# Um algoritmo *Iterated Local Search* para o Problema de Sequenciamento de Tarefas em Máquinas Idênticas para minimização de *makespan* e consumo total de energia

**Klevison D. O. Ribeiro** [*] **Luciano P. Cota** [**] **Helen Costa** [***]
**Frederico G. Guimarães** [****] **Marcone J. F. Souza** [†]

[*] *Programa de Pós-Graduação em Instrumentação, Controle e Automação de Processos de Mineração, Universidade Federal de Ouro Preto e Instituto Tecnológico Vale (e-mail: klevison.ribeiro@aluno.ufop.edu.br).*
[**] *Instituto Tecnológico Vale (e-mail: luciano.p.cota@itv.org).*
[***] *Departamento de Computação e Sistemas, Universidade Federal de Ouro Preto (e-mail: helen@ufop.edu.br).*
[****] *Departamento de Engenharia Elétrica, Universidade Federal de Minas Gerais (e-mail: fredericoguimaraes@ufmg.br).*
[†] *Departamento de Computação, Universidade Federal de Ouro Preto (e-mail: marcone@ufop.edu.br).*

**Abstract:** This work deals with an identical machine scheduling problem. In this problem, a set of jobs must be allocated to a set of identical machines at a given instant of time, aiming at minimizing the weighted sum of the makespan and the total energy consumption involved in this operation. Problems of this class, known as green scheduling, have been emerging in the literature given the growing concern about the environment and sustainable development. In order to solve it, an algorithm based on *Iterated Local Search* (ILS) was developed, having as the *Random Variable Neighborhood Descent* (RVND) as local search method. The results of the ILS were compared with those produced by the CPLEX optimizer, and it was observed that ILS is able to produce good quality solutions in a short processing time.

**Resumo**: Este trabalho trata um problema de sequenciamento de tarefas em máquinas idênticas. No problema abordado, tem-se um conjunto de tarefas que devem ser atribuídas a um conjunto de máquinas idênticas em um determinado instante de tempo, tendo como objetivo minimizar a soma ponderada do *makespan* e do consumo total de energia envolvido nessa operação. Problemas desta classe, conhecidos como sequenciamento verde (*green scheduling*), têm surgido na literatura, dada a crescente preocupação com o meio ambiente e o desenvolvimento sustentável. Para resolvê-lo desenvolveu-se um algoritmo baseado na metaheurística *Iterated Local Search* (ILS), tendo como busca local o procedimento *Random Variable Neighborhood Descent* (RVND). Os resultados do algoritmo proposto foram comparados com aqueles produzidos pelo otimizador CPLEX, sendo possível observar que o ILS é capaz de produzir soluções de boa qualidade em baixo tempo de processamento.

*Keywords:* Green scheduling; Identical Machine Scheduling; Iterated Local Search; Random Variable Neighborhood Descent.

*Palavras-chaves:* Sequenciamento verde; Sequenciamento em Máquinas Paralelas Idênticas; Busca Local Iterada; Descida Aleatória em Vizinhança Variável.


**DOI:** 10.17648/sbai-2019-111233

## 1. INTRODUÇÃO

Este trabalho trata um problema de sequenciamento em máquinas idênticas (PSMI) que surgiu recentemente na área de sequenciamento de tarefas, conhecido como sequenciamento verde (*green scheduling*) (Bampis et al., 2015; Mansouri, 2016a; Mansouri et al., 2016b). Tal como em Wang et al. (2018), procura-se atribuir um dado número de tarefas a um conjunto de máquinas disponíveis, em determinado instante de tempo de forma a minimizar o tempo total de processamento (*makespan* ou $C_{\max}$) e o consumo total de energia (TEC, *Total Energy Consumption*). No entanto, ao contrário de Wang et al. (2018), que tratou o problema de forma multiobjetivo, neste trabalho utiliza-se uma função objetivo mono-objetivo, dada pela soma ponderada dos dois objetivos considerados.

O tema *green scheduling* vem sendo amplamente estudado atualmente, dada a crescente preocupação com o meio ambiente e o desenvolvimento sustentável, assim como pelas inúmeras aplicações na indústria de manufatura e automobilística (Wang et al., 2018).

Nessa versão de sequenciamento, além da busca pela minimização dos custos relacionados ao tempo de processamento (*makespan*, tempo total de atraso, entre outros), procura-se também minimizar o consumo de recursos, como água, energia elétrica, ou mesmo a redução da emissão de poluentes, uma vez que todos esses fatores impactam diretamente o meio ambiente (Sauer et al., 2015).

Trabalhos desta natureza estão cada vez mais presentes na literatura. Wang et al. (2018) tratam o PSMI objetivando a minimização do *makespan* e do TEC em uma abordagem biobjetivo. O consumo de energia é calculado com base nos tempos de uso das máquinas e no preço de utilização delas no período de uso. Para resolvê-lo, os autores aplicaram os métodos $\epsilon$-restrito aumentado e NSGA-II.

Cota et al. (2018) trataram o problema de sequenciamento em máquinas paralelas, não relacionadas, com tempos de preparação dependentes da sequência. Assim como neste trabalho, os autores buscam minimizar o *makespan* e o TEC. Diferentemente de Wang et al. (2018), o TEC é calculado com base nos modos de operação das máquinas, que podem ter sua velocidade aumentada para reduzir o tempo de processamento das tarefas; entretanto, consumindo mais energia nesse processo.

Em Li et al. (2011) o PSMI é tratado com o objetivo de se reduzir o *makespan*, restringido pelo consumo limitado de recursos. Para resolvê-lo é aplicado um algoritmo *Simulated Annealing*. Ji et al. (2013), por sua vez, utilizaram um algoritmo *Particle Swarm Optimization* (PSO) para tratar o problema de sequenciamento em máquinas paralelas uniformes. Neste caso, os autores abordam o problema considerando um limite superior de *makespan* predefinido, e buscam a minimização do consumo de energia elétrica, água e emissão de gás carbônico.

Já em Ding et al. (2016), o problema de sequenciamento em máquinas paralelas é tratado para minimizar o custo total de eletricidade com tempo limite de execução. Os autores propõem uma abordagem exata baseada em programação linear inteira mista e também uma heurística de geração de colunas para resolvê-lo.

Um modelo biobjetivo para o problema de sequenciamento em máquinas paralelas é proposto em Zeng et al. (2018). Os autores buscam minimizar o custo total da energia elétrica utilizada, bem como o número de máquinas usadas durante os horários com tarifação mais cara.

Neste trabalho trata-se o modelo de Wang et al. (2018) em uma abordagem mono-objetivo, em que os objetivos de minimização do *makespan* e do TEC são ponderados. Dado o fato de que o problema aqui tratado tem como subproblema o PSMI com o objetivo de minimizar o *makespan*, e este é NP-difícil (Karp, 1972), para encontrar soluções aproximadas para ele, foi desenvolvido um algoritmo baseado no método *Iterated Local Search* - ILS (Lourenço et al., 2003). O ILS foi escolhido tendo em vista sua aplicação bem sucedida na resolução de vários outros problemas combinatórios (Lourenço et al., 2003).

O restante deste trabalho está estruturado como segue. Na Seção 2, as particularidades e restrições do problema são descritas. Na Seção 3 é detalhado o algoritmo desenvolvido. Na Seção 4 são reportados os resultados dos experimentos realizados. Por fim, na Seção 5 são apresentadas as conclusões sobre a aplicação do algoritmo proposto e apresentadas sugestões para trabalhos futuros.

## 2. CARACTERIZAÇÃO DO PROBLEMA

O problema de sequenciamento em máquinas idênticas (PSMI) considerado neste trabalho tem as seguintes características:

(1) Tem-se um conjunto de $n$ tarefas a serem executadas em um conjunto de $m$ máquinas durante um horizonte $T_{\max}$ de processamento;
(2) As máquinas são idênticas, dessa forma o tempo de processamento de cada tarefa é sempre o mesmo, qualquer que seja a máquina usada para processá-la;
(3) Cada máquina $j$ tem uma taxa de consumo de energia, dada por $e_j$;
(4) A cada instante $k$ do horizonte de planejamento há um custo $c_k$ para processar uma tarefa;
(5) Uma vez iniciada uma tarefa, ela deve ser concluída, não podendo sua execução ser interrompida e retomada posteriormente;
(6) Cada tarefa deve ser executada uma única vez;
(7) O objetivo é minimizar a soma ponderada do *makespan* e do TEC, normalizados, dada pela expressão $\alpha \cdot C'_{\max} + \beta \cdot TEC'$, sendo $\alpha$ e $\beta$ pesos de ponderação tais que $\alpha + \beta = 1$, $\alpha, \beta \in [0, 1]$, e $C'_{\max}$ e $TEC'$ representam os valores de $C_{\max}$ e $TEC$ normalizados no intervalo [0,1], respectivamente.

De acordo com a notação de Pinedo (2008), o PSMI aqui descrito pode ainda ser representado pela seguinte tripla: $P_m \,||\, (C_{\max}, TEC)$, na qual $P_m$ representa a característica do problema de utilizar máquinas paralelas idênticas e o par $(C_{max}, TEC)$ indica os objetivos a serem minimizados.

## 3. DESCRIÇÃO DO ALGORITMO PROPOSTO

Nesta seção, o algoritmo ILS desenvolvido é descrito. Na Seção 3.1 mostra-se como uma solução do PSMI é representada. Na Seção 3.2 descreve-se como uma solução inicial é gerada, enquanto na Seção 3.3 mostra-se como uma solução é avaliada. Na Seção 3.4, são apresentados os


DOI: 10.17648/sbai-2019-111233

movimentos utilizados para explorar o espaço de soluções do problema, enquanto na Seção 3.5 o ILS é detalhado.

### *3.1 Representação da Solução*

Uma solução $s$ do problema é representada tal como em Wang et al. (2018), isto é, por um vetor $s$ de dimensão $2n$. Nesse vetor, cada posição $j$, com $j \in [1, n]$, indica a máquina que executa a tarefa $j$, enquanto que a posição $k = j + n$, com $k \in [n + 1, 2n]$ indica o instante de início dessa tarefa.

Uma representação auxiliar também foi utilizada para facilitar a implementação. Mais precisamente, a cada máquina associa-se um vetor com $T_{\max}$ posições, sendo que cada posição desse vetor representa um instante da máquina no horizonte de planejamento, e cada célula desse vetor indica a tarefa que está sendo executada. No caso de a máquina estar ociosa naquele instante, atribui-se o número 0 àquela célula. A Figura 1 ilustra um exemplo de como uma solução é representada em um problema com 6 tarefas e 3 máquinas. Nessa figura, percebe-se, por exemplo, que na solução $s$ a tarefa 6 (posição 6 do vetor $s$) é executada na máquina 2 (valor da célula $s_6$ do vetor $s$) e no instante 5 (valor da célula $s_{12}$ do vetor $s$). Na estrutura auxiliar mostra-se um vetor de 10 posições para cada máquina, já que neste exemplo $C_{\max} = 10$. Assim, na máquina M1, por exemplo, em sua quinta posição a célula contém o valor 5, indicado que no instante 5 essa máquina está processando a tarefa 5. Nesse mesmo vetor da máquina M1, tem-se o conteúdo 0 nas posições 8, 9 e 10, indicando que nesses instantes a máquina está ociosa.

### *3.2 Solução Inicial*

Uma solução inicial é gerada por um procedimento guloso, que consiste em atribuir cada tarefa, iniciando-se da primeira, na máquina que tem o menor tempo total de processamento. Em caso de empate, a tarefa é atribuída à máquina com o menor índice.

### *3.3 Função de Avaliação*

Uma solução $s$ do problema é avaliada pela função mono-objetivo $f(s)$, dada por (1), que é a soma ponderada do makespan e do consumo total de energia, ambos normalizados no intervalo [0,1].

$$f(s) = \alpha \cdot C'_{\max}(s) + \beta \cdot TEC'(s) \quad (1)$$

em que $C'_{\max}$ e $TEC'$ representam os valores de $C_{\max}$ e $TEC$ normalizados no intervalo [0,1], e $\alpha$ e $\beta$, com $\alpha, \beta \in [0, 1]$, $\alpha + \beta = 1$, são pesos de ponderação que refletem a importância relativa de cada grandeza avaliada.

A determinação de $C'_{\max}(s)$ em (1) é obtida de acordo com (2), em que $C_{\max}(s)$ é o maior tempo total de processamento dentre todas as máquinas, e $T_{\max}$ o horizonte de planejamento, isto é, um limite superior para $C_{\max}$.:

$$C'_{\max}(s) = C_{\max}(s)/T_{\max} \quad (2)$$

Para estimar $T_{\max}$, basta minimizar apenas a parcela $TEC'(s)$ da solução. Isto é possível pois as duas parcelas que compõem a função de avaliação são conflitantes. Sendo assim, ao se minimizar uma, a outra atinge o seu maior valor possível. Então, para estimar o valor de $T_{\max}$, basta minimizar a parcela $TEC'(s)$ alocando as tarefas nos instantes em que o preço de utilização de energia for mais baixo. Isso criará lacunas no horizonte de planejamento, pois os períodos de tempo com preço alto de utilização de energia não terão tarefas alocadas. Assim, a parcela $C'_{\max}(s)$ alcançará o seu valor máximo, isto é, uma estimativa do limite superior $T_{\max}$ para o *makespan*.

Por sua vez, a determinação de $TEC'(s)$ em (1) é feita de acordo com (3):

$$TEC'(s) = TEC(s)/TEC_{\max} \quad (3)$$

em que $TEC(s)$ mensura o consumo total de energia de todas as máquinas envolvidas na solução $s$, e é obtida de acordo com (4):

$$TEC(s) = \sum_{j=1}^{n} EC_j(s) \quad (4)$$

sendo $EC_j$ o consumo de energia na máquina $j$, calculado por meio de (5):

$$EC_j(s) = e_j \cdot \sum_{i=1}^{n} \sum_{k=1}^{T_{\max}} c_k x_{ijk} \quad (5)$$

Em (5), $e_j$ é a taxa de consumo de energia da máquina $j$, $c_k$ representa o custo de uso da energia no instante de tempo $k$, $T_{\max}$ é o horizonte de planejamento, e, portanto, o limite superior para $C_{\max}$.

Em (3), $TEC_{\max}$ representa o consumo máximo de energia e pode ser estimado atribuindo todas as tarefas na máquina de maior taxa de consumo de energia, propiciando um total de consumo de energia grande o suficiente para normalizar a parcela $TEC(s)$ no intervalo [0,1].

A Figura 1 ilustra a representação de uma solução considerada para mostrar o cálculo das parcelas $C_{\max}(s)$ e $TEC(s)$. Os tempos de processamento das máquinas $M1$, $M2$ e $M3$ nessa solução são iguais a $C_{M1} = 7$, $C_{M2} = 10$ e $C_{M3} = 5$, respectivamente. Como $C_{\max}(s)$ é o maior valor de $C_j$ dentre todas as máquinas, então $C_{\max}(s) = 10$.

A energia consumida por cada máquina $j$, por sua vez, é calculada com base na taxa de consumo de cada máquina e nos instantes de tempo em que cada máquina foi utilizada. Dessa forma, considerando as taxas de consumo das máquinas 1, 2 e 3, respectivamente iguais a 1, 3 e 1 unidades e os custos de uso nos instantes de tempo de 1 a 10 iguais a 6, 6, 5, 5, 5, 2, 2, 2, 2, 2 respectivamente, temos que:

$$TEC(s) = EC_1 + EC_2 + EC_3 = 31 + 111 + 27 = 169$$

É importante ressaltar que, quando movimentos são realizados para explorar o espaço de soluções do problema, é necessário atualizar o valor da função de avaliação dessa solução. Para isso, um cálculo inteligente é utilizado para evitar que todas as máquinas tenham seu consumo de energia recalculados. Considerando que o movimento realizado envolve apenas as máquinas $j_1$ e $j_2$, a atualização da parcela $TEC(s)$ da função $f(s)$ é calculado conforme (6), em que $TEC^a(s)$ representa o consumo de energia da solução $s$ antes da aplicação do movimento, $EC^a_{j_1}$ e $EC^a_{j_2}$ representam, respectivamente, o consumo de energia nas


DOI: 10.17648/sbai-2019-111233

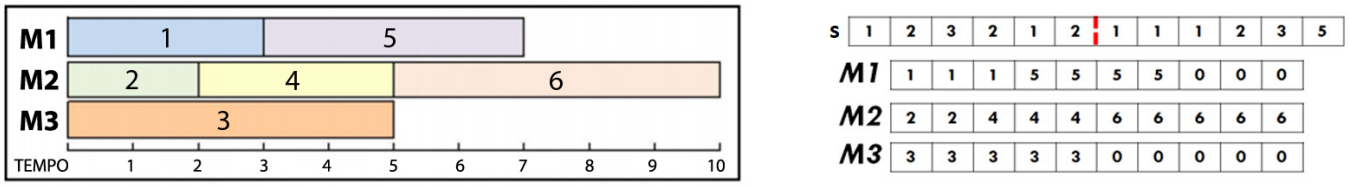


Figura 1. Representação de uma solução do problema

máquinas $j_1$ e $j_2$ antes do movimento e $EC^d_{j_1}$ e $EC^d_{j_2}$ representam o consumo de energia nessas mesmas máquinas depois do movimento.

$$TEC(s) = TEC^a(s) - (EC^a_{j_1} + EC^a_{j_2}) + (EC^d_{j_1} + EC^d_{j_2}) \quad (6)$$

**Algoritmo 1:** ILS

```
   entrada: s_inicial, ILS_max
   saída  : s
1  s ← RVND(s_inicial);
2  iterSemMelhora ← 0;
3  nivel ← 1;
4  enquanto (iterSemMelhora ≤ ILS_max) faça
5  |  iterSemMelhora ← iterSemMelhora + 1;
6  |  s' ← perturbacao(s, nivel);
7  |  s'' ← RVND(s');
8  |  se (f(s'') < f(s)) então
9  |  |  s ← s'';
10 |  |  iterSemMelhora ← 0;
11 |  |  nivel ← 1;
12 |  fim
13 |  senão
14 |  |  nivel ← nivel + 1;
15 |  fim
16 fim
17 retorne s;
```

### 3.4 Estruturas de vizinhança

Foram utilizadas três estruturas de vizinhança, $N_1$, $N_2$ e $N_3$, para explorar o espaço de soluções do PSMI, as quais são geradas a partir dos seguintes movimentos, nesta ordem:

- $N_1$) **Troca de tarefas entre máquinas:** Duas tarefas de máquinas diferentes são trocadas de posição.
- $N_2$) **Realocação de tarefa para outra máquina:** Uma tarefa de uma máquina é realocada para outra posição de outra máquina.
- $N_3$) **Realocação de tarefa na mesma máquina:** Uma tarefa é realocada para outra posição na mesma máquina, isto é, ela passa a iniciar em um instante de tempo diferente. Tal movimento é importante pois permite a redução do $TEC$ sem alterar o $C_{\max}$.

Em todas as estruturas de vizinhança existe um rearranjo de tarefas nas máquinas, após a aplicação do movimento. O rearranjo ocorre sempre que a tarefa realocada é inserida numa posição que já tenha alguma tarefa sendo executada. Neste caso, a tarefa sobreposta e as tarefas subsequentes são remanejadas para o término da tarefa que foi realocada primeiramente, corrigindo a sobreposição. Entretanto, o rearranjo de tarefas não ocorre quando surgem lacunas no horizonte de planejamento das máquinas. Nesses casos, o rearranjo não ocorre porque essas lacunas podem gerar soluções melhores que a solução anterior ao movimento.

### 3.5 Iterated Local Search (ILS)

Nesta seção, descrevemos o algoritmo ILS desenvolvido para resolver o PSMI. Seu pseudocódigo segue o *framework* de Lourenço et al. (2003), e é mostrado no Algoritmo 1.

Inicialmente, na linha 1, a solução inicial $s_{inicial}$ é submetida a um procedimento de busca local pelo RVND, descrito pelo Algoritmo 2. Na linha 2, o número de iterações sem melhora é iniciado em 0 e na linha 3 o nível de perturbação é iniciado em 1. Na linha 4, inicia-se o laço de repetição do algoritmo ILS, que é encerrado quando o número de iterações sem melhora for maior que o limite máximo $ILS_{\max}$, recebido como entrada.

O número de iterações sem melhora é incrementado na linha 5. Na linha 6, é aplicada uma perturbação na solução corrente, tal como descrito a seguir, gerando-se uma solução intermediária $s'$. Uma nova busca local é aplicada sobre essa solução intermediária $s'$, gerando-se possivelmente uma solução ótima local $s''$. Essa etapa ocorre na linha 7. Das linhas 8 a 12 tem-se um laço condicional, no qual as soluções $s''$ e $s$ são avaliadas de acordo com (1). Se $s''$ for melhor do que $s$, então a solução $s$ é atualizada, o número de iterações sem melhora retorna para seu valor inicial, isto é, 0, e o nível de perturbação é reiniciado para seu valor mínimo, isto é, 1. Caso $s''$ não melhore a solução corrente $s$, o algoritmo executa então o trecho entre as linhas 13 e 15, incrementando o nível de perturbação em uma unidade.

O procedimento de perturbação (linha 6 do Algoritmo 1) funciona como segue. Dada uma solução $s$, sobre ela são aplicadas $nivel + 1$ modificações, sendo que cada modificação consiste em escolher aleatoriamente uma máquina e uma tarefa dessa máquina e, em seguida, posicionar essa tarefa ao final do processamento de outra máquina, também escolhida de forma aleatória. Assim, quando *nivel* é igual a 1, são feitas duas modificações na solução corrente $s$. Quando *nivel* é igual a 2 são feitas três modificações na solução, e, assim por diante. Observa-se que sempre que há melhora na solução corrente, a variável *nivel* retorna ao seu valor mínimo.

A busca local do ILS, linhas 1 e 7 do Algoritmo 1, é feita pelo procedimento *Random Variable Neighborhood Descent* – RVND (Souza et al., 2010). O Algoritmo 2 mostra seu pseudocódigo.

No Algoritmo 2, linha 1, um vetor $v$ de três posições é criado com os números ordenados 1, 2 e 3, cada qual indicando as vizinhanças $N_1$, $N_2$ e $N_3$, respectivamente. Em seguida, na linha 2, esse vetor é embaralhado para


DOI: 10.17648/sbai-2019-111233

permitir que as buscas locais ocorram em ordens diferentes à cada chamada do procedimento. Partindo-se da vizinhança de índice $k = 1$ (linha 3), busca-se o primeiro vizinho de melhora nessa vizinhança $N_{v_k}$. Havendo melhora, esse vizinho passa a ser a solução corrente e o índice da vizinhança retorna a 1 (linha 7); caso contrário, passa-se para a próxima vizinhança (linha 10). Esse procedimento é repetido até que todas as três vizinhanças sejam exploradas consecutivamente sem que ocorra melhora na solução corrente em nenhuma delas, situação que configura o final do método.

**Algoritmo 2:** RVND

```
   entrada: s
   saída  : s
1  v ← [1 2 3];
2  embaralha(v);
3  k ← 1;
4  enquanto (k ≤ 3) faça
5  |  s' ← FirstNeigh(s, N_{v_k});
6  |  se (f(s') < f(s)) então
7  |  |  s ← s'; k ← 1;
8  |  fim
9  |  senão
10 |  |  k ← k + 1;
11 |  fim
12 fim
13 retorne s;
```

## 4. RESULTADOS

O algoritmo ILS foi implementado na linguagem de programação C/C++, utilizando-se a IDE *NetBeans 8.2*. Os testes foram realizados em um computador Dell Inspirion 5558, com processador Intel Core i5-5200U, 2.2 GHz, 8 GB de memória RAM e sistema operacional Ubuntu 16.04 LTS de 64 bits. A função objetivo, dada por (1), considerou $\alpha = \beta = 0.5$, isto é, tanto o *makespan* quanto o consumo total de energia tiveram o mesmo peso.

Para testá-lo, foi usado um conjunto de 30 instâncias de pequeno porte, disponibilizado em Wang et al. (2018). Cada instância foi executada 30 vezes, sendo coletados o tempo médio de execução do algoritmo, o melhor valor de função objetivo retornado pelo ILS, e a média desses valores. Nesses testes, os valores utilizados para $T_{\max}$ foram fornecidos nas próprias instâncias, não sendo necessária a utilização do procedimento para estimá-lo, descrito na Seção 3.3.

Para calibrar o único parâmetro do ILS, o $ILS_{\max}$, foi utilizado o pacote *irace* (López-Ibáñez et al., 2016), aplicado a um subconjunto das instâncias, com diferentes números de tarefas, máquinas e faixas de preço de energia. Foram testados os valores $ILS_{\max} \in \{500, 1000, 1500, 2000, 2500\}$ e a configuração de melhor desempenho foi $ILS_{\max} = 1000$.

Além dos testes com o ILS, utilizou-se também o resolvedor CPLEX, versão 12.63, para resolver as instâncias com o objetivo de avaliar o desempenho do algoritmo ILS. Ao limitar o tempo de processamento por instância a duas horas, o CPLEX foi capaz de encontrar 13 soluções ótimas.

A Tabela 1 reporta os resultados da aplicação do CPLEX e do algoritmo ILS ao conjunto das 30 instâncias. A coluna # indica o número da instância, a coluna $n$ representa o número de tarefas da instância, a coluna $m$ o número de máquinas, e a coluna $T_{\max}$ a quantidade de instantes de tempo disponíveis, isto é, o horizonte de planejamento. A coluna $\text{CPLEX}_{\text{FO}}$ indica o valor da função objetivo gerada pelo CPLEX, e a coluna $\text{CPLEX}_{\text{Tempo}}$ representa o tempo de processamento, em segundos, requerido pelo resolvedor. Na coluna $\text{ILS}_{\text{MelhorFO}}$ é exibido o melhor valor da função objetivo pelo ILS. O valor médio dessa função e o tempo médio demandado pelo ILS são apresentados nas colunas $\text{ILS}_{\text{MédiaFO}}$ e $\text{ILS}_{\text{TempoMédio}}$, respectivamente. Por fim, nas colunas $Gap_{Melhor}$ e $Gap_{Médio}$ são exibidas as diferenças percentuais entre os melhores resultados e os resultados médios, respectivamente, obtidos via ILS, e o valor da solução ótima (ou do limite superior), encontrado pelo CPLEX. Valores percentuais negativos indicam que o ILS obteve resultados iguais ou melhores que o CPLEX. Os valores nos quais o ILS obteve desempenho igual ou melhor que o do CPLEX estão destacadas em negrito.

Pela Tabela 1, observa-se que o ILS gerou soluções com *gaps* da melhor solução muito baixos, e de forma bem rápida, uma vez que as instâncias foram resolvidas em, no máximo, 191,49 segundos em média e a qualidade delas foram iguais ou muito próximas às das soluções geradas pelo CPLEX. Nos casos em que o ILS não alcançou a solução ótima, os valores da função objetivo encontrados foram, em geral, bem próximos aos obtidos pelo CPLEX, variando de 0,46% a 10,39% de diferença. Nas instâncias 1 a 7 e na instância 10, o ILS foi capaz de encontrar as soluções ótimas, enquanto nas instâncias 21, 25, 27 e 29, o ILS obteve resultados melhores que os do CPLEX, chegando a percentuais de melhora em torno de 25%.

## 5. CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, tratou-se o problema de sequenciamento de tarefas em máquinas idênticas, tendo por objetivo a minimização da soma ponderada do *makespan* e do consumo de energia (TEC). Para resolvê-lo, foi desenvolvido um algoritmo baseado na metaheurística ILS, tendo o RVND como método de busca local.

O algoritmo ILS foi testado em instâncias da literatura e seus resultados comparados com aqueles produzidos pelo otimizador CPLEX aplicado à formulação adaptada de Wang et al. (2018). Das 30 instâncias usadas para teste, o ILS foi capaz de encontrar a solução ótima em 8, além de superar o CPLEX em 4 instâncias, em um tempo de processamento baixo, de, no máximo, 191,49 segundos.

Como trabalhos futuros, pretende-se aplicar uma versão *multi-start* (Avci and Topaloglu, 2017) ao algoritmo ILS, além de outras formas de perturbar as soluções ótimas locais com vistas a diminuir a variabilidade das soluções finais.

## AGRADECIMENTOS

Os autores agradecem ao Instituto Tecnológico Vale (ITV), à Universidade Federal de Ouro Preto (UFOP) e às agências de fomento CAPES, FAPEMIG e CNPq pelo apoio ao desenvolvimento deste trabalho.

## REFERÊNCIAS

Avci, M. and Topaloglu, S. (2017). A multi-start iterated local search algorithm for the generalized quadratic


DOI: 10.17648/sbai-2019-111233

**Tabela 1. Resultados do CPLEX e do ILS**

| Instâncias | | | | CPLEX | | ILS | | | Gap | |
|---|---|---|---|---|---|---|---|---|---|---|
| # | $n$ | $m$ | $T_{\max}$ | FO | Tempo (s) | Melhor FO | FO Média | Tempo (s) | Melhor (%) | Médio (%) |
| **1** | **6** | **3** | **50** | 0,1775 | 4,28 | **0,1775** | 0,1776 | 2,72 | **0,00** | 0,08 |
| **2** | **6** | **3** | **80** | 0,0845 | 1,70 | **0,0845** | 0,0864 | 4,43 | **0,00** | 2,33 |
| **3** | **6** | **5** | **50** | 0,1203 | 1,79 | **0,1203** | 0,1238 | 3,76 | **0,00** | 2,97 |
| **4** | **6** | **5** | **80** | 0,0875 | 4,60 | **0,0875** | 0,0875 | 6,65 | **0,00** | 0,05 |
| **5** | **6** | **7** | **50** | 0,1051 | 6,41 | **0,1051** | 0,1053 | 5,06 | **0,00** | 0,15 |
| **6** | **6** | **7** | **80** | 0,1247 | 17,45 | **0,1247** | 0,1376 | 8,78 | **0,00** | 10,38 |
| **7** | **10** | **3** | **50** | 0,2405 | 29,02 | **0,2405** | 0,2426 | 3,43 | **0,00** | 0,84 |
| **8** | **10** | **3** | **80** | 0,1730 | 154,85 | 0,1775 | 0,1834 | 6,25 | 2,64 | 6,02 |
| **9** | **10** | **5** | **50** | 0,3144 | 924,14 | 0,3270 | 0,3371 | 3,88 | 4,01 | 7,21 |
| **10** | **10** | **5** | **80** | 0,1294 | 79,47 | **0,1294** | 0,1358 | 7,71 | **0,00** | 4,92 |
| **11** | **10** | **7** | **50** | 0,1384 | 11,55 | 0,1405 | 0,1490 | 5,17 | 1,56 | 7,68 |
| **12** | **10** | **7** | **80** | 0,1470 | 1567,75 | 0,1476 | 0,1611 | 10,68 | 0,46 | 9,63 |
| **13** | **15** | **3** | **50** | 0,4432 | 7200,00 | 0,4541 | 0,4725 | 18,43 | 2,44 | 6,60 |
| **14** | **15** | **3** | **80** | 0,2341 | 7200,00 | 0,2400 | 0,2521 | 8,01 | 2,53 | 7,66 |
| **15** | **15** | **5** | **50** | 0,3432 | 7200,00 | 0,3789 | 0,3826 | 31,49 | 10,39 | 11,46 |
| **16** | **15** | **5** | **80** | 0,1679 | 7200,00 | 0,1703 | 0,1799 | 10,19 | 1,41 | 7,13 |
| **17** | **15** | **7** | **50** | 0,2103 | 161,36 | 0,2124 | 0,2367 | 7,12 | 1,03 | 12,57 |
| **18** | **15** | **7** | **80** | 0,1551 | 7200,00 | 0,1557 | 0,1765 | 13,19 | 0,44 | 13,82 |
| **19** | **20** | **3** | **50** | 0,4253 | 7200,00 | 0,4370 | 0,4383 | 33,79 | 2,75 | 3,05 |
| **20** | **20** | **3** | **80** | 0,3889 | 7200,00 | 0,4283 | 0,4330 | 9,27 | 10,12 | 11,34 |
| **21** | **20** | **5** | **50** | 0,3027 | 7200,00 | **0,3003** | 0,3446 | 56,07 | **-0,80** | 13,83 |
| **22** | **20** | **5** | **80** | 0,2341 | 7200,00 | 0,2454 | 0,2675 | 11,55 | 4,80 | 14,27 |
| **23** | **20** | **7** | **50** | 0,1819 | 7200,00 | 0,1835 | 0,1988 | 23,34 | 0,89 | 9,27 |
| **24** | **20** | **7** | **80** | 0,2392 | 7200,00 | 0,2578 | 0,2785 | 18,84 | 7,77 | 16,45 |
| **25** | **25** | **3** | **50** | 0,5416 | 7200,00 | **0,4992** | 0,5533 | 52,29 | **-7,83** | 2,16 |
| **26** | **25** | **3** | **80** | 0,3024 | 7200,00 | 0,3040 | 0,3116 | 87,10 | 0,53 | 3,05 |
| **27** | **25** | **5** | **50** | 0,3500 | 7200,00 | **0,2607** | 0,3842 | 87,79 | **-25,52** | 9,77 |
| **28** | **25** | **5** | **80** | 0,2998 | 7200,00 | 0,3105 | 0,3368 | 28,39 | 3,55 | 12,34 |
| **29** | **25** | **7** | **50** | 0,3319 | 7200,00 | **0,3168** | 0,3955 | 191,49 | **-4,56** | 19,15 |
| **30** | **25** | **7** | **80** | 0,3870 | 7200,00 | 0,4160 | 0,4736 | 19,24 | 7,49 | 22,37 |

multiple knapsack problem. *Computers & Operations Research*, 83, 54–65.

Bampis, E., Letsios, D., and Lucarelli, G. (2015). Green scheduling, flows and matchings. *Theoretical Computer Science*, 579, 126–136.

Cota, L.P., Coelho, V.N., Guimarães, F.G., and Souza, M.J.F. (2018). Bi-criteria formulation for green scheduling with unrelated parallel machines with sequence-dependent setup times. *International Transactions in Operational Research*. Disponível em https://doi.org/10.1111/itor.12566.

Ding, J.Y., Song, S., Zhang, R., Chiong, R., and Wu, C. (2016). Parallel machine scheduling under time-of-use electricity prices: New models and optimization approaches. *IEEE Transactions on Automation Science and Engineering*, 13(2), 1138–1154.

Ji, M., Wang, J.Y., and Lee, W.C. (2013). Minimizing resource consumption on uniform parallel machines with a bound on makespan. *Computers & Operations Research*, 40(12), 2970–2974.

Karp, R.M. (1972). Reducibility among combinatorial problems. In R.E. Miller, J.W. Thatcher, and J.D. Bohlinger (eds.), *Complexity of computer computations*, 85–103. Springer.

Li, K., Shi, Y., Yang, S.l., and Cheng, B.y. (2011). Parallel machine scheduling problem to minimize the makespan with resource dependent processing times. *Applied Soft Computing*, 11(8), 5551–5557.

Lourenço, H.R., Martin, O.C., and Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics*, 320–353. Springer.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., and Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3, 43–58.

Mansouri, S.A. e Aktas, E. (2016a). Minimizing energy consumption and makespan in a two-machine flowshop scheduling problem. *Journal of the Operational Research Society*, 67(11), 1382–1394.

Mansouri, S.A., Aktas, E., and Besikci, U. (2016b). Green scheduling of a two-machine flowshop: Trade-off between makespan and energy consumption. *European Journal of Operational Research*, 248(3), 772–788.

Pinedo, M. (2008). *Scheduling: theory, algorithms and Systems*. Springer, 3ª edition.

Sauer, I.L., Tatizawa, H., Salotti, F.A., and Mercedes, S.S. (2015). A comparative assessment of brazilian electric motors performance with minimum efficiency standards. *Renewable and Sustainable Energy Rev.*, 41, 308–318.

Souza, M.J., Coelho, I.M., Ribas, S., Santos, H.G., and Merschmann, L.H.C. (2010). A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European J. of Op. Res.*, 207(2), 1041–1051.

Wang, S., Wang, X., Yu, J., Ma, S., and Liu, M. (2018). Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan. *Journal of Cleaner Production*, 193, 424–440.

Zeng, Y., Che, A., and Wu, X. (2018). Bi-objective scheduling on uniform parallel machines considering electricity cost. *Engineering Optimization*, 50(1), 19–36.


DOI: 10.17648/sbai-2019-111233