XXVIII Congresso {virtual} de Iniciação Científica da Unicamp 2020

Algoritmos para o Problema do Flow Shop

Autor

Mateus Fontanari Scolastici

mateusfontascp2704@gmail.com

Orientadora

Profa. Dra. Priscila Berbert Rampazzo

Universidade Estadual de Campinas / Faculdade de Ciências Aplicadas

Resumo das Atividades

Dentre os vários problemas causados nas indústrias pela alocação indevida de recursos, a área da otimização busca resolver a alocação e sequenciamento de tarefas a um conjunto de máquinas utilizando a classe de *Scheduling Problems.* O intuito desse projeto é pesquisar, propor e implementar métodos para solucionar o problema do *Flow Shop*. A primeira parte do projeto consiste em um modelo matemático resolvido por um solver disponível no pacote *ortools* (linguagem de programação Python) que, devido ao *Flow Shop* tratar-se de um problema do tipo NP-difícil, a solução exata levaria a um tempo computacional impraticável para instâncias de grande porte. Com isso, na segunda parte deste projeto, uma abordagem meta-heurística foi proposta e implementada como um método alternativo de solução para o problema: um Algoritmo Genético, com o propósito de disponibilizar uma solução de qualidade para os problemas de médio e grande porte em um tempo computacional razoável.

Palavras-chave: Scheduling, Tarefas, Máquinas, *Flow Shop*, Algoritmos, Indústria 4.0.

1. Introdução

No cenário atual da indústria 4.0, com a automação e robotização da produção em larga escala, a agilidade na geração de bens é muito superior comparado ao dos tempos antigos de manufatura e maquinário simples. Entretanto, a cadeia produtiva de uma indústria possui diversos processos que envolvem alocação de tarefas em um conjunto de máquinas que

necessitam ser otimizados para que seja possível a indústria operar próxima de seu potencial máximo e com isso produzir de forma mais eficiente sem que ocorra desperdício de tempo e recursos por sua parte. Essa cadeia produtiva com processos subsequentes em máquinas enquadra-se no problema do *Flow Shop* da Pesquisa Operacional (Taha, 2007), em que a solução visa a forma mais eficiente de alocação das tarefas em sua respectiva ordem de processos nas máquinas com o objetivo de reduzir o tempo total do processo, denominado *makespan*.

Os problemas de *Flow Shop* podem ser modelados e resolvidos através de modelos matemáticos, como será apresentado nos Modelos 1 e 2, os quais se mostraram eficientes para as instâncias de pequeno porte, mas exigem um tempo computacional impraticável para instâncias de médio e grande porte, devido à complexidade dos problemas de *Flow Shop*, que se classificam como problemas do tipo NP-difícil (Não Polinomial Difícil)(Brucker, 2006). Por esse motivo, uma opção alternativa para solução dessas instâncias de maior porte, foi implementado uma meta-heurística alternativa, o Algoritmo Genético, com o objetivo de disponibilizar uma solução de qualidade em um tempo computacional razoável.

2. Métodos e Metodologias

Para a solução dos problemas do *Flow Shop* neste projeto, foram implementados dois modelos matemáticos, Modelo 1 e Modelo 2 e uma abordagem alternativa, o Algoritmo Genético.

2.1 Exploração de Modelos Matemáticos

Considere o problema com *n* tarefas e *m* máquinas, sendo que cada tarefa *i* tem m operações O_{ij} com tempos de processamento p_{ij}. A operação O_{ij} deve ser processada na máquina M_{ij} e as restrições de precedência $O_{ij} \rightarrow O_{i,j+1}$ devem ser respeitadas para toda tarefa *i*. O objetivo do problema é ordenar as tarefas para cada máquina *j* (Brucker, 2006).

2.1.1 Modelo 1

As variáveis de decisão de um modelo clássico podem ser definidas como:

$$x_{ij} = \text{início da tarefa } i \text{ na máquina } j.$$

$$w_{ikj} = \begin{cases} 1 & \text{se tarefa } i \text{ precede tarefa } k \text{ na máquina } j. \\ 0 & \text{caso contrário.} \end{cases}$$

Assim, o seguinte problema que minimiza o makespan:

$$\begin{aligned}
\min \quad & Cmax && (1)\\
\text{s.a} \quad & Cmax \geqslant x_{im} + p_{im} \quad \forall i = 1,...,n && (2)\\
& x_{ij} + p_{ij} \leqslant x_{i(j+1)} \quad \forall i = 1,...,n \quad j = 1,...,m-1 && (3)\\
& x_{ij} + p_{ij} \leqslant x_{kj} + M(1 - w_{ikj}) \quad \forall i,k = 1,...,n \quad j = 1,...,m && (4)\\
& w_{ikj} + w_{kij} = 1 \quad \forall i = 1,...,n \quad \forall k = 1,...,n \quad i \neq k \quad \forall j = 1,...,m && (5)\\
& x_{ij} \geqslant 0 \quad \forall i = 1,...,n \quad \forall j = 1,...,m \\
& w_{ikj} \in \{0,1\} \quad \forall i = 1,...,n \quad \forall k = 1,...,n \quad i \neq k \quad \forall j = 1,...,m
\end{aligned}$$

A restrição (2) estabelece que o *makespan* deve ser o maior tempo de término de uma tarefa na última máquina. A restrição (3) garante que, para toda tarefa, a *(j+1)*-ésima operação inicie após o processamento da *j*-ésima operação. As restrições (4) e (5) garantem que se duas tarefas *i* e *k* passam por uma mesma máquina *j*, ou a tarefa *i* é processada antes da tarefa *k* ou vice-versa. A função-objetivo (1) minimiza o *makespan*.

2.1.2 Modelo 2

O modelo anterior pode ser modificado, alterando os limites de definição da restrição (8) e adicionando a restrição complementar (9). Com isso, diminui-se o número de variáveis do modelo.

$$\min \quad Cmax \tag{6}$$

$$\text{s.a} \quad Cmax \geqslant x_{ij} + p_{ij} \quad \forall i = 1, ..., n \quad j = 1, ..., m \tag{7}$$

$$x_{ij} + p_{ij} \leqslant x_{i(j+1)} \quad \forall i = 1, ..., n \quad j = 1, ..., m-1 \tag{8}$$

$$x_{ij} + p_{ij} \leqslant x_{kj} + M(1 - w_{ikj}) \quad \forall i = 1, ..., n-1 \quad \forall k > i \quad j = 1, ..., m \tag{9}$$

$$x_{kj} + p_{kj} \leqslant x_{ij} + Mw_{ikj} \quad \forall i = 1, ..., n-1 \quad \forall k > i \quad j = 1, ..., m \tag{10}$$

$$x_{ij} \geqslant 0 \quad \forall i = 1, ..., n \quad \forall j = 1, ..., m$$

$$w_{ikj} \in \{0, 1\} \quad \forall i = 1, ..., n-1 \quad \forall k > i \quad j = 1, ..., m$$

Neste segundo modelo, as restrições (8) e (9) garantem que se duas tarefas *i* e *k* passam por uma mesma máquina *j*, ou a tarefa *i* é processada antes da tarefa *k* ou vice-versa. A função-objetivo (1) também minimiza o *makespan*. As demais restrições são as mesmas do primeiro modelo.

2.2 Algoritmo Genético

O algoritmo genético é um método de otimização inspirado na evolução estudada pela biologia, desenvolvido por John Henry Holland no ano de 1975. O método refere-se à formação de diversos indivíduos que representam possíveis soluções os quais são submetidos inicialmente à uma seleção natural para identificar os que possuem melhores desempenho de acordo com o problema designado. O processo consiste na recombinação de código entre os indivíduos com melhores resultados de forma sucessiva e ordenada, de modo a obter, na última geração, uma solução factível para o problema (Barbosa, 1997).

2.2.1 Algoritmo Genético para o Problema do *Flow Shop*

Para o caso específico do *Flow Shop* o Algoritmo Genético foi elaborado em linguagem Python. O fluxograma a seguir explica de forma resumida cada etapa do algoritmo até a obter a solução.

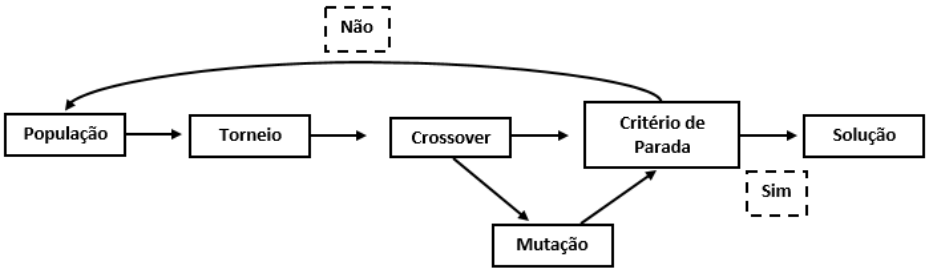

Figura 1: Fluxograma explicativo do Algoritmo Genético para o problema do *Flow Shop*

A primeira etapa consiste na inicialização da população, nela são gerados indivíduos capazes de armazenar dados necessários para representar uma solução do problema. Os indivíduos são decodificados de modo a serem capazes de alocar de forma aleatória a primeira tarefa na primeira máquina do processo, realizando todo o processo da cadeia subsequente. Definidos os indivíduos, ocorre a seleção por torneio entre pares de pais selecionando os com

melhores resultados para compor a geração paternal. Essa geração passa por um processo de recombinação, denominado crossover, formando novos indivíduos resultantes do processo, os quais formam a primeira geração filial. Para maior variedade de resultados nos indivíduos, ocorre em 10% dos filhos um processo de mutação, o qual consiste em trocas duplas de tarefas, produzindo um resultado diversificado em relação aos demais indivíduos de sua geração. Esse processo ocorre sucessivamente ao longo de um número de gerações definidos no início do algoritmo.

3. Apresentação dos Resultados

As instâncias de teste escolhidas para implementação dos modelos estão disponíveis em um repositório on-line (soa.iti.es/problem-instances): VRF benchmarks sets for Flow Shop Scheduling Problems.

O solver (OR-Tools, 2010) foi limitado em 10 minutos de execução. Os testes foram executados em sistema operacional Linux, Intel(R) Core(TM) i5-3337U CPU 1.80GHz, 12Gb de memória RAM.A Tabela a seguir, resume os resultados obtidos com M1 (Modelo 1) e M2 (Modelo 2).

	M1	M1	M1	M1	M1	M1	M2	M2	M2	M2	M2	M2
Instâncias	var	restr	obj	gap	BB	tempo	var	restr	obj	gap	BB	tempo
10 x 5	501	990	808	0	808	614.883	276	540	808	0.314	554	614.268
10 x 10	1001	1990	1370	0.359	878	606.321	551	1090	1370	0.377	854	608.635
10 x 15	1501	2990	1620	0.351	1051	620.646	826	1640	1627	0.352	1054	627.312
10 x 20	2001	3990	1954	0.325	1319	625.807	1101	2190	1939	0.316	1326	615.82
20 x 5	2001	3980	1519	0.71	441	604.139	1051	2080	1540	0.655	532	607.569
20 x 10	4001	7980	1991	0.605	787	610.914	2101	4180	2046	0.611	795	608.502
20 x 15	6001	11980	2361	0.541	1083	614.25	3151	6280	2344	0.532	1097	624.127
20 x 20	8001	15980	2929	0.518	1413	628.279	4201	8380	2956	0.526	1402	630.238

Tabela 1: Resultados obtidos pelos Modelo 1 e Modelo 2

Os resultados obtidos pelo Algoritmo Genético para instâncias de pequeno, médio e grande porte está apresentado na Tabela a seguir. Os testes foram executados em sistema operacional Windows, Intel(R) Core(TM) i7-8700K CPU 3.70GHz, 8Gb de memória RAM.

N. Tarefas	N. Maquinas	Tamanho Populaca	N. Geracoes	Makespan Medio	Desvio do Makespan	Tempo de Processamento
10	5	100	50	505	4	0.1764 (s)
10	20	100	50	1707	2.52	0.4922 (s)
20	10	100	50	1738	6.03	0.5177 (s)
20	20	100	50	2547	7.55	1.00 (s)
30	10	100	100	2274	12.06	1.5311 (s)
30	20	100	100	3115	25.24	2.7862 (s)
40	10	100	100	2895	10.12	1.9154 (s)
40	20	100	100	3853	7.57	3.6368 (s)
50	10	100	100	3506	10.54	2.4398 (s)
50	20	100	100	4429	31.01	4.4525 (s)
60	10	100	100	4011	21.73	2.8652 (s)
60	20	100	100	5088	23.63	5.3119 (s)
100	20	100	100	7482	41.88	9.2131 (s)
200	60	100	100	11002	26.85	27.4945 (s)
300	20	100	100	18787	30.35	31.7205 (s)
400	60	100	100	29617	21.51	123.1111 (s)
500	20	100	100	29930	42.89	55.9255 (s)
800	60	100	100	52740	67.62	267.6049 (s)

Tabela 2: Resultados obtidos pelo Algoritmo Genético

4. Conclusão

Após análise dos resultados obtidos pelos Modelos Matemáticos e pelo Algoritmo Genético, foram analisados três principais pontos, que consistem em: valor obtido para o makespan final, tempo de execução do código e tamanho de instâncias capazes de serem resolvidas. No quesito valor obtido, os resultados dos Modelo 1 e Modelo 2 comparados aos valores obtidos pelo Algoritmo Genético não foram discrepantes entre si, ambos os métodos resultaram em valores factíveis e próximos da solução ótima. Quanto ao tempo de execução, o Algoritmo Genético revelou um tempo de processamento muito inferior ao tempo executado pelos Modelo 1 e Modelo 2, os quais foram limitados em 10 minutos de processamento enquanto o Algoritmo Genético levou segundos para obter resultados muito próximos aos dos modelos matemáticos. Por fim, em relação ao tamanho das instâncias capazes de serem executadas, o Algoritmo Genético não possui limitações de tamanho por tempo de processamento, enquanto os Modelo 1 e Modelo 2 são inviáveis de executar instâncias de médio e grande porte pela complexidade dos cálculos envolvidos.

Desse modo, considerando esses três pontos, observamos vantagens e desvantagens em ambos os métodos. Os modelos matemáticos são capazes de obter a solução ótima para instâncias de pequeno porte, mas em compensação são incapazes de rodar instâncias de grande porte e exigem muito tempo computacional para calcular o makespan. Já o Algoritmo Genético não possui limitações para instâncias e exige menos tempo computacional comparado aos anteriores mas necessita de uma análise mais cautelosa para se obter um resultado próximo à solução ótima, uma vez que devem ser analisadas quantas gerações e o número de indivíduos necessários para cada instância.

Referências

BRUCKER, P.Scheduling Algorithms. 5. New York: Springer-Verlag B. H., 2006.

OR-TOOLS, G.Route. Schedule. Plan. Assign. Pack. Solve. ORToolsis fast and portable software for combinatorial optimization.2019. Https://developers.google.com/optimization/. PINEDO, M. L.Scheduling: Theory, Algorithms and Systems. 4. New York: Springer,2010.

TAHA, H. A.Pesquisa Operacional. 8aed. São Paulo: Pearson (Prentice Hall), 2007.

BARBOSA, Hélio J. C. , Introdução aos algoritmos genéticos, 1997.