Documentação

Trabalho Prático - Algoritmos Bioinspirados

Guilherme Souza Barros, Vítor Oliveira Amorim, Leonardo Guimarães de Oliveira

Universidade Federal de São João del Rei
7 de julho de 2025

1 Introdução

Neste Trabalho Prático foi implementado um **Algoritmo Genético (AG)**[1] para minimizar a solução do **Problema de Agendamento de Tarefas em Linha (Flow Shop Scheduling Problem - FSSP)**. O problema consiste em um conjunto de `N` tarefas que devem ser processadas em um conjunto de `M` máquinas na mesma ordem. O objetivo é determinar a ordem das tarefas para todas as máquinas que minimize o *makespan*, ou seja, o tempo total para que todas as tarefas sejam concluídas.

O **FSSP** possui diversas aplicações práticas em setores onde a eficiência no processo de produção é crucial, dentre elas:

- **Logística de centros de distribuição**: onde pedidos devem ser processados por várias etapas com diferentes recursos disponíveis.
- **Indústria**: onde múltiplas máquinas podem executar diferentes tarefas em tempos diferentes, e o objetivo é otimizar a produção.

2 Modelagem do Programa

2.1 População Inicial

Para gerar a população inicial foi utilizada uma estratégia que envolvia **heurística** e geração de **ndivíduos aleatórios**. A heurística escolhida foi uma adaptação da NEH (Nawaz, Enscore, Ham)[2], amplamente utilizada para problemas de escalonamento de fluxo de trabalho, combinada com uma mutação de suas soluções, para garantir que os indivíduos gerados por heurística também tenham variabilidade. Os demais indivíduos da população foram gerados de forma aleatória, mas garantindo que nenhuma tarefa deixasse de ser executada ou fosse executada duas vezes.

2.2 Cálculo dos Fitness

O *fitness* de cada um dos indivíduos é equivalente ao tempo de término da última operação da última tarefa na última máquina. Ou seja, o *fitness* de cada um dos indivíduos é o tempo total necessário para completar todas as tarefas em todas as máquinas, desde o início da primeira tarefa até o fim da última tarefa.

2.3 Seleção dos Pais

A fim de selecionar os pais dos indivíduos da próxima geração, foram implementados dois operadores de seleção de pais: torneio e roleta.

- **Torneio local**: Semelhante ao torneio tradicional, mas população é dividida em várias ilhas que não competem entre si, para cada pai, são selecionados *npop* (tamanho da população) vezes dois indivíduos aleatórios distintos e da mesma ilha. Depois disso, é calculada a probabilidade do indivíduo mais apto se tornar um pai: caso assim seja, o mais apto é copiado para o vetor de pais, caso contrário, o outro indivíduo é o inserido no vetor de pais.
- **Roleta**: foi adotada uma espécie de "ranking"dos indivíduos com base em seus valores de aptidão ($fitness$): os melhores indivíduos (menor valor de $fitness$) recebem *ranks* maiores, provocando maior probabilidade de seleção, enquanto os menos aptos (maior valor de $fitness$) obtêm *ranks* menores, o que os deixa com menos probabilidade de serem selecionados.

2.4 Cruzamento

Foram implementados dois tipos de cruzamento: *OX* (*Order Crossover*) e *PMX* (*Partially Mapped Crossover*).

- **Cruzamento PMX:** são selecionados dois pontos de corte. Os genes da região entre os dois pontos de corte do primeiro pai são copiados para o segundo filho, e os genes desta região do segundo pai são copiados para o primeiro filho. Após isso, é criado um mapeamento de cada gene de um pai para o gene correspondente do outro pai na mesma posição no cromossomo, para os genes pertencentes à região entre os pontos de corte. Por fim, os genes dos filhos fora da região de corte são preenchidos com o gene de acordo com as relações mapeadas, desde que os genes mapeados ainda não foram inseridos na região de corte.
- **Cruzamento OX:** são selecionados dois pontos de corte. A região entre os dois pontos de corte do primeiro pai é copiada para o segundo filho e a do segundo pai é copiada para o primeiro filho. Após isso, os demais genes dos filhos são preenchidos em ordem, a partir do segundo ponto de corte, e são atribuídos aos filhos os genes do outro pai que participou do cruzamento, levando apenas em consideração apenas os genes que não foram recebidos do seu primeiro pai.

2.5 Mutação

A mutação consiste em trocar duas tarefas de posição, que representa a ordem em que cada uma delas deve ser processada. Isso garante que nenhuma tarefa aparecerá duas vezes no vetor.

A taxa de mutação é ajustada para evitar estagnação em mínimos locais: caso não haja melhora na solução em 10 gerações, a probabilidade de mutação é incrementada em 1% até que ocorra alteração no mínimo local.

2.6 Elitismo

Por fim, no elitismo, os indivíduos de maior aptidão são copiados diretamente para as primeiras posições da nova geração, garantindo a preservação dos melhores indivíduos ao longo das gerações.

2.7 3-OPT

Devido às semelhanças ao PCV, por ser um problema sequencial e permutativo, o 3-OPT foi utilizado para otimizar a busca local e permitir uma mutação extra dos indivíduos a uma certa taxa de probabilidade.

Para a execução, t tentativas ocorrem, onde em cada uma, 3 pontos são selecionados e reconectados, a melhor permutação das t tentativas é salva. A combinação final obtida é definida como o novo indivíduo caso seu fitness seja mais otimizado do que o indivíduo antigo ou independente do fitness anterior (caso uma mutação esteja ocorrendo).

3 Avaliação dos Resultados Obtidos

Com o propósito de avaliar a eficiência da solução, foi analisado o comportamento do algoritmo para os seguintes arquivos de entrada: *fssp_instance_05.txt* e *fssp_instance_07.txt*. Os arquivos de entrada utilizados foram os arquivos padrão disponibilizados para testes dos resultados obtidos.

Para a execução do algoritmo, foi adotado um conjunto de parâmetros obtidos por meio da execução de experimentos fatoriais para cada um dos parâmetros do **Algoritmo Genético**, conforme detalhado abaixo:

Probabilidade de Cruzamento	Elites	Probabilidade de Mutação	Gerações	Indivíduos
0.9	10	0.01	100	400

Com base nos experimentos realizados, foi possível observar que o melhor operador de **seleção de pais** foi o **Torneio**, tanto em questão de desempenho quanto em tempo de execução. Além disso, o melhor operador de cruzamento, de acordo com os testes, foi o operador *PMX*, que apresentou melhores resultados do que o operador *OX*.

3.0.1 Evolução dos Valores de Fitness ao Longo das Gerações

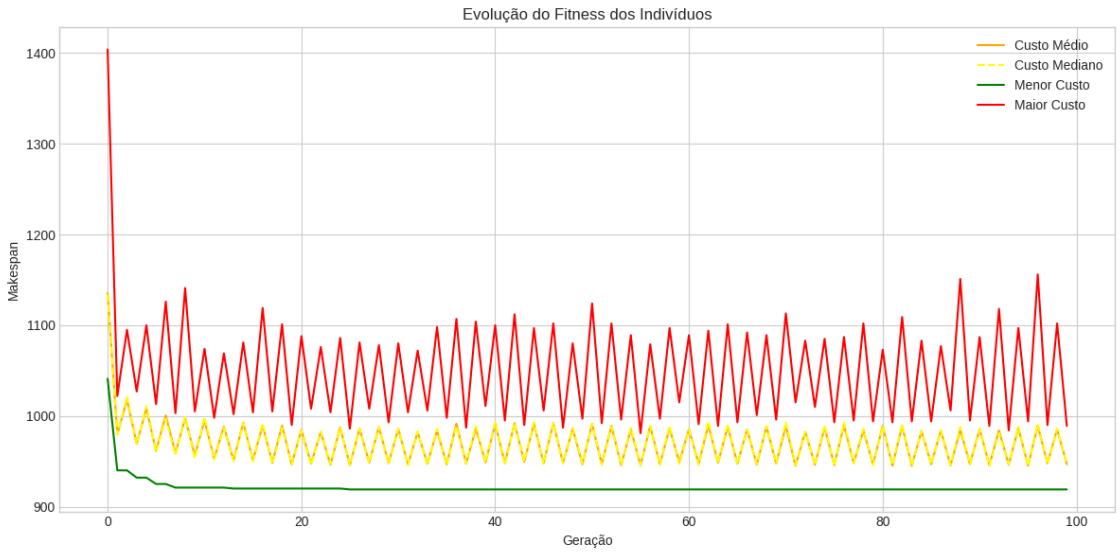

Figura 1: Comportamento do fitness dos indivíduos da população ao longo das gerações para a instância 05.

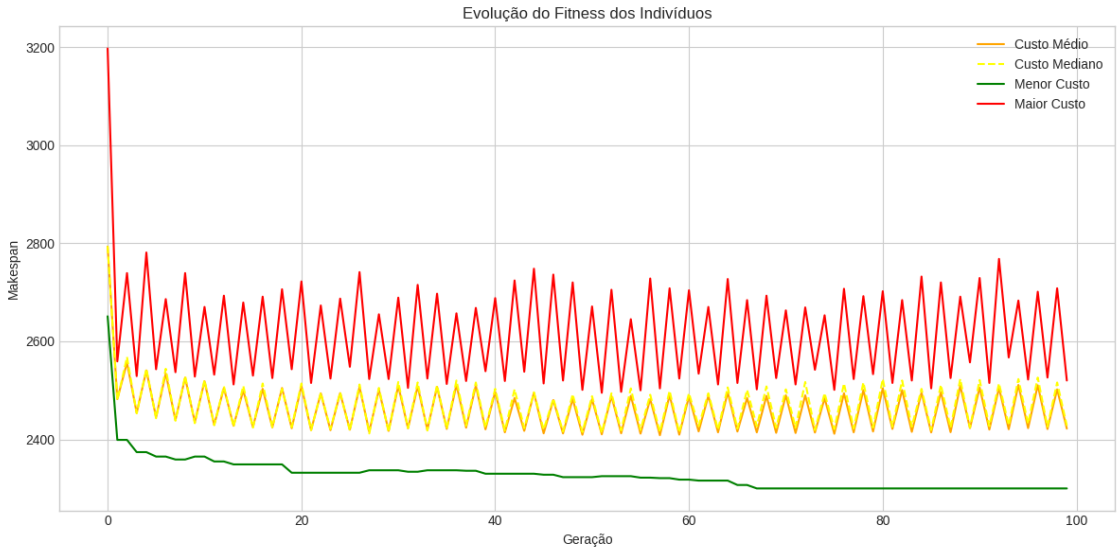

Figura 2: Comportamento do fitness dos indivíduos da população ao longo das gerações parar a instância 07.

Ambas as execuções demonstram o mesmo comportamento, enfrentando dificuldade para encontrar soluções mais otimizadas após as primeiras gerações, tendo quedas em uma frequência menor, mas ainda presentes devido à busca local e comportamento mutatório gerado pelo 3-OPT.

Durante a execução também pôde-se observar pequenos intervalos de queda após um tempo estagnado em ambas as instâncias, mostrando que o algoritmo consegue escapar de mínimos locais dado tempo suficiente para que isso ocorra, tendo como exemplo a segunda instância, na qual foi obtido um resultado melhor mesmo após um tempo considerável preso em permutações piores.

3.0.2 Makespan Mínimo Obtido em Diferentes Execuções

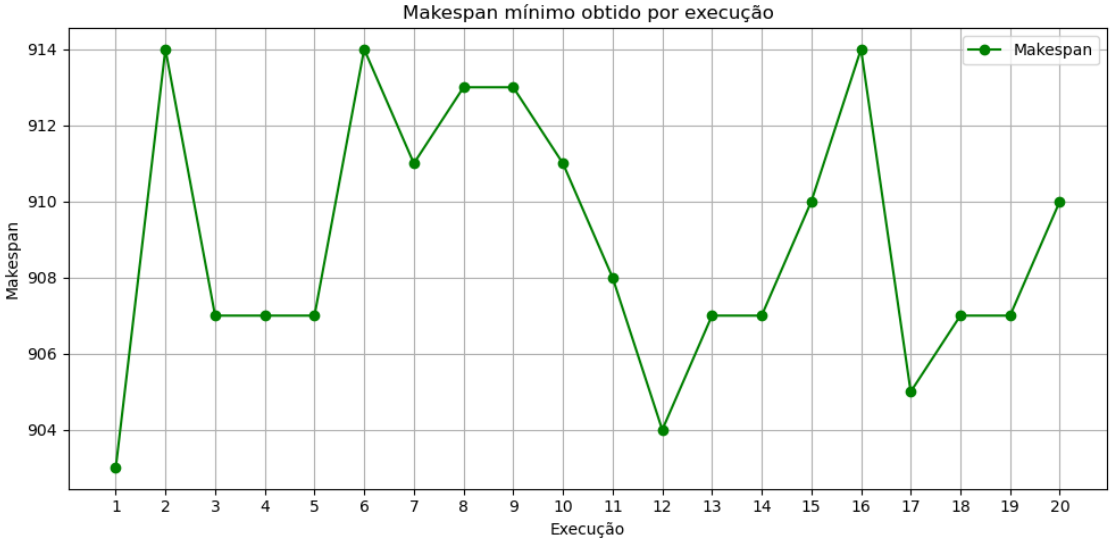

Figura 3: *Makespan* mínimo obtido pelo Algoritmo Genético em diferentes execuções da instância 5.

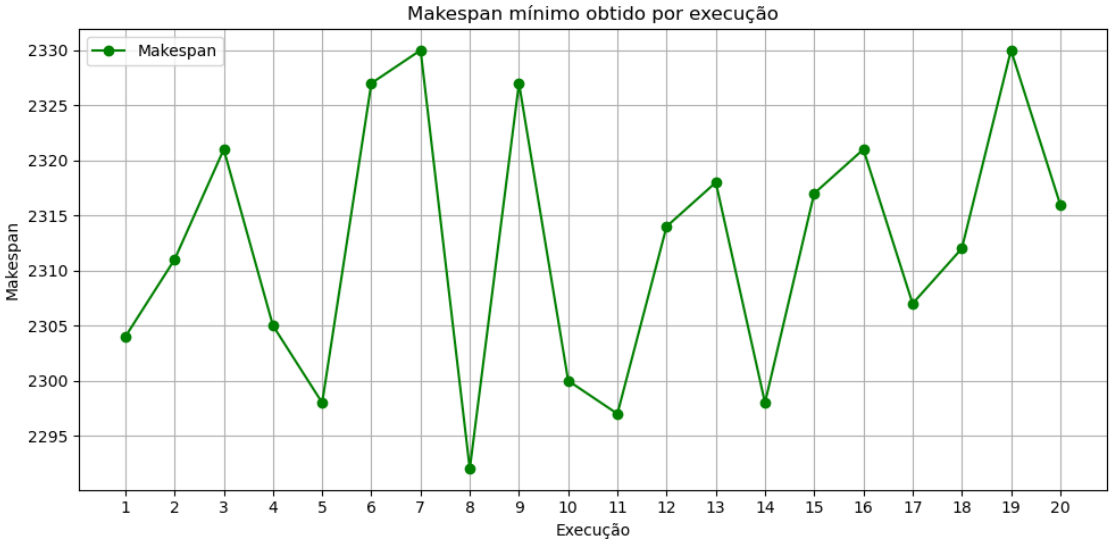

Figura 4: *Makespan* mínimo obtido pelo Algoritmo Genético em diferentes execuções da instância 7.

Ao analisar as figuras 3 e 4, contendo os *makespans* obtidos em diferentes execuções do algoritmo, observa-se que os resultados tendem a se concentrar em torno de valores próximos, indicando uma certa estabilidade no desempenho geral do método. No entanto, essa proximidade não garante que o algoritmo tenha sempre convergido para a melhor solução global ou a melhor solução conhecida para o problema.

Em várias execuções, o algoritmo encontrou soluções satisfatórias, mas ligeiramente inferiores à melhor conhecida, indicando que ele pode ficar preso em ótimos locais.

3.0.3 Tempo de Execução do Algoritmo

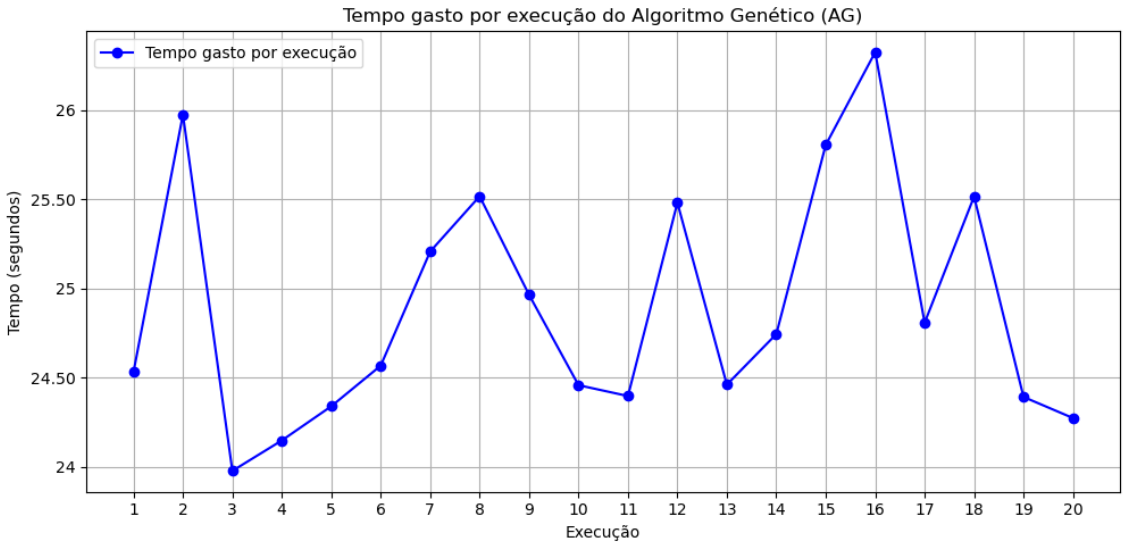

Figura 5: Tempo gasto pelo Algoritmo Genético em diferentes execuções da instância 5.

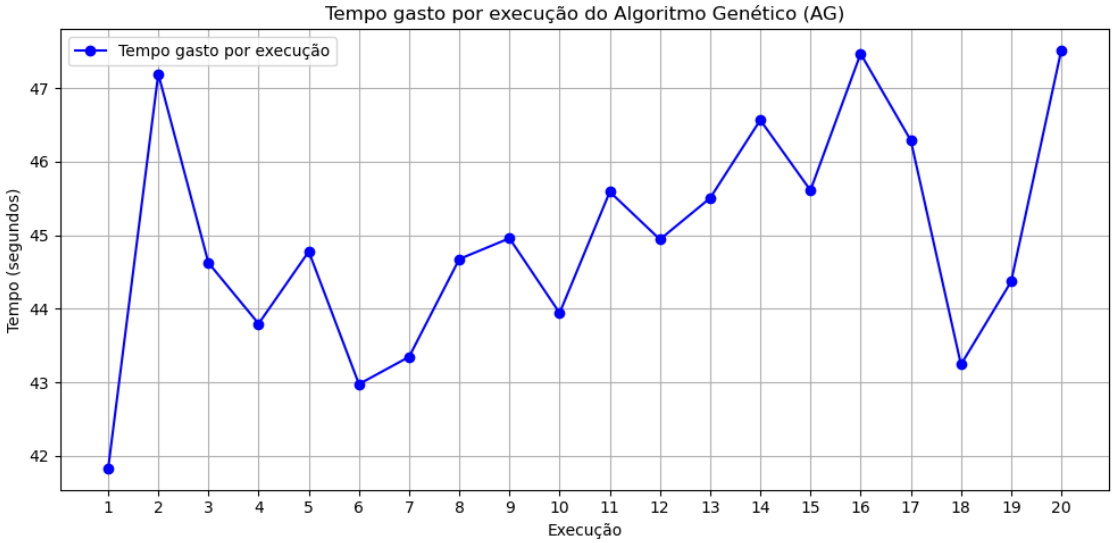

Figura 6: Tempo gasto pelo Algoritmo Genético em diferentes execuções da instância 7.

Com base nos tempos de execução em diferentes execuções do algoritmo, observados nas figuras 5 e 6, constatamos uma leve variação no tempo total. Isso ocorre porque o operador 3-OPT sempre é chamado exatamente 50 vezes, quantidade correspondente à metade do número de gerações executadas.

3.1 Especificações da Máquina de Testes

Especificação das peças relevantes e sistema operacional da máquina utilizada para o teste empírico:

- Sistema Operacional: Ubuntu 24.04.1 LTS (Linux 6.11.0-26-generic);
- Processador: i5-10400F;
- Memória: 12GiB DIMM DDR4 2666MHz.

4 Conclusão

Os experimentos realizados demonstram que o algoritmo é capaz de produzir soluções com *makespans* consistentes em diferentes execuções, com variações relativamente pequenas. No entanto, apesar dessa estabilidade, nem sempre foi possível atingir a melhor solução global, o que indica a presença de ótimos locais e limitações do processo de intensificação de busca.

Portanto, podemos concluir que o algoritmo é bastante capaz de obter boas soluções com ótimas médias de tempo, sem variar muito entre suas diferentes execuções.

Referências

[1] HOLLAND, John H. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Cambridge, MA: MIT Press, 1992.

[2] NAWAZ, Muhammad; ENSCORE JR., E. Emory; HAM, Inyong. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega, Oxford, v. 11, n. 1, p. 91-95, 1983.