EXACTA Engenharia de Produção eISSN 1983-9308

Artigos

https://doi.org/10.5585/exactaep.2021.11626

UM ALGORITMO HÍBRIDO BUSCA TABU/ILS PARA O PROBLEMA DE SEQUENCIAMENTO DA PRODUÇÃO EM AMBIENTE OPEN SHOP

A HYBRID TABU SEARCH/ILS ALGORITHM FOR OPEN SHOP SCHEDULING PROBLEM

Recebido em: 13 fev. 2019
Aprovado em: 23 maio 2020
Versão do autor aceita publicada online: 23 maio 2020
Publicado online: 12 maio 2021

Como citar esse artigo - American Psychological Association (APA):

Costa Lima, V. H. L., & Prata. B. de A. (2021, out./dez.). Um algoritmo híbrido busca tabu/ILS para o problema de sequenciamento da produção em ambiente open shop. *Exacta*. *19*(4), 729-744. https://doi.org/10.5585/exactaep.2021.11626.

EXACTA Engenharia de Produção eISSN 1983-9308

Artigos

https://doi.org/10.5585/exactaep.2021.11626

Check for updates

UM ALGORITMO HÍBRIDO BUSCA TABU/ILS PARA O PROBLEMA DE SEQUENCIAMENTO DA PRODUÇÃO EM AMBIENTE OPEN SHOP

A HYBRID TABU SEARCH/ILS ALGORITHM FOR OPEN SHOP SCHEDULING PROBLEM

Vitor Hugo Lopes Costa Lima[1]

Bruno de Athayde Prata[2]

[1] Graduação
Universidade Federal do Ceará – UFC.
Fortaleza, Ceará – Brasil.
vitorhugo_mec@alu.ufc.br

[2] Doutorado
Universidade Federal do Ceará – UFC.
Fortaleza, Ceará – Brasil.
baprata@ufc.br

Recebido em: 13 fev. 2019
Aprovado em: 23 maio 2020

Resumo: O *Open Shop Scheduling Problem* (OSPP) é um ambiente no qual a produção é realizada por m máquinas, em que todas as máquinas podem realizar todas as n tarefas existentes e cada máquina possui um tempo específico para realizar cada tarefa. Tendo em vista que o OSSP é NP-hard, uma meta-heurística híbrida busca tabu/iterated local search (ILS) é proposta. Foram realizados testes computacionais em 140 instâncias disponíveis na literatura. Foram avaliadas as regras de prioridade LPT, SPT, LAPT e LTRPOM como algoritmos para construção de uma solução inicial. A função objetivo utilizada foi a minimização do makespan e o desvio percentual relativo foi a medida de desempenho adotada. Os resultados computacionais apontam para a competitividade da meta-heurística proposta nas instâncias avaliadas.

Palavras-chave: Programação da Produção. Meta-heurísticas. Otimização Combinatória.

Abstract: The open shop scheduling problem is a production environment in which the jobs are processed in m machines where all the machines can process any of the n jobs and each machine presents a specific processing time for a given job. Since the OSSP is NP-complete, a tabu search/iterated local search (ILS) metaheuristic is proposed. We carried out computational experiments on 140 literature problem instances in order to test the performance of the proposed algorithm. The well-known priority rules LPT, SPT, LAPT and LTRPOM were evaluated as the initial solution for the proposed meta-heuristic. The objective function adopted was minimizing the makespan, and we used the relative deviation as performance criteria. Computational results point to a very well performance of the proposed meta-heuristic for the tested instances.

Keywords: Production Programming. Meta-heuristics. Combinatorial Optimization.

1 Introdução

Problemas de programação da produção são de suma relevância em sistemas nos quais se tem um conjunto de tarefas a serem executadas com uma quantidade escassa de recursos, os quais devem ter um agendamento da sua utilização para uma melhor realização das tarefas em um dado horizonte de planejamento.

Nas últimas décadas, problemas de sequenciamento em ambientes flow shop, job shop e open shop têm sido amplamente estudados na literatura, tendo em vista as suas importâncias do ponto de vista prático para a resolução de problemas reais, segundo Pinedo (2016).

Pode-se observar que a grande maioria dos trabalhos publicados na área de programação da produção tem focado em variantes de problemas em ambientes flow shop e job shop, enquanto problemas em ambiente open shop tem recebido uma menor atenção como afirmam Reza Hejazi e Saghafian (2005), Cheng, Gen e Tsujimura (1996) e Anand e Panneerselvam (2015). Neste contexto, existem diversas variantes de problemas em ambiente open shop que ainda requerem um estudo aprofundado.

O problema de programação em ambiente *open shop* (em inglês, *Open Shop Scheduling Problem* – OSSP), proposto por Gonzalez e Sahni (1976), consiste em um conjunto de n tarefas as quais têm de ser processadas em m máquinas, consumindo uma quantidade de tempo nesse processamento.

No caso de instâncias com apenas uma máquina (m=1), a obtenção da solução ótima pode ser obtida sem grande dificuldade. No caso de instâncias com 2 máquinas (m=2), existem algoritmos polinomiais que garantem a otimalidade da solução obtida (Gonzalez & Sahni, 1976; Pinedo, 2016). Para instâncias de maior porte ($m \geq 3$), o OSSP é NP-Completo (Garey & Johnson, 1974).

Diante do exposto, a proposição de algoritmos aproximativos é de grande importância para a obtenção de boas soluções para o OSSP em tempo computacional admissível. A obtenção de soluções de boa qualidade em tempos acessíveis é fundamental para a aplicação destas em ambientes industriais, notadamente na resolução de problemas reais.

Heurísticas e meta-heurísticas são abordagens que têm sido utilizadas em OSSP, em que os seguintes trabalhos podem ser destacados. Liaw (1999) apresenta uma busca tabu que usa blocos de operações sobre um caminho crítico como estrutura de vizinhança. Adicionalmente, um procedimento eficiente de avaliação de soluções é apresentado. Liaw (2000), Prins (2000) e Hosseinabadi, Vahidi, Saemi, Sangaiah e Elhoseny (2018) apresentam algoritmos genéticos híbridos para o OSSP, utilizando operadores específicos para o problema em estudo. Sha e Hsu (2008) apresentam um novo algoritmo de otimização por enxame de partículas com uma forma inovadora de codificação e de tipo de movimento das partículas. Low e Yeh (2009) propõem um algoritmo genético híbrido para o problema de programação da produção em ambiente *open shop* considerando tempos de *setup* das máquinas e

de remoção dos trabalhos. Naderi, Nafati e Yazdani (2012) apresentam uma meta-heurística eletromagnética para o *open shop* considerando tempos de *setup* dependentes da sequência de processamento. Bai e Tang (2013) apresentam uma heurística *on-line* para o *open shop* com datas de entrega. Naderi, Mousakhani e Khalili (2013) propõem um algoritmo imunológico para o *open shop* multiobjetivo. Liaw (2014) apresenta uma heurística para a minimização de tarefas com atrasos em um *open shop* com possibilidade de interrupção das tarefas. Naderi e Zandieh (2014) apresentam um algoritmo genético para o *open shop* com restrições de espera (*no-wait*). Azadeh, Farahani e Kalantari (2015) apresentam meta-heurísticas multiobjetivo, baseadas em algoritmos genéticos e em otimização por enxame de partículas, para o problema *open shop* com manutenção preventiva de múltiplos processadores. Bai, Zhang e Zhang (2016) apresentam uma meta-heurística evolução diferencial para o problema *open shop* flexível, no qual existem máquinas paralelas em cada estágio de produção.

Anand e Panneerselvam (2015) apresentam um levantamento bibliográfico sobre o OSSP, classificando os trabalhos presentes na literatura conforme a medida de desempenho (função objetivo) e o método de resolução adotado, identificando as áreas promissoras de pesquisa no problema em estudo. Uma das sugestões apresentadas pelos autores citados é o desenvolvimento de algoritmos híbridos para o OSSP.

Lal, Vishnu, Haq e Jeyapaul (2019) implementam um algoritmo híbrido para o OSSP. Uma meta-heurística incorporando elementos de SA (*Simmulated Annealing*) e DFA (*Discrete Firefly Algorithm*), que segundo os autores vem apresentando bons resultados em outros trabalhos, e a métrica de otimização utilizada foi o "tempo médio de fluxo", que calcula a média do tempo total de produção (*makespan*) mais o tempo ocioso, para cada tarefa.

Trabalhos recentes sobre o OSSP têm se focado mais em suas variantes. Pempera e Smutinicki (2018) apresentam um algoritmo TS-PS (derivado da busca tabu) para problema *open shop cyclic*, onde o sequenciamento das operações é otimizado para um sistema contínuo, as operações são organizadas de forma que repitam todo o sequenciamento da mesma forma continuamente. O autor ainda relata a pouca quantidade de trabalhos referente à variante *open shop cyclic* na literatura. Abreu, Cunna, Prata e Framinan (2019) apresentam um algoritmo genético híbrido para OSSP com tempo de *setup*. Nessa variante do *open shop* antes que uma tarefa seja executada por uma máquina há um certo tempo de "ajuste" entre a tarefa executada e a tarefa que será executada. O algoritmo proposto por Abreu *et al* (2019) é um algoritmo genético, cujo a solução inicial é gerada pela junção de BICH (*Bouded Isertion Construtive Heuristic*) e MIH (*Mínimal Idleness Heuristic*). Os autores também relatam o problema do número de soluções redundantes que aparecem, e reforçam a necessidade de filtrar essas soluções. Meddourene, Bouzoia, Abbou e Farraa (2019) apresentam um algoritmo híbrido integrando SA e DATC (*Distributed Arrival Time Control*) para um problema *open shop* JIT (*Just-in-time*) onde cada operação possui um tempo pré-determinado para se completar (data de vencimento), o tempo de execução não

pode nem ser menor nem maior. Dessa forma os autores inserem penalidades de atraso e adiantamento para avaliar a qualidade das soluções.

Sendo assim, este artigo tem como objetivo apresentar um algoritmo híbrido busca tabu/ILS para o problema de sequenciamento da produção em ambiente *open shop*. O artigo é estruturado em outras quatro seções, descritas a seguir. Na segunda seção, é definido o problema em estudo. Na terceira seção, são apresentados os algoritmos propostos para resolução da variante abordada. Na quarta seção, são apresentados os resultados computacionais em testes realizados com instâncias disponíveis na literatura. Por fim, na quinta seção, são apresentadas as principais conclusões seguidas de sugestões para estudos futuros.

2 Definição do problema

Como citado anteriormente ambiente de produção *flow shop e job shop* são amplamente estudados. No *flow shop* cada tarefa tem que ser processada em cada máquina e todas devem seguir o mesmo sequenciamento de passagem nas máquinas. O *job shop* é um problema de sequenciamento muito parecido com o *flow shop*, porém cada tarefa possui uma rota pré-definida das máquinas pelo quais vai passar (Pinedo, 2016).

Diferentemente dos problemas de sequenciamento em ambientes *flow shop* e *job* shop, no OSSP não existem restrições de ordenamento no sequenciamento das tarefas nas máquinas e cada máquina pode executar todas as tarefas, de modo que uma tarefa pode ser processada mais de uma vez em uma mesma máquina. Deve-se observar que o processamento das tarefas ocorre em momentos distintos, ou seja, uma dada tarefa não pode ser processada por mais de uma máquina simultaneamente.

O OSSP possui diversas aplicações práticas industriais, como, por exemplo, nas áreas de moldagem de plástico, processos químicos, produção de alimentos e ambientes avançados de fabricação, conforme apresentado por Naderi, Fatemi Ghomi, Aminnayeria e Zandieh (2010) e Naderi e Zandieh (2014). O OSSP possui também aplicações no setor de serviços, como, por exemplo, na manutenção de veículos (Gonzalez & Sahni, 1976, p. 666).

A função objetivo mais usual para o OSSP é a minimização do *makespan*. O *makespan* é o tempo de término do projeto, ou seja, a diferença entre o início e o término do processamento dos trabalhos. Sejam *n* trabalhos, tem-se um vetor de tempos de término das tarefas $C = \{C1, C2,..,Cn\}$, em que *Cj* é o término do trabalho *j*, o *makespan* $Cmax = \max\{C\}$.Tendo em vista a complexidade do OSSP, refletida pela grande quantidade de soluções a serem avaliadas, a obtenção de um limite inferior (*lower bound*) mostra-se de grande importância. Seja um problema de programação em ambiente *open shop* com *m* máquinas e *n* trabalhos, pode-se definir um vetor p_i como a soma das execuções (sem pausas) das

operações de processamento do trabalho *i*, assim como um vetor z_j com a carga de trabalho da máquina *j* (tempo de processamento acumulado, sem pausas). O limite inferior *LB* pode ser definido como o maior valor dentre os elementos dos vetores p_i e z_j, conforme ilustrado na Equação (1).

$$\mathrm{LB} = \max\left(\max\left(\sum_{i=1,n} p_{ij}\right), \max\left(\sum_{j=1,m} p_{ij}\right)\right) \quad (1)$$

Na Tabela 1, é apresentado um exemplo com três máquinas e três trabalhos, com o cálculo do limite inferior (no caso, *LB*=1327). Na Figura 1, é ilustrado o gráfico de Gantt de uma solução viável para a instância em foco, apresentando um *makespan* igual 1399. Nesta solução, a máquina M_1 processa os trabalhos J_3, J_2 e J_1; a máquina M_2 processa os trabalhos J_1, J_3 e J_2 e a máquina M_3 processa os trabalhos J_2, J_1 e J_3.

Tabela 1

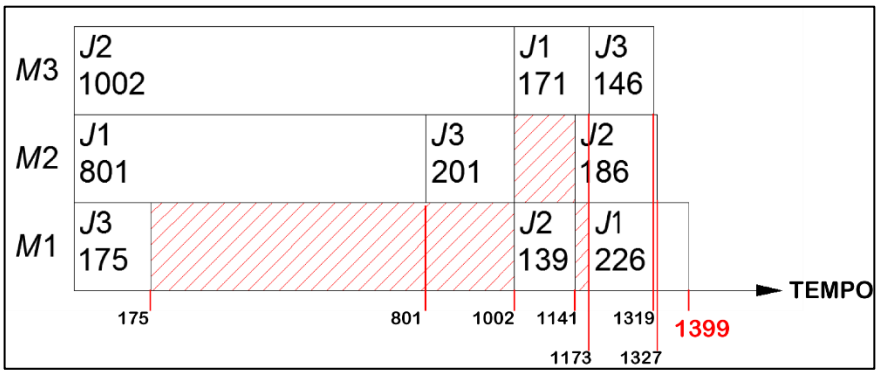
Tempos de processamento para uma instância com três máquinas e três trabalhos

	M_1	M_2	M_3	p_i
J_1	226	801	171	1198
J_2	139	186	1002	1327
J_3	175	201	146	522
z_j	540	1188	1319	**1327**

Fonte: Autores.

Figura 1

Um exemplo de uma solução viável para o open shop *clássico*



Fonte: Autores.

3 Algoritmo proposto

3.1 Considerações iniciais

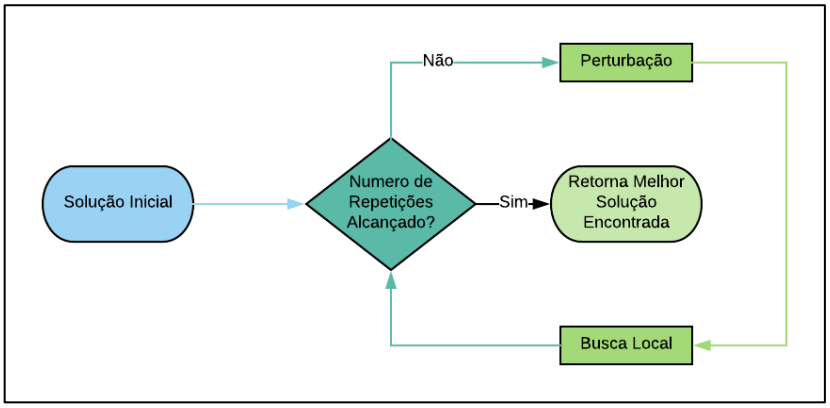
Tendo em vista a natureza combinatória do problema em estudo, expressa pela grande quantidade de soluções viáveis a serem avaliadas, foi desenvolvida uma meta-heurística para solucionar o mesmo. Foi buscada uma meta-heurística com baixa dependência de parâmetros e com fácil implementação computacional. *Iterated Local Search* (ILS) é uma das meta-heurísticas mais simples de serem implementadas, com uma baixa variação em seu desempenho em função dos parâmetros utilizados. Os conceitos básicos do ILS foram propostos independentemente por Martin, Otto e Felten (1991) e Johnson e McGeogh (1997). Lourenço, Martin e Stützle (2002) apresentaram o ILS como um método que incorporou essas ideias, unificando a nomenclatura existente sobre essa classe de algoritmos.

O ILS é um algoritmo de busca de vizinhança que se baseia no conceito de geração de uma sequência de soluções durante o processo de busca. Um algoritmo construtivo gera estas soluções, o qual encontra ótimos locais. Em seguida, o algoritmo realiza um *loop* iterativo no qual as soluções ótimas locais são aleatoriamente perturbadas, aplicando-se procedimentos de busca local nas soluções modificadas. A perturbação não pode ser fraca a ponto de gerar uma solução similar à sua predecessora, tampouco forte o bastante para gerar uma solução aleatória. Se o movimento de perturbação é bem projetado, o ILS apresenta a capacidade de escapar de ótimos locais, guiando a busca para regiões promissoras do espaço de soluções viáveis do problema.

Como a vizinhança do problema *open shop* tende a apresentar uma grande quantidade de soluções, percebeu-se que o processo de busca local do ILS poderia ser aprimorado se ele incorporasse elementos da meta-heurística Busca Tabu, do inglês *Tabu Search – TS* (Glover & Laguna, 1997). No TS, cria-se uma lista de soluções já visitadas, reduzindo a possibilidade de curto prazo do algoritmo avaliar uma solução previamente avaliada. Deste modo, a busca local torna-se mais eficiente, visto que cada algoritmo traz uma melhor performance onde o outro não trabalha muito bem. Por exemplo, na meta-heurística ILS é possível aumentar sua eficiência aumentando alguns fatores de busca, porém isso acarreta um maior custo computacional. Com o uso da Busca Tabu, a busca realizada pelo algoritmo é reduzida, consequentemente se torna possível aumentar os fatores de busca sem muito custo computacional. Na Figura 2, é ilustrado o fluxograma da meta-heurística híbrida ILS + TS proposta para solução do problema de sequenciamento em ambiente *open shop*.

Figura 2

Fluxograma da meta-heurística híbrida proposta



Fonte: Autores.

Como descrito na figura o algoritmo entra com uma solução inicial, depois essa solução de entrada passa pelo loop de perturbação e busca local até que atinja o número limite de repetições, no final a melhor solução encontrada no loop é retornada. A codificação do algoritmo proposto foi matricial, na qual cada linha representa a sequência de trabalhos processada naquela máquina na sua ordem numérica (a linha *i* representa a *i*-ésima máquina). Essa codificação se assemelha um pouco com a ordem das operações utilizada por Pempera e Smutnicki (2018), onde existe uma permutação $\pi_k = (\pi_k(1), \pi_k(2),..., \pi_k)$ operações em O^k, onde $k \in M$, e M é um conjunto de máquinas {1, 2,...,m}. A matriz de uma solução de uma instância 4x4 é representada na Tabela 2.

Tabela 2

Tempos de processamento para uma instância com quatro máquinas e quatro trabalhos

	1ª Tarefa	2ª Tarefa	3ª Tarefa	4ª Tarefa
Máquina 1	2	1	4	3
Máquina 2	3	2	1	4
Máquina 3	1	4	3	1
Máquina 4	4	1	3	2

Fonte: Autores.

3.2 Geração da solução inicial

Para geração de uma solução inicial, foram implementadas quatro regras de prioridade apresentadas por Pinedo (2012) para a resolução do *open shop*, a saber: LPT (*Longest Processing Time*), SPT (*Shortest Processing Time*), LAPT (*Longest Alternative Processing Time*) e LTRPOM (*Longest Total Remaining Processing on Other Machines*). As regras de prioridade LPT e SPT são bem conhecidas na

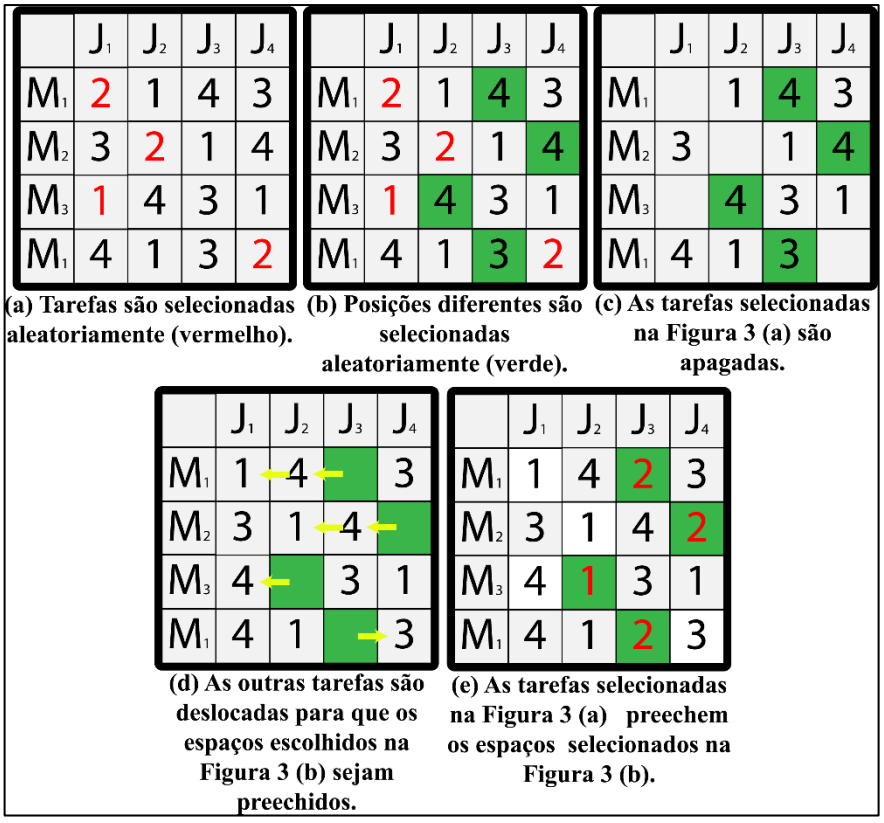
área de programação da produção e consistem, respectivamente, em ordenar as tarefas em ordem não-decrescente e não-crescente de tempos de processamento. A regra LAPT ordena as tarefas alocando aquelas que possuem maior tempo de processamento em outra máquina, primeiro. Já a regra LTRPOM ordena as tarefas alocando aquela que tiver o maior somatório de tempos em todas as outras máquinas, primeiro.

3.3 Perturbação

A perturbação desempenha um papel de suma relevância para o desempenho do algoritmo ILS. Após experimentos computacionais preliminares, foi adotado o procedimento ilustrado na Figura 3. Na matriz de solução é efetuada uma perturbação em cada linha da matriz (representando uma dada máquina), de modo que uma tarefa aleatória é inserida em uma posição aleatória. Esta perturbação se mostrou eficaz, no sentido de permitir que o algoritmo escape de ótimos locais e que o processo de busca migre para outras regiões promissoras do espaço de soluções viáveis do problema.

Figura 3

Representação da Perturbação usada



Fonte: Autores.

3.4 Busca local

A busca local consiste em um movimento de rotação em cada sequência da ordem de tarefas para cada máquina. Depois que uma rotação acontece, ocorre outra sequência de rotações para cada máquina. Para cada solução gerada é verificado se ela já foi encontrada (lista Tabu). Caso ainda não tenha sido encontrada, é calculado seu custo (*makespan*) e verificado se é melhor do que a melhor solução obtida até então. No final da verificação de cada máquina, a solução é restaurada a sua forma original.

Essa é uma incorporação do algoritmo de Busca Tabu na nossa busca local, onde o resultado das rotações é guardado dentro de uma lista mesmo que seu resultado seja pior, e se na mesma busca local essa solução for encontrada novamente ela é ignorada. Porém, como diz Liaw (1999), a lista deve ser grande o suficiente para evitar soluções redundantes e pequena suficiente para não guardar soluções demais.

O pseudocódigo da busca local é apresentado no Algoritmo 1. A seguir a Figura 4 ilustra o movimento de rotação.

Algoritmo 1

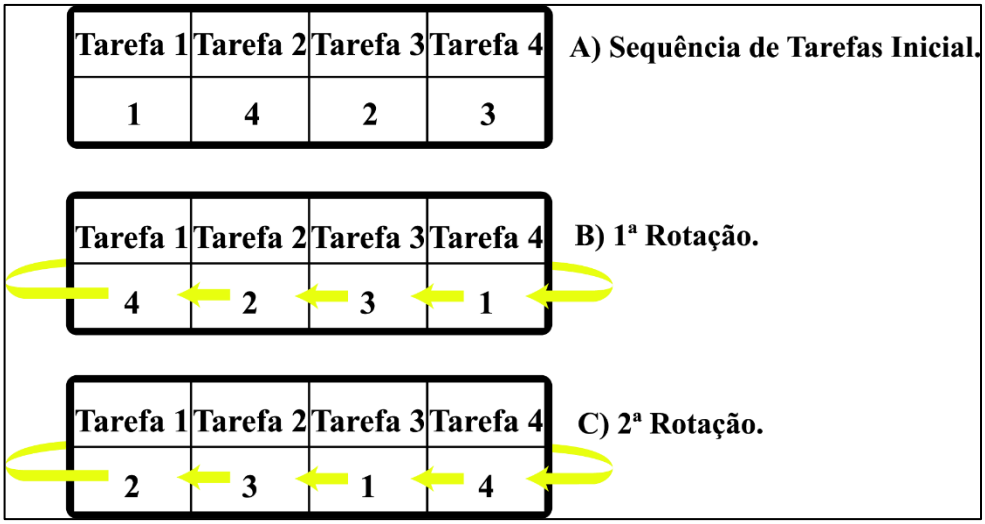
Pseudocódigo da Busca Local Tabu implementada

```
Algoritmo Busca Local Tabu
Recebe Lista Tabu, solução s
Repetir para M máquinas
      Rotaciona a sequência de tarefas da solução s, gerando solução s*
      Se nova solução s* existe dentro da Lista Tabu
            Vá para próxima máquina
      Senão
            Se Custo (s*) é menor, é nova melhor solução
            Solução entra na lista Tabu
      Restaure a solução inicial
Retorne a melhor solução
```

Fonte: Autores.

Figura 4

Representação do movimento de rotação



Fonte: Autores.

4 Resultados computacionais

Os testes computacionais foram realizados em um computador com processador Intel Core i5-7200U 2.5GHZ/3.1 GHz com 8GB RAM, 64 bits. O sistema operacional utilizado foi o Windows 10. O algoritmo foi escrito em linguagem de programação Python utilizando a IDE *Intel Distribution for Python**. A meta-heurística proposta foi testada nas 60 instâncias propostas por Taillard e nas 80 instâncias propostas por Guéret e Prins.

Para cada instância, a meta-heurística foi executada 10 vezes, sendo registrados o melhor *makespan*, o *makespan* médio, o pior *makespan*, a solução inicial e o tempo médio para processar uma instância.

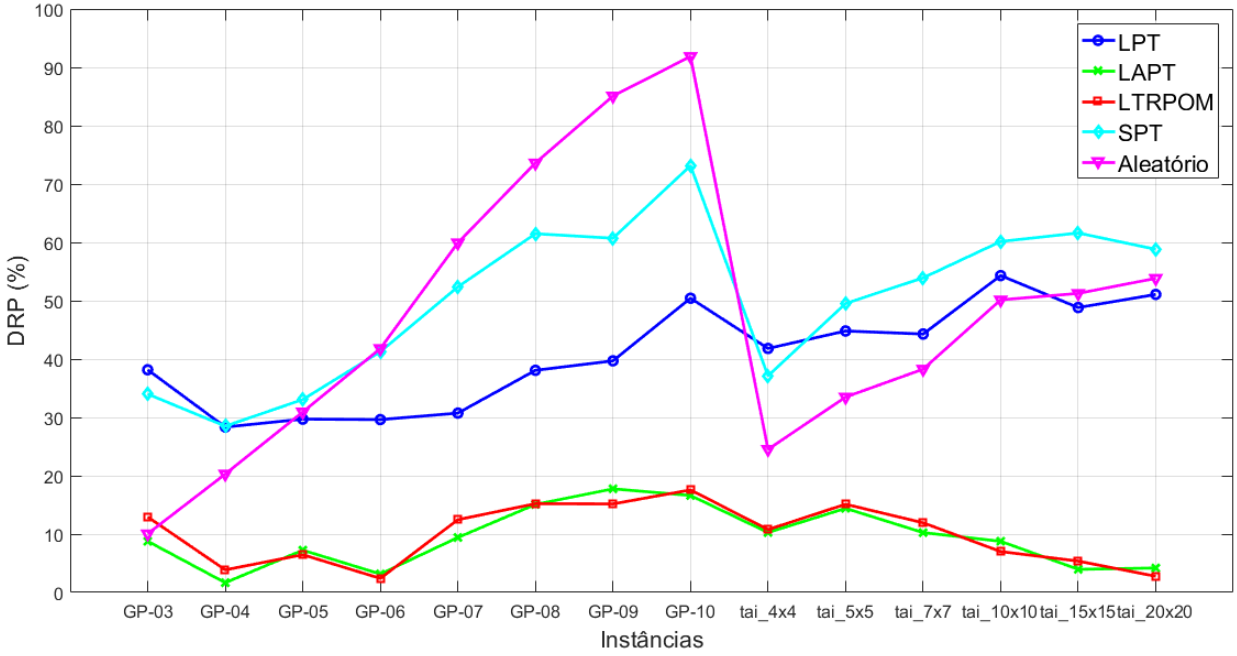
Como estatística para avaliação dos resultados obtidos pelos métodos em análise, foi considerado o Desvio Percentual Relativo (DPR) do resultado de cada método em relação às melhores soluções conhecidas reportadas por Sha e Hsu (2008). Na Figura 6 são reportados os resultados do algoritmo híbrido proposto expressos em termos do DPR médio das melhores soluções obtidas pelo algoritmo proposto.

Foram realizados experimentos computacionais preliminares com as regras de prioridade LPT, SPT, LAPT e LTRPOM, objetivando definir qual destas seria adotada como solução inicial das meta-heurísticas propostas. Em testes realizados com as 80 instâncias de Guéret e Prins e com as 60 instâncias de Taillard, foi comparado o DPR entre o *makespan* obtido por cada regra e as melhores soluções conhecidas, onde os resultados médios são ilustrados na Figura 5. Os tempos computacionais

requeridos para a execução das regras de prioridade são desprezíveis, totalizando menos de um minuto para o processamento de todas as instâncias em cada heurística. Na Figura 6 são apresentados os resultados médios do algoritmo híbrido proposto, expressos em termos de DPR, para cada conjunto de instâncias utilizado.

Figura 5

DPR médio das regras de prioridade com instâncias Guéret and Prins e Taillard



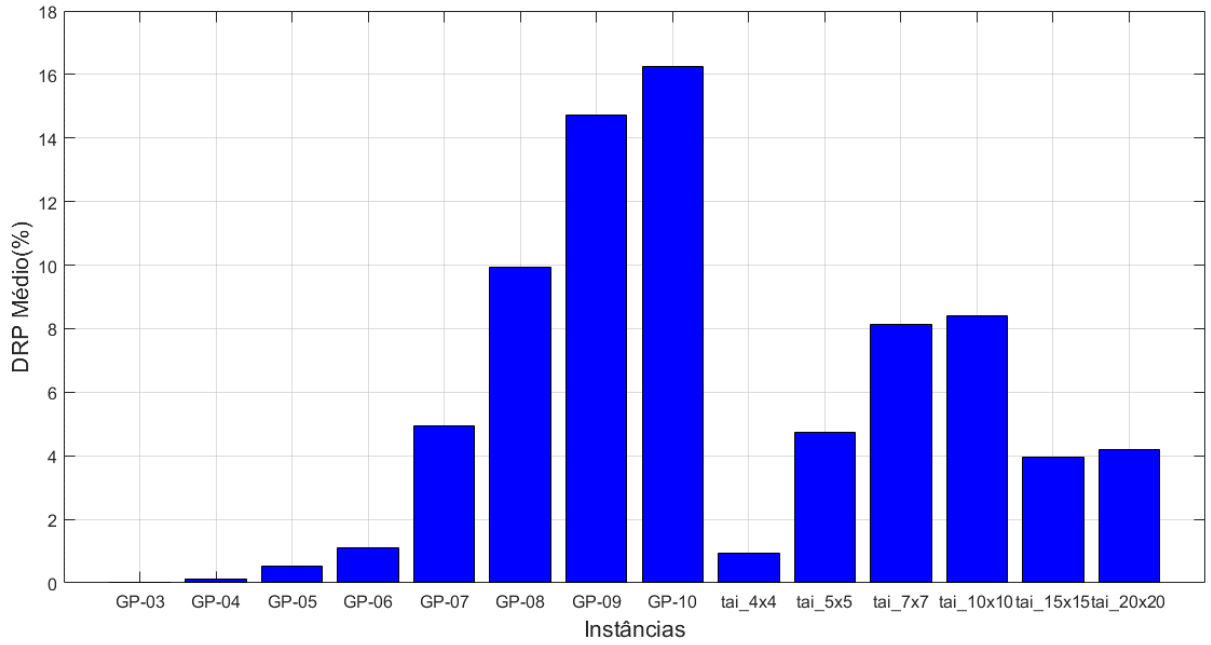
Fonte: Autores.

A média dos DPR's para as regras de prioridade LPT, SPT, LAPT e LTRPOM foram de 46,2%, 53,8%, 10,9% e 11,4%, respectivamente. Deste modo, a heurística LAPT foi escolhida para gerar a solução inicial, por apresentar um *makespan* médio menor em comparação com as melhores soluções conhecidas.

Como critério de parada da meta-heurística híbrida proposta, foi utilizado um número máximo de iterações igual à 1500*n*. Nos experimentos computacionais realizados, cada instância foi testada 10 vezes, sendo aferido o tempo médio de execução. Na Figura 7 são apresentados os tempos de processamento médio para cada conjunto de instâncias.

Figura 6

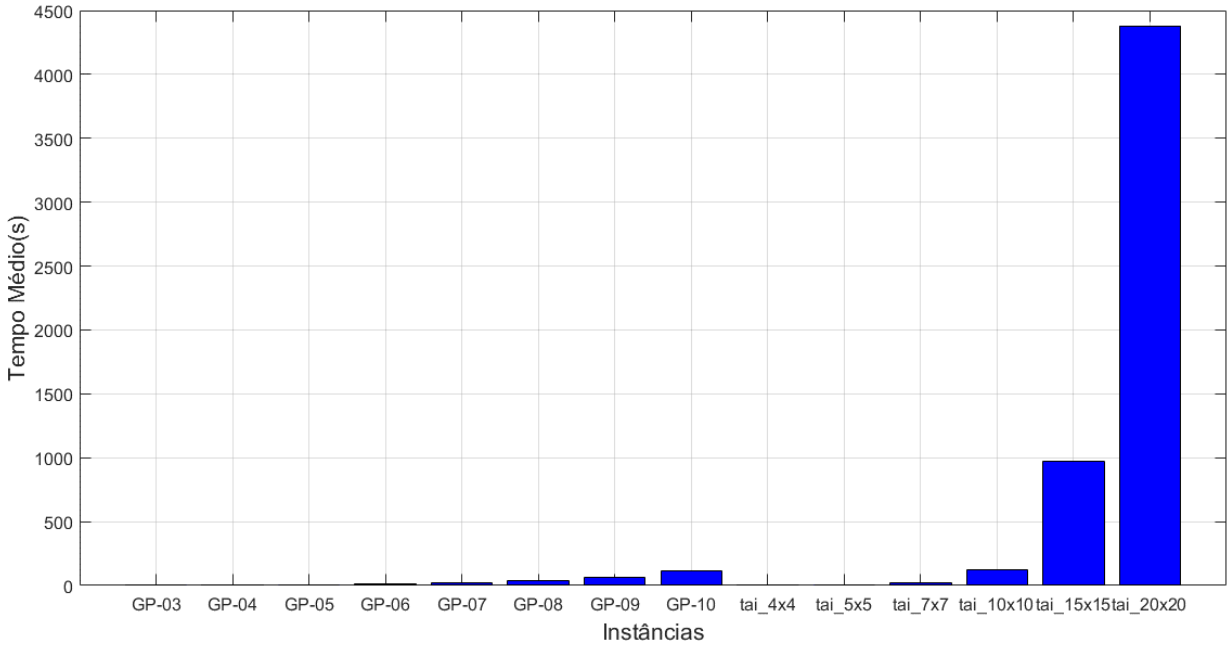
DPR médio do ILS/Tabu com instâncias Guéret and Prins e Taillard



Fonte: Autores.

Figura 7

Tempo médio de processamento (custo computacional) para instâncias Guéret and Prins e Taillard



Fonte: Autores.

Deve-se observar que, para as instâncias maiores (problemas de dimensões 15x15 e 20x20), a busca local não conseguiu melhorar as soluções iniciais construídas pela regra de prioridade LAPT. Entretanto, mesmo sem haver melhoria, o algoritmo proposto conseguiu chegar a resultados próximos às melhores soluções conhecidas (em alguns casos, com valores inferiores à 1%). Pode-se constatar que, para as instâncias maiores, as regras de prioridade tornam-se mais competitivas, pois obtém soluções de boa qualidade com tempo computacional desprezível. Ademais, a dificuldade em solucionar esses conjuntos de instâncias também foi encontrada com as demais abordagens presentes na literatura revisada.

No que concerne aos testes computacionais realizados nas instâncias de Guéret e Prins, pode-se destacar que a meta-heurística híbrida proposta conseguiu obter as melhores soluções conhecidas em diversas instâncias, tendo obtido *gaps* de baixa magnitude, notadamente para os conjuntos de problemas de porte inferior a 7x7. Estes resultados apontam para a capacidade da abordagem proposta solucionar satisfatoriamente problemas práticos encontrados em sistemas industriais. De acordo com a experiência dos autores, em ambientes industriais de manutenção (uma das aplicações clássicas do ambiente *open shop*), não é usual se ter um número elevado de estações de trabalho.

No que se refere aos testes computacionais realizados nas instâncias de Taillard, a meta-heurística proposta, em geral, não conseguiu encontrar as melhores soluções conhecidas, dado o maior porte destes problemas.

Deve-se observar que a meta-heurística híbrida proposta não obteve melhorias em relação às melhores soluções conhecidas reportadas na literatura. Nesse sentido, os seguintes comentários podem ser salientados. Embora não possa ser realizada uma comparação justa, pois os algoritmos apresentados na literatura foram executados em computadores distintos, o algoritmo apresentado consegue retornar resultados com baixos tempos computacionais para instâncias de pequeno e médio porte. Adicionalmente, a meta-heurística híbrida proposta tem baixa dependência de parâmetros, ao contrário de grande parte dos algoritmos presentes na literatura revisada, os quais possuíam uma ampla gama de parâmetros a serem calibrados (notadamente, os algoritmos evolucionários). Assim, o método proposto consegue apresentar soluções de alta qualidade em tempos computacionais admissíveis com uma baixa dependência dos parâmetros a serem calibrados.

5 Conclusões

Neste trabalho foi apresentada uma meta-heurística híbrida busca tabu/*iterated local search* (ILS) para o problema de sequenciamento da produção em ambiente *open shop*. Foi realizada uma experimentação computacional intensiva, envolvendo 140 instâncias largamente estudadas por autores

da área, como em alguns dos trabalhos citados aqui, considerando tanto a meta-heurística híbrida proposta como quatro regras de prioridade (LPT, SPT, LAPT e LTRPOM).

A abordagem proposta conseguiu obter resultados competitivos, como pode ser observado na Figura 6 com instâncias menores. Com relação às instâncias de Taillard, foi obtido um *gap* médio de 5,06%. Considerando as instâncias de Guéret e Prins, foi obtido um *gap* médio de 4,69%, sendo obtida a melhor solução conhecida em 26,2% das instâncias e sendo obtidos *gaps* inferiores a 2% em 51,2% das instâncias.

Embora o custo computacional para instâncias de maior porte tenha sido elevado, para as instâncias de menor porte os tempos computacionais se mostraram competitivos. Efetuamos as comparações com os melhores resultados reportados na literatura (Sha & Hsu, 2007). Como não havia um algoritmo ILS na literatura revisada, concluímos que não seria necessária a comparação com outros algoritmos, uma vez que já compramos com a melhor abordagem. O algoritmo proposto é flexível e possui uma baixa dependência de parâmetros, chegando a soluções de boa qualidade em tempo computacional admissível. Portanto, pode-se ressaltar que a meta-heurística híbrida proposta pode ser utilizada para solucionar problemas reais, visto que, na prática, um grande número de máquinas e tarefas, em geral, não é observado.

Como sugestões para desenvolvimento de futuros estudos, recomenda-se: (i) aprimorar a busca local para que esta se torne eficiente para a resolução de problemas de grande porte; (ii) aplicar a abordagem proposta na solução de problemas reais; (iii) adaptar a abordagem proposta para solucionar outras variantes do OSSP.

Agradecimentos

O primeiro autor agradece à Pró-Reitoria de Assuntos Estudantis da Universidade Federal do Ceará pela concessão da bolsa de estudos. O segundo autor agradece ao Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq pelo fomento (Processo 404232/2016-7).

Referências

Abreu, L. R, Cunha, J. O., Prata, B. A., (2019). A genetic algorithm for scheduling open shops with sequence-dependent setup times. *Computers and Operations Research*. *113*. DOI: https://doi.org/10.1016/j.cor.2019.104793

Anand, E., & Panneerselvam, R. (2015). Literature review of open shop scheduling problems. *Intelligent Information Management*, *7*(1), 32-52. DOI: https://doi.org/10.4236/iim.2015.71004

Azadeh, A., Farahani, M. H., & Kalantari, S. S. (2015). Solving a multi-objective open shop problem for multi-processors under preventive maintenance. *International Journal of Advanced*

Manufacturing Technology, *78*(5-8), 707-722. DOI: https://doi.org/10.1007/s00170-014-6660-3

Bai, D., & Tang, L. (2013). Open shop scheduling problem to minimize makespan with release dates. *Applied Mathematical Modeling, 37*(4), 2008-2015. DOI: https://doi.org/10.1016/j.apm.2012.04.037

Bai, D., Zhang, Z. H., & Zhang, Q. (2016). Flexible open shop scheduling problem to minimize makespan. *Computers & Operational Research, 67*(1), 207-215. DOI: https://doi.org/10.1016/j.cor.2015.10.012

Cheng-, R., Gen, M., & Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms-I. representation. *Computers & Industrial Engineering*, *30*(4), 983-997. DOI: https://doi.org/10.1016/0360-8352(96)00047-2

Garey, M. R., & Johnson, D. S. (1974). *Computers and intractability: a guide to the theory of np-completeness*. New York: Freeman.

Glover, F., & Laguna, M. (1997). *Tabu search*. Norwell, MA: Kluwer Academic.

Gonzalez, T., & Sahni, S. (1976). Open-shop scheduling to minimize finish time. *Journal of the Association for Computing Machinery, 23*(4), 665-679. DOI: https://doi.org/10.1145/321978.321985

Hosseinabadi, A. A. R., Vahidi, J., Saemi, B., Sangaiah, A. K., & Elhoseny, M. (2019). Extended genetic algorithm for solving open-shop scheduling problem. *Soft Computing, 23*, 5099-5116. DOI: https://doi.org/10.1007/s00500-018-3177-y

Johnson, D. S., & Mcgeoch, L. A. (1997). The travelling salesman problem: a case study in local optimization. In Aarts, E. H. L. & Lenstra, J. K. (Eds). *Local search in combinatorial optimization* (Chap. 2, pp. 215-310). New York: John Wiley.

Lal, A. H., Vishnu, K. R., Haq, A. N. & Jeyapaul, R. (2019). The mean flow time in open shop scheduling. *Journal of Advances in Management Research, 17*(2), 251-261. DOI: https://doi.org/10.1108/JAMR-05-2019-0081

Liaw, C. -F. (1999). A tabu search algorithm for the open shop scheduling problem. *Computers and Operations Research, 26*(2), 109-126. DOI: https://doi.org/10.1016/S0305-0548(98)00056-2

Liaw, C. -F. (2000). A hybrid genetic algorithm for the open shop scheduling problem. *European Journal of Operational Research, 124*(1), 28-42. DOI: https://doi.org/10.1016/S0377-2217(99)00168-X

Liaw, C. -F. (2014). A fast heuristic to minimize number of tardy jobs in preemptive open shops. *Journal of Industrial and Production Engineering, 31*(1), 27-35. DOI: https://doi.org/10.1080/21681015.2013.879395

Lourenço, H. R., Martin, O., & Stützle, T. (2002). Iterated local search. In: Glover, F. & Kochenberger, G. (Eds). *Handbook of metaheuristics* (pp. 321-353). Norwell, MA: Kluwer Academic Publishers.

Low, C., & Yeh, Y. (2009). Genetic algorithm-based heuristics for an open shop scheduling problem with setup, processing, and removal times separated. *Robotics and Computer-Integrated Manufacturing, 25*(2), 314-322. DOI: https://doi.org/10.1016/j.rcim.2007.07.017

Martin, O., Otto, S. W., & Felten, E. W. (1991). Large-step Markov chains for the traveling salesman problem. *Complex Systems, 5*(3), 299-326. Disponível em: https://www.complex-systems.com/abstracts/v05_i03_a03/

Meddourene, A., Bouzouia, B., Abbou, R., & Farraa, B. B. (2019). An hybrid SA-DATC approach for JIT open-shop scheduling problem with earliness and tardiness penalties. *IFCA-Paper Online*, *52*(13), 2396-2401. DOI: https://doi.org/10.1016/j.ifacol.2019.11.565

Naderi, B., Fatemi Ghomi, S. M. T., Aminnayeria, M., & Zandieh, M. (2010). A contribution and new heuristics for open shop scheduling. *Computers & Operations Research, 37*(1), 213-221. DOI: https://doi.org/10.1016/j.cor.2009.04.010

Naderi, B., Mousakhani, M., & Khalili, M. (2013). Scheduling multi-objective open shop scheduling using a hybrid immune algorithm. *International Journal of Advanced Manufacturing Technology, 66*(5-8), 895-905. DOI: https://doi.org/10.1007/s00170-012-4375-x

Naderi, B., Nafati, E., & Yazdani, M. (2012). An electromagnetism-like metaheuristic for open-shop problems with no buffer. *Journal of Industrial Engineering International, 8*(1), 1- 8. DOI: https://doi.org/10.1186/2251-712X-8-29

Naderi, B., & Zandieh, M. (2014). Modeling and scheduling no-wait open shop problems. *International Journal of Production Economics, 158*(1), 256-266. DOI: https://doi.org/10.1016/j.ijpe.2014.06.011

Pempera, J., & Smutnicki, C. (2018). Open Shop cyclic scheduling. *European Journal of Operational Research*, *269*(2), 773-781. DOI: https://doi.org/10.1016/j.ejor.2018.02.021

Pinedo, M. L. (2016). *Scheduling: theory, algorithms, and systems*. New York: Prentice-Hall.

Prins, C. (2000). Competitive genetic algorithms for the open-shop scheduling problem. *Mathematical Methods of Operations Research, 52*(3), 389-411. DOI: https://doi.org/10.1007/s001860000090

Reza Hejazi, S., & Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: a review. *International Journal of Production Research*, *43*(14), 2895-2929. DOI: https://doi.org/10.1080/00207540500056417

Sha, D. Y., & Hsu, C.-Y. (2008). A new particle swarm optimization for the open shop scheduling problem. *Computers and Operations Research, 35*(10), 3243-3261. DOI: https://doi.org/10.1016/j.cor.2007.02.019