UMA VIZINHANÇA DE GRANDE PORTE PARA O PROBLEMA DE PROGAMAÇÃO DE TAREFAS EM MÁQUINAS PARALELAS NÃO RELACIONADAS COM TEMPO DE PREPARAÇÃO DEPENDENTE DA SEQUÊNCIA

Fernando Stefanello[1]
stefanello@inf.ufsm.br

Olinto C. B. Araújo[2]
olinto@smail.ufsm.br

Felipe Martins Müller[1]
felipe@inf.ufsm.br

Vinícius Jacques Garcia[3]
viniciusgarcia@unipampa.edu.br

[1]Programa de Pós-Graduação em Informática – Universidade Federal de Santa Maria – Centro de Tecnologia Av. Roraima nº. 1000 – Cidade Universitária Bairro Camobi – Santa Maria – RS – 97105-900.

[2]Colégio Técnico Industrial de Santa Maria – Universidade Federal de Santa Maria.

[3]Universidade Federal do Pampa – Campus Alegrete – Av. Tiarajú 810 – Alegrete – RS – 97546-550.

RESUMO

Este trabalho apresenta uma vizinhança de grande porte baseada em programação inteira mista para o problema de programação de tarefas em máquinas paralelas não relacionadas com tempo de preparação dependente da sequência. A vizinhança é explorada por um otimizador de programação inteira mista genérico. Testes computacionais são realizados utilizando a vizinhança proposta em uma busca local para instâncias de grande porte e como parte de um algoritmo de múltiplos inícios para instâncias de pequeno porte. A qualidade do conjunto de soluções obtido é avaliada e comparada com os resultados de um algoritmo genético recentemente reportado na literatura. Os experimentos computacionais demonstram que a metodologia apresentada é robusta e eficiente.

PALAVRAS-CHAVE: Máquinas paralelas não relacionadas. Vizinhança de grande porte. Tempos de *setup*. Otimização combinatória.

ABSTRACT

This paper presents a large scale neighborhood based on mixed integer programming (MIP) to the unrelated parallel machine scheduling problem with sequence dependent setup times. A MIP solver is used to explore the neighborhood for improved solutions. Computational tests are conducted using the proposed neighborhood within a local search for large instances size and as part of a multistart algorithm for small instances size. The quality of the set of solutions obtained is evaluated and compared with results from a genetic algorithm recently reported in the literature. Computational experiments show that this methodology is robust and efficient.

KEYWORDS: Unrelated parallel machines. Very large-scale neighborhood. Setup times. Combinatorial optimization.

1. Introdução

O problema de programação de tarefas em máquinas não relacionadas com tempo de preparação dependente da sequência considera um conjunto de *n* tarefas que devem ser, uma a uma, processadas sem interrupção por exatamente uma única máquina dentre *m* disponíveis. No caso em que as máquinas são não relacionadas, o tempo de processamento de uma tarefa depende da máquina na qual a mesma é processada. Os tempos de preparação de máquina (*setup time*) para este problema são dependentes da máquina e da sequência, isto é, o *setup time* computado na máquina *i* entre as tarefas *j* e *k* é diferente do *setup time* computado na mesma máquina entre as tarefas *k* e *j*. Ainda, o *setup time* entre as tarefas *j* e *k* na máquina *i* é diferente do *setup time* entre as tarefas *j* e *k* na máquina *i'*. O objetivo é encontrar uma distribuição das tarefas às máquinas de tal modo que o tempo de processamento total da máquina com maior carga entre todas (*makespan*) seja o menor possível. Esse problema é denotado por $R|S_{ijk}|C_{max}$ em Pinedo (2008), seguindo a classificação de três campos introduzida por Graham et al. (1979).

Máquinas não uniformes representam mais adequadamente problemas encontrados na prática e são generalizações dos problemas de programação em máquinas idênticas e relacionadas. Tempos de preparação de máquinas durante a produção envolve trabalho e tempo relativo à limpeza de ferramentas e do ambiente. O *setup time* para essas atividades são frequentemente negligenciadas e, consequentemente, ignoradas. No entanto, em alguns casos, devido ao impacto na solução final, devem ser calculados separadamente do tempo de processamento das tarefas. Allahverdi et al. (1999) apresentam o estado da arte de sequenciamento de tarefas em máquinas com *setup time*. Em Yu et al. (2002), os autores afirmam que programação de tarefas em máquinas não relacionadas é um dos problema mais difícil, com respeito à complexidade, considerando programação de tarefas em máquinas idênticas, relanionadas e não-relacionadas.

Existem vários trabalhos na literatura que abordam o problema de programação de tarefas em máquinas não relacionadas, como Franca et al. (1996), Weng et al. (2001), Kim et al. (2002), Chen (2005), Low (2005), Chen (2006), Chen and Wu (2006), Rabadi et al. (2006), Rocha de Paula et al. (2007), Logendran et al. (2007) e Armentano e Felizardo (2007), para citar alguns. Estes estudos consideram a otimização de um ou mais critérios de avaliação, como minimização do adiantamento e/ou atraso e *makespan*. Especificamente para o problema em questão, Vallada e Ruiz (2009) apresentam um algoritmo genético e propõem um conjunto de instâncias com características diversas.

Neste trabalho é proposta uma vizinhança de grande porte baseada em programação inteira mista (PIM) que é explorada por otimizador de programação matemática genérico. A intensidade da busca na vizinhança pode variar alterando o valor dos parâmetros de tempo e tamanho associados aos respectivos problemas de PIM. Resultados comparativos são apresentados com relação ao algoritmo genético proposto em Vallada e Ruiz (2009).

O uso de busca local com um otimizador PIM para explorar a vizinhança tem recebido considerável atenção nos últimos anos. Esta direção de pesquisa procura formas de combinar ideias e métodos oriundos de abordagens exatas e de abordagens baseadas em meta-heurísticas. Puchinger e Raidl (2005) descrevem algumas combinações existentes, e diferenciam as duas principais categorias:

- Combinação colaborativa: quando os algoritmos trocam informações, mas não fazem parte um do outro. Algoritmos exatos e heurísticos podem ser executados sequencialmente, entremeados ou em paralelo.
- Combinação integrativa: quando uma técnica é um componente subordinado de outra técnica. Assim, existe um algoritmo mestre distinto, que pode ser um algoritmo exato ou meta-heurístico e, ao menos, um escravo integrado.

Uma descrição de vários trabalhos relacionados com estas duas categorias pode ser encontrada em Lourenço e Fernandes (2007). A proposta deste trabalho recai na segunda categoria.

A apresentação do trabalho está organizada na forma que segue. Na seção seguinte é apresentado o modelo matemático proposto por Vallada e Ruiz (2009) para o problema de programação de tarefas em máquinas não relacionadas com tempo de preparação dependente da sequência. Na seção 3 é estudada a vizinhança de grande porte baseada em PIM e, na sequência, é definida uma busca local e um algoritmo de múltiplos inícios. Na seção 4 os resultados computacionais são discutidos. Por fim, na seção 5, conclusões e algumas direções para trabalhos futuros são apresentadas.

2. Modelo matemático

Nesta seção é apresentado um modelo de programação inteira mista para o problema de programação de tarefas em máquinas não relacionadas com tempo dependente da sequência de acordo com Vallada e Ruiz (2009).

Para simplificar a exposição do modelo, faz-se necessário definir a notação utilizada.

Parâmetros

n : número de tarefas.
m : número de máquinas.
M : $\{1,...,m\}$, representa o conjunto das máquinas.
N : $\{1,...,n\}$, representa o conjunto das tarefas.
V : um número suficientemente grande.
p_{ij} : tempo de processamento da tarefa $j, j \in N$, na máquina i, $i \in M$.
s_{ijk} : tempo de preparação (*setup time*) da máquina i, $i \in M$, para execução da tarefa k após o processamento da tarefa j, $k, j \in N$.

Variáveis

$$x_{ijk} = \begin{cases} 1 & \text{se a tarefa } j \text{ precede a tarefa } k \text{ na máquina } i \\ 0 & \text{caso contrário} \end{cases}$$

$C_{ij} \geq 0$: Tempo de execução das tarefas j na máquina i, $i,j \in N$.
$C_{max} \geq 0$: Tempo máximo de execução.

$$\text{Min } C_{\max} \quad (1)$$

Sujeito a:

$$\sum_{i \in M} \sum_{\substack{j \in \{0\} \cup \{N\} \\ j \neq k}} x_{ijk} = 1 \qquad \forall k \in N \quad (2)$$

$$\sum_{i \in M} \sum_{\substack{k \in N \\ j \neq k}} x_{ijk} \leq 1 \qquad \forall j \in N \quad (3)$$

$$\sum_{k \in N} x_{i0k} \leq 1 \qquad \forall i \in M \quad (4)$$

$$\sum_{\substack{h \in \{0\} \cup \{N\} \\ h \neq k, h \neq j}} x_{ihj} \geq x_{ijk} \qquad \forall j,k \in N, j \neq k, \forall i \in M \quad (5)$$

$$C_{ik} + V(1 - x_{ijk}) \geq C_{ij} + s_{ijk} + p_{ik} \qquad \forall j \in \{0\} \cup \{N\}, \forall k \in N, j \neq k, \forall i \in M \quad (6)$$

$$C_{i0} = 0 \qquad \forall i \in M \quad (7)$$

$$C_{ij} \geq 0 \qquad \forall j \in N, \forall i \in M \quad (8)$$

$$C_{\max} \geq C_{ij} \qquad \forall j \in N, \forall i \in M \quad (9)$$

$$X_{ijk} \in \{0,1\} \qquad \forall j \in \{0\} \cup \{N\}, \forall k \in N, j \neq k, \forall i \in M \quad (10)$$

O objetivo é minimizar o tempo máximo de execução ou *makespan* (C_{max}). O conjunto de restrições (2) garante que cada tarefa seja atribuída a uma única máquina e tenha exatamente um antecessor. As restrições (3) determinam que cada tarefa só pode ter um sucessor. É importante observar que uma tarefa artificial 0 identifica o início do sequenciamento das tarefas nas máquinas, e o conjunto de restrições (4) limita o número de sucessores da tarefa artificial em, no máximo, um para cada máquina. O conjunto (5) garante que as tarefas são ligadas corretamente nas maquinas. As restrições (6) determinam o tempo final do processamento das tarefas de cada máquina. Os conjuntos de restrições (7) e (8) definem o tempo de conclusão como 0 para as tarefas artificiais e valores não-negativos para as demais tarefas, respectivamente. As restrições (9) definem o tempo máximo de execução ou *makespan*. Finalmente, o conjunto de restrições (10) define as variáveis binárias.

3. Vizinhança de grande porte

As técnicas de hibridização de métodos exatos e heurísticos têm se mostrado cada vez mais aplicáveis a problemas de otimização combinatória, em especial diante da evolução dos computadores e dos otimizadores de problemas de programação inteira mista. Em Stefanello e Müller (2009), os autores testaram reduções heurísticas para um modelo matemático referente ao problema de p-medianas capacitado com bons resultados.

Esta seção descreve uma vizinhança de grande porte baseada em programação inteira mista para definir um conjunto de sequências de movimentos atômicos denominados inserção e expulsão. O propósito reside em utilizar um modelo matemático para identificar conjuntos destas sequências de movimentos e obter uma melhora no valor da função objetivo.

No movimento de inserção, indexado por uma variável de decisão y_{ij}, a tarefa j é inserida como sucessora da tarefa i. Já no movimento de expulsão, representado pela variável de decisão x_{ij}, a tarefa j deve tomar o lugar da tarefa i, e esta última pertencerá a outro movimento que poderá ser uma inserção ou uma nova expulsão.

A vizinhança é definida por dois tipos de sequência de movimentos denominados cadeia e ciclo, representados na Figura 1 a partir de uma solução do problema em estudo com duas máquinas, ver (a). Um movimento expulsão é ilustrado em (b) e uma sequência ciclo, definida apenas por movimentos deste tipo, pode ser observada envolvendo as primeiras tarefas de cada máquina em (d). Em uma sequência de movimentos em cadeia, após a expulsão da primeira tarefa, esta pode ser tanto inserida como sucessora de outra tarefa (c) quanto expulsar outra tarefa e assim sucessivamente até que a sequência finalize com uma inserção, como pode ser observado em (d). É importante entender que a vizinhança considera uma solução estática, e, para que o custo de cada movimento atômico seja válido, algumas tarefas não podem ser movimentadas, aquelas assinaladas com um "x" em (d). Estas tarefas são sucessoras ou antecessoras das tarefas que estão sendo movimentadas.

Um grande número de sequências independentes de movimentos pode ser definido, incluindo movimentos entre tarefas de uma mesma máquina e entre máquinas diferentes.

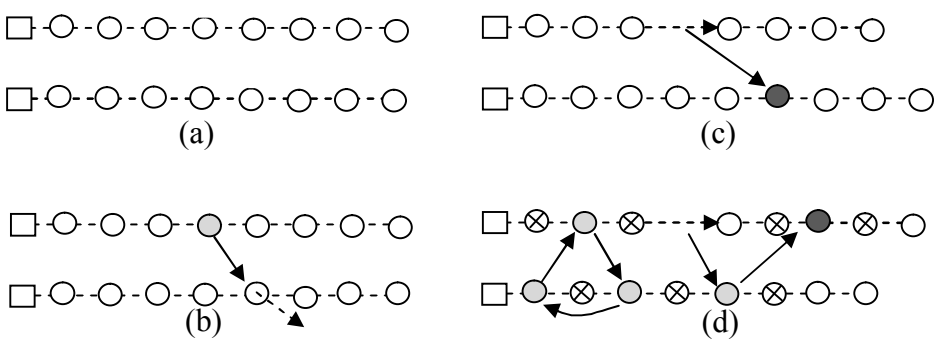

Figura 1 – Ilustração de movimentos de troca, inserção, ciclo e cadeia

Formalmente, o modelo matemático associado à vizinhança é definido a seguir:

Parâmetros:

N^* : $\{1,...,n,n+1,...,n+m\}$, representa o conjunto das tarefas a serem executadas, bem como as tarefas artificiais que representam cada máquina.

D : $N^* \setminus N = \{n+1,...,n+m\}$, representa o conjunto de tarefas artificiais.

D_m : Representa a tarefa artificial da máquina m, $m \in M$.

U_m : Conjunto das tarefas da solução inicial designadas à máquina m, $m \in M$.

cO_i : Custo da retirada da tarefa i de sua posição.

cX_{ij} : Custo da inserção da tarefa j na posição i.

cY_{ij} : Custo da inserção da tarefa j como sucessora da tarefa i.

cW_i : Custo para definir o sucessor da tarefa i como sucessor da tarefa que precede i.

CM_i : Custo total da máquina i da solução atual.

$suc(i)$: Representa a tarefa sucessora da tarefa i no sequenciamento de uma das máquinas.

Variáveis:

$$x_{ij} = \begin{cases} 1 & \text{se a tarefa } j \text{ é inserida na posição } i, \text{ e a tarefa } i \text{ é expulsa de sua posição.} \\ 0 & \text{caso contrário} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{se a tarefa } j \text{ é inserida como sucessora } i \\ 0 & \text{caso contrário} \end{cases}$$

$$w_i = \begin{cases} 1 & \text{o sucessor da tarefa } j \text{ passa a suceder o predecessor da tarefa } j. \\ 0 & \text{caso contrário} \end{cases}$$

$C_{max} \geq 0$: Tempo máximo de execução da máquina mais carregada (*makespan*).

Os parâmetros que representam os custos de retirada e inserção de tarefas são definidos a partir da solução atual, onde *pre(i)* representa a tarefa que precede a tarefa i, e $Q_{i,j} = p_{ij} + s_{kij}$, representa o custo total de alocação da tarefa j como sucessora da tarefa i na maquina k. Assim, $cO_j = -Q_{pre(j),j} - Q_{j,suc(j)}$ é o custo da retirada da tarefa j de sua posição na solução atual. Esta situação ocorre tanto com as variáveis x_{ij} onde a tarefa i é expulsa, quanto nas variáveis y_{ij} onde a tarefa j é retirada de sua posição. Já o custo denotado por $cX_{ij} = Q_{pre(i),j} + Q_{j,suc(i)}$, é aplicável apenas nas variáveis x_{ij}, pois representa o custo de inserção da tarefa j na posição da tarefa i. O custo da inserção de uma tarefa j como sucessora de uma tarefa i é dado por $cY_{ij} = Q_{i,j} + Q_{j,suc(i)}$, e é associado exclusivamente as variáveis y_{ij}. Por fim, o custo $cW_i = Q_{pre(i),suc(i)}$ é o custo de associar a tarefa antecessora e a sucessora de uma tarefa i. Este custo é associado às variáveis w_i.

As variáveis denotadas por w auxiliam na definição de uma sequência de movimentos em cadeia, na qual a primeira tarefa expulsa é indexada pela variável w_i. Além disso, esta variável indica que o arco entre a tarefa predecessora e sucessora da tarefa i deve ser fechado, ou seja, a tarefa sucessora da tarefa i deve passar a ser sucessora da tarefa que precede a tarefa i.

$$\text{Min } C_{max} \qquad (1)$$

Sujeito a:

$$\sum_{j\in N} x_{ij} + \sum_{j\in N} y_{ij} + w_i \le 1 \qquad \forall i \in N \qquad (2)$$

$$\sum_{j\in N} y_{ij} \le 1 \qquad \forall i \in D \qquad (3)$$

$$\sum_{i\in N} x_{ij} + \sum_{i\in N^*} y_{ij} \le 1 \qquad \forall j \in N \qquad (4)$$

$$\sum_{\substack{j\in N \\ i\notin D}} x_{ij} + \sum_{j\in N} y_{ij} + \sum_{j\in N} x_{j,suc(i)} + \sum_{j\in N^*} y_{j,suc(i)} \le 1 \qquad \forall i \in N^* \qquad (5)$$

$$\sum_{i\in N} x_{ij} + \sum_{i\in N^*} y_{ij} + \sum_{i\in N} x_{i,suc(j)} + \sum_{i\in N^*} y_{i,suc(j)} \le 1 \qquad \forall j \in N \qquad (6)$$

$$\sum_{j\in N} x_{ij} \le \sum_{j\in N} x_{ji} + \sum_{j\in N^*} y_{ji} \qquad \forall i \in N \qquad (7)$$

$$\sum_{j\in N^*} y_{ji} + \sum_{j\in N} x_{ji} \le \sum_{j\in N} x_{ij} + w_i \qquad \forall i \in N \qquad (8)$$

$$\sum_{j\in N} y_{ij} + \sum_{j\in N} x_{ji} + \sum_{j\in N^*} y_{ji} \le 1 \qquad \forall i \in N \qquad (9)$$

$$\sum_{\substack{i\in N \\ j\neq D_m}} cO_j x_{ij} + \sum_{\substack{i\in N \\ j\neq D_m}} cX_{ji} x_{ji} + \sum_{\substack{i\in N^* \\ j\neq D_m}} cO_j y_{ij} + \\ + \sum_{i\in N} cY_{ji} y_{ji} + \sum_{i\in U_m} cW_i w_i - CM_m \le C_{\max} \qquad \forall j \in U_m \cup D_m,\ \forall m \in M \qquad (10)$$

$$\sum_{j\in N} x_{ji} + \sum_{j\in N} y_{ji} - w_i \ge 0 \qquad \forall i \in N \qquad (11)$$

$$C_{\max} \ge 0,\ x_{ij}, y_{kj}, w_i \in \{0,1\} \qquad \forall i, j \in N, \forall k \in N^* \qquad (12)$$

O objetivo é obter movimentos que minimizem o tempo máximo de execução ou *makespan*. As restrições (2), (3) e (4) obrigam que uma tarefa participe apenas de um movimento. Os conjuntos de restrições (5) e (6) definem que se uma tarefa for movimentada, as tarefas sucessoras e antecessoras devem se manter fixas. Esta restrição é necessária para que os custos de inserção e expulsão de uma tarefa sejam calculados corretamente, visto que os mesmos são dados em função das tarefas que precedem e sucedem a tarefa. Em (7) e (8) são definidas as restrições que forçam uma tarefa que compõem um movimento ser alocada em outra posição. Especificamente sobre as restrições (8), se uma tarefa for retirada da sua posição original uma outra tarefa deve ocupar essa posição ou o antecessor e o sucessor devem ser conectados na sequência. Este fato é representado pelas variáveis denotadas por x e w no lado direito da desigualdade. As restrições (9) implicam que, se uma tarefa for inserida como sucessora de outra, a tarefa sucessora da tarefa inserida deve permanecer fixa, novamente, assim como nas restriões (5) e (6), isto é necessário para que o custo de um movimento atômico seja avaliado corretamente. De outra forma, também implica que se uma tarefa for movimentada, nenhuma tarefa pode ser inserida como sua predecessora. As restrições (10) são responsáveis por avaliar os custos dos movimentos. As restrições (11), a rigor, podem ser suprimidas, uma vez que tem como função evitar que variáveis w assumam valor 1 sem que a correpondente tarefa faça parte de algum movimento. De fato, caso sejam suprimidas o único inconveniente que decorre é a necessidade de analisar quais variáveis w com valor 1 não compõe a solução do modelo. Por último, no conjunto de restrições (12), tem-se a restrição que impõe as condições para o domínio das variáveis do problema.

4. Busca local e Algoritmo de Múltiplos Inícios

A vizinhança proposta foi originalmente projetada para compor um procedimento de intensificação ou pós-otimização de um método meta-heurístico, visto que, principalmente em instâncias de grande porte, a resolução do modelo matemático associado pode requerer um tempo maior de processamento. No entanto, dada a fase em que se encontra a pesquisa, o objetivo deste trabalho é avaliar a capacidade de obtenção de boas soluções sem uma excessiva preocupação com o tempo computacional. Desde modo, esta seção descreve uma busca local e um algoritmo de múltiplos inícios que utilizam a vizinhança de grande porte descrita na seção 3.

Tanto para a busca local quanto para o algoritmo de múltiplos inícios é necessário gerar uma solução inicial. Para a obtenção dessa solução inicial foram implementadas duas heurísticas construtivas. A primeira heurística gera uma solução denominada solução inicial trivial, e consiste em tomar as tarefas na ordem dos índices atribuídos a cada uma e inseri-las sequencialmente nas máquinas, de forma que a tarefa com índice k seja designada para a máquina com índice (k mod m).

Embora este método gere soluções de baixa qualidade, ele permite demonstrar a robustez que o uso da vizinhança de grande porte propicia. Ou seja, é possível encontrar soluções de boa qualidade mesmo partindo de soluções triviais.

No segundo método, as tarefas são designadas para as máquinas nas quais produzem menor acréscimo de carga, seguindo uma ordenação pré-estabalecida. Para a definição da ordem em que as tarefas são alocadas às máquinas é avaliada a soma de todos os custos de designação de cada tarefa para qualquer máquina, sendo esta a primeira tarefa ou sucessora de outra tarefa. A ordem que as tarefas são alocadas é a ordem decrescente de classificação. Deste modo, quando o aumento de *makespan* é inevitável, a tarefa é inserida na máquina que fornecer o menor acréscimo de *makespan*. A solução gerada por esse método foi denominada solução inicial gulosa. A ideia subjacente consiste em favorecer que as últimas tarefas a serem designadas tenham custos menores, de forma a aumentar as chances de inserção de uma tarefa em uma máquina sem que ocorra incremento no valor do *makespan*.

Dois parâmetros são definidos para a busca local: o primeiro é o tempo máximo (*tmax*) que o otimizador permanece na busca dos movimentos; o segundo é o tempo mínimo (*tmin*).

O tempo máximo é necessário principalmente para as instâncias de grande porte, uma vez que o otimizador pode ter dificuldade de provar a otimalidade num tempo viável. Este procedimento é adotado para manter o tempo computacional dentro de um limite aceitável, no entanto, caso seja necessário tentar soluções de melhor qualidade, nada impede que mais tempo seja utilizado. Partindo da solução inicial, a vizinhança é iterada até que seja provada a otimalidade da vizinhança sem melhora de função objetivo ou o tempo máximo seja excedido.

O segundo parâmetro, *tmin*, determina o tempo mínimo que o otimizador deve ficar buscando por melhores soluções. Após este tempo, caso o otimizador tenha conseguido encontrar movimentos que melhorem a solução inicial, a busca é interrompida, os movimentos são executados de forma a obter a nova solução, partindo para uma nova iteração da busca local. Existe a possibilidade de interromper a busca logo que a primeira solução é encontrada, no entanto, verificou-se que um pequeno tempo adicional em geral possibilita a obtenção de melhoras significativas que justificam sua utilização.

Para instâncias pequenas, a busca local nem sempre encontra a solução ótima global, motivo pelo qual foi implementado um algoritmo de múltiplos inícios. Alia-se a este fato o tempo computacional relativamente baixo da exploração da vizinhança neste caso. As soluções iniciais fornecidas ao algoritmo são, além da solução inicial gulosa, uma variação de $k*n$ soluções iniciais gulosas em que a escolha da tarefa a ser inserida é aleatória. Também são utilizadas $k*n$ soluções iniciais variantes do método da solução inicial trivial, para as quais as tarefas também são escolhidas aleatoriamente. Define-se que k é um parâmetro que determina o número de iterações do algoritmo de múltiplos inícios.

5. Resultados computacionais

Para os testes computacionais, foi usado um computador com processador Intel Quad-Core Xeon X3360 2.83 GHz. Também foi utilizado o otimizador CPLEX 12.1[1]. O conjunto de instâncias utilizadas nos testes foram propostas em Vallada e Ruiz (2009) e estão disponíveis em http://soa.iti.es. O conjunto de instâncias é dividido em dois grupos. O primeiro grupo contém 640 instâncias consideradas de pequeno porte, enquanto o segundo grupo contém 1000 instâncias consideradas de grande porte. Cada um dos conjuntos é subdividido em grupos de 10 instâncias com diferentes quantidades de máquinas e tarefas, além de diferentes intervalos para os tempos de *setup*.

Para os testes, foi adotado $tmax = (n\text{-}m/2)/2$, que demonstrou uma boa relação entre qualidade de solução e tempo computacional, e $tmim = 0{,}10 * tmax$.

Para avaliação da qualidade das soluções obtidas foi definido o desvio médio relativo (*DMR*), de acordo com a fórmula $DMR = (S_h - S_g)/\min(S_h,S_g)*100$, em que S_h corresponde à solução obtida através da busca local ou algoritmo de múltiplos inícios e S_g corresponde à melhor solução do algoritmo genético de Vallada e Ruiz (2009).

Os primeiros testes consideram a busca local aplicada à resolução do conjunto de instâncias de grande porte. A Tabela 1 apresenta os resultados médios obtidos a cada conjunto de 40 instâncias, em que a primeira e a segunda coluna correspondem, respectivamente, ao número de tarefas e máquinas do grupo de instâncias. A coluna *DMR* traz o desvio médio relativo referente ao conjunto de cada configuração. A coluna MelhorSol traz o percentual de instâncias do conjunto para o qual a busca local conseguiu igualar ou melhorar a solução final, e a coluna Tempo mostra o tempo computacional médio em segundos para a obtenção dos resultados das colunas anteriores.

No algoritmo genético proposto em Vallada e Ruiz (2009), o critério de parada é o tempo computacional definido por $n*(m/2)*t$, com um valor máximo para t=50 milissegundos. Desta forma, o tempo de CPU varia de 12,5 a 187,5 segundos em um computador com processador Intel Core 2 Duo, 2.4 Ghz com 2 GB de memória RAM.

Os resultados mostram que a busca local apresenta bom desempenho, pois conseguiu melhorar 85,90% das melhores soluções conhecidas até então, obtendo um DMR de -5,72% para todos os conjuntos de instâncias com os parâmetros citados anteriormente, e inclusive com reduções de até -33,33% para uma determinada instância.

Durante os testes como também na avaliação dos resultados, foi possível observar que quando a relação n/m é menor a busca local tem desempenho melhor, tanto em tempo computacional como em valor de função objetivo. Isso pode ser visualizado a partir dos dados da Tabela 1 para as diferentes configurações de número de tarefas e máquinas, observando-se que a busca local apresentou os piores resultados para os conjuntos de instâncias 50x10 e 100x10.

Na maioria dos casos em que a busca local parte de uma solução de baixa ou média qualidade, o otimizador consegue encontrar movimentos que melhoram a função objetivo em um curto espaço de tempo, no entanto, há situações que isso não acontece. Em especial, isto ocorre na primeira iteração, na qual o otimizador pode extrapolar o tempo limite máximo sem conseguir melhorar a solução inicial. Nos testes realizados com os parâmetros anteriores, houve 13 destas situações, sendo uma no conjunto 200x25, três no conjunto 250x20, quatro no conjunto 250x25 e cinco no conjunto 250x30. Isso causou um impacto nos valores médios do *DMR*, pois, retirando esses casos, o *DMR* do conjunto restante de instâncias fica reduzido a -7,20%.

Nestas instâncias, o fato de não conseguir melhorar a solução inicial num tempo inferior ao parâmetro de tempo máximo, não implica que a vizinhança não consiga melhorar a solução ou que a solução inicial seja um ótimo local, pois, em testes adicionais, permitindo um

[1] http://www-01.ibm.com/software/integration/optimization/cplex/

tempo extra, em todos os casos foi possível melhorar a solução inicial e inclusive melhorar a melhor solução conhecida, o que reduz o *DMR* para -7,30%.

Tabela 1: Resultados computacionais médios para as instancias de grande porte:

Tarefas	Máquinas	*DMR* (%)	Melhor Sol (%)	Tempo(s)
50	10	6,07	15,00	35,45
50	15	-0,97	60,00	18,52
50	20	-3,03	77,50	7,54
50	25	-5,32	95,00	3,44
50	30	-7,20	92,50	2,13
100	10	3,00	37,50	98,45
100	15	-1,11	70,00	97,07
100	20	-4,64	87,50	93,40
100	25	-10,50	97,50	84,23
100	30	-14,68	100,00	70,86
150	10	-1,82	77,50	177,43
150	15	-3,34	87,50	164,00
150	20	-6,66	95,00	158,52
150	25	-12,71	100,00	152,47
150	30	-13,76	100,00	137,99
200	10	-3,84	97,50	277,26
200	15	-5,57	92,50	270,23
200	20	-9,58	100,00	246,30
200	25	-8,92	97,50	220,77
200	30	-15,47	100,00	208,27
250	10	-5,47	97,50	386,37
250	15	-8,24	100,00	369,51
250	20	-2,11	92,50	379,14
250	25	-3,54	90,00	336,23
250	30	-3,65	87,50	314,58
Média		-5,72	85,90	172,41

Em outro experimento realizado, foi verificado, como esperado, que quando a busca local parte da solução inicial trivial o otimizador despende mais tempo para encontrar as sequências de movimentos dentro dos parâmetros estabelecidos. No entanto, considerando somente as instâncias para as quais as sequências de movimentos são encontradas dentro do tempo máximo permitido, a média total do *DMR* se mantém a 0,61% dos resultados obtidos com a busca local partindo da solução gulosa. Isto demonstra que o método é robusto. Outro resultado digno de nota é o fato da busca local a partir da solução trivial conseguir soluções de melhor qualidade para 356 instâncias do que quando executada a partir da solução gulosa.

Até o presente momento da pesquisa, do conjunto de 1000 instâncias de larga escala, foi possível obter soluções com valor de função objetivo inferior ou igual a obtida pelo algoritmo genético em 924 instâncias, sendo 871 estritamente melhores. As soluções estão disponíveis para consulta no endereço http://www-usr.inf.ufsm.br/~stefanello/research.

Para as instâncias de pequeno porte foi realizada uma bateria de testes para diferentes valores do parâmetro k descrito na seção 4, conforme ilustrado na Tabela 2. Entende-se que para k =0 tem-se a busca local partindo da solução inicial gulosa.

Na Tabela 2, a primeira coluna mostra o valor do parâmetro k. Os valores da coluna Tarefas representam um conjunto de 160 instâncias que possuem o mesmo número de tarefas e diferentes quantidades de maquinas, cujo *DMR* médio para este conjunto, é mostrado na terceira coluna. A coluna seguinte apresenta o percentual de instâncias de cada grupo para as quais o

algoritmo de múltiplos inícios conseguiu encontrar a melhor solução conhecida e, por fim, a coluna Tempo(s) traz o tempo computacional médio em segundos.

A partir dos dados da Tabela 2, é possível observar que a utilização da busca local para as instâncias pequenas, na média, possibilitou obter boas soluções. Além disso, em 54,69% das instâncias foi possível obter o valor das melhores soluções conhecidas.

Fazendo uso do algoritmo de múltiplos inícios é possível perceber que com um pequeno acréscimo de esforço e tempo computacional foi possível obter uma boa melhora nos valores do *DMR*, e do percentual das melhores soluções conhecidas. Além disso, com a utilização do fator de multiplicação igual a dois, foi possível encontrar as melhores soluções conhecidas para todas as instâncias analisadas.

Tabela 2: Resultados computacionais médios para instancias de pequeno porte:

Fator	Tarefas	*DMR*	Melhor Sol (%).	Tempo(s)
0	6	2,10%	79,38	0,016
	8	5,22%	57,50	0,048
	10	3,81%	47,50	0,088
	12	4,01%	34,38	0,187
Média		3,79%	54,69	0,085
0,5	6	0,09%	99,38	0,131
	8	0,12%	98,13	0,435
	10	0,16%	93,13	0,997
	12	0,22%	88,75	2,413
Média		0,15%	94,84	0,994
1	6	0,02%	99,38	0,223
	8	0,00%	100,00	0,772
	10	0,03%	98,13	1,833
	12	0,03%	96,88	4,553
Média		0,02%	98,59	1,845
2	6	0,00%	100,00	0,405
	8	0,00%	100,00	1,428
	10	0,00%	100,00	3,468
	12	-0,04%	100,00	8,908
Média		-0,01%	100,00	3,552

Outro fato a ser considerado é que na utilização do algoritmo de múltiplos inícios com $k \geq 0,5$ foi possível observar a melhora no valor da função objetivo de uma instância reportado como ótimo global por Vallada e Ruiz (2009). Isto explica o *DMR* negativo para último conjunto de instâncias de pequeno porte.

Apesar de na maioria dos casos o tempo computacional total médio ser superior ao tempo utilizado pelo algoritmo genético, vale lembrar que o objetivo principal deste trabalho é avaliar a capacidade de exploração da vizinhança. Além disso, originalmente, a vizinhança foi proposta como um método de diversificação, ou pós-otimização, e aqui foi utilizada numa busca local e num algoritmo de múltiplos inícios, os quais requerem várias iterações.

Outra observação é que, principalmente nas instâncias de grande porte, as quais o otimizador tem dificuldade de provar a otimalidade do modelo matemático associado à vizinhança, invariavelmente, somente a última iteração da busca local é limitada pelo tempo máximo de execução (*tmax*). Isto implica que a solução final de cada instância de larga escala é encontrada a, aproximadamente, 60% do tempo total utilizado. Este comportamento sugere a possibilidade de testar formas adaptáveis de atribuição do valor para o parâmetro *tmax*.

6. Conclusão

Neste trabalho foi proposta uma vizinhança de grande porte baseada em programação inteira mista, que é explorada com um otimizador comercial. Os experimentos realizados demonstram que esta vizinhança é eficiente na obtenção de boas soluções para o problema de programação de tarefas em máquinas paralelas não relacionadas com tempo de preparação dependente da sequência. Em comparação com o algoritmo genético proposto em Vallada e Ruiz (2009), os resultados obtidos foram notadamente superiores para os conjuntos de instâncias consideradas.

Além disso, a metodologia proposta mostrou-se robusta, pois, na resolução de uma instância, o resultado final apresentou pouca dependência com relação à qualidade da solução inicial, ou seja, soluções com boa qualidade foram obtidas independendo se a solução inicial era bem estrutura ou de baixa qualidade.

Em trabalhos futuros, pretende-se observar o comportamento da vizinhança como um método de intensificação e pós-otimização para o problema aqui tratado, além de outros problemas similares encontrados na literatura. Também pretende-se avaliar o desempenho da vizinhança redução heurística do número de variáveis e ainda inclusão de planos de cortes no modelo matemático.

Agradecimento:
Os autores agradecem à CAPES pelo apoio financeiro recebido.

Referências

Allahverdi, A., Gupta, J. N. D., Aldowaisan, T. (1999). A review of scheduling involving setup considerations. Omega, v. 27, p. 219-239.

Armentano, V. e Felizardo, M. (2007). Minimizing total tardiness in parallel machine scheduling with setup times: An adaptive memory-based GRASP approach. *European Journal of Operational Research*, 183: 100-114.

Chen, J. (2005). Unrelated parallel machine scheduling with secondary resource constraints. International Journal of Advanced Manufacturing Technology, 26: 285-292.

Chen, J. (2006). Minimization of maximum tardiness on unrelated parallel machines with process restrictions and setups. *International Journal of Advanced Manufacturing Technology*, 29: 557-563.

Chen, J. e Wu, T. (2006). Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints. OMEGA, *The International Journal of Management Science*, 34: 81-89.

Fernandes S. e Lourenço, H. R. (2008). Optimised Search Heuristics: A Mapping of procedures and Combinatorial Optimisation Problems, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.2032.

Franca, P., Gendreau, M., Laporte, G. e Müller, F. (1996). A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *International Journal of Production Economics*, 43:79-89.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326.

Kim, D.W., Kim, K.H., Jang, W., Chen, F.F., (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, 18: 223–231.

Logendran, R., McDonell, B. e Smucker, B. (2007). Scheduling unrelated parallel machines with sequence-dependent setups. *Computers & Operations Research*, 34: 3420-3438.

Low, C. (2005). Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. *Computers & Operations Research*, 32: 2013-2025.

Pinedo, M. (2008). Scheduling: Theory, Algorithms and Systems. Prentice Hall, New Jersey, third edition.

Puchinge, J. e Raidl, G. R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In J. Mira and J. Alvarez, editors, Proceedings of the First International Work-Conference on the Interplay Between Natural and Artificial Computation, volume 3562 of LNCS, pages 41-53. Springer.

Rabadi, G., Moraga, R., e Salem, A. (2006). Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing*, 17: 85-97.

Rocha de Paula, M., Gómez-Ravetti, M., Robson-Mateus, G., e Pardalos, P. (2007). Solving parallel machines scheduling problems with sequence-dependent setup times using variable neighbourhood search. IMA, *Journal of Management Mathematics*, 18: 101-115.

Stefanello, F., Müller, F.M. (2009) Um Estudo Sobre Problemas de Agrupamento Capacitado. *XLI SBPO - Simpósio Brasileiro de Pesquisa Operacional* - Porto Seguro.

Vallada, E. e Ruiz, R. (2009). Genetic algorithms for the unrelated parallel machine scheduling problem with sequence dependent setup times. *DEIOAC*.

Yu, L., Shih, H. M., Pfund, M., Carlyle, W. M., Fowler J.W. (2002). Scheduling of unrelated parallel machines: an application to PWB manufacturing, *IIE Transactions*, Springer.

Weng, M. X., Lu, J., Ren, H. (2001). Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *International Journal of Production Economics*, 70: 215–226.