Algoritmos para o Problema de Sequenciamento de Tarefas em Máquinas Paralelas

CONGRESSO DE INICIAÇÃO CIENTÍFICA

PALAVRAS CHAVE. ***Scheduling*, Tarefas, Máquinas Paralelas, Algoritmos, Indústria 4.0.**

Henrique Bussom de Castro, FCA-UNICAMP
Orientadora: Profa. Dra. Priscila Cristina Berbert Rampazzo, FCA-UNICAMP

1. Introdução

Diversos problemas operacionais de indústrias de produtos e serviços são causados pela alocação indevida de recursos. A área de otimização apresenta os problemas de alocação e sequenciamento de tarefas a um conjunto de máquinas disponíveis através da classe de *Scheduling Problems*. Dentro das diversas classificações que esta classe representa, o intuito deste projeto é pesquisar, propor e implementar soluções para o problema de Máquinas Paralelas [Pinedo, 2010; Brucker, 2006].

O problema de Máquinas Paralelas consiste em um conjunto de máquinas que devem processar um conjunto de tarefas. Cada tarefa possui características específicas em relação ao tempo. O problema deve considerar restrições na alocação: uma máquina pode processar apenas uma tarefa por vez e uma tarefa só pode ser processada por uma das máquinas. Diferentes funções-objetivo podem ser consideradas nesse tipo de problema: a minimização do *makespan*, a minimização do tempo de espera, a minimização do tempo de atraso total, a minimização do atraso máximo, etc.

Sabe-se que um problema de Máquinas Paralelas pertence à classe dos Problemas NP-Difícil, o que aumenta significativamente o tempo e custo computacional dependendo de sua dimensão, dificultando a determinação de uma solução ótima para o problema. Para esses casos, a busca por uma solução ótima é substituída por uma solução satisfatória em tempo mais adequado, fazendo isso através dos processos heurísticos ou metaheurísticos.

2. Métodos e Metodologia

Inicialmente, o problema foi modelado matematicamente e resolvido por três *solvers* de Programação Matemática: CBC, CPLEX e CPSAT. Concomitantemente, foi implementado um gerador de instâncias sintéticas, em linguagem de Programação Python. Por fim, comparações entre os diferentes *solvers* e impacto da dimensionalidade do problema na obtenção de uma solução em tempo computacional razoável foram realizadas. Na segunda etapa do projeto foi implementada uma metaheurística evolutiva baseada em Algoritmos de Estimação da Distribuição, como alternativa para a obtenção de uma solução de qualidade em baixo tempo computacional.

2.1. Modelo matemático

Considera-se um problema com n tarefas, m máquinas e os seguintes parâmetros:

e_i: instante de chegada da tarefa i.

d_i: prazo de término da tarefa i.

p_{ij}: tempo total de processamento da tarefa i na máquina j.

As variáveis de decisão do modelo foram definidas como:

$$z_{ij} = \begin{cases} 1 & \text{se a tarefa } i \text{ está na máquina } j. \\ 0 & \text{caso contrário.} \end{cases} \tag{1}$$

$$w_{ik} = \begin{cases} 1 & \text{se a tarefa } i \text{ é alocada antes da tarefa } k. \\ 0 & \text{caso contrário.} \end{cases} \tag{2}$$

$$x_i = \text{início do processamento da tarefa } i. \tag{3}$$

Um modelo de Programação Linear Inteira (PLI) que minimiza o tempo de espera total pode ser descrito como:

$$\textit{Minimizar} \quad f_1 = \sum_{i=1}^{n} (x_i - e_i) \tag{4}$$

$$\text{s.a} \quad \sum_{j=1}^{m} z_{ij} = 1 \; \forall i \tag{5}$$

$$z_{ij} + z_{kj} - w_{ik} - w_{ki} \leq 1 \; \forall i, \forall k, i \neq k, \tag{6}$$

$$x_i + \sum_{j=1}^{m} z_{ij} \, p_{ij} \leq x_k + (1 - w_{ik})M \quad \forall i, \forall k, i \neq k \tag{7}$$

$$z_{ij} + z_{kh} + w_{ik} + w_{ki} \leq 2 \quad \forall i, \forall k, \forall j, \forall h, i \neq k, j \neq h \tag{8}$$

$$x_i \geq e_i \; \forall i. \tag{9}$$

$$x_i + \sum_{j=1}^{m} z_{ij} \, p_{ij} \leq d_i \; \forall i. \tag{10}$$

$$x_i \geq 0 \; \forall i. \tag{11}$$

$$w_{ik}, z_{ij} \in \{0, 1\} \; \forall i, k, j \tag{12}$$

A Restrição 5 estabelece que cada tarefa será alocada em apenas uma máquina. A Restrição 6 garante que se as tarefas i e k estão na mesma máquina, então, ou a tarefa i é processada antes da tarefa k ou vice-e-versa. A Restrição 7 não permite que as tarefas estejam sobrepostas. A Restrição 8 define que a variável w_{ik} só existe para tarefas que estão na mesma máquina. A Restrição 9 não permite que as tarefas sejam alocadas antes de sua chegada. E finalmente, a Restrição 10 não permite que as tarefas sejam finalizadas após o prazo de término. As demais restrições limitam os valores para as variáveis de decisão.

2.2. EDA

Os Algoritmos de Estimação da Distribuição (EDA) [Eiben e Smith, 2015] pertecem à classe das meta-heurísticas evolutivas. Os EDA se baseiam em modelos de distribuição para realizar a estimação de novas soluções, decidindo em qual caminho prosseguir através de modelos probabilísticos realizados pelo próprio algoritmo e extraídos através das soluções candidatas promissoras. Em relação a outras meta-heurísticas evolutivas, nota-se que o EDA é um tipo de algoritmo mais direcionado em sua busca por uma solução de qualidade, visto que, se houver um modelo de distribuição bem definido, o mesmo consegue encontrar relações e características existentes entre as soluções candidatas, o que faz com que as regiões em torno dessas soluções sejam menos aleatórias e mais prováveis de serem exploradas.

Será apresentado a seguir o pseudocódigo da implementação do EDA realizada:

```
Definir Parâmetros:  TP, G, Pm
Leitura da Instância:  N, M, p, e, d
Pop = InicializaPopulação (TP, N, M)
   para cada geração g = 1 até G
      x = DecodificaPopulacao (TP, N, M, p, e)
      fo = AvaliaPopulacao(TP, x, N, e)
      mat1, mat2 = Modelos de distribuicao(PopOrd, Pm)
Pop = novaPop(mat1, mat2, TP, N, M)
```

Primeiramente, são definidos os parâmetros do EDA: tamanho da população `TP`, número de gerações `G` e Operador de Elitismo `Pm` (que seleciona uma porcentagem de melhores soluções candidatas). Em seguida, é realizada a leitura da instância, a qual fornece os parâmetros realcionados às tarefas `N` e máquinas `M`: tempo de processamento da tarefa `p`, instante de chegada da tarefa `e` e prazo de término da tarefa `d`. Após, inicializa-se a População através dos parâmetros: `TP`, `N` e `M`.

E para cada geração `G`, são realizados os seguintes passos: A decodificação da População, na qual cada indíviduo é associado a um número real, sendo sua parte inteira determinante para a alocação da tarefa em uma determinada máquina, e sua parte decimal determinando a sua posição em relação às outras tarefas as quais se encontram na mesma máquina. Esta codificação (Figura 1) segue a proposta de Ting et al. [2014]. Após a decodificação, são determinados o tempo de início de cada tarefa `x` e a função objetivo `fo` associada a cada indivíduo da população.

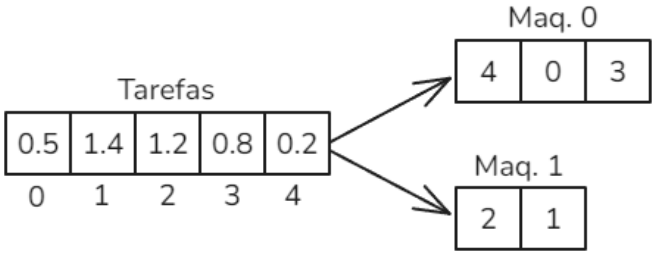

Figura 1: Esquema da Decodificação

Após isso, através do Operador de Elitismo, são selecionados as `Pm%` melhores soluções da população. A partir dessas soluções, criamos as matrizes de distribuição `mat1` e `mat2`. A `mat1` nos mostra quantas vezes cada tarefa se encontra em cada máquina, já a `mat2` nos mostra a relação de precedência das tarefas quando as mesmas se encontram em uma mesma máquina.

Por último, é criada uma nova população, baseada nos modelos representados pelas matrizes de distribuição, onde cada novo indivíduo terá maior probabilidade de carregar as características definidas pelas matrizes, ou seja, alocando determinada tarefa em determinada máquina e definindo a ordem de precedência entre as tarefas em uma mesma máquina. Dessa forma, os indivíduos da nova população tendem a ter funções objetivo mais promissoras.

3. Resultados e Discussões

A implementação dos códigos foi realizada através da linguagem de programação Python. Após realização da modelagem matemática, foi utilizado o *solver* comercial CPLEX[1], e também os de código aberto, CBC e CPSAT[2] para resolver o problema. Os *solvers* tiveram tempo de execução limitado em 30 minutos.

Seguindo a mesma proposta dos trabalhos Silva et al. [2014] e Barbosa et al. [2022], desenvolvemos um gerador de dados utilizando Distribuição Beta devido à sua versatilidade. A dificuldade de resolução do Problema de Sequenciamento de Tarefas em Máquinas Paralelas varia de acordo com a forma como cada parâmetro da distribuição é gerado; as janelas de tempo podem ser pequenas ou grandes. Gerando os parâmetros de diferentes formas, para obter diferentes classes de problemas, podemos fazer uma avaliação mais sólida de nossa metodologia.

As Tabelas a seguir, resumem os resultados obtidos até o momento. A coluna ”obj”indica o valor da função-objetivo. A coluna ”gap”apresenta a tolerância relativa entre a melhor solução inteira obtida (”obj”) e o limitante inferior; se uma solução ótima for encontrada, tem-se que o ”gap”é igual a zero. Os restultados da coluna ”tempo”estão representados em segundos. A primeira coluna representa a dimensão das instâncias, na ordem: (tarefas - máquinas). Os resultados do EDA referem-se à melhor rodada de um conjunto 10 rodadas.

Instância	**CBC**			**CPLEX**			**CPSAT**			**EDA**	
tar - maq	obj	gap	tempo	obj	gap	tempo	obj	gap	tempo	obj	tempo
15 - 5	189	0	1.37	189	0	0.17	189	0	0.30	195	52.5
20 - 5	213	0	579.30	213	0	21.31	213	0	17.53	230	105.7
30 - 10	832	0.76	1812.07	298	0.29	1800.01	295	0.28	1800.01	315	167.3
40 - 10	1556	0.82	1835.32	372	0.26	1800.01	364	0.24	1800.01	384	215.8

Tabela 1: Resultados do modelo obtidos com os *solvers*.

Analisando a Tabela 1 percebemos que o CPLEX e o CPSAT têm melhor desempenho que o CBC e que isto se torna ainda mais claro em relação ao tempo computacional despendido na obtenção de uma solução ótima (1^a e 2^a instâncias, com ”gap”= 0) e ao menor ”gap”apresentado (portanto melhor solução viável) quando a solução ótima não é obtida no limite de tempo computacional. Além disso, realizando a comparação entre os *solvers* e o EDA, nota-se que para

[1]IBM ILOG CPLEX Optimization Studio, versão acadêmica.

[2]utilizado com a importação do pacote *ortools*. O OR-Tools [OR-Tools, 2019] é uma biblioteca de código aberto para escrever os modelos de otimização.

instâncias menores, os *solvers* realmente possuem melhor desemepnho, chegando em soluções melhores em menos tempo computacional que o EDA, porém, aumentando a escala das instâncias (a qual assemelha-se com problemas reais), o EDA consegue soluções satisfatórias, de mesma ordem de grandeza que os *solvers*, em tempo computacional consideravelmente menor, o que torna a abrodagem promissora para o Problema de Alocação de Tarefas em Máquinas Paralelas.

Referências

Barbosa, F., Rampazzo, P. C. B., de Azevedo, A. T., e Yamakami, A. (2022). The impact of time windows constraints on metaheuristics implementation: a study for the discrete and dynamic berth allocation problem. *Applied Intelligence*, 52(2):1406–1434.

Brucker, P. (2006). *Scheduling Algorithms*. Springer-Verlag B. H., New York, 5 edition.

Eiben, A. E. e Smith, J. E. (2015). *Introduction to evolutionary computing*. Springer.

OR-Tools, G. (2019). Route. schedule. plan. assign. pack. solve. or-tools is fast and portable software for combinatorial optimization. https://developers.google.com/optimization/.

Pinedo, M. L. (2010). *Scheduling: Theory, Algorithms and Systems*. Springer, New York, 4 edition.

Silva, E., Oliveira, J. F., e Wascher, G. (2014). 2dcpackgen: A problem generator for two-dimensional rectangular cutting and packing problems. *European Journal of Operational Research*, 237(3):846 – 856. ISSN 0377-2217.

Ting, C.-J., Wu, K.-C., e Chou, H. (2014). Particle swarm optimization algorithm for the berth allocation problem. *Expert Systems with Applications*, 41(4):1543–1550.