Uma heurística híbrida para o problema de escalonamento de tarefas em uma máquina com tempos de setup dependentes da sequência e datas de liberação

Rafael Morais[1], **Teobaldo Bulhões**[1], **Anand Subramanian**[1]

[1]Centro de Informática – Universidade Federal da Paraíba (UFPB)
Rua dos Escoteiros s/n, Mangabeira – 58058-600 – João Pessoa – PB – Brasil

rafaelsobralm@gmail.com, tbulhoes@ci.ufpb.br, anand@ci.ufpb.br

Abstract. *This paper addresses the problem of minimizing the makespan on a single machine scheduling, considering release dates and non-anticipatory, sequence dependent setup times. A hybrid heuristic approach that combines iterated local search with beam search is proposed. To reduce the computational complexity of the algorithm, an efficient move evaluation strategy is employed during the local search. Experiments were performed on 1800 instances from the literature, demonstrating that the algorithm is capable of obtaining high-quality solutions when compared to existing methods.*

Resumo. *Este trabalho aborda o problema de minimizar o tempo de conclusão em um escalonamento para uma máquina, considerando datas de liberação e tempos de configuração dependentes da sequência. Uma abordagem heurística híbrida que combina iterated local search e beam search é proposta. Para reduzir a complexidade computacional do algoritmo uma estratégia de avaliação eficiente de movimentos é utilizada durante a busca local. Foram realizados experimentos em 1800 instâncias da literatura, demonstrando que o algoritmo é capaz de obter soluções de alta qualidade em comparação com métodos existentes.*

1. Introdução

O escalonamento de tarefas, como um processo de tomada de decisão, desempenha um papel importante na maioria dos ambientes de produção e manufatura. O gerenciamento dos recursos interfere na eficiência do processo produtivo, e pode levar a diminuição de custos e aumento da produtividade [Pinedo 2012].

O presente trabalho aborda o problema de minimizar o tempo total de conclusão (*makespan*) em uma única máquina, considerando datas de liberação e tempos de configuração dependentes da sequência, sendo denominado $1|r_j, s_{ij}|C_{max}$ na notação definida em [Graham et al. 1979]. Na literatura este problema foi explorado nos trabalhos de [Montoya-Torres et al. 2012], [Velez-Gallego et al. 2016], [Fan et al. 2019] e [Fernandez-Viagas and Costa 2021].

Sendo o $1|r_j, s_{ij}|C_{max}$ um problema $\mathcal{NP}$-difícil [Pinedo 2012], é proposto uma heurística baseada em *iterated local search* (ILS), incorporando elementos de *beam search* (BS). É apresentado um método para avaliação eficiente de movimentos baseado em concatenação de subsequencias, inspirado nos trabalhos de [Kindervater and Savelsbergh 1997] e [Vidal et al. 2011].

2. Definição do Problema

Seja $J = \{1, 2, \ldots, n\}$ o conjunto de n tarefas a ser executada em uma única máquina. Cada tarefa $j \in J$ possui uma data de liberação r_j e tempo de processamento p_j. A notação s_{ij} representa o tempo de *setup* não-antecipatório que ocorre se a tarefa j é processada imediatamente após a tarefa i. Se j é a primeira tarefa na sequência, um tempo de *setup* inicial denotado como s_{0j} é gasto preparando a máquina. Como os tempos de *setup* são não-antecipatórios, a preparação da máquina só pode começar após a data de liberação da tarefa a ser processada. Neste problema preempção não é permitida.

Uma solução é definida por uma sequência $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ em que cada tarefa $j \in J$ é executada exatamente uma vez. O tempo de conclusão de uma tarefa $j \in J$, denotado por C_j, é dado por $C_j = r_j + s_{0j} + p_j$, se j for a primeira tarefa da sequência, ou seja, $j = \sigma_1$. Caso contrário, tem-se $C_j = \max\{C_i, r_j\} + s_{ij} + p_j$, sendo i a tarefa que antecede j no escalonamento. O problema consiste em encontrar uma solução que minimize o tempo de conclusão da última tarefa (*makespan*) dado por $C_{max} = \max_{j \in J}\{C_j\}$.

3. Algoritmo proposto

O método desenvolvido para abordar o problema proposto é uma meta-heurística denotada como ILS-BS, que combina elementos de *iterated local search* (ILS) (ver [Silva et al. 2012]) e do algoritmo *beam search* (BS).

Em cada iteração do ILS é construída uma solução inicial por meio de uma adaptação do método *beam search*. Essa solução é então iterativamente melhorada ao passar pelo procedimento de busca local que é realizado com o método de descida em vizinhança variável com escolha aleatória (RVND), e auxiliado por um mecanismo de perturbação (*double bridge*) que evita que o algoritmo fique retido em resultados ótimos locais.

3.1. *Beam Search*

O procedimento construtivo gerador de soluções iniciais é uma adaptação do *beam search* apresentado no trabalho de [Velez-Gallego et al. 2016]. Uma árvore é construída a partir de um nó raiz no nível 0, onde todas as variáveis de decisão são livres. Esta árvore é expandida um nível por vez até o nível $n - 1$. Para cada nó, são criadas até w novas ramificações ao se fixar uma tarefa j ao final da sequência. São criados os w ramos que acarretam no menor tempo ocioso gerado pela fixação de j. Se o número de nós em um nível ultrapassa o limite N, cada nó é avaliado de acordo com o *lower bound* do *makespan*, e são mantidos apenas os N melhores nós selecionados aleatoriamente dentre os $\lfloor(1 + \alpha)N\rfloor$.

3.2. Busca Local

O procedimento de busca local é baseado no *randomized variable neighborhood descent* (RVND), similar ao descrito em [Silva et al. 2012]. São consideradas duas estruturas de vizinhança: *swap*, que consiste na permutação de duas tarefas; e o *l-block-insertion*, em que um bloco de l tarefas contíguas é removido e reinserido em outra posição.

3.2.1. Avaliação Eficiente de Movimentos

Durante a fase de busca local é necessário avaliar os custos de cada solução explorada nas estruturas de vizinhança. Este cálculo pode ser feito diretamente ao percorrer a sequência de tarefas, checando as datas de liberação e computando os tempos de conclusão de cada tarefa, o que é uma operação com complexidade $\mathcal{O}(n)$. Para reduzir essa complexidade é proposto um método de avaliação baseado em concatenação de subsequências, inspirado nos trabalhos de [Kindervater and Savelsbergh 1997] e [Vidal et al. 2011].

Para cada subsequência $\sigma = (\sigma_1, \sigma_2, ..., \sigma_{|\sigma|})$ presente em uma solução, são armazenados os seguintes atributos:

- $P(\sigma) =$ primeira tarefa;
- $U(\sigma) =$ última tarefa;
- $E(\sigma) =$ tempo mínimo de início para processar as tarefas sem tempo ocioso;
- $D(\sigma) =$ somas dos tempos de processamento e dos tempo de preparação.

Em uma subsequência unitária σ, composta por uma única tarefa $j \in J$, o tempo mínimo de início $E(\sigma)$ é a data de liberação r_j da tarefa, e o tempo de processamento $D(\sigma)$ corresponde a p_j. Qualquer subsequência σ com $|\sigma| > 1$ pode ser formada pela concatenação de duas subsequências menores.

O tempo de conclusão de uma subsequência é dado por $C(\sigma) = E(\sigma) + D(\sigma)$, no entanto, este valor não corresponde ao *makespan* de uma solução composta pela subsequência, pois $C(\sigma)$ não considera o *setup* inicial da sequência. Assim, para obter o *makespan* é necessário realizar a concatenação da subsequência σ correspondente com uma subsequência unitária contendo uma tarefa artificial 0, tal que $r_0 = 0$, $p_0 = 0$. Com este método é possível calcular o custo de uma solução em tempo constante amortizado, ao considerar que qualquer movimento pode ser descrito como concatenação de subsequências guardadas na memória.

4. Resultados

O ILS-BS proposto foi programado em C++ (g++ 5.4.0) e todos os experimentos foram executados em uma máquina com processador Intel i7-2600 3.40 GHz e 16 GB de RAM com sistema operacional Linux Ubuntu 16.04 64 bits. Testes foram realizados no conjunto de 1800 instâncias proposto por [Ovacikt and Uzsoy 1994]. Nesse conjunto as instâncias são divididas pelo número de tarefas $n \in \{25, 50, 75, 100, 125, 150\}$ e pelo fator de dispersão das datas de liberação no tempo $R \in \{0.2, 0.4, 1.0, 1.4, 1.8\}$. Para cada instância o ILS-BS foi executado 10 vezes, com parâmetros I_R = 10, $I_{ILS} = 100$, $\alpha = 0.01$.

A performance do ILS-BS é comparada as melhores heurísticas conhecidas para o $1|r_j, s_{ij}|C_{max}$. Sendo estas o *beam search* (BS) de [Velez-Gallego et al. 2016], e os algoritmos VBIH e IG apresentados em [Fan et al. 2019], que só reportou resultados para instâncias com $n \leq 75$.

Também é incluído na comparação outro método baseado em ILS, aqui denotado GILS, que utiliza um procedimento construtivo baseado em *greedy randomized adaptative search procedure* (GRASP) [Feo and Resende 1995], similar ao apresentado em [Silva et al. 2012].

A Tabela 1 reporta os *gaps* entre a melhor solução encontrado por cada um dos algoritmos supracitados e a melhor solução conhecida, incluindo aquelas encontradas pelos próprios algoritmos, para cada par de $n \leq 75$ e R. Pode-se observar que em média o ILS-BS e GILS foram capazes de obter melhores soluções que os algoritmos encontrados literatura em todos os tipos de instância.

Tabela 1. *Gaps* médios considerando a melhor solução encontrada por cada algoritmo: $n \in \{25, 50, 75\}$

	$n = 25$					$n = 50$					$n = 75$				
R	**ILS-BS**	**GILS**	**BS**	**VBIH**	**IG**	**ILS-BS**	**GILS**	**BS**	**VBIH**	**IG**	**ILS-BS**	**GILS**	**BS**	**VBIH**	**IG**
0.2	0.00	0.00	0.67	0.47	0.03	0.03	0.07	0.78	0.71	0.68	0.02	0.20	0.55	0.92	1.03
0.6	0.00	0.00	0.12	0.00	0.11	0.02	0.03	0.55	0.01	0.93	0.12	0.18	0.68	0.22	1.64
1.0	0.01	0.00	0.10	0.17	0.09	0.03	0.07	0.37	0.23	1.03	0.05	0.24	0.49	0.54	1.72
1.4	0.01	0.00	0.08	0.27	0.06	0.08	0.07	0.36	0.56	0.78	0.04	0.23	0.42	0.78	1.39
1.8	0.00	0.00	0.14	0.24	0.04	0.05	0.07	0.78	0.65	0.62	0.03	0.20	0.65	0.86	1.14

A Tabela 2 apresenta os *gaps* médios entre as melhores soluções encontradas pelos algoritmos ILS-BS, GILS, BS e a melhor solução conhecida para instâncias com $n \geq 100$. Nota-se que em instâncias $n \in \{125, 150\}$ o GILS apresenta piores resultados comparado ao BS, porém com a adoção do procedimento construtivo baseado em *beam search* foi possível obter melhores soluções em média para todos os casos.

Tabela 2. *Gaps* médios considerando a melhor solução encontrada por cada algoritmo: $n \in \{100, 125, 150\}$

	$n = 100$			$n = 125$			$n = 150$		
R	**ILS-BS**	**GILS**	**BS**	**ILS-BS**	**GILS**	**BS**	**ILS-BS**	**GILS**	**BS**
0.2	0.02	0.33	0.56	0.01	0.54	0.49	0.03	0.70	0.44
0.6	0.09	0.21	0.58	0.06	0.21	0.51	0.03	0.31	0.37
1.0	0.01	0.32	0.41	0.00	0.45	0.36	0.00	0.59	0.32
1.4	0.02	0.44	0.36	0.00	0.54	0.38	0.01	0.80	0.30
1.8	0.01	0.36	0.62	0.01	0.45	0.60	0.00	0.66	0.47

5. Conclusão

Neste trabalho foi proposta uma heurística híbrida ILS-BS para o problema $1|r_j, s_{ij}|C_{max}$.

O ILS-BS foi testado em um conjunto de 1800 instâncias da literatura e comparado com as melhores heurísticas conhecidas para o problema abordado. Os resultados obtidos mostraram que o algoritmo proposto é capaz de produzir solução de alta qualidade com consistência, encontrando a melhor solução conhecida para 1596 instâncias.

Durante a análise dos movimentos na busca local, utilizou-se uma estratégia de avaliação eficiente que consistia na concatenação de subsequências. Essa abordagem resultou em uma significativa melhora nos tempos de execução, sendo que o impacto dessa estratégia foi ainda maior em instâncias de maior tamanho. Ao aplicar a avaliação eficiente em instâncias com 150 tarefas, observou-se uma média de melhora de velocidade de execução em 7,76 vezes.

Trabalhos futuros podem estender a solução proposta para outras variantes do problema, com novas restrições encontradas em cenários reais e desafiadores ou uma função objetivo alternativa.

Referências

Fan, J., Öztop, H., Tasgetiren, M., and Gao, L. (2019). A variable block insertion heuristic for single machine with release dates and sequence dependent setup times for makespan minimization. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1676–1683, Xiamen, China. IEEE.

Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.

Fernandez-Viagas, V. and Costa, A. (2021). Two novel population based algorithms for the single machine scheduling problem with sequence dependent setup times and release times. *Swarm and Evolutionary Computation*, 63.

Graham, R., Lawler, E., Lenstra, J., and Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In P.L. Hammer, E. J. and Korte, B., editors, *Discrete Optimization II Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium co-sponsored by IBM Canada and SIAM Banff, Aha. and Vancouver*, volume 5 of *Annals of Discrete Mathematics*, pages 287 – 326. Elsevier.

Kindervater, G. and Savelsbergh, M. (1997). Vehicle routing: handling edge exchanges. *In: Aarts, E., Lenstra, J. (Eds.), Local Search in Combinatorial Optimization. Wiley, New York*, page 337–360.

Montoya-Torres, J., González-Solano, F., and Soto-Ferrari, M. (2012). Deterministic machine scheduling with release times and sequence-dependent setups using random-insertion heuristics. *International Journal of Advanced Operations Management*, 4(1/2):4–26.

Ovacikt, I. and Uzsoy, R. (1994). Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent setup times. *International Journal of Production Research*, 32(6):1243–1263.

Pinedo, M. L. (2012). *Scheduling: Theory, Algorithms, and Systems*. Springer-Verlag, Nova Iorque.

Silva, J. M. P., Teixeira, E., and Subramanian, A. (2021). Exact and metaheuristic approaches for identical parallel machine scheduling with a common server and sequence-dependent setup times. *Journal of the Operational Research Society*, 72(2):444–457.

Silva, M. M., Subramanian, A., Vidal, T., and Ochi, L. S. (2012). A simple and effective metaheuristic for the minimum latency problem. *European Journal of Operational Research*, 221(3):513–520.

Velez-Gallego, M. C., Maya, J., and Montoya-Torres, J. (2016). A beam search heuristic for scheduling a single machine with release dates and sequence dependent setup times to minimize the makespan. *Computers & Operations Research*, 73:132–140.

Vidal, T., Crainic, T., Gendreau, M., and Prins, C. (2011). A unifying view on timing problems and algorithms. Cirrelt tech.rep. 2011-43, CIRRELT.