Formulação matemática e algoritmo híbrido para o problema de escalonamento em máquinas paralelas idênticas com servidor único de *setup*

João Marcos P. Silva, Ewerton Victor P. P. Teixeira
Centro de Tecnologia - Universidade Federal da Paraíba
Cidade Universitária, s/n - Castelo Branco III, João Pessoa - PB, 58059-900
joaomarkco@gmail.com, ewertonvppt@gmail.com

Anand Subramanian
Centro de Informática - Universidade Federal da Paraíba
Rua dos Escoteiros, s/n - Mangabeira, João Pessoa - PB, 58055-000
anand@ci.ufpb.br

RESUMO

Este artigo aborda o problema de escalonamento em m máquinas paralelas idênticas com servidor único e tempos de *setup* dependentes da sequência. Nessa variante, cujo objetivo considerado é o de minimizar o tempo máximo de término (*makespan*), tem-se disponível apenas uma máquina, indivíduo ou equipe responsável pelas operações de *setup*. Logo, não pode haver conflitos entre essas operações. Para resolvê-lo, é apresentado um modelo matemático baseado em arcos indexados no tempo e um algoritmo híbrido baseado nas meta-heurísticas *Iterated Local Search* (ILS), *Greedy Randomized Adaptive Search Procedure* (GRASP) e na formulação matemática proposta. Experimentos computacionais foram realizados em 150 instâncias com até 100 tarefas, variando entre 2 e 10 máquinas. Os resultados obtidos sugerem que os métodos exato e heurístico desenvolvidos foram capazes de encontrar soluções competitivas quando comparadas com as aquelas determinadas por outras abordagens propostas na literatura.

PALAVRAS CHAVE. Escalonamento com único servidor de *setup*, Formulação matemática com arcos indexados no tempo, Algoritmo híbrido.

Tópicos (AD&GP – PO na Administração e Gestão da Produção, MH – Metaheuristicas, OC – Otimização Combinatória)

ABSTRACT

This paper deals with the parallel machine scheduling problem with a common server and sequence-dependent setup times. In this variant, whose objective considered is to minimize the makespan, there is only a single server, which can be a machine, individual or a team responsible for performing the setup operations. Therefore, there might not be conflicts while conducing them. In order to solve it, an arc-time-indexed formulation is proposed as well as a hybrid algorithm based on the metaheuristics Iterated Local Search (ILS) and *Greedy Randomized Adaptive Search Procedure* (GRASP), and on the proposed model. Computational experiments were carried out on 150 instances involving up to 100 jobs ranging from 2 to 10 machines. The results obtained suggest that the and heuristic methods developed were capable of finding competitive solutions when compared to those determined by other approaches proposed in the literature.

KEYWORDS. Scheduling problem with a common server. Arc-time-indexed formulation. Hybrid algorithm.

Paper topics (AD&GP – OR in Administration and Production Management, MH – Metaheuristics, OC – Combinatorial Optimization)

https://proceedings.science/p/85437?lang=pt-br

1. Introdução

O objeto de estudo desse trabalho é o problema de escalonamento de produção (*production scheduling*), em que o ambiente de produção é composto por máquinas paralelas idênticas e as operações de preparação (*setup*) devem ser realizadas por um servidor único de *setup* e os tempos de *setup* são dependentes da sequência das tarefas. O objetivo é minimizar o *makespan*, ou seja, o tempo máximo necessário para o processamento de todas as tarefas.

Nessa variante, tem-se disponível apenas um indivíduo ou equipe responsável pelas atividades de preparação das máquinas, que recebe a denominação de "servidor único de *setup*". Desta forma, apenas uma máquina pode passar pela operação de preparação por vez. Dentre os ambientes de produção onde ocorre esse tipo de problema podemos destacar a fabricação de componentes automotivos [Koulamas, 1996] e indústrias de impressão [Huang et al., 2010].

O problema de sequenciamento da produção pode ser classificado de acordo com a nomenclatura dos três campos $\alpha|\beta|\gamma$ proposta por [Graham et al., 1979]. Essa nomenclatura possibilita a representação da configuração das máquinas (α), das características das tarefas (β) e da função objetivo a ser minimizada/maximizada (γ) de maneira simplificada. De acordo com essa notação, o problema de escalonamento em m máquinas paralelas com servidor único e tempos de *setup* dependentes da sequência é representado por $P, S_1|s_{ij}|C_{max}$.

Kravchenko e Werner [1997] mostraram que os problemas $P, S_1|s_i = 1|C_{max}$ e $P_2, S_1|s_i = s|IT$ são fortemente NP-difícil e apresentaram um algoritmo pseudo-polinomial para o problema $P_2, S_1|s_i = 1|C_{max}$. [Kravchenko e Werner, 1999] definiram a complexidade computacional para generalizações de alguns problemas de máquinas paralelas com m máquinas e k servidores, onde $1 \leq k < m$. Kravchenko e Werner [2001] também propuseram um algoritmo heurístico para o problema $P, S_1|s_i = 1|\sum C_j$.

Werner e Kravchenko [2010] consideraram o problema com m máquinas paralelas idênticas e múltiplos servidores. Os autores forneceram resultados de complexidade de alguns desses problemas para minimização do *makespan* e minimização do atraso máximo. Além disso, eles propuseram um algoritmo polinomial para o problema com tempos de processamento iguais e tempos de *setup* iguais, e um algoritmo pseudo-polinomial para o problema com tempos de *setup* unitários, m máquinas e $m - 1$ servidores.

Ou et al. [2010] desenvolveram algoritmos heurísticos e com estratégias de *branch-and-bound*, com objetivo de minimizar o $\sum C_j$ no problema de sequenciamento de máquinas paralelas com múltiplos servidores de descarregamento. Kim e Lee [2012] formularam MIPs e um algoritmo heurístico híbrido, que combina as meta-heurísticas busca tabu e *Simulated Annealing* (SA), para resolver o problema $P, S_1|s_i|C_{max}$.

Hasani et al. [2014] abordaram o problema $P_2, S_1|s_i|C_{max}$. Para este, propuseram duas formulações de programação linear inteira mista, baseada na ideia de decompor o escalonamento das tarefas em um conjunto de blocos. Os resultados computacionais mostraram que ambos os modelos possuiram melhor desempenho que todas as heurísticas propostas em [Gan et al., 2012].

Hamzadayi e Yildiz [2016] abordaram o problema $P, S_1|pmtn, s_{ij}|C_{max}$ dinâmico. Propuseram uma heurística SA e regras de despacho com base em abordagens de reprogramação completa. O SA obteve melhor performance.

Hamzadayi e Yildiz [2017] propuseram, para o problema $P, S_1|s_{ij}|C_{max}$, um modelo de programação linear inteira mista e duas meta-heurísticas: SA e *Genetic Algorithm* (GA). De acordo com os resultados reportados, o modelo de programação conseguiu resolver apenas 4 das 30 instâncias propostas, dentre as heurísticas, o GA obteve o melhor desempenho em todas elas. Na literatura, esse foi o único trabalho identificado que lida com o mesmo problema estudado nesse artigo.

Nesse trabalho, para lidar com o dado problema, é proposto um novo modelo matemático baseado em arcos indexados no tempo e um algoritmo híbrido baseado nas meta-heurísticas *Iterated Local Search* (ILS), *Greedy Randomized Adaptive Search Procedure* (GRASP) e na formulação

https://proceedings.science/p/85437?lang=pt-br

proposta neste trabalho.

A disposição do trabalho dá-se da seguinte forma. A Seção 2 introduz o modelo de programação linear inteira desenvolvido. A Seção 3 descreve o algoritmo híbrido proposto. A Seção 4 contém os resultados computacionais. Por fim, na Seção 5 são expostas as conclusões e proposta de trabalhos futuros.

2. Modelo proposto

Formalmente, o problema aqui tratado pode ser definido da seguinte forma. Seja $J = \{1, \ldots, n\}$ o conjunto de tarefas a serem escalonadas em um conjunto de $M = \{1, \ldots, m\}$ máquinas paralelas idênticas de forma a minimizar o *makespan*. Cada tarefa $i \in J$ possui um tempo de processamento p_i. Um tempo de *setup* s_{ij} é considerado caso uma tarefa $j \in J$ seja processada imediatamente após a tarefa i na mesma máquina. Contudo, como já mencionado, as atividades de *setup* não podem ser realizadas de forma simultânea. Assim sendo, é possível que haja a ocorrência de tempos ociosos nas máquinas para que uma solução seja viável.

O modelo matemático proposto, denominado ATIF (*Arc-Time Indexed Formulation*), é baseado na formulação de Silva et al. [2018] originalmente desenvolvida para uma variante do problema $1|s_{ij}|\sum w_j T_j$. A formulação assume um horizonte de tempo de 0 a T. As variáveis binárias $x^t_{i,j}, i \neq j$, indicam que a tarefa i termina em um instante de tempo igual a $t - s_{ij}$ e a tarefa j inicia seu processamento no tempo t, na mesma máquina, e nenhum outro *setup* pode ser realizado entre $t - s_{ij}$ e t; ou quando a tarefa $i = j$ termina em um instante de tempo menor ou igual a $t - 1$ e o tempo de *setup* da próxima tarefa começará a ser computado em um instante de tempo maior ou igual a t.

As variáveis x^t_{jj} representam o tempo ocioso entre $t-1$ e t. Essas variáveis são necessárias porque a operação de *setup* da próxima tarefa da sequência, caso alguma outra máquina esteja passando por essa operação no instante t, só poderá ser realizada em um tempo maior que $t + p_j$ e, consequentemente, a tarefa j iniciará seu processamento em um tempo maior que t, permanecendo ociosa até lá. Variáveis x^t_{jj}, portanto, armazenam as informações referentes a tarefa j, enquanto esta permanece ociosa, permitindo que o tempo de *setup* da tarefa j para a próxima, da sequência, seja computado corretamente. Observe que esse tempo de inatividade ocorre antes do processamento da tarefa j.

Na Figura 1, o diagrama de Gantt mostra uma solução viável para uma instância exemplo com 6 tarefas e 2 máquinas, onde os retângulos sombreados e claros representam os tempos de *setup* e processamento, respectivamente. No ATIF, a solução é dada pelas seguintes variáveis não nulas: x^0_{04}, x^3_{42}, x^6_{25}, x^8_{50}, x^0_{03}, x^1_{33}, x^4_{31}, x^5_{11}, x^8_{16} e x^9_{60}. Percebe-se que as tarefas 3 e 1, na máquina $M2$, permanecem ociosas por uma unidade de tempo cada, para evitar conflitos de *setup* de suas tarefas sucessoras com as tarefas processadas na máquina $M1$.

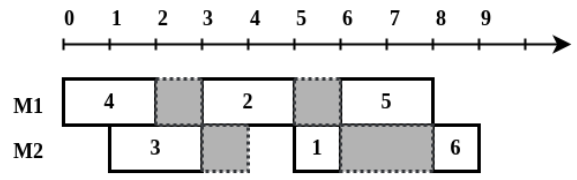


Figura 1: Diagrama de Gantt de uma instância exemplo com 6 tarefas e 2 máquinas.

Seja $J_+ = J + \{0\}$, onde a tarefa *dummy* 0 representa a primeira e a última tarefa em cada solução viável, com $p_0 = 0$ e $s_{0,j} = s_{j,0} = 0, \forall j \in J$ e m igual o número de máquinas. Cada variável é associada a um arco em um grafo, onde cada nó possui um índice de tarefa e tempo, do mesmo modo como descrito por [Pessoa et al., 2010]. Nesse grafo, o arco associado com a variável x^t_{ij} parte do nó $(i, t - p_i - s_{i,j})$ para o nó (j, t) e a sequência em cada máquina é representada

https://proceedings.science/p/85437?lang=pt-br

por um caminho, que começa no nó $(0, 0)$ e termina em um nó $(0, t)$. A Figura 2 representa o grafo associado ao exemplo da Figura 1. Assim sendo, a formulação pode ser escrita da seguinte maneira.

$$\text{Minimizar} \quad C_{max} \tag{1}$$

Sujeito a:

$$\sum_{\substack{i \in J_+, \\ i \neq j}} \sum_{t=p_i+s_{ij}}^{T-p_j-1} x_{ij}^t = 1 \qquad \forall j \in J \tag{2}$$

$$\sum_{\substack{j \in J, \\ t=s_{0j}}} x_{0j}^t = m \tag{3}$$

$$\sum_{\substack{j \in J_+, \\ i \neq j, \\ t \geq p_j+s_{ji}, \\ t<T-p_i}} x_{ji}^t + \sum_{\substack{j \in J_+, \\ i=j, \\ t \geq 1+s_{ji}, \\ t<T-p_i}} x_{ji}^t - \sum_{\substack{j \in J_+, \\ i \neq j, \\ t+p_i+s_{ij}+p_j<T}} x_{ij}^{t+p_i+s_{ij}} - \sum_{\substack{j \in J_+, \\ i=j, \\ t+1+p_j<T}} x_{ij}^{t+1} = 0 \qquad \forall i \in J; t = 0, .., T-1 \tag{4}$$

$$C_{max} \geq \sum_{\substack{i \in J, \\ i \neq j}} \sum_{t=p_i+s_{ij}}^{T-p_j-1} (t + p_j) x_{ij}^t \qquad \forall j \in J \tag{5}$$

$$\sum_{(i,j,t') \in D_t} x_{ij}^{t'} \leq 1 \qquad t = 0, \ldots, T-1 \tag{6}$$

$$x_{ij}^t \in \{0, 1\} \qquad \forall i, j \in J_+, i \neq j; \quad t = p_i + s_{ij}, \ldots, T - p_j - 1. \tag{7}$$

$$x_{ii}^t \in \{0, 1\} \qquad \forall i \in J_+; t = 0, \ldots, T - p_i - 1. \tag{8}$$

A função objetivo (1) minimiza o *makespan*. As restrições (2) indicam que uma tarefa deve ser executada apenas uma vez. A restrição (3) garante que a quantidade de tarefas que iniciam uma sequencia é igual ao número de máquinas. As restrições (4) asseguram a preservação de fluxo no grafo previamente definido. As restrições (5) determinam o *makespan*. Em (6), dado D_t o conjunto de variáveis $x_{ij}^{t'}$, que quando ativas implicarão na ocorrência de *setup* no tempo t, no máximo uma tarefa poderá realizar *setup* naquele instante. As restrições (7) e (8) definem o domínio das variáveis de decisão.

3. Algoritmo heurístico

O algoritmo híbrido proposto, aqui denominado HILS-CSvr, utiliza os princípios da meta-heurística ILS [Lourenço et al., 2003] para resolver uma versão relaxada do problema que não considera as restrições de conflito de *setup*. Uma versão restrita do modelo descrito na Seção 2 ou um procedimento heurístico baseado na fase construtiva da meta-heurística GRASP [Feo e Resende, 1995] é, então, empregado para gerar soluções viáveis para o problema original a partir da solução relaxada.

O HILS-CSvr é apresentado no Algoritmo 1. O método recebe como parâmetros: I_{max}, que representa o número máximo de *restarts*, onde em cada um dos quais uma solução inicial (s) é gerada; I_{ILS}, que corresponde ao numero máximo de iterações ILS consecutivas sem melhorias; α, que corresponde a um parâmetro utilizado no procedimento heurístico que gera uma solução viável

https://proceedings.science/p/85437?lang=pt-br

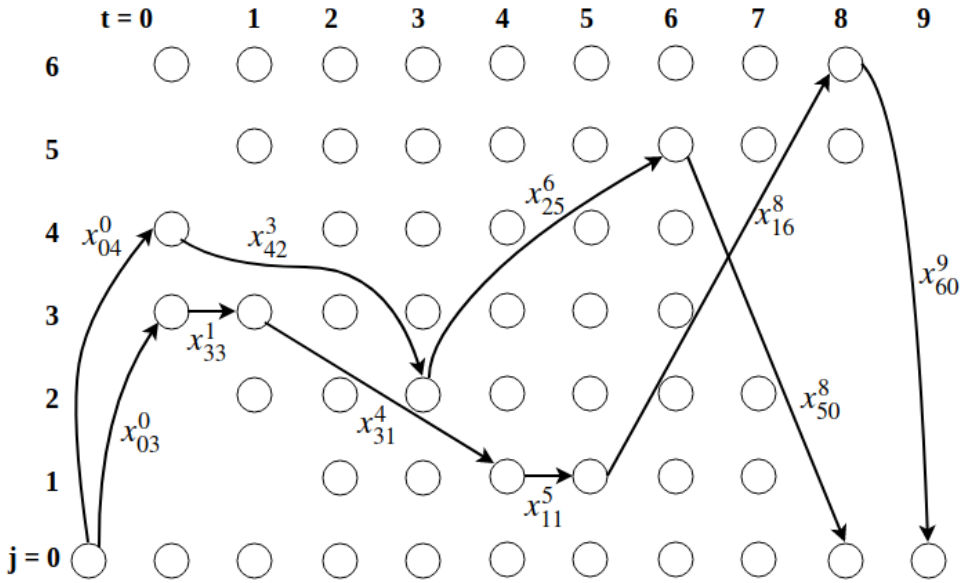


Figura 2: Representação dos caminhos em um grafo da solução ATIF para a instância exemplo.

Algoritmo 1: HILS-CSvr

```
1  Procedimento HILS-CSvr(I_max, I_ILS, α, I_GSV)
2  f* ← ∞
3  P = ∅;  // P é o conjunto de soluções já visitadas
4  para i ← 1,...,I_max faça
5      s ← GeraSolucaoInicial(α)
6      s' ← s
7      Iter_ILS ← 0
8      enquanto Iter_ILS < I_ILS faça
9          s ← BuscaLocal(s)
10         se f(s) ≤ f(s') e f(s) < f* então
11             se f(s) < f(s') então
12                 Iter_ILS ← 0
13             s' ← s
14             se s' ∉ P então
15                 P ← P ∪ s'
16                 se instância é de médio/grande porte então
17                     f'' ← GeraSolucaoViavel(s', α, I_GSV)
18                 senão
19                     se f* = ∞ então
20                         T ← f(s') + n  // T é o horizonte de tempo
21                     senão
22                         T ← f*
23                     f'' ← ATIF(s', T)
24                 se f'' < f* então
25                     s* ← s'  f* ← f''
26         s ← Perturbacao(s)
27         Iter_ILS ← Iter_ILS + 1
28 retorna s*
29 fim HILS-CSvr
```

dada uma solução relaxada, nesse mesmo procedimento, o parâmetro I_{GSV} tem função semelhante ao I_{ILS}.

A solução inicial (linha 5) é melhorada por meio de um procedimento de busca local (linha 9) combinado com o mecanismo de perturbação (linha 26). Sempre que a busca local retornar uma

https://proceedings.science/p/85437?lang=pt-br

solução s com o valor objetivo $f(s) \leq f(s')$, a melhor solução corrente (s') é atualizada (linhas 10–25). Uma vez que isso ocorre, um método para gerar soluções viáveis é utilizado caso a solução já não tenha sido visitada (linhas 14–25). Por fim, a heurística retorna a melhor solução viável s^* entre todas as iterações.

A solução viável é obtida de duas maneiras: ATIF (linha 23) ou o procedimento `GeraSolucaoViavel` (linha 17). A chamada desses métodos depende do porte da instância (linha 16), nesse trabalho considerou-se instâncias de médio/grande porte aquelas com $n \geq 49$ ou $m \geq 10$. Ambos recebem a sequências pré-definidas, bastando realizar os ajustes para evitar os conflitos de *setup*. Para o ATIF, descrito na Seção 2, uma vez que a sequência está pré-definida, é declarado no modelo apenas as variáveis que preservam a sequência, resultando em uma diminuição considerável da quantidade de variáveis e no tempo de execução do modelo. Ainda para o ATIF, em máquinas com até de 4 tarefas alocadas é permitida a permutação na sequência.

O procedimento `GeraSolucaoViavel`, descrito pelo Algoritmo 2, é empregado para gerar a solução viável das instâncias de médio/grande porte, pois nessas instâncias o modelo ATIF requer um tempo CPU relativamente elevado . Os termos usados nesse algoritmo estão descritos na Tabela 1. O algoritmo recebe como parâmetros a solução relaxada s' e α, que controla o nível de gulosidade e aleatoriedade durante a geração da solução viável. I_{GSV} é semelhante ao I_{max} do ILS. As tarefas são inseridas na solução uma a uma, de acordo com a sequência pré-definida s'_k, em s', para cada máquina k (linhas 7–26).

A cada iteração, é calculado o tempo de conclusão da próxima tarefa da sequência de cada máquina (linhas 8–18). Se a tarefa j' é a primeira da sequência na máquina k, seu tempo de conclusão será igual ao seu tempo de processamento $p^k_{j'}$ (linha 10), caso contrário, são analisadas as possibilidades de conflitos na sua inclusão, caso haja, seu tempo de início é ajustado de modo a evitá-los (linhas 11–18) e então o tempo de conclusão é computado (linha 18). De modo semelhante a fase construtiva do GRASP, uma máquina dentre aquelas com o menor tempo de término é aleatoriamente escolhida (linhas 19–20), e então o tempo em CT_k é atualizado (linha 21). Uma vez que todas as tarefas já tenham sido computadas na solução viável, o *makespan* será igual ao maior tempo de conclusão dentre todas as máquinas (linha 27). Por fim, é retornado o valor objetivo f'' da solução viável.

Tabela 1: Termos usados no algoritmo `GeraSolucaoViavel`

Termos	Definição
TP_t	Se $TP_t \neq 0$ indica que no instante t está sendo realizado *setup*, $(t = 1, \ldots, T_{max})$.
CT_k	O tempo de conclusão na máquina k, $\forall k \in M$
LCR	Lista de candidatos restrita
M_a	O conjunto de máquinas com tarefas não alocadas
ST	Armazena temporariamente o instante de tempo que uma tarefa poderá começar sem ocasionar conflitos de *setup*
$p^k_{j'}$	O tempo de processamento da atual tarefa na sequência da máquina k
$d^k_{jj'}$	O tempo de *setup* da atual tarefa j' ao suceder a última tarefa j, alocadas na máquina k
$g(k)$	Armazena o tempo de término temporário da máquina k

3.1. Geração da Solução Inicial

O procedimento para geração da solução inicial funciona da seguinte forma. Inicialmente m tarefas pertencentes a J são escolhidas, aleatoriamente, e atribuídas uma a cada máquina. Em seguida, procedimentos de inserção são aplicados até que todas as tarefas tenham sido adicionadas a solução. A cada chamada da função de construir uma solução inicial, o método decide de forma aleatória a estratégia (sequencial ou paralela) e critério de inserção (mais barata ou mais próxima) a serem adotados. A estratégia de inserção sequencial consiste em considerar uma máquina diferente a cada iteração do procedimento construtivo, ao passo que a estratégia paralela leva em conta todas as máquinas simultaneamente ao avaliar a inserção mais barata ou próxima. Seja J' o conjunto de

https://proceedings.science/p/85437?lang=pt-br

Algoritmo 2: GeraSolucaoViavel

```
1  Procedimento GeraSolucaoViavel(s', α, I_GSV)
2  f'' ← ∞
3  enquanto iter_GSV < I_GSV faça
4      TP_t ← 0,  t = 1, ..., T_max  // T_max é um valor suficientemente grande
5      CT_k ← 0,  ∀k ∈ M
6      M_a ← k,  ∀k ∈ M
7      enquanto M_a ≠ ∅ faça
8          para k ∈ M_a faça
9              se CT_k = 0 então
10                 g(k) ← p^k_{j'}
11             senão
12                 repita
13                     ST ← CT_k
14                     para t ← ST, ..., ST + d^k_{jj'} faça
15                         se TP_t ≠ 0 então
16                             CT_k ← t
17                 até CT_k = ST;
18                 g(k) ← ST + d^k_{jj'} + p^k_{j'}
19         Insira em LCR os α × |M_a| menores valores g(k), k ∈ M_a
20         Selecione, aleatoriamente, um elemento k ∈ LCR
21         CT_k ← g(k)
22         Limpe LCR
23         Atualize TP
24         s'_k ← s'_k − {j'}
25         se s'_k = ∅ então
26             M_a ← M_a − {k}
27     C_max ← max{CT_k | k ∈ M}
28     se C_max < f'' então
29         f'' ← C_max
30         iter_GSV ← 0
31     iter_GSV ← iter_GSV + 1
32 retorna f''
33 fim GeraSolucaoViavel
```

tarefas escalonadas e J'' o conjunto das tarefas não escalonadas. A inserção mais barata de uma tarefa $k \in J''$ entre duas tarefas $i \in J'$ e $j \in J'$ é dada por $\min\{s_{ik} + s_{kj} - s_{ij} + p_k \mid i, j \in J,' k \in J''\}$. Já a inserção mais próxima é dada por $\min\{s_{ik} + p_k \mid i \in J', k \in J''\}$.

3.2. Busca Local

A fase de busca local é baseada no procedimento *Variable Neighborhood Descent* (VND) Mladenovic e Hansen [1997] com ordem aleatória na escolha das vizinhanças. Foram considerados dois grupos de estruturas de vizinhança: Vizinhança entre Máquinas (VE) e intra Máquinas (VI). Na primeira, os movimentos de busca são realizados entre duas máquinas, enquanto que na segunda, os movimentos são realizados entre as tarefas da própria máquina. Nesse caso, seleciona-se uma estrutura de vizinhança VE aleatoriamente, caso a solução seja melhorada, todas as vizinhanças VE serão reconsideradas para uma nova seleção, caso contrário, a estrutura é eliminada de VE. A cada melhora da solução retornada pelas estruturas VE, é também realizada uma busca local utilizando as estruturas VI, que ocorre do mesmo modo que o descrito para VE. Uma vez que VE esteja vazia, é realizada uma última busca intra máquinas.

3.2.1. Estruturas VE

As seguintes estruturas VE foram implementadas.

1. *Cross* — A sequência de duas máquinas são divididas em dois segmentos, em seguida, o segmento inicial de cada uma é unido no segmento final da outra, no mesmo ponto de corte, como mostrado na Figura 3(a).

https://proceedings.science/p/85437?lang=pt-br

2. *Reallocation* (k) — Um número k de tarefas adjacentes é retirado de uma máquina e realocado em outra, com $0 \leq k \leq 3$. Na Figura 3(b) é exemplificado o movimento para $k = 1$.

3. *Swap* (k_1, k_2) — Um número k_1 de tarefas adjacentes é trocado de máquina com um número k_2 de tarefas adjacentes de outra máquina, com $1 \leq k_1 \leq 3$ e $1 \leq k_2 \leq k_1$. A Figura 3(c) exemplifica o funcionamento dessa vizinhança para $k_1 = 3$ e $k_2 = 2$.

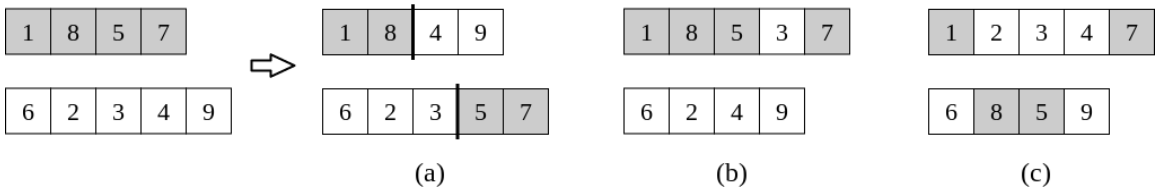


Figura 3: Vizinhanças VE: *Cross* (a), *Reallocation*(1) (b) e *Swap*$(3, 2)$ (c).

3.2.2. Estruturas VI

As seguintes estruturas VI foram consideradas.

1. *Reinsertion* (k) — k tarefas adjacentes são reinseridas em outra posição na máquina, com $1 \leq k \leq 4$. Na figura Figura 4(a) é mostrado um exemplo para essa vizinhança com $k = 2$.

2. *Block Reverse* — A ordem de processamento de um bloco de tarefas é invertida, conforme ilustrado (para um bloco de tamanho 3) na Figura 4(b).

3. *Job Exchange* — Duas tarefas são trocadas de posição na sequência, assim como o demonstrado na Figura 4(c).

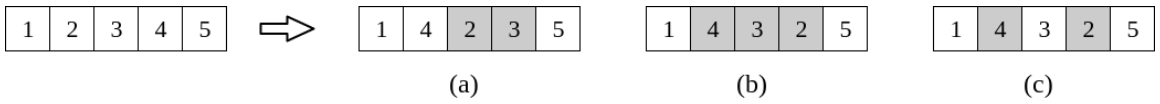


Figura 4: Vizinhanças VI: *Reinsertion*(2) (a), *Block Reverse* (b), *Job Exchange* (c).

3.3. Mecanismos de Perturbação

De acordo com Lourenço et al. [2003], o movimento de perturbação é responsável por transformar uma solução de boa qualidade em um bom ponto de partida para se realizar a busca local, evitando assim ótimos locais. Em oposição aos movimentos realizados na busca local, os movimentos de perturbação não possuem como objetivo a melhoria da solução. As perturbações adotadas pelo algoritmo são as seguintes.

1. *Reallocation* $(1, 1)$ — Duas máquinas são selecionadas aleatoriamente assim como uma tarefa em cada. Em seguida são reinseridas em posições arbitrárias da outra máquina. O movimento é realizado múltiplas vezes.

2. *Swap* $(1, 1)$ — Duas máquinas são selecionadas aleatoriamente assim como uma tarefa em cada. Em seguida, as duas tarefas são trocadas. O movimento é realizado múltiplas vezes.

https://proceedings.science/p/85437?lang=pt-br

4. Experimentos Computacionais

Como na literatura, apenas [Hamzadayi e Yildiz, 2017] estudaram o problema $P, S_1|St_{sd}|C_{max}$, foram solicitadas as instâncias utilizadas em seus testes, porém, não houve resposta. Por isso, tendo em vista a comparação justa dos resultados, as instâncias usadas nesse trabalho foram geradas de acordo com a descrição fornecida pelos autores. O MILP, SA e GA, propostos, também foram reimplementados.

O HILS-CSvr, SA e GA foram implementados em linguagem de programação C++ e os experimentos foram executados em um Intel Core i7, com 3,4 GHz, 16 GB de memória RAM e sistema operacional Linux Ubuntu 16.04. O ATIF e o MILP, sob o mesmo ambiente computacional, foram executados no *solver* CPLEX 12.7 considerando 8 *threads*. Porém, para a chamada do ATIF pelo HILS-CSvr foi utilizada apenas uma *thread*.

As instâncias foram geradas com seis diferentes níveis de máquinas, $m = 2$, 3, 4, 5, 7 e 10. E o número de tarefas em cinco níveis, $n = 3m$, $4m$, $5m$, $7m$ e $10m$. Tendo assim 30 grupos de instâncias. Para cada grupo foram geradas 5 instâncias, totalizando 150 instâncias. Os tempos de processamento foram uniformemente gerados no intervalo (10,100). Os tempos de *setup* também foram gerados uniformemente no intervalo (5,50), o que corresponde a 20–40% do tempo de processamento.

4.1. Parâmetros das abordagens

Para a HILS-CSvr, em relação ao parâmetro I_{ILS}, adotou-se o mesmo valor que [Subramanian et al., 2014], com $I_{ILS} = 4 \times n$. De acordo com os referidos autores, estes valores parecem fornecer uma boa relação entre qualidade da solução e o tempo de CPU. Para o número máximo de *restarts*, $I_{max} = 20$. Em relação aos parâmetros do procedimento `GeraSolucaoViavel`, tomou-se $\alpha = 0,5$ e $I_{GSV} = 1000$, pois, considerando a rapidez a cada chamada de tal procedimento pelo HILS-CSvr, esses valores forneceram melhores resultados.

Para o SA e GA reimplementados, considerou-se os parâmetros propostos por Hamzadayi e Yildiz [2017]. Porém, tendo em vista que esses algoritmos possuem um tempo de CPU muito pequeno, decidiu-se manter o tempo de execução igual a média dos tempos de execução do HILS-CSvr para cada grupo de instâncias. Permitindo assim, um número maior de gerações para o GA, enquanto que no SA, o parâmetro de temperatura sofre um *restart* quando atinge a temperatura final, mantendo seu decréscimo até o tempo estabelecido ser atingido.

Do mesmo modo que Hamzadayi e Yildiz [2017], para obter uma avaliação consistente do desempenho dos algoritmos heurísticos e para reduzir o efeito de sua natureza estocástica, todos os problemas foram resolvidos dez vezes. Os modelos exatos tiveram seu tempo de execução limitados a 18000 segundos.

O horizonte de tempo T, do ATIF, é inicializado com o melhor LB (*Lower Bound*) encontrado pelo HILS-CSvr acrescido de uma unidade. Para a versão simplificada do ATIF, usado pelo HILS-CSvr para viabilizar a solução, o horizonte de tempo adotado é demonstrado no Algoritmo 1 (linhas 19-23) e seu tempo de execução, em cada chamada, foi limitado a 30 segundos.

4.2. Resultados

Na Tabela 2 estão sumarizados os resultados agregados dos grupos de instâncias obtidos em cada abordagem adotada. Para os métodos exatos são reportados a média (Avg.) dos *gaps* retornados pelo CPLEX. Os métodos heurísticos tem seus resultados dados pela média (Avg.), mínimo (Min.) e máximo (Max.) dos *gaps* (%) entre o melhor *Lower Bound* (LB) cohecido e o *Upper Bound* (UB) obtido pelo respectivo método para cada instância do grupo. O melhor LB conhecido é dado pelo $min\{LB_{HILS-CSvr}, LB_{SA}, LB_{GA}\}$ de cada instância. São reportados, também, a média dos tempos de execução dos modelos exatos e das heurísticas (H).

https://proceedings.science/p/85437?lang=pt-br

Tabela 2: Média dos resultados agregados de cada método para os grupos de intâncias

m	n	Gap (%)												CPU (s)		
		ATIF	MILP	HILS-CSvr			SA			GA				ATIF	MILP	H
				Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.				
2	6	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**		0,48	0,99	0,22
	8	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	**0,00**	0,21	**0,00**	0,37		2,82	67,79	0,09
	10	**0,00**	2,89	**0,00**	**0,00**	**0,00**	0,61	**0,00**	1,76	4,04	2,47	5,21		6,52	12834,39	0,08
	14	**0,00**	80,02	0,03	**0,00**	0,24	1,51	0,31	2,39	5,48	3,70	7,30		226,71	18000,00	0,28
	20	**0,00**	-	0,12	**0,00**	0,37	2,81	1,67	3,64	7,92	6,47	9,45		10506,81	18000,00	1,00
3	9	**0,00**	**0,00**	0,08	**0,00**	0,27	**0,00**	**0,00**	**0,00**	0,35	**0,00**	0,91		0,93	423,73	0,45
	12	**0,00**	61,74	0,30	0,20	0,33	1,10	**0,00**	2,01	4,37	2,00	6,24		15,05	18000,00	0,40
	15	**0,00**	74,08*	0,04	**0,00**	0,36	1,63	0,66	2,87	6,69	4,34	8,58		102,17	14057,55	0,44
	21	**0,00**	-	0,55	**0,00**	1,09	2,74	1,90	3,45	7,51	5,23	10,20		8203,60	18000,00	1,99
	30			0,53	**0,00**	1,09	3,08	2,27	3,81	8,11	5,67	9,99				4,84
4	12	**0,00**	46,27	0,01	**0,00**	0,10	**0,00**	**0,00**	**0,00**	1,57	**0,00**	3,55		4,32	18000,00	3,85
	16	**0,00**	64,88*	0,28	**0,00**	0,76	1,38	0,43	2,51	6,91	3,51	9,12		58,89	11360,09	1,09
	20	**0,00**	-	0,62	**0,00**	1,44	2,61	0,93	3,73	9,71	6,36	13,16		2459,81	18000,00	1,68
	28			0,91	**0,00**	1,49	2,94	2,18	3,55	8,42	5,47	10,78				11,01
	40			0,53	**0,00**	0,95	3,76	2,95	4,31	8,53	6,19	10,65				24,83
5	15	**0,00**	56,13*	0,12	**0,00**	0,63	0,10	**0,00**	0,42	4,14	1,37	6,71		14,08	15092,65	8,32
	20	**0,00**	71,13*	0,74	**0,00**	1,65	2,28	1,30	3,26	8,29	5,66	10,99		604,50	18000,00	3,23
	25			1,09	**0,00**	1,90	2,85	1,87	3,82	9,84	6,65	13,34				4,07
	35			1,02	**0,00**	1,59	3,61	2,36	4,36	9,17	6,26	11,75				18,68
	50			0,70	**0,00**	1,33	4,58	3,66	5,34	12,02	9,80	14,25				9,61
7	21	**0,00**	-	1,09	0,39	2,08	0,28	**0,00**	0,69	4,53	1,77	6,62		210,26	18000,00	106,03
	28			1,14	0,16	2,07	1,25	**0,08**	2,07	7,63	4,99	10,35				39,54
	35			1,18	**0,00**	2,12	2,74	1,17	3,48	8,94	6,01	12,41				34,79
	49			1,84	**0,00**	2,98	3,58	2,29	4,50	12,19	9,61	14,57				10,71
	70			1,38	**0,00**	2,36	4,98	3,42	5,96	13,03	9,97	15,84				21,89
10	30			10,96	6,05	15,15	1,77	**0,00**	3,05	9,02	5,01	13,64				7,81
	40			10,71	7,16	13,06	1,78	**0,00**	2,97	10,29	6,21	14,02				11,76
	50			9,55	6,48	11,84	1,76	**0,00**	2,92	10,95	6,95	14,40				16,23
	70			7,34	5,59	8,86	1,11	**0,00**	2,23	10,42	7,10	13,60				30,20
	100			3,80	1,67	5,08	1,66	**0,10**	3,01	10,74	7,89	13,44				59,58

https://proceedings.science/p/85437?lang=pt-br

Como pode ser notado na Tabela 2, comparando os menores valores indicados, dentre os métodos heurísticos o HILS-CSvr se mostrou mais eficiente, retornando as melhores soluções em 22 dos 30 grupos de instâncias. Percebe-se que os grupos de instâncias nos quais tal método não obteve melhores resultados, são aqueles no qual o procedimento `GeraSolucaoViavel` foi chamado para realizar a construção da solução viável. Nessas instâncias o SA obteve melhor performance. No total, o SA e o GA obtiveram melhores ou iguais resultados que o HILS-CSvr em 14 e 4 grupos de instâncias, respectivamente.

Em relação aos métodos exatos, os testes foram realizados para instâncias com até 21 tarefas. Os grupos de instâncias marcadas com (*), indica que naquele grupo pelo menos uma instância não retornou uma solução ou, devido a restrições de memória da máquina utilizada,, o CPLEX interrompeu o teste. Para esse último caso, foram reportados os valores dos *gaps* (%) retornados na última iteração.

Analisando a média dos *Gaps* (%), o MILP obteve a solução ótima de todas as instâncias de apenas 3 grupos. Para os demais problemas uma solução ótima ou viável não foi encontrada dentro do limite de tempo. Em contrapartida, o ATIF proposto obteve melhor performance, sendo capaz de retornar o ótimo de 15 grupos.

5. Conclusões

Neste trabalho foram propostos uma abordagem exata e um algoritmo heurístico para a resolução do problema sequenciamento de m máquinas paralelas com servidor único e tempos de *setup* dependentes da sequência. O primeiro método consiste em formulação de arcos indexados no tempo (do inglês, ATIF). O algoritmo heurístico, chamado de HILS-CSvr, é um método híbrido que combina o ILS, que fornecem soluções relaxadas, com o modelo ATIF reduzido e um procedimento heurístico semi-guloso que retornam a solução viável não-relaxada.

Todos os métodos foram testados em 150 instâncias, divididas em 30 grupos, cada um com uma configuração de número de tarefas e máquinas distintas. Os resultados foram comparados com os resultados dos métodos exato e heurísticos propostos por [Hamzadayi e Yildiz, 2017]. Dentre os métodos exatos, o ATIF mostrou melhor desempenho em termos do número de instâncias que foi capaz de resolver. Dentre os métodos heurísticos, o HILS-CSvr obteve os melhores resultados em 73,3% das instâncias, nas demais 26,7%, percebe-se que a maioria são instâncias com 10 máquinas.

Como trabalhos futuros, sugere-se o desenvolvimento de procedimentos heurísticos de viabilização mais eficientes, capazes de gerar melhores soluções para instâncias de médio/grande porte.

Agradecimentos

Os autores agradecem a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro concedido.

Referências

Feo, T. A. e Resende, M. G. C. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6(2):109–133. ISSN 1573-2916.

Gan, H.-S., Wirth, A., e Abdekhodaee, A. (2012). A branch-and-price algorithm for the general case of scheduling parallel machines with a single server. *Computers & Operations Research*, 39(9):2242–2247.

Graham, R. L., Lawler, E. L., Lenstra, J. K., e Kan, A. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. In *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, p. 287–326. Elsevier.

https://proceedings.science/p/85437?lang=pt-br

Hamzadayi, A. e Yildiz, G. (2016). Hybrid strategy based complete rescheduling approaches for dynamic m identical parallel machines scheduling problem with a common server. *Simulation Modelling Practice and Theory*, 63:104–132.

Hamzadayi, A. e Yildiz, G. (2017). Modeling and solving static m identical parallel machines scheduling problem with a common server and sequence dependent setup times. *Computers & Industrial Engineering*, 106:287–298.

Hasani, K., Kravchenko, S. A., e Werner, F. (2014). Block models for scheduling jobs on two parallel machines with a single server. *Computers & Operations Research*, 41(1):94–97.

Huang, S., Cai, L., e Zhang, X. (2010). Parallel dedicated machine scheduling problem with sequence-dependent setups and a single server. *Computers & Industrial Engineering*, 58(1): 165–174.

Kim, M.-Y. e Lee, Y. H. (2012). MIP models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server. *Computers & Operations Research*, 39(11):2457–2468.

Koulamas, C. P. (1996). Scheduling two parallel semiautomatic machines to minimize machine interference. *Computers & Operations Research*, 23(10):945–956.

Kravchenko, S. A. e Werner, F. (1997). Parallel machine scheduling problems with a single server. *Mathematical and Computer Modelling*, 26(12).

Kravchenko, S. A. e Werner, F. (1999). Scheduling on Parallel Machines with Single and Multiple Servers.

Kravchenko, S. A. e Werner, F. (2001). A heuristic algorithm for minimizing mean flow time with unit setups. *Information Processing Letters*, 79(6):291–296.

Lourenço, H. R., Martin, O. C., e Stützle, T. (2003). *Iterated Local Search*, p. 320–353. Springer US, Boston, MA. URL `https://doi.org/10.1007/0-306-48056-5{\_}11`.

Mladenovic, N. e Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Ou, J., Qi, X., e Lee, C.-Y. (2010). Parallel machine scheduling with multiple unloading servers. *Journal of Scheduling*, 13(3):213–226.

Pessoa, A., Uchoa, E., de Aragão, M. P., e Rodrigues, R. (2010). Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation*, 2(3-4):259–290.

Silva, Y. L. T. V., Subramanian, A., e Pessoa, A. A. (2018). Exact and heuristic algorithms for order acceptance and scheduling with sequence-dependent setup times. *Computers & Operations Research*, 90:142–160.

Subramanian, A., Battarra, M., e Potts, C. N. (2014). An Iterated Local Search heuristic for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times. *International Journal of Production Research*, 52(9):2729–2742.

Werner, F. e Kravchenko, S. A. (2010). Scheduling with multiple servers. *Automation and Remote Control*, 71(10):2109–2121.

https://proceedings.science/p/85437?lang=pt-br