posteriormente, outra máquina w é selecionada aleatoriamente da lista L, sendo $L = \{w \in M : \forall\ w \neq i\}$; então, é aplicado um método de descida (BI ou FI) nas máquinas i e w, retirando tarefas da máquina i e inserindo nas posições da máquina w. Caso a busca resultar em algum movimento de melhora, a solução é retornada; caso contrário, a busca é reiniciada, selecionando outra máquina w em L. A busca termina ao encontrar um movimento de melhora em alguma máquina w selecionada de L, ou se todas as máquinas pertencentes a L tenham sido exploradas.

3.3.4. Troca Externa - N_{te}

Sejam k e j as tarefas que ocupam respectivamente as posições p e p' das máquinas i e k, sendo $i \neq w$. Um movimento de troca externa consiste na tarefa k ser processada na posição p' da máquina w; já a tarefa j passa a ser processada na posição p da máquina i. Esta estrutura também é processada de forma restrita, considerando os mesmos passos utilizados na estrutura N_{ie} (Seção 3.3.3).

3.3.5. Inserção *Iterated Greedy Search* - N_{igsi}

A estrutura N_{igsi} é baseada na fase de destruição e construção da metaheurística Iterated Greedy Search (IGS). Esta estrutura consiste em remover aleatoriamente rj tarefas, de uma solução s, gerando uma solução s'. Estas tarefas podem estar presentes em qualquer máquina, mas devem ser inseridas pela ordem de remoção de uma lista L_{rj}; em seguida, cada tarefa $j \in L_{rj}$ é removida de forma ordenada de L_{rj}, sendo inserida, na máquina/posição de s', que introduza o menor custo de atraso e avanço ponderados. Deve-se notar que, s' pode retornar uma solução com pior avaliação de função objetivo em relação a s, mas o próprio VND se encarrega de não aceitar soluções de piora. Percebe-se também que não há como utilizar estratégias de busca BI ou FI, mas o parâmetro rj deve ser atribuído.

3.3.6. Troca *Iterated Greedy Search* - N_{igst}

Apesar de também ser baseada na fase de destruição e construção da metaheurística IGS, a N_{igst} apresenta algumas diferenças na exploração da estrutura em relação à N_{igsi}. A exploração de N_{igst} consiste em selecionar aleatoriamente, r_j tarefas, pertencentes a qualquer máquina. As tarefas não são removidas como em N_{igsi}. Em seguida, são avaliadas, para cada tarefa selecionada no passo anterior, todas as possíveis trocas (interna - Seção 3.3.2 e externa 3.3.4) com todas as demais tarefas. Os movimentos que acarretarem em maior redução da soma ponderada dos atrasos e avanços serão efetivados e retornados pelo método de exploração da estrutura.

4. Metodologia experimental e resultados

Para avaliar a performance da abordagem proposta, foram realizados experimentos sobre o conjunto de instâncias propostas por Nogueira et al. (2014). Nos experimentos são utilizadas 260 instâncias, subdivididas em dois grupos, denominados *small* e *large*, que representam, respectivamente, instâncias de pequena e grande dimensão. As instâncias *small* possuem resultados ótimos obtidos pela resolução de um *Mixed-Integer Programming* pelo software ILOG-IBM CPLEX 12.2. Já as instâncias *large* possuem *upper bounds* obtidos por uma metaheurística híbrida, a qual integra o GRASP com as metaheurísticas ILS e PR, denominada aqui como GRASP+ILS+PR. O subconjunto *small* possui 60 instâncias distribuídas pela combinação do números de tarefas n e máquinas m, dada por: $n = \{8, 9, 10\}$ e $m = \{3, 5\}$. Os tempos de processamento estão distribuídos uniformemente no intervalo $[50, 100]$. Já os tempos de preparação são distribuídos de forma aleatória, com distribuição uniforme, dentro do intervalo $[\frac{2}{3}\eta\bar{p}, \frac{4}{3}\eta\bar{p}]$, sendo $\bar{p}$ a média dos tempos de processamento e $\eta = 0.25$. As penalidades de atraso (β_j) e avanço(α_j) são geradas aleatoriamente com distribuição uniforme no intervalo $[1, 100]$. As datas de entrega d_j são geradas aleatoriamente, sendo $d_j \in [(1 - R)\bar{d}, \bar{d}]$, com probabilidade τ; já com probabilidade $1 - \tau$, $d_j \in [\bar{d}, \bar{d} + (C_{max} - \bar{d})R]$, sendo o *due date* estimado médio $\bar{d} = C_{max}(1 - \tau)$, e o makespan estimado $C_{max} = ((0, 4+10/(n/m)^2 - \eta/7)\bar{s}+\bar{p})\times(n/m)$, sendo $\bar{s}$ a média dos tempos de preparação, $R = 0, 25$ e $\tau = 0, 3$. O segundo subconjunto é formado por 200 instâncias de grande porte ($large$), distribuídas na combinação dos conjuntos $n = \{70, 80, 90, 100\}$ e $m = \{10, 15, 20, 25, 30\}$. As instâncias são geradas sob as mesmas condições adotadas para as instâncias de pequeno porte. Para cada uma das combinações dos parâmetros número de tarefas (n) e número de máquinas (m), foram geradas 10 instâncias distintas.

Os experimentos realizados neste trabalho foram conduzidos em um processador Intel Pentium E5200 @ 2.50GHz com 2GB of RAM e em sistema operacional Linux Ubuntu 16.04. Os algoritmos foram desenvolvidos em linguagem Java como Java Development Kit 1.8. Segundo o *Single Thread Performance* dado por CPU *benchmarks* do PassMark Software[1], o hardware utilizado por Nogueira et al. (2014) é cerca de $1,18$ vezes superior ao utilizado neste trabalho. Para a execução do algoritmo proposto, foi utilizado o critério de parada baseado no tempo de processamento de $n \times m \times 50$ milissegundos proposto por Nogueira et al. (2014). Cada instância foi processada dez vezes e os resultados agrupados por conjunto de máquinas e tarefas.

Os resultados foram comparados com as soluções obtidas pelo MIP, no caso do conjunto *small*, e, para o conjunto *large*, a metaheurística híbrida GRASP+ILS+PR foi utilizada como referência. A comparação dos resultados foi realizado pelo GAP médio dos grupos $n \times m$. O GAP (y_i) para uma execução de uma instância é dado por:

$$y_i = \frac{f(i) - \min f(i)}{\min f(i)} \tag{3}$$

sendo $f(i)$ o resultado obtido pelo método para uma instância; e $\min f(i)$ o melhor resultado alcançado pelos experimentos de Nogueira et al. (2014) para a mesma instância.

Como, para o conjunto *large*, os resultados são obtidos por abordagens estocásticas, foi realizado um *Pareid t-test* considerando a hipótese do atraso e avanço médio obtido pela abordagem GRASP-VND ser inferior ao obtido pela metaheurística híbrida GRASP+ILS+PR. As hipótese validadas pelo teste foram: (i) $H_0 : \mu_1 - \mu_2 = 0$; e (ii) $H_1 : \mu_1 - \mu_2 < 0$, sendo μ_1 e μ_2, respectivamente, os valores médios obtidos pelas abordagens GRASP-VND e GRASP+ILS+PR. Os testes estatísticos consideram nível de confiança de 95% ($\alpha = 0.05$) e foram realizados na versão 3.2.3 do software estatístico *R-Project*.

[1]Disponível em `https://www.cpubenchmark.net/singleThread.html` e acessado em 26 de março de 2018.

4.0.7. Analise de parâmetros e estruturas de vizinhança

Neste trabalho é analisado o algoritmo GRASP integrado com uma busca local VND, devendo responder a inúmeros cenários distintos, podendo haver inúmeras variações nas características do problema, como dimensionalidade das instâncias, fatores de definição das datas de entrega, dentre outros. Para reduzir a influência destes fatores não controláveis na resposta obtida pela metaheurística, é utilizada, para a definição dos parâmetros do GRASP-VND, a metodologia *Robust Parameter Design*, proposta por Taguchi (1987), que é uma técnica de projetos de experimentos (*Design of Experiments* – DoE) para a definição dos fatores controláveis de um processo, de modo de tornar a resposta deste menos sensível (mais robusta) a fatores não controláveis, denominados como *Noise Factors*.

Foram avaliados sete fatores controláveis para a definição do GRASP-VND. Para o fator de ponderação do tamanho da lista restrita de candidatos LRC, foram avaliados seis níveis $\alpha_{grasp} = \{0,075;\ 0,1;\ 0,125;\ 0,15;\ 0,175 \text{ e } 0,2\}$. Já os seis demais fatores representam cada estrutura de vizinhança e consideram três níveis cada. O primeiro nível (NA) indica que a estrutura não deve ser utilizada, já o segundo (FI) e o terceiro (BI) indicam, respectivamente, que a estrutura deve usar as estratégias de busca *First Improvement* e *Best Improvement*. A estrutura N_{igsi} e N_{igst} não utilizam as estratégias FI e BI. Assim, o segundo e o terceiro níveis indicam o valor do parâmetro rj, sendo, respectivamente, $[2, 3]$ e $[1, 2]$.

O método de Taguchi utiliza um DoE do tipo fatorial fracionado. Foi utilizado um arranjo ortogonal $L_{18}(6^1 \times 3^6)$. Assim, são executados $18 * r$ experimentos para a calibração dos fatores controláveis, dado que r é o número de replicações para cada variação de parâmetro. Como o conjunto de instâncias possui 26 combinações possíveis de $m \times n$, optou-se por selecionar 26 instâncias ($r = 26$) de forma aleatória, sendo cada instância representando uma combinação possível dos fatores $m \times n$. O desenho experimental foi gerado no software `R` pelo pacote *Quality Tools* [2].

As respostas resultantes de cada uma das combinações dos parâmetros são transformadas em um *signal-to-noise ratio* (S/N_{ratio}). Este fator é uma medida de qualidade dos fatores de controle, de modo que a combinação ótima dos fatores deve minimizar a resposta média e a variabilidade dos resultados. Neste trabalho, como se deseja atingir o menor valor possível, utilizou-se para cálculo do S/N_{ratio} a função de perda "*smaller the better*", expressa por:

$$S/N_{ratio} = -10 \log \left(\frac{1}{r} \sum_{i=1}^{r} y_i^2 \right) \tag{4}$$

sendo y_i a resposta de uma execução do algoritmo proposto para uma determinada replicação. Como há variação na escala dos valores de resposta pelos grupos de instâncias estudadas, optou-se por normalizar as respostas pelo desvio percentual relativo (GAP) - Equação 3, sendo $f(i)$ a resposta de uma instância i e $\min f(i)$ a resposta mínima para a instância i alcançada por todos os experimentos de calibração.

Para validar se os níveis dos fatores controláveis têm influência real na resposta, foi realizado uma ANOVA (Analysis of variance) do GAP expresso na Tabela 1. Considerou-se um nível de confiança de 95% ($\alpha = 0,05$), sendo as premissas ANOVA de normalidade, independência e homocedasticidade dos resíduos verificadas e atendidas em todos os cenários. Pelo resultado do ANOVA, pode-se observar que todos os fatores avaliados influenciam na resposta da metodologia proposta, já que para todos os fatores a hipótese nula de igualdade nos níveis dos fatores foram descartadas (p-valor $< 0,05$). Considerando que maiores valores de S/N_{ratio} indicam os níveis dos fatores controláveis que minimizam a interferência dos *noise factors* na resposta, pela Figura 1 é possível observar que a combinação dos fatores que minimiza o GAP e a variação no mesmo são: (i) $\alpha_{grasp} = 0,15$; (ii) as estruturas N_{igsi} e N_{igst} usam, respectivamente, para o parâmetro r_j, os valores 3 e 2; (iii) já as estruturas N_{ii}, N_{ie} e N_{te} utilizam a estratégia *Best Improvement*; (iv) por fim, a estrutura N_{ti} utiliza a estratégia de busca *First Improvement*.

[2] Disponível em `http://www.r-qualitytools.org/`

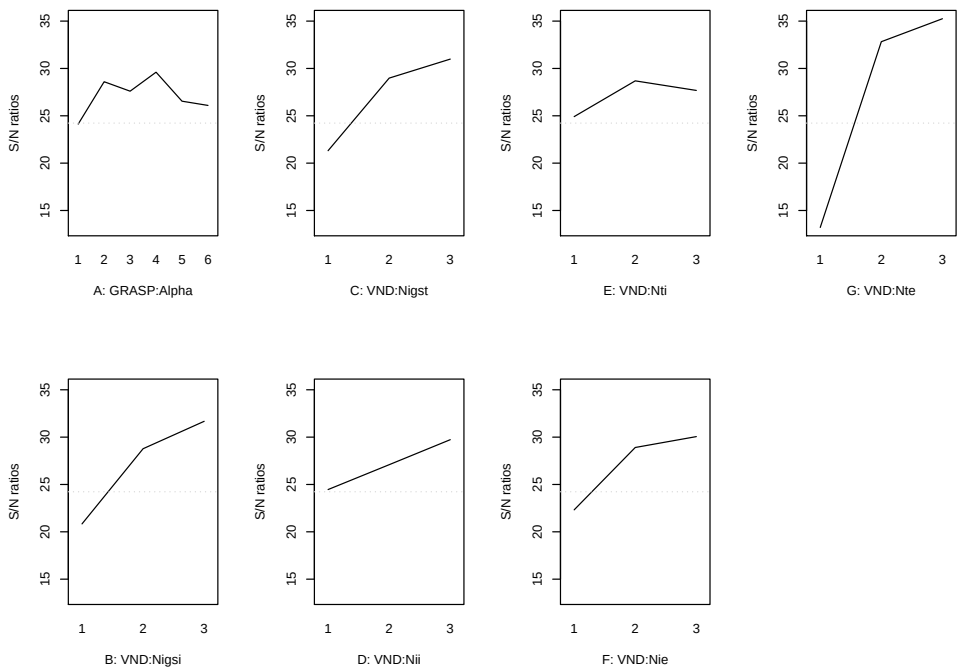

Figura 1: S/N_{ratio} para os fatores controláveis do GRASP e VND.

Tabela 1: ANOVA dos GAPs para os fatores controláveis das abordagem GRASP-VND.

Factor	Df	Sum Sq	Mean Sq	F	p-valor
GRASP.Alpha	5	72,45	14,491	86,375	**<2e-16**
VND.Nigsi	2	30,19	15,093	89,967	**<2e-16**
VND.Nigst	2	31,29	15,645	93,254	**<2e-16**
VND.Nii	2	28,44	14,221	84,766	**<2e-16**
VND.Nti	2	28,49	14,243	84,900	**<2e-16**
VND.Nie	2	30,44	15,218	90,712	**<2e-16**
VND.Nte	2	39,91	19,953	118,936	**<2e-16**
Instâncias	25	4,23	0,169	1,008	0,454
Residuals	425	71,30	0,168		

4.0.8. Resultados

Como pode ser observado na Tabela 2, a abordagem GRASP+VND proposta encontrou os resultados ótimos, definidos pelo MIP e resolvido pelo ILOG-IBM CPLEX 12.2 em Nogueira et al. (2014), para todos os experimentos realizados para os conjuntos com 8 e 9 tarefas. Já para os conjuntos com 10 tarefas, a abordagem propostas encontrou resultados ligeiramente inferiores aos ótimos. Para este conjunto de instâncias, os resultados da abordagem proposta são similares aos encontrados pelo GRASP+ILS+PR.

Para o conjunto de instâncias *large*, observa-se na Tabela 3 que, com o mesmo critério de parada, mas com uma CPU inferior à utilizada nos experimentos do GRASP+ILS+PR, a abordagem proposta superou esta em todos os cenários avaliados, inclusive alcançando para todas as instâncias resultados inferiores aos melhores resultados obtidos por Nogueira et al. (2014). Observa-se também que os resultados são todos estatisticamente significantes, já que, para todos os cenários, o teste estatístico encontrou evidências suficientes para aceitar a hipótese alternativa (p-valor < 0.05), invalidando assim a hipótese nula, o teste indica assim que os resultados apresentados pelo GRASP+VND são superiores aos alcançados pelo GRASP+ILS+PR.

Tabela 2: Resultado do GAP para os conjuntos de instâncias *small*.

m	n	GRASP+VND	GRASP+ILS+PR
3	8	**0,00%**	**0,00%**
	9	**0,00%**	0,35%
	10	0,23%	**0,00%**
5	8	**0,00%**	**0,00%**
	9	**0,00%**	0,01%
	10	0,19%	0,83%

Tabela 3: Resultado para os conjuntos de instâncias *large*.

m	n	GRASP+VND		GRASP+ILS+PR		$H_1 : \mu_1 - \mu_2 < 0$
		$f(x)$ médio	GAP	$f(x)$ médio	GAP	p-valor
10	70	155826,20	**-4,16%**	165979,02	2,13%	**6,62e-08**
	80	199649,44	**-5,19%**	215933,10	2,66%	**5,93e-08**
	90	246649,96	**-6,11%**	268225,74	2,17%	**1,82e-07**
	100	322718,04	**-4,49%**	344688,28	2,04%	**7,42e-08**
15	70	104841,77	**-4,82%**	112058,04	1,75%	**1,26e-09**
	80	134218,21	**-4,83%**	144329,86	2,38%	**1,33e-09**
	90	161112,46	**-6,64%**	176099,50	2,10%	**1,93e-08**
	100	216658,40	**-5,83%**	233955,84	1,70%	**1,12e-09**
20	70	67221,07	**-5,83%**	73657,22	3,19%	**1,57e-09**
	80	83062,10	**-6,66%**	91741,00	3,16%	**1,33e-10**
	90	110411,07	**-5,21%**	119827,16	3,01%	**3,19e-08**
	100	136480,83	**-5,48%**	148101,80	2,57%	**4,64e-09**
25	70	50367,33	**-6,34%**	55287,78	2,90%	**6,07e-09**
	80	57185,91	**-8,30%**	64001,78	2,95%	**1,49e-08**
	90	89531,99	**-6,13%**	97693,00	2,46%	**2,97e-10**
	100	104847,42	**-7,47%**	115365,32	1,83%	**1,67e-11**
30	70	32523,01	**-9,92%**	37238,64	3,25%	**2,28e-08**
	80	54106,65	**-7,92%**	60323,48	2,63%	**3,04e-10**
	90	65070,62	**-9,28%**	73460,36	2,47%	**4,00e-12**
	100	85111,06	**-8,14%**	95316,26	3,03%	**3,39e-09**

5. Conclusões e Trabalhos Futuros

Neste trabalho é proposto, para o problema de minimização dos atrasos e avanços em um ambiente UPMSP, um estudo do VND com seis estruturas de vizinhança como busca local da metaheurística GRASP. O problema de minimização dos atrasos e avanços apresenta duas etapas. Primeiramente obtém-se uma sequencia de tarefas para o conjunto de máquinas; posteriormente, para a avaliação da função objetivo, é necessário resolver um subproblema de otimização, o qual consiste na inserção dos *idle times* e consequente definição dos instantes iniciais de tempo de preparação/processamento de cada tarefa. Para a inserção dos *idle times*, foi adaptado um método exato proposto na literatura para sequenciamento em máquina única.

Na análise das estruturas de vizinhança pelo método de Taguchi, observou-se que todas estruturas de vizinhança contribuem no processo de busca, já que a análise estatística não indicou a utilização do primeiro nível para nenhuma estrutura. Este nível indicava a não utilização da estrutura. Além disto, das quatro estruturas avaliadas, três apresentam melhor desempenho com a utilização da estratégia BI e apenas uma pela utilização do FI. Ao comparar o GRASP+VND proposto com um método GRASP integrado ao ILS e *Path Reliking* proposto na literatura, observa-se que a abordagem proposta supera esta em todos os cenários avaliados, considerando o mesmo critério de parada e um hardware com menor poder de processamento.

Apesar da performance observada pelo método, é necessário realizar novos estudos para confirmar o desempenho do GRASP+VND. As instâncias propostas por Nogueira et al. (2014) apresentam apenas uma variação nos parâmetros R e τ que definem as *due dates*. Em trabalhos propostos na literatura para minimização ponderada dos atrasos, estes parâmetros apresentam variações

que tendem a comprometer o desempenho de algumas abordagens propostas. Além disto, apesar de poucos trabalhos terem sido propostos para minimização dos atrasos e avanços ponderados para ambientes UPMSP, existem trabalhos propostos para ambientes e objetivos similares que podem ser adaptados a este, afim de verificar o real desempenho do GRASP+VND.

6. Agradecimentos

Os autores agradecem ao CEFET/MG, CAPES, CNPq e FAPEMIG pelo apoio ao desenvolvimento nesta pesquisa e ao Prof. Dr. José Elias C. Arroyo. pela disponibilização do conjunto de instâncias.

Referências

Chen, J. F. (2009). Scheduling on unrelated parallel machines with sequence- and machine-dependent setup times and due-date constraints. *The International Journal of Advanced Manufacturing Technology*, 44(11):1204 – 1212.

Feo, T. A. e Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 9:849–859.

Liao, C. J., Lee, C. H., e Tsai, H. T. (2016). Scheduling with multi-attribute set-up times on unrelated parallel machines. *International Journal of Production Research*, 54(16):4839 – 4853.

Mladenovic, N. e Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100.

Nogueira, J. P. C. M., Arroyo, J. E. C., Villadiego, H. M. M., e Gonçalves, L. B. (2014). Hybrid grasp heuristics to solve an unrelated parallel machine scheduling problem with earliness and tardiness penalties. *Electronic Notes in Theoretical Computer Science*, 302:53 – 72.

Randall, P. A. e Kurz, M. E. (2007). Effectiveness of adaptive crossover procedures for a genetic algorithm to schedule unrelated parallel machines with setups. *International Journal of Operations Research*, 4:1–10.

Rosa, B. F., Souza, M. J. F., de Souza, S. R., de F. Filho, M. F., Ales, Z., e Michelon, P. Y. P. (2017). Algorithms for job scheduling problems with distinct time windows and general earliness/tardiness penalties. *Computers and Operations Research*, 81(Supplement C):203 – 215.

Ruiz, R. e Andrés-Romano, C. (2011). Scheduling unrelated parallel machines with resource-assignable sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, 57(5):777 – 794.

Sen, H. e Bülbül, K. (2015). A strong preemptive relaxation for weighted tardiness and earliness/tardiness problems on unrelated parallel machines. *INFORMS Journal on Computing*, 27: 135–150.

Taguchi, G. (1987). *System of Experimental Design: Engineering Methods to Optimize Quality and Minimize Cost*. UNIPUB/Kraus International. White Plains.

Zhu, Z. e Heady, R. B. (2000). Minimizing the sum of earliness/tardiness in multi-machine scheduling: a mixed integer programming approach. *Computers and Industrial Engineering*, 38(2): 297 – 305.