# PROTÓTIPO DE OTIMIZADOR RKO APLICADO PARA O PROBLEMA DE SEQUENCIAMENTO EM MÁQUINAS PARALELAS COM TEMPOS DE PREPARAÇÃO DEPENDENTES DA SEQUÊNCIA (PMSP-SDS)

João Vitor Fonseca [(1)]; Diego Mello da Silva [(2)]; Everthon Valadão [(3)]

(1) Bolsista IFMG, Bacharelando em Ciências da Computação, IFMG Campus Formiga, Formiga–MG; jovifsc@gmail.com

(2) Professor orientador, Mestre em Ciências da Computação, IFMG Campus Formiga, Formiga–MG; diego.silva@ifmg.edu.br

(3) Professor orientador, Mestre em Ciências da Computação, IFMG Campus Formiga, Formiga–MG; everthon.valadao@ifmg.edu.br

## RESUMO

O problema de sequenciamento em máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência (UPMSP-SDS) envolve decidir a atribuição e sequenciamento de tarefas em uma estação composta por máquinas que operam em paralelo de modo a minimizar o tempo total necessário para finalizar todo o lote de tarefas (*makespan*) (VALLADA; RUIZ, 2011). Este trabalho apresenta resultados preliminares, comparando a recente meta-heurística *Random Key Optimizer* (RKO) com métodos alternativos de resolução do UPMSP-SDS, a saber: *Simulated Annealing* (SA)*, Biased Random-key Genetic Algorithm* (BRKGA), *Random Key Genetic Algorithm* (RKGA) e um modelo de Programação Inteira Mista (MIP). Ensaios em 400 instâncias mostraram que as meta-heurísticas superaram o MIP em cenários desafiadores (muitas tarefas, poucas máquinas), similares aos cenários industriais. Já para cenários de menor complexidade, com mais máquinas disponíveis e menos tarefas, o MIP se sobressai, encontrando a solução ótima dentro de 20 minutos.

**Palavras-chave**: Meta-heurísticas. RKO. UPMSP-SDS.

## 1 INTRODUÇÃO

Problemas de sequenciamento de máquina são comuns em ambientes de indústrias, manufatura e serviços. Graves (1981) define o sequenciamento como a alocação de recursos ao longo do tempo para satisfazer algum critério, como minimizar o *makespan* (*i.e.*, o tempo necessário para completar todas as tarefas), maximizar o número de tarefas concluídas por unidade de tempo, ou minimizar o atraso quando a tarefa termina depois do prazo, por exemplo. Segundo Pinedo (2008) e Graham *et al*. (1979), os problemas de sequenciamento de tarefas em máquinas se distinguem em três aspectos: ambiente de processamento, restrições do sequenciamento e o objetivo a otimizar. Dentre estes, o problema de sequenciamento em máquinas com tempo de preparação dependente da sequência (PMSP-SDS) modela cenários onde o tempo de preparo de cada máquina varia conforme a tarefa precedente, aumentando a complexidade combinatória (KIM, 2002). Neste trabalho estuda-se a variante UPMSP-SDS (VALLADA e RUIZ, 2011), da classe *NP*-Difícil, com tempo de processamento dependente

de máquina, e tempo de preparação (ou *setup*) que depende da sequência na qual as tarefas são realizadas. A otimização dos tempos de preparação é industrialmente relevante, pois seu consumo de capacidade produtiva justifica a pesquisa sobre o tema.

Diversas heurísticas e meta-heurísticas da literatura abordam o problema, como Vallada e Ruiz (2011). Recentemente, Chaves et al. (2025) propuseram o *Random-Key Optimizer* (RKO)[1] para integrar meta-heurísticas em problemas que codificam soluções como chaves aleatórias (*i.e.*, vetores com números aleatórios entre zero e um) via um *pool* comum de soluções elites. Cada algoritmo usa primitivas *blending*, *shaking* e *local search* para perturbar esses vetores e gerar novas soluções. O RKO pode potencializar os resultados obtidos em problemas de sequenciamento de máquinas (aplicação original de algoritmos de chaves aleatórias). Considerando isso, este estudo explora a efetividade do RKO para o problema UPMSP-SDS através de uma análise comparativa de *makespan* com meta-heurísticas *Simulated Annealing*, *Biased Random Key Genetic Algorithm* (BRKGA), e *Random Key Genetic Algorithm* (RKGA), além de um modelo de Programação Inteira Mista (MIP) (VALLADA e RUIZ, 2011). Este trabalho consiste na primeira fase do projeto, que contemplará também uma interface gráfica que permitirá aos usuários configurar parâmetros sobre as tarefas a sequenciar, como tempos de processamento e de *setup*, importar os dados de planilha eletrônica ou *benchmarks*.

## 2 METODOLOGIA

Os algoritmos foram implementados em Python 3. Os experimentos foram realizados em um servidor Intel(R) Xeon(R) E5-1650 v3 @ 3.50GHz de seis núcleos e com 32 GB de RAM, no Ubuntu 24.04.2 LTS. O *benchmark* foi composto de 400 instâncias com 50 e 100 tarefas; 10, 15, 20, 25 e 30 máquinas; tempos de *setup* de até 9, 49, 99 e 124; com 10 instâncias por cenário (tarefas, máquinas, *setup*). O tempo limite para resolver cada instância foi de 200 segundos para o RKO, 300 segundos para o BRKGA, SA e RKGA, e 1200 segundos para o modelo MIP via *solver* comercial Gurobi 12. Optou-se por tempo limite menor para o RKO porque ele executa suas meta-heurísticas integradas via *multi-threading*, tornando a comparação mais justa. Parâmetros do BRKGA e SA foram calibrados através do pacote `irace` da linguagem R, enquanto RKO e RKGA não passaram por nenhum tipo de

[1] Biblioteca Python disponível em https://github.com/RKO-solver. Acessado em Outubro/2025.

refinamento em seus parâmetros. Como resultado, para o SA obtiveram-se os parâmetros: temperatura inicial (605), final (1), iterações por temperatura (178), tamanho dos blocos (10), probabilidade de aplicar movimento em bloco (6%) e de aplicar movimento de deslocamento (46%); para o BRKGA: número de gerações (400), indivíduos (1500), proporção de elite (18%), taxa de mutação (9,5%) e cruzamento (75%). O RKGA utilizou os mesmos parâmetros definidos para o BRKGA por ter funcionamento similar, e o RKO utilizou parâmetros *default* da biblioteca disponibilizada por Chaves et al. (2025) para cada agente. O RKO e BRKGA utilizam o mesmo *decoder* para converter chaves aleatórias em cronogramas baseando-se na política *Early Completion Time* (ECT): primeiramente, as tarefas são ordenadas conforme o valor das chaves aleatórias que codificam a solução para o problema; em seguida, elas são alocadas nesta ordem na máquina disponível na qual se obtém o tempo de conclusão mais breve, considerando-se o momento em que ela está livre somado aos tempos de *setup* e de processamento.

## 3 RESULTADOS E DISCUSSÕES

A Figura 1 apresenta o intervalo de confiança (IC) de 95% sobre o valor médio do *makespan* (eixo Y) para os diferentes cenários de tarefas e máquinas disponíveis (eixo X), ordenados pela razão (nº de tarefas / nº de máquinas), calculado para os cinco procedimentos de resolução do UPMSP-SDS propostos. O IC foi estimado via *bootstrap* por não requerer suposições sobre a distribuição estatística dos valores de *makespan* em amostras pequenas. Cada amostra foi organizada agrupando-se as 40 instâncias com igual número de tarefas e máquinas, independentemente dos tempos de *setup*. Observa-se que as meta-heurísticas apresentam, em regra, desempenho superior ao MIP nos cenários mais à direita, quando a razão tarefas/máquinas aumenta (situação na qual o *makespan* tende a ser maior). Essa diferença ocorre porque, em instâncias de maior complexidade (menos máquinas disponíveis para alocar muitas tarefas), o tempo limite do MIP parece insuficiente para encontrar soluções ótimas ou competitivas. Isso impactou no intervalo de confiança (IC) do MIP que, em vários cenários, possui ponto médio (do IC) mais alto do que os pontos médios das meta-heurísticas.

Ao comparar os métodos entre si, nota-se que BRKGA, RKO e SA resultam frequentemente com *makespan* próximos, ICs sobrepostos em boa parte dos cenários, sinal de que, estatisticamente, não é possível afirmar com segurança uma vantagem consistente entre

elas sem um teste formal. Ainda assim, existe uma tendência visível: nos cenários de menor complexidade, BRKGA e RKO possuem desempenho um pouco melhor do que o SA, enquanto nos cenários mais difíceis (muitas tarefas, poucas máquinas) o SA se aproxima ou supera os algoritmos baseados em chaves aleatórias.

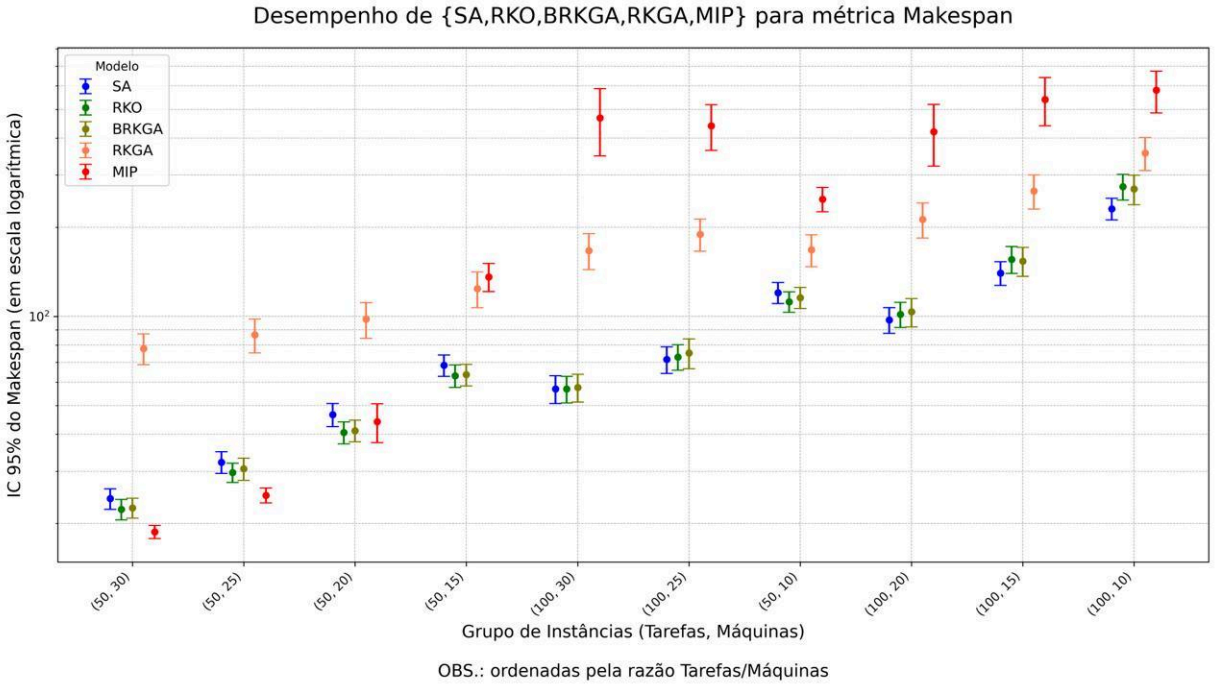


Figura 1 — Desempenho relativo dos algoritmos para o critério *makespan*
Fonte: Elaborado pelos autores (2025).

O RKGA não apresentou desempenho satisfatório, em parte porque não se investigou a melhor calibragem de seus parâmetros via **`irace`**. Em muitos cenários, os ICs do BRKGA e RKO se sobrepõem, o que requer mais experimentos com casos maiores e testes de hipótese para concluir sobre diferenças estatísticas. Ainda sobre os ICs, intervalos com amplitudes menores demonstram algoritmos mais robustos para o problema, com menor variância entre instâncias do mesmo cenário. Um algoritmo com IC consistentemente menor é mais previsível e, portanto, preferível em cenários operacionais, mesmo que sua média de desempenho seja similar à de outro. O eixo vertical está em escala logarítmica, portanto diferenças verticais aparentemente pequenas podem representar diferenças multiplicativas relevantes no *makespan* absoluto; é preciso ter isso em mente ao interpretar a magnitude das diferenças. O RKO, mesmo sem ajuste de parâmetros, foi competitivo com o SA e BRKGA em menor tempo de execução, já o SA teve melhor desempenho no cenário mais difícil.

## 4 CONCLUSÕES

Para instâncias com mais tarefas e poucas máquinas disponíveis, é preferível usar meta-heurísticas (RKO, BRKGA, RKGA, SA) para resolver o problema UPMSP-SDS, pois elas entregam soluções melhores ou comparáveis ao MIP com menor tempo de execução. Entre as heurísticas, não existe um vencedor universal, porém vale ressaltar que até o momento somente BRKGA e SA passaram por refinamento de parâmetros. Para casos pequenos com maior quantidade de máquinas, o MIP encontra a solução ótima no tempo limite, superando as meta-heurísticas. A natureza *NP*-Difícil do problema torna o MIP impraticável para instâncias de escala industrial, reforçando as meta-heurísticas como a melhor abordagem para esses cenários. Para trabalhos futuros, é recomendável realizar testes estatísticos formais (Friedman, Nemenyi ou Wilcoxon pareado) para confirmar quais diferenças são expressivas em casos maiores. Adicionalmente, o RKO deve ser aprimorado, pois sua competitividade em menor tempo de execução o posiciona como um forte solucionador para o problema. O refinamento dos parâmetros do RKO é sugerido, visto que este trabalho usou sua configuração *default*, podendo sobrepor o SA em cenários complexos a depender do ajuste fino de seus parâmetros.

## REFERÊNCIAS

CHAVES, A. A.; RESENDE, M. G. C.; SCHUETZ, M. J. A. *et al*. *A Random-Key optimizer for combinatorial optimization*. **Journal of Heuristics**, v. 31, n. 32, 2025.

GRAHAM, Ronald Lewis *et al*. ***Optimization and approximation in deterministic sequencing and scheduling: a survey***. In: *Annals of discrete mathematics*. Elsevier, 1979. p. 287-326.

GRAVES, Stephen C. ***A review of production scheduling***. **Operations research**, v. 29, n. 4, p. 646-675, 1981.

KIM, Dong-Won *et al*. ***Unrelated parallel machine scheduling with setup times using simulated annealing.*** *Robotics and Computer-Integrated Manufacturing*, v. 18, n. 3-4, p. 223-231, 2002.

PINEDO, M. ***Deterministic Models: Preliminaries.*** *In:* ***Scheduling: theory, algorithms and systems***. 3. ed. New York: Springer, 2008. p. 13-34.

VALLADA, E.; RUIZ, R. ***A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times***. European Journal of Operational Research, v. 211, n. 3, p. 612-622, 2011.