Trabalho apresentado no XXXIX CNMAC, Uberlândia - MG, 2019.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Aplicação do AG com Operadores Otimizados para Resolver o Problema de Escalonamento do Tipo Job Shop

Daniel Marques da Silva Sousa[1]
Dayvid Wesley Pereira Martins[2]
Maria José Pereira Dantas[3]
PUC GOIÁS[1,2,3]

1 Introdução

O problema de escalonamento *Job Shop* (JSP) é um problema de otimização combinatória e é bastante estudado pela comunidade de pesquisa de operacional [1]. Na literatura, os algoritmos heurísticos têm mostrado resultados promissores, com destaque para o algoritmo genético (AG). O problema JSP consiste em distribuir operações de um conjunto de tarefas com tempo fixo de execução entre um conjunto finito de máquinas, de forma que o tempo de processamento necessário para terminar a execução de todas elas (*makespan*), seja o menor possível. É preciso destacar que operações de uma mesma tarefa não podem ser atribuídas a uma mesma máquina duas vezes, uma operação não pode ser interrompida, e cada máquina pode processar apenas uma operação de cada vez [1]. Vários pesquisadores já publicaram seus trabalhos abordando o JSP. Um dos pioneiros foi [2], que criou um algoritmo com solução ótima para o problema de duas máquinas e minimização do *makespan*. Os autores em [3] realizaram um levantamento dos problemas JSP e os classificaram por técnicas de solução. Mais recentemente, os autores [4, p. 425 – 436], publicaram uma pesquisa que aborda o problema de programação da produção JSP com tempos de *setup* explícitos e independentes da sequência de processamento das tarefas visando minimizar a duração total da programação (*makespan*).

Este trabalho teve como objetivo fazer uma revisão de literatura de meta-heurísticas baseadas em população, em seguida, propor um AG com operadores otimizados e aplicar na solução de instâncias de *benchmark* compiladas em [5]. O AG foi modelado para minimizar o tempo de finalização (*makespan*) utilizando operadores genéticos específicos e criados para a proposta, atuando em um cromossomo unidimensional (vetor) construído com referência no gráfico de Gantt de uma possível solução de escalonamento e, elementos como seleção por roleta, elitismo e migração.

O algoritmo foi incorporado em um ambiente experimental *online*, um app *web* (em desenvolvimento) e disponível no Laboratório `LEMM` da PUC GOIÁS, com seleção de instâncias e parâmetros configuráveis pelo usuário. Utilizou-se linguagem Python para o desenvolvimento do núcleo da aplicação. Foram realizados testes com variações de

[1]danielmarques.20@hotmail.com.
[2]dayvidwesley@gmail.com
[3]mjpdantas@gmail.com.

instâncias de *benchmark* dos tipos `abz`, `ft` e `la`, com os melhores valores obtidos até o momento referenciados em trabalho da literatura recente [5].

Os resultados foram promissores, uma vez que a implementação não utilizou hibridização do AG com outros algoritmos como em [6], e os valores encontrados circundaram os valores ótimos já publicados em [5]. Os resultados das instâncias foram detalhados com apresentação do gráfico com a evolução do *makespan* médio e melhor *makespan* em cada geração; diagrama de dispersão dos *makespans* da população inicial e da população final; mapa genético com a diversidade de cromossomos em cada geração e gráfico de Gantt com a melhor solução (melhor escalonamento) e, ainda, um resumo dos resultados da instância. Para as instâncias `ft6` 6×6, `la01` 10×5 e `la05` 10×5 foram obtidos valores iguais aos melhores valores encontrados em [5]. Para as instâncias `abz5` 10×10 e `abz6` 10×10 os *gaps* foram de até 4,98%. Para as instâncias `la01` 10×5, `la02` 10×5, `loa03` 10×5, `la04` 10×5, `la05` 10×5 *gaps* de até 2,88% e para as instâncias `ft10` 10×10 e `la16` 10×10 *gaps* de até 5,29%.

Conclui-se que o algoritmo proposto no núcleo da aplicação do app *web* circunda os valores ótimos das instâncias avaliadas. O usuário pode executar diferentes configurações de parâmetros e/ou repetir experimentações sem alterar parâmetros. Outra vantagem do app *web* é a disponibilização de muitas instâncias de teste com diferentes graus de dificuldade, preenchendo uma lacuna para ambientes experimentais com algoritmo AG. Espera-se que a aplicação contribua para a curva de aprendizado do problema *Job Shop*. Estudos estão em desenvolvimento para melhorar o desempenho do AG, diminuindo os *gaps* em relação aos valores já publicados na literatura.

Referências

[1] C. Koulamas and S. S. Panwalkar. The proportionate two-machine no-wait job shop scheduling problem, *European Journal of Operational Research*, 252(1):131–135, 2016.

[2] J. R. Jackson. An extension of Johnson's results on job IDT scheduling, *Naval Research Logistics Quarterly*, 3(3):201–203, 1956.

[3] J. Blazewicz, W. Domschke, and E. Pesch. The job shop scheduling problem: Conventional and new solution techniques, *European Journal of Operational Research*, 93(1):1–33, 1996.

[4] H. Y. Fuchigami and S. Rangel. A survey of case studies in production scheduling: Analysis and perspectives, *Journal of Computational Science*, 2017.

[5] J. J. van Hoorn. The Current state of bounds on benchmark instances of the job-shop scheduling problem, *Journal of Scheduling*, 21:127–128, 2018.

[6] N. Kundakcı and O. Kulak. Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem, *Computers & Industrial Engineering*, 96:31–51, 2016.