SEQUENCIAMENTO DE TAREFAS COM RESTRIÇÕES DE COMPATIBILIDADE EM MÁQUINAS PARALELAS COM TEMPOS DE PREPARAÇÃO DEPENDENTES DA SEQUÊNCIA

Edson Luiz França Senne
Universidade Estadual Paulista – UNESP, Faculdade de Engenharia de Guaratinguetá
Av. Dr. Ariberto Pereira da Cunha, 333, CEP 12516-410, Guaratinguetá, SP, Brasil
elfsenne@feg.unesp.br

Antônio Augusto Chaves
Universidade Federal de São Paulo – UNIFESP, Instituto de Ciência e Tecnologia
Rua Talin, 330, CEP 12231-280, São José dos Campos, SP, Brasil
antonio.chaves@unifesp.br

RESUMO

Neste trabalho considera-se o problema de programar um conjunto de máquinas paralelas com tempos de preparação dependentes da sequência de programação, para um conjunto de tarefas para as quais existem restrições de compatibilidade, de tal forma que o tempo de conclusão máximo (*makespan*) seja minimizado. Este problema é baseado em um problema real encontrado em uma indústria de autopeças, que produz peças estampadas em aço para caminhões, carros e tratores, com prensas de alta capacidade. Para este problema NP-difícil, apresenta-se uma formulação matemática e uma heurística para encontrar uma boa solução para o problema. O desempenho da heurística é testada em 20 casos, com diferentes tamanhos e um problema-teste real, com 194 tarefas e 3 máquinas.

PALAVARAS CHAVE. Sequenciamento de tarefas; Máquinas paralelas; Heurística.

Área principal: Metaheurísticas

ABSTRACT

In this paper we consider the problem of scheduling a set of independent jobs with sequence-dependent setup times and jobs compatibility constraints on a set of parallel machines such that maximum completion time (makespan) is minimized. This problem is based on a real problem found in an auto parts industry, which produces steel stamped parts for trucks, cars and tractors, using high capacity presses. For this NP-hard problem, we present a mathematical formulation and a heuristic to find a good solution to the problem. The performance of the heuristic is tested on 20 instances with different sizes and a real instance with 194 jobs and 3 machines.

KEYWORDS. Scheduling; Parallel machines; Heuristics.

Main area: Metaheuristics

1. Introdução

Este artigo considera um problema de sequenciamento em máquinas paralelas (do inglês, PMSP - *parallel machine scheduling problem*), em que a preparação das máquinas é necessária no início do processamento de cada tarefa e entre o processamento de tarefas diferentes. Além disso, existem algumas restrições de compatibilidade entre tarefas. O objetivo é produzir no prazo mais curto, isto é, minimizar o tempo de realização da última tarefa processada (*makespan*). O problema de agendamento de tarefas em máquinas paralelas, sem preempção, para minimizar o *makespan* é NP-difícil (Tahar *et al*., 2006).

O problema considerado neste trabalho é um problema real, como ocorre na maioria dos estudos sobre PMSP tendo em conta os tempos de preparação, a fim de minimizar o *makespan*. Nosso problema é baseado em um problema real encontrado em uma indústria de autopeças, que produz peças estampadas em aço para caminhões, carros e tratores, com prensas de alta capacidade. As prensas utilizadas fazem uso de um conjunto de ferramentas que precisa ser montado antes que a produção de uma peça comece. Devido a isto, o tempo de preparação não pode ser considerado como negligenciável. A preparação de uma prensa pode variar de acordo com a sequência de produção realizada, isto é, os tempos de preparação dependem tanto da peça que acabou de ser processada como da peça que deve ser processada em seguida. Além disso, como o conjunto de ferramentas é único para cada tipo de peça, quando uma peça está sendo processada em uma prensa, uma outra peça do mesmo tipo não pode ser processada em qualquer outra prensa.

O PMSP com tempos de preparação dependentes da sequência de produção tem sido investigado por pesquisadores para ambientes *job-shop* em diversos segmentos industriais, como metalúrgico, químico, semicondutor, vidro, papel, têxtil, madeira, aeroespacial, entre outros. Boas revisões sobre o problema podem ser vistas em Allahverdi *et al*. (1999), Allahverdi *et al*. (2008).

Muitos trabalhos de pesquisa têm sido realizados no desenvolvimento de soluções para o PMSP, levando-se em conta uma variedade de situações e objetivos. Meyr (2002) apresenta uma heurística que combina procedimentos de busca local com a metaheurística de *Simulated Annealing* com reotimização *dual* para resolver simultaneamente os problemas de tamanho de lote e de sequenciamento de vários produtos em máquinas heterogêneas com tempos de preparação dependentes da sequência de produção, com o objetivo de minimizar os custos de produção, os custos de estoque e os custos de preparação de máquinas.

Gupta e Magnusson (2005) consideram o problema capacitado de tamanho de lote e sequenciamento em uma única máquina (do inglês, CLSP - *capacitated lot-sizing and scheduling problem*) com custos de preparação não-nulos e dependentes da sequência de produção, com a característica adicional de que as preparações podem transitar de um período para o outro, e que as preparações são preservadas ao longo de períodos ociosos. Os autores apresentam uma formulação exata do problema usando programação inteira mista e uma heurística para a solução de grandes problemas-teste do CLSP.

Tahar *et al*. (2006) propuseram um novo algoritmo heurístico baseado em técnicas de programação linear para resolver o problema de programar um conjunto de tarefas independentes envolvendo tempos de preparação dependentes da sequência e divisão de tarefas em um conjunto de máquinas paralelas idênticas de tal forma que o tempo de conclusão máximo (*makespan*) é minimizado.

Omar *et al*. (2007) exploram os problemas de programação e sequenciamento que surgem em plantas de múltiplos produtos químicos que envolvem sequenciamento de tarefas oriundas de famílias incompatíveis, com tempos de preparação dependentes da sequência. Os autores sugerem duas formulações de programação inteira mista para este problema. A primeira formulação considera o caso de uma única máquina e visa minimizar o atraso total, enquanto a segunda formulação tenta minimizar a soma dos tempos de antecipação ou de atraso para o caso de máquinas paralelas.

Almada-Lobo *et al*. (2008) apresentam uma nota sobre o modelo proposto por Gupta e

Magnusson (2005) para o CLSP e mostram que este modelo apresenta alguns problemas e propõem um novo conjunto de restrições a ser adicionado ao modelo, a fim de proporcionar uma formulação melhor para este problema.

Behnamian *et al*. (2009) propõem uma metaheurística híbrida para a minimização do *makespan* em problemas de sequenciamento em máquinas paralelas e tempos de preparação dependentes da sequência de produção. A abordagem proposta por estes autores é composta por três componentes: um método de geração de população inicial baseado no método de otimização de colônia de formigas, um algoritmo de *Simulated Annealing* para a evolução da solução, e uma busca em vizinhança variável que envolve três procedimentos de busca local para melhorar a população.

Chen e Chen (2009) propõem várias metaheurísticas híbridas para o problema de sequenciamento em máquinas paralelas não-relacionadas com tempos de preparação dependentes da sequência, de modo a minimizar o número ponderado de tarefas atrasadas. As metaheurísticas propostas compreendem geradores de soluções iniciais efetivos e abordagens de melhoria que integram os métodos de descida em vizinhança variável e busca tabu. Esses autores também propõem quatro estruturas de vizinhança e duas estratégias de busca para melhorar a eficácia e eficiência das metaheurísticas.

Kovacs *et al*. (2009) introduzem um novo modelo de programação inteira mista para o problema capacitado de dimensionamento de lotes e sequenciamento em máquina única com tempos de preparação dependentes de sequência. Os autores utilizam um procedimento heurístico para gerar parâmetros para algumas restrições deste modelo, a fim de estabelecer uma formulação apertada. Os autores relatam que a abordagem proposta supera todas as abordagens de otimização previamente conhecidas e que podem resolver casos de tamanho relevante industrialmente.

Shim *et al*. (2011) também consideram o problema capacitado de dimensionamento de lotes e sequenciamento em máquina única com tempos de preparação dependentes de sequência, com o objetivo de minimizar a soma dos custos de estoque e preparação ao longo do horizonte de planejamento. Esses autores sugerem uma heurística de duas fases em que uma solução inicial é obtida e, em seguida, é melhorada usando um método de melhoria que utiliza várias regras de prioridade para selecionar os itens a serem movidos.

Neste artigo apresenta-se uma formulação matemática para o problema abordado e uma metaheurística *Simulated Annealing* (SA) para encontrar uma boa solução para o problema. O trabalho está organizado da seguinte forma. Na Seção 2 descreve-se o problema e apresenta-se a formulação matemática proposta. A Seção 3 apresenta a metaheurística aplicada na solução do problema. A Seção 4 é dedicada aos testes computacionais. A Seção 5 apresenta as considerações finais do trabalho.

2. Descrição e formulação do problema

No problema abordado neste trabalho, N tarefas devem ser executadas em M máquinas paralelas idênticas. O número de máquinas (M) é, no mínimo, igual a 2, e o número de tarefas (N) é maior do que ou igual ao número de máquinas (M). Considera-se que um tempo de preparação (*setup*) $S_{ij} > 0$ (independente da máquina) é requerido sempre que uma máquina deixa de processar uma tarefa i ($i = 1, ..., N$) e passa a processar uma tarefa j ($j = 1, ..., N, j \neq i$). Sem perda de generalidade, pode-se fazer $S_{ii} = 0$ ($i = 1, ..., N$). Assume-se também que um tempo de preparação inicial $t_i > 0$ é necessário se uma tarefa i ($i = 1, ..., N$) é programada para o início de uma sequência de produção. Além disso, assume-se que certas tarefas não podem ser processadas ao mesmo tempo. Para representar essa restrição, introduz-se uma matriz de incompatibilidade R (N x N), tal que $R_{ij} = 1$ se as tarefas i e j ($i, j = 1, ..., N$) não podem ser processadas ao mesmo tempo, e $R_{ij} = 0$, caso contrário. O tempo de processamento de uma tarefa i ($i = 1, ..., N$) é dado por $p_i > 0$. Assume-se também que, inicialmente, todas as tarefas estão disponíveis para processamento em qualquer máquina, que as máquinas podem processar apenas uma tarefa de cada vez e que o processamento das tarefas, uma vez iniciado, não pode ser interrompido. A decisão consiste em determinar, para cada máquina, e que ordem as tarefas devem ser processadas, de modo a minimizar o *makespan*.

Se C_i denota o tempo de conclusão de uma tarefa i (a ser decidido), o objetivo é minimizar $C_{\max} = \max\{C_i\}$. Se D_m denota o tempo no qual a máquina m (m = 1, ..., M) termina o seu processamento, esse objetivo é equivalente a minimizar C_{max} = Max (m = 1, ..., M) D_m. Note que D_m é a soma dos tempos de processamento das tarefas alocadas à máquina m (que não depende da sequência de tarefas) e dos tempos de preparação e de ociosidade (que são dependentes da sequência).

A formulação deste problema como um modelo de programação linear inteira foi proposta na dissertação de mestrado (Elisei, 2012), orientada pelo primeiro autor. No entanto, o grande número de variáveis e restrições torna o uso desse modelo impraticável para problemas-teste com mais de 7 tarefas e 3 máquinas.

Considerando a seguinte notação:

I – Conjunto de tarefas;

K – Conjunto de máquinas;

P – Conjunto de períodos;

p_i – Tempo de processamento da tarefa i ($i \in I$);

t_i – Tempo de preparação inicial da tarefa i ($i \in I$);

S_{ij} – Tempo de preparação da tarefa j processada imediatamente após a tarefa i ($i, j \in I$, $i \neq j$);

R_{ij} – Matriz de incompatibilidade entre as tarefas i e j ($i, j \in I$, $i \neq j$);

z_{ikp} – Variável binária tal que z_{ikp} = 1 se a tarefa i é processada no período p pela máquina k; z_{ikp} = 0, caso contrário ($i \in I$, $k \in K$, $p \in P$);

y_{kp} – Variável binária tal que y_{kp} = 1 se alguma tarefa é processada no período p pela máquina k; y_{kp} = 0, caso contrário ($k \in K$, $p \in P$);

w_{ijkp} – Variável binária tal que w_{ijkp} = 1 se a tarefa i é processada no período p e a tarefa j é processada no período (p+1) pela máquina k; w_{ijkp} = 0, caso contrário ($i, j \in I$, $i \neq j$, $k \in K$, $p \in P$).

O problema pode ser formulado como:

$$Min\ C_{\max} \tag{1}$$

sujeito a:

$$C_{\max} \geq \sum_{\substack{i \in I \\ k \in K}} t_i z_{ik1} + \sum_{\substack{i,j \in I \\ i \neq j \\ k \in K \\ p \in P}} S_{ij} w_{ijkp} + \sum_{\substack{i \in I \\ k \in K \\ p \in P}} p_i z_{ikp} \tag{2}$$

$$\sum_{i \in I} z_{ikp} \leq 1 \quad \forall k \in K, p \in P \tag{3}$$

$$\sum_{k \in K} z_{ikp} \leq 1 \quad \forall i \in I, p \in P \tag{4}$$

$$z_{ikp} + z_{jkp+1} - 2w_{ijkp} \geq 0 \quad \forall i, j \in I, i \neq j, k \in K, p \in P \tag{5}$$

$$z_{ikp} + z_{jkp+1} - 2w_{ijkp} \leq 1 \quad \forall i, j \in I, i \neq j, k \in K, p \in P \tag{6}$$

$$y_{kp} = \sum_{i \in I} z_{ikp} \quad \forall k \in K, p \in P \tag{7}$$

$$y_{kp} \geq y_{kp+1} \quad \forall k \in K, p \in P \tag{8}$$

$$z_{ikp} + z_{jk'p} + R_{ij} \leq 2 \quad \forall i, j \in I, i \neq j, k, k' \in K, p \in P \tag{9}$$

$$z_{ikp} \in \{0,1\} \quad \forall i \in I, k \in K, p \in P \tag{10}$$

$$y_{kp} \in \{0,1\} \quad \forall k \in K, p \in P \tag{11}$$

$$w_{ijkp} \in \{0,1\} \quad \forall i, j \in I, k \in K, p \in P \tag{12}$$

Nesta formulação, (1) representa a função-objetivo, que é minimizar o *makespan*, o qual é estabelecido pela restrição (2). As restrições (3) garantem que, no máximo, uma tarefa é processada por uma máquina em cada período. As restrições (4) previnem a mesma tarefa de ser processada em máquinas diferentes no mesmo período. As restrições (5) e (6) requerem que w_{ijkp} = 1 se existirem diferentes tarefas sendo processadas em períodos consecutivos numa certa máquina, e w_{ijkp} = 0, caso contrário, respectivamente. As restrições (7) garantem que y_{kp} = 1 se alguma tarefa é processada pela máquina *k* no período *p*. Isto é importante para as restrições (8), para prevenir a existência de períodos vazios entre os processamentos de duas tarefas diferentes numa dada máquina. As restrições (9) garantem que duas tarefas podem ser processadas (em diferentes máquinas) ao mesmo tempo somente se essas tarefas forem compatíveis. As restrições (10)-(12) fornecem as condições de integralidade para as variáveis de decisão.

3. A heurística proposta

O problema considerado neste artigo consiste de duas decisões: atribuição de tarefas a máquinas e sequenciamento das tarefas atribuídas a cada máquina. A melhor ordem na qual as tarefas são processadas deve levar em conta os tempos de *setup* e a compatibilidade entre tarefas. O *makespan* é a soma de todos os tempos de processamento e tempos de *setup* na máquina mais crítica.

Para ilustrar o problema, considere um exemplo com seis tarefas (N = 6) e três máquinas (M = 3). Os tempos de processamento e os tempos de *setup* iniciais para cada tarefa são dados pela Tabela 1. Os tempos de *setup* entre tarefas são dados pela Tabela 2. A matriz de incompatibilidade de tarefas é dada pela Tabela 3.

Tabela 1. Tempos de processamento e *setup* iniciais

Tarefa *i*	1	2	3	4	5	6
Tempo de processamento p_i	2	2	3	1	2	1
Tempo de *setup* inicial t_i	2	3	4	4	3	2

Tabela 2. Matriz de tempos de *setup* S_{ij}

i/j	1	2	3	4	5	6
1	0	1	2	3	1	1
2	3	0	4	2	1	2
3	5	6	0	3	4	3
4	3	4	5	0	2	2
5	2	2	4	3	0	2
6	3	3	3	4	3	0

Tabela 3. Matriz de incompatibilidade R_{ij}

i/j	1	2	3	4	5	6
1	0	1	0	0	1	0
2	1	0	0	0	0	0
3	0	0	0	1	0	0
4	0	0	1	0	0	0
5	1	0	0	0	0	0
6	0	0	0	0	0	0

Para calcular o *makespan* de qualquer solução viável do problema deve-se levar em conta os tempos de processamento, os tempos de *setup* e a compatibilidade entre tarefas. Considere, por exemplo, a solução viável mostrada na Tabela 4:

Tabela 4. Uma solução viável

m	Sequência de tarefas
1	2–5–6
2	1–3
3	4

Para essa solução, pode-se considerar o gráfico de Gantt mostrado na Figura 1:

	períodos																
m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	2	2	2	2	5	5	5	6	6	6						
2									1	1	1	1	3	3	3	3	3
3	4	4	4	4	4												

Figura 1 – Gráfico de Gantt de uma solução

Neste caso, *makespan* = 17. Note que a sequência 2-2-2-2-2 para a tarefa 2 nos períodos 1 a 5 refere-se a três unidades para o tempo de *setup* inicial e duas unidades para o tempo de processamento. A sequência 5-5-5 para a tarefa 5 nos períodos 6 a 8 refere-se a uma unidade devida ao tempo de *setup* requerido entre as tarefas 2 e 5, e duas unidades para o tempo de processamento da tarefa 5. Note que a tarefa 1 pode começar a ser processada somente no período 9 devido à incompatibilidade com as tarefas 2 e 5.

Portanto, uma solução para o PMSP pode ser representada por um vetor de *m* máquinas tal que cada elemento do vetor armazena uma possível sequência de tarefas (Tabela 4). A função-objetivo pode ser calculada como o número total de períodos (*makespan*) do gráfico de Gantt associado a uma dada solução.

A solução inicial para o algoritmo SA (*Simulated Annealing*) é balanceada: os números de tarefas atribuídas a quaisquer duas máquinas diferem em, no máximo, uma unidade. Em primeiro lugar, as tarefas são atribuídas aleatoriamente às máquinas, e então as tarefas são distribuídas nas máquinas até que nenhuma máquina contenha mais do que uma tarefa do que qualquer outra máquina.

3.1. O algoritmo SA (*Simulated Annealing*)

Nesta seção, apresenta-se a heurística SA (*Simulated Annealing*) proposta para o PMSP. A heurística *Simulated Annealing* foi proposta por Kirkpatrick *et al*. (1983). O algoritmo SA proposto parte de uma solução inicial aleatória. Os próximos passos do algoritmo SA seguem o esquema tradicional da heurística *Simulated Annealing*. Dada uma temperatura T, o algoritmo seleciona aleatoriamente um dos movimentos para uma vizinhança da solução corrente e calcula a variação do valor da função-objetivo. Se esse movimento melhora a solução corrente, o

movimento é aceito. Caso contrário, existe uma probabilidade de aceitação do movimento que é tanto menor quanto menor for a temperatura.

Três diferentes movimentos foram definidos para compor as vizinhanças de uma solução, denominadas N^1, N^2 e N^3. A vizinhança N^1 é obtida trocando-se a alocação de duas tarefas de diferentes máquinas. A vizinhança N^2 é obtida retirando-se uma tarefa alocada a uma máquina e alocando-se essa tarefa a uma outra máquina. Finalmente, a vizinhança N^3 é obtida trocando-se a posição de duas tarefas da mesma máquina.

Os parâmetros de controle do procedimento são a taxa de resfriamento ou fator de decremento α, o número de iterações para uma temperatura fixa (SA_{max}) e a temperatura inicial T_0. Neste trabalho, foram usados α=0.95, SA_{max}=1000 e T_0=1000000. O algoritmo SA encerra o processamento quando a temperatura é muito baixa ($T \leq 0.0001$). A Figura 2 mostra o pseudocódigo do algoritmo SA implementado.

```
algoritmo SA (T0, SAmax, α )
    gere uma solução inicial s
    IterT ← 0
    T ← T0
    enquanto (critério de parada não for satisfeito ) faça
        enquanto ( IterT < SAmax ) faça
            IterT ← IterT + 1
            gere um vizinho s' aleatoriamente ( s' ∈ N(s) )
            calcule Δ = f (s') – f (s)
            se (Δ ≤ 0) então
                s ← s'
            senão
                s ← s' com probabilidade e^(-Δ/T)
        fim-enquanto
        T ← α x T
        IterT ← 0
    fim-enquanto
    retornar s*
fim-algoritmo
```

Figura 2 – Pseudocódigo do algoritmo SA

4. Resultados computacionais

A seguir, descreve-se o experimento computacional realizado com o algoritmo SA. O algoritmo SA foi codificado em C++ e executado em um PC com processador Pentium Core 2 Duo de 2.0 GHz com 3 GB de RAM sob o sistema operacional Windows.

O algoritmo SA foi testado para problemas-teste do PMSP gerados com base em dados reais de uma indústria de autopeças, em que as restrições e a incompatibilidade entre peças foram mantidas e os tempos de produção e *setup* foram gerados aleatoriamente (proporcional aos valores reais). Esses problemas-teste contêm 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 70, 80, 90, 100, 120, 150, 170 e 200 tarefas, e 3, 4 e 5 máquinas. Existe um problema-teste para cada par de valores (tarefas, máquinas), resultando num total de 60 problemas-teste. O algoritmo SA foi executado 20 vezes para cada problema-teste. Também foi utilizado um problema-teste que corresponde ao caso real de uma indústria de autopeças que tem 194 tarefas com uma demanda

de 36323 peças para 3 máquinas. Estes problemas-teste, assim como as melhores soluções encontradas, estão disponíveis em *http://www.sjc.unifesp.br/docente/ chaves/problem-instances.*

Os resultados dos testes computacionais estão resumidos na Tabela 5. Esta tabela contém as seguintes colunas: **Instância** (nome do problema-teste, que indica o número de tarefas), **Melhor** (a melhor solução encontrada), **Média** (a média das soluções encontradas), **Desvio** (a diferença percentual entre a média das soluções e a melhor solução encontrada, calculada como Desvio = 100%(Média − Melhor)/Melhor) e **Tempo** (o tempo computacional, em segundos).

A Tabela 5 apresenta os resultados para cada problema-teste para 3, 4 e 5 máquinas. O algoritmo SA obtém boas soluções para o PMSP, com um desvio relativo médio de, no máximo, 0,52% (para 3 máquinas), 0,59% (para 4 máquinas) e 0,88% (para 5 máquinas). Os tempos computacionais do algoritmo SA são razoáveis (poucos segundos ou minutos). Para o problema-teste correspondente ao caso real, a solução adotada atualmente pela empresa tem um *makespan* de 12446. O melhor resultado obtido pelo algoritmo SA para este problema-teste tem um *makespan* de 10523, ou seja, cerca de 15% melhor.

Tabela 5. Resultados obtidos pelo algoritmo SA

	SA											
Instância	3 máquinas				4 máquinas				5 máquinas			
	Melhor	Média	Desvio (%)	Tempo (s)	Melhor	Média	Desvio (%)	Tempo (s)	Melhor	Média	Desvio (%)	Tempo (s)
p005	401	401	0,00	22,15	401	401	0,00	26,95	**401**	401	0,00	32,93
p010	911	911,45	0,05	76,58	688	688,9	0,13	106,79	**576**	576	0,00	139,62
p015	2201	2208,45	0,34	152,02	1652	1661,3	0,56	215,53	1340	1351,8	0,88	282,50
p020	2737	2744,05	0,26	227,70	2055	2066,8	0,57	341,85	1649	1662,3	0,81	456,12
p025	2028	2034,65	0,33	130,16	1524	1531,1	0,47	179,68	1222	1230,7	0,71	237,76
p030	1706	1714,4	0,49	99,36	1282	1290,95	0,70	133,27	1029	1036,55	0,73	173,68
p035	2597	2607,9	0,42	193,90	1954	1964,4	0,53	279,48	1573	1579,3	0,40	373,95
p040	3745	3764,5	0,52	258,21	2817	2828,2	0,40	365,87	2256	2268,65	0,56	493,71
p045	4979	4995,15	0,32	340,28	3743	3758,75	0,42	487,76	3001	3011,2	0,34	655,67
p050	4061	4076,6	0,38	251,73	3058	3070,05	0,39	345,97	2446	2458,9	0,53	463,48
p055	4538	4547,85	0,22	271,47	3403	3415,5	0,37	372,61	2724	2740,7	0,61	490,18
p060	6953	6978,55	0,37	447,61	5216	5246,9	0,59	625,11	4196	4206,95	0,26	840,91
p070	6339	6364,25	0,40	396,23	4765	4785,8	0,44	540,63	3824	3834,6	0,28	721,22
p080	7890	7920,05	0,38	500,88	5931	5954,8	0,40	686,76	4746	4768,6	0,48	920,31
p090	8747	8781,4	0,39	555,91	6569	6599,45	0,46	769,75	5273	5291,8	0,36	1001,66
p100	9149	9189,55	0,44	599,61	6878	6902,15	0,35	823,21	5519	5533,15	0,26	1069,38
p120	12719	12767,15	0,38	826,70	9584	9597,85	0,14	1081,57	7666	7683,2	0,22	1414,80
p150	14389	14441,55	0,37	1040,49	10832	10852,75	0,19	1383,46	8666	8688,45	0,26	1739,44
p170	16618	16643	0,15	1224,74	12466	12494,4	0,23	1552,60	9983	10011,75	0,29	1782,96
p200	18674	18713,4	0,21	1687,48	14058	14077,65	0,14	1768,32	11262	11278,8	0,15	1800,66
real194	10523	10643,45	1,14	1800,06	-	-	-	-	-	-	-	-
Média	**6569,10**	**6590,25**	**0,32**	**465,16**	**4943,80**	**4959,44**	**0,37**	**604,36**	**3967,60**	**3980,72**	**0,41**	**754,55**

5. Conclusão

Neste trabalho, considerou-se uma variação para o problema de sequenciamento de tarefas em máquinas paralelas com tempos de preparação dependentes da sequência de produção, que inclui restrições de compatibilidade entre tarefas. Foram apresentados uma formulação matemática e um algoritmo *Simulated Annealing* para o problema. A formulação foi executada com *solver* CPLEX, mas este consegue resolver apenas exemplares muito pequenos (cerca de 7 tarefas e 3 máquinas). Os resultados dos testes computacionais realizados mostram que o algoritmo heurístico implementado obtém boas soluções em um tempo razoável, tendo sido a metaheurística com melhores resultados entre as implementadas pelos autores até o momento. Para um problema-teste correspondente a um caso real de uma indústria de autopeças, o algoritmo implementado obteve uma solução melhor do que a atualmente usada pela empresa.

Novos estudos podem ser feitos considerando outras metaheurísticas, como: *Variable Neighborhood Search* (Hansen e Mladenovic, 2003), *Iterated Local Search* (Lourenço *et al.*, 2003) e Algoritmo Genético (Goldberg, 1989), ou então analisando outras formas de aplicar

busca local, como *Clustering Search* (CS) (Chaves e Lorena, 2010). A ideia do CS é evitar a aplicação de heurísticas de busca local a todas as soluções geradas por uma metaheurística. O CS detecta regiões promissoras no espaço de busca e aplica busca local somente nessas regiões. Isto é interessante porque previne a aplicação indiscriminada de métodos de busca local que, em geral, são computacionalmente caros.

Agradecimentos

Os autores agradecem ao CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) pelo suporte financeiro parcial para a realização deste trabalho e aos revisores anônimos pelas contribuições.

Referências

Allahverdi, A.; Gupta, J.N.D.; Aldowaisan, T. (1999), A review of scheduling research involving setup considerations, *Omega International Journal of Management Science*, 27, 219-239.

Allahverdi, A.; Ng, C.T.; Cheng, T.C.E.; Kovalyov, M.Y. (2008), A survey of scheduling problems with setup times or costs, *European Journal of Operational Research*, 187, 985-1032.

Almada-Lobo, B,; Oliveira, J.F.; Carravilla, M.A. (2008), A note on the capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times, *Computers & Operations Research*, 35, 1374-1376.

Behnamian, J.; Zandieh, M.; Ghomi, S.M.T.F. (2009), Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA an VNS hybrid algorithm, *Expert System with Applications*, 36, 9637-9644.

Chaves, A.A.; Lorena, L.A.N. (2010), Clustering search algorithm for the capacitated centered clustering problem, *Computers & Operations Research*, 37, 552-558.

Chen, C.; Chen, C. (2009), Hybrid metaheuristics for unrelated parallel machine scheduling with sequence-dependent setup times, *The International Journal of Advanced Manufacturing Technology*, 46, 161-169.

Elisei, J.L. Sequenciamento de lotes em prensas de alta capacidade com tempo de *setup* dependente da sequência. Dissertação (mestrado), Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2012.

Goldberg, D. E. *Genetic algorithms in Search: optimization and machine learning*. Addison-Wesley, Berkeley, 1989.

Gupta, D.; Magnusson, T. (2005), The Capacitated Lot-sizing and Scheduling Problem with Sequence-Dependent Setup Cost and Setup Times, *Computers and Operations Research*, 32, 727-747.

Hansen, P.; Mladenovic, N., Variable neighborhood search, em Glover, F.; Kochenberger, G. A. (Eds.), *Handbook of Metaheuristics*. New York: Springer, (International Series in Operations Research & Management Science, v. 57), 145-184, 2003.

Kirkpatrick, S.; Gellat, D. C.; Vecchi, M. P. (1983), Optimization by simulated annealing. *Science*, 220 (4598), 671-680.

Kovacs, A.; Brown, K.N.; Tarim, S.A. (2009), An efficient MIP model for the capacitated lot-sizing and scheduling problem with sequence-dependent setups, *International Journal of Production Economics*, 118, 282-291.

Lourenço, H.; Martin, O.; Stützle, T., Iterated local search, em Glover, F.; Kochenberger, G. A. (Ed.). *Handbook of Metaheuristics*. New York: Springer , (International Series in

Operations Research & Management Science, v. 57), 320-353, 2003.

Meyr, H. (2002), Simultaneous Lotsizing and Scheduling on Parallel Machines. *European Journal of Operational Research*, 139, 277-292.

Omar, M.K.; Teo, S.C.; Yasothei, S. Scheduling with Setup Considerations: An MIP Approach, em Levner, E. (Ed.), *Multiprocessor Scheduling, Theory and Applications*, I-Tech Education, 242-254, 2007.

Shim, I.S.; Kim, H.C.; Doh, H.H.; Lee, D.H. (2011), A two-stage heuristic for single machine capacitated lot-sizing and scheduling with sequence-dependent setup costs, *Computers & Industrial Engineering*, 61, 920-929.

Tahar, D.N.; Yalaoui, F.; Chu, C.; Amodeo, L. (2006), A linear programming approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times, *International Journal of Production Economics*, 99, 63–73.