

**Programação por restrições e algoritmo genético de chaves aleatórias
enviesadas para o problema de escalonamento de tarefas acopladas em uma
única máquina minimizando o *makespan***

Vítor A. Barbosa, Rafael A. Melo

Instituto de Computação, Universidade Federal da Bahia
Salvador, BA, Brasil
vitor.barbosa@ufba.br, rafael.melo@ufba.br

Tiago Januario

Department of Computer Science, Boston University
Boston, Massachusetts, USA
januario@bu.edu

RESUMO

Neste artigo, consideramos o problema de escalonamento de tarefas acopladas em uma única máquina minimizando o *makespan*. Neste problema, um conjunto de trabalhos deve ser escalonado, cada um composto por duas tarefas intercaladas por um atraso exato. Não é permitida preempção e o objetivo consiste em minimizar o tempo de conclusão da última tarefa escalonada. Propomos um modelo de programação por restrições e um algoritmo genético de chaves aleatórias enviesadas (BRKGA) para o problema. Resultados preliminares mostram que o modelo de programação por restrições pode alcançar soluções de alta qualidade e limites duais fortes dentro do limite de tempo fornecido. Além disso, ele encontra uma solução pelo menos tão boa quanto a melhor conhecida na literatura para todas as instâncias de nosso conjunto preliminar, exceto uma. As soluções obtidas são estritamente melhores do que as disponíveis na literatura para mais de 70% das instâncias testadas. Ademais, o BRKGA consegue encontrar soluções com *gaps* consideravelmente baixos em pouco tempo computacional.

PALAVRAS CHAVE. Escalonamento de tarefas. Programação por restrições. BRKGA.

Tópicos (Logística e Transportes, Metaheurísticas)

ABSTRACT

In this paper, we consider a single-machine coupled task scheduling problem to minimize the makespan. In this problem, a set of jobs has to be scheduled, each composed of two tasks interspersed by an exact delay. No preemption is allowed and the goal consists of minimizing the completion time of the last scheduled task. We propose a constraint programming model and a biased random-key genetic algorithm (BRKGA) for the problem. Preliminary results show that the constraint programming model can achieve high-quality solutions and strong dual bounds within the provided time limit. Moreover, it can find a solution that at least matches the best-known available in the literature for all the instances but one in our preliminary set. The obtained solutions are strictly better than the best known ones for more than 70% of the tested instances. Additionally, the BRKGA can find solutions with reasonably low gaps within short computational times.

KEYWORDS. Task scheduling. Constraint programming. BRKGA.

Paper topics (Logistics and Transportation, Metaheuristics)

1. Introdução

O problema abordado neste estudo consiste no escalonamento de trabalhos em uma única máquina, cada um deles composto por duas tarefas acopladas. Em cada trabalho, a segunda tarefa (tarefa final) deve ser iniciada após o término da primeira tarefa (tarefa inicial) com um atraso exato. O objetivo consiste em minimizar o tempo de conclusão da última tarefa. Dentre as aplicações do escalonamento de tarefas acopladas, podemos mencionar: sistemas de radar pulsado nos quais um radar transmite um pulso e recebe sua reflexão para rastrear alvos [Shapiro, 1980]; processos de fabricação de produtos químicos [Ageev e Baburin, 2007]; agendamentos de pacientes em serviços médicos [Pérez et al., 2011; Condotta e Shakhlevich, 2014; Liu et al., 2019]. Outra potencial aplicação consiste no escalonamento de tarefas operadas por robôs em casas inteligentes, principalmente em cozinhas inteligentes [Sharma e Reddy, 2024]. Neste cenário, robôs de cozinha [Zhai et al., 2015; Mepani et al., 2022; Jiang e Zhou, 2022] podem aproveitar momentos de ociosidade entre sub-tarefas para executar outras sub-tarefas, reduzindo, portanto, o tempo total de preparo [Bautista et al., 2023; Yi et al., 2022]. Yi et al. [2022] estudou um problema similar de múltiplas tarefas operadas por robôs de cozinha com dois braços em um ambiente controlado.

Diversos artigos trataram questões de complexidade [Orman e Potts, 1997; Sherali e Smith, 2005; Condotta e Shakhlevich, 2012] e complexidade parametrizada [Bessy e Giroudeau, 2019] para o problema de escalonamento de tarefas acopladas em uma única máquina para minimizar o *makespan*. Dentre as abordagens mais recentes para o problema, podemos mencionar algumas. Li e Zhao [2007] propuseram uma busca tabu. Békési et al. [2014] apresentaram formulações de programação inteira (PI) e um *branch and bound* combinatório. Khatami et al. [2020] elaboraram uma revisão da literatura e discutiram o uso de formulações de PI para diversos problemas de escalonamento de tarefas acopladas, incluindo o problema estudado neste trabalho. Khatami e Salehipour [2021] apresentaram uma heurística de PI baseada em busca binária. Békési et al. [2022] desenvolveram um algoritmo aproximativo para um caso especial do problema no qual os trabalhos só possuem dois tipos diferentes de atraso. Recentemente, Khatami e Salehipour [2024] propuseram uma nova formulação de PI e uma heurística *relax-and-solve* para o problema.

Neste artigo, apresentamos um modelo de programação por restrições e um algoritmo genético de chaves aleatórias enviesadas (BRKGA, do inglês *biased random-key genetic algorithm*) para o problema de escalonamento de tarefas acopladas em uma única máquina minimizando o *makespan*. O restante do artigo está organizado como segue. A Seção 2 apresenta as notações usadas e o modelo de programação por restrições. A Seção 3 descreve os componentes do BRKGA. A Seção 4 resume os experimentos computacionais preliminares. A Seção 5 conclui o artigo com alguns comentários finais.

2. Notações e modelo de programação por restrições

Considere um conjunto de trabalhos $J = \{1, \dots, n\}$. Cada trabalho é representado por (a_j, L_j, b_j) , onde a_j e b_j denotam, respectivamente, os tempos de processamento de suas tarefas inicial e final, e L_j fornece o intervalo exato entre elas. Preempções não são permitidas e o objetivo do problema consiste em minimizar o tempo de execução C_{max} , ou seja, o tempo de conclusão do último trabalho agendado. O problema é denotado por $1|(a_j, L_j, b_j)|C_{max}$. Assume-se que os tempos de processamento e os intervalos entre as tarefas são valores inteiros. Além disso, denote o conjunto de tarefas por $T = \{1, \dots, 2n\}$, onde as tarefas $2j - 1$ e $2j$ identificam, respectivamente, as tarefas inicial e final do trabalho j . Denote por p_h o tempo de processamento da tarefa $h \in T$.

Considere as variáveis s_h e e_h representando, respectivamente, os tempos de início e fim da tarefa $h \in T$. Defina a variável C_{max} como o tempo de fim da última tarefa escalonada. O problema pode ser formulado como o seguinte modelo de programação por restrições:

$$\min C_{max} \quad (1)$$

$$s_{2j} = e_{2j-1} + L_j, \quad \forall j \in J, \quad (2)$$

$$e_h = s_h + p_h, \quad \forall h \in T, \quad (3)$$

$$\text{noOverlap}(T), \quad (4)$$

$$C_{max} = \max_{h \in T} e_h, \quad (5)$$

$$s_h \in [0, UB], \quad \forall h \in T, \quad (6)$$

$$e_h \in [0, UB], \quad \forall h \in T, \quad (7)$$

$$C_{max} \in [0, UB]. \quad (8)$$

A função objetivo (1) minimiza o tempo de fim da última tarefa escalonada, i.e., o *makespan*. As restrições (2) asseguram o intervalo exato entre as duas tarefas de cada um dos trabalhos. As restrições (3) relacionam os tempos de início e de fim de cada tarefa. A restrição global noOverlap (4) garante que as tarefas em T não se sobrepõem. As restrições (5) definem o *makespan* como o tempo de fim da última tarefa. As restrições (6)-(8) definem os domínios das variáveis de decisão.

3. Algoritmo genético de chaves aleatórias enviesadas

O BRKGA foi proposto por Gonçalves e Resende [2011] baseado no algoritmo genético de chaves aleatórias (RKGA, do inglês *random-key genetic algorithm*) [Bean, 1994]. O BRKGA é uma metaheurística de busca que simplifica os algoritmos genéticos, em geral, tornando os aspectos de representação e o mecanismo de intensificação-diversificação independentes do problema. Primeiramente, as soluções são representadas por vetores de números reais gerados aleatoriamente no intervalo $[0, 1]$, sendo esses números denotados chaves. Cada vetor codifica uma única solução para o problema. Para intensificação, a criação de uma nova geração de soluções em um BRKGA envolve um processo de cruzamento. Nesse processo, um dos pais é sempre uma solução elite, definida como uma com valor alto de aptidão (no nosso caso, com *makespan* reduzido). O outro pai é uma solução que não está na população elite. Além disso, o pai elite tem uma maior probabilidade de passar suas características, i.e., chaves, para os descendentes. O BRKGA usa a estratégia de cruzamento uniforme paramétrico [Spears e De Jong, 1991] para combinar as soluções dos pais e gerar uma nova. Para diversificação, a cada nova geração, o BRKGA adiciona um conjunto de soluções geradas aleatoriamente, chamadas mutantes, na população. Ademais, uma implementação do BRKGA pode permitir seu reinício, i.e., uma nova população inicial é gerada a partir da qual o algoritmo deve continuar gerando soluções.

O BRKGA possui parâmetros que necessitam ser definidos para guiar o seu funcionamento. Eles são: o número de chaves aleatórias na solução (n), o tamanho da população (p), o tamanho da população elite (p_e), o tamanho da população mutante (p_m), a probabilidade de herança de uma chave do pai elite (ρ_e), e o número de gerações (n_g). Além disso, utilizamos um parâmetro (n_{nimp}) definindo o número máximo de gerações sem aprimoramento da melhor solução conhecida antes de o algoritmo reiniciar. Nossa implementação do BRKGA sempre reinicia o algoritmo inserindo na população inicial o melhor cromossomo obtido até o momento.

3.1. Codificação

No BRKGA para o problema de escalonamento de tarefas acopladas em uma única máquina minimizando o *makespan*, cada solução é codificada como um vetor X composto por $|J|$ chaves aleatórias. Neste vetor, a chave de índice j corresponde ao trabalho $j \in J$.

3.2. Decodificador

No BRKGA apresentado neste artigo, as chaves são usadas para definir uma sequência $\sigma = (j_1, \dots, j_n)$ dos trabalhos. Os trabalhos são escalonados consecutivamente utilizando uma heurística *first fit*, conforme a sequência σ . O k -ésimo trabalho na sequência é inserido na solução no momento mais cedo possível após o término da tarefa inicial do $(k - 1)$ -ésimo trabalho, sem sobreposição com as tarefas dos trabalhos em $(j_1, \dots, j_{k-1})$ já escalonadas.

4. Experimentos computacionais preliminares

Nesta seção, apresentamos alguns resultados computacionais preliminares realizados para avaliar o desempenho do modelo de programação por restrições e do BRKGA. Todos os experimentos foram executados em uma máquina rodando Ubuntu x86-64 GNU/Linux, com processador Intel Core i7-10700 Octa-Core 2.90 Ghz e 16Gb de RAM. O modelo de programação por restrições foi implementado em Minizinc versão 2.8.4 [Nethercote et al., 2007; Stuckey et al., 2014] e resolvido com o OR Tools CP-SAT versão 9.9.3963 [Perron e Didier, 2024]. O BRKGA foi implementado em Julia, baseado no arcabouço BrkgMpIpr [Andrade, 2019; Andrade et al., 2021].

4.1. Instâncias

Os testes utilizaram um subconjunto das instâncias do conjunto geral propostas em Khatami et al. [2019, 2020]. No conjunto geral, as instâncias são categorizadas em diferentes números de trabalhos, $n \in \{5, 10, 15, 20, 25, 40, 50, 100\}$. Para cada valor de n , há 30 instâncias geradas aleatoriamente em três categorias: pequenas (S), médias (M) e grandes (L). Ao todo são 240 instâncias. Cada uma das categorias possui 10 instâncias, nas quais o tempo de processamento das tarefas e a duração dos atrasos são retirados da distribuição uniforme discreta com os seguintes parâmetros: para a categoria S, $a_j, b_j \sim U(1, 20)$ e $L_j \sim U(10, 80)$; para a categoria M, $a_j, b_j \sim U(1, 50)$ e $L_j \sim U(25, 200)$; e para a categoria L, $a_j, b_j \sim U(1, 100)$ e $L_j \sim U(50, 400)$. As instâncias são representadas como $n_id_categoria_gen$, onde $id \in [1, 10]$ e “gen” indica que as instâncias fazem parte do conjunto geral proposto por Khatami et al. [2020]. Nos experimentos, consideramos as 24 instâncias do conjunto com $id = 2$.

4.2. Resultados do modelo de programação por restrições

A Tabela 1 apresenta uma comparação entre os resultados dos experimentos com o modelo de programação por restrições apresentado na Seção 2 e os melhores resultados das formulações de PI (i.e., métodos exatos) analisados recentemente na literatura. Os resultados reportados correspondem aos seguintes métodos exatos: modelo de programação por restrições (PR), formulação de PI de Békési et al. [2014] (MIP_P1), e formulação de PI de Khatami e Salehipour [2024] (MIP_P2). As colunas exibem o valor objetivo, o *gap* de otimalidade (em %), e o tempo em segundos para cada método exato. Os valores reportados para MIP_P1 e MIP_P2 são aqueles disponíveis em Khatami e Salehipour [2024]. A coluna “MSC” exibe o valor da melhor solução conhecida da literatura para cada instância. Os valores em negrito indicam os melhores valores conhecidos (incluindo aqueles encontrados por PR).

Conforme visto na Tabela 1, o modelo de programação por restrições provou otimalidade para todas as instâncias com até dez trabalhos, com tempo máximo de 3.1 segundos. PR também encontrou soluções melhores ou iguais às disponíveis na literatura para 23 das 24 instâncias testadas (95.8%). Além disso, PR obteve soluções estritamente melhores do que as da literatura para 17 das 24 instâncias testadas (70.8%). Observe que PR alcançou soluções viáveis de alta qualidade mesmo para instâncias com 100 trabalhos, para as quais as formulações de PI não conseguiram encontrar pelo menos uma solução viável. Além disso, embora PR tenha atingido o tempo limite para todas as instâncias com pelo menos 15 trabalhos, o modelo obteve um *gap* médio de 3.06%, sendo que o maior *gap* foi 5.97%.

Tabela 1: Comparação do modelo PR com os métodos exatos

instância	valor objetivo			gap (%)			tempo (s)			MSC
	PR	MIP_P1	MIP_P2	PR	MIP_P1	MIP_P2	PR	MIP_P1	MIP_P2	
5_2.S_gen	154	154	154	0.00	0.00	0.00	0.0	0.1	0.1	154
5_2.M_gen	302	302	302	0.00	0.00	0.00	0.0	0.0	0.0	302
5_2.L_gen	488	488	488	0.00	0.00	0.00	0.0	0.0	0.0	488
10_2.S_gen	217	217	217	0.00	0.00	0.00	1.2	23.4	6.7	217
10_2.M_gen	515	515	515	0.00	0.00	0.00	3.1	7.5	2.3	515
10_2.L_gen	1,004	1,004	1,004	0.00	0.00	0.00	2.8	9.2	6.5	1,004
15_2.S_gen	372	379	386	1.61	48.44	40.18	3,600.0	3,600.0	3,600.0	379
15_2.M_gen	965	969	978	2.69	46.52	38.86	3,600.0	3,600.0	3,600.4	969
15_2.L_gen	1,748	1,753	1,750	3.66	42.10	32.81	3,600.0	3,600.0	3,601.2	1,750
20_2.S_gen	419	431	428	1.67	66.13	58.93	3,600.0	3,600.0	3,600.0	428
20_2.M_gen	1,082	1,105	1,092	3.88	61.97	56.07	3,600.0	3,600.1	3,600.0	1,092
20_2.L_gen	1,949	2,028	1,978	4.16	58.90	63.82	3,600.0	3,600.0	3,600.0	1,978
25_2.S_gen	552	599	571	3.62	70.16	67.42	3,600.0	3,600.0	3,600.0	570
25_2.M_gen	1,524	1,607	1,604	4.53	73.96	69.65	3,600.0	3,600.1	3,600.0	1,573
25_2.L_gen	2,822	3,035	2,899	4.64	71.60	71.77	3,600.0	3,600.7	3,600.0	2,899
40_2.S_gen	757	-	-	3.57	-	-	3,600.0	3,600.0	3,600.0	763
40_2.M_gen	2,068	2,403	2,280	4.88	88.08	86.33	3,600.0	3,600.0	3,600.0	2,070
40_2.L_gen	4,202	13,357	-	5.57	83.22	-	3,600.0	3,600.1	3,600.1	4,160
50_2.S_gen	1,030	1,285	-	3.98	96.82	-	3,600.0	3,600.1	3,600.0	1,041
50_2.M_gen	2,568	7,848	2,875	4.32	91.56	85.97	3,600.0	3,600.1	3,600.0	2,593
50_2.L_gen	5,142	-	-	4.90	-	-	3,600.0	3,600.0	3,600.0	5,185
100_2.S_gen	2,266	-	-	4.59	-	-	3,600.0	3,600.0	3,600.0	2,306
100_2.M_gen	5,391	-	-	5.12	-	-	3,600.0	3,600.0	3,600.0	5,499
100_2.L_gen	10,425	-	-	5.97	-	-	3,600.0	3,600.0	3,600.0	10,607
Média				3.06	-	-	2,700.3	2,701.7	2,700.7	

A Tabela 2 apresenta os tempos para o modelo PR alcançar o melhor valor objetivo (obj) e o limite inferior (LB), além de exibir o limite inferior combinatório proposto por Khatami e Salehipour [2021] (LBc). Nota-se que o método PR conseguiu melhorar o limite inferior para todas as instâncias com até 15 trabalhos. No entanto, o modelo não conseguiu aprimorar o limite inferior para instâncias com pelo menos 20 trabalhos, embora tenha alcançado o limite inferior combinatório instantaneamente.

4.3. Resultados do BRKGA

Como mencionado na Seção 3, o BRKGA possui parâmetros que precisam ser configurados. Portanto, testes de treinamento foram realizados em 12 das 24 instâncias com $id = 1$, variando os valores dos parâmetros. Essas instâncias variam em tamanho (5 a 100 trabalhos), e categoria (S, M, L). A Tabela 3 apresenta os valores escolhidos nos testes de treinamento. Combinando os valores dos parâmetros, testamos ao todo 32 configurações. Dada a natureza estocástica do algoritmo, foram realizadas dez execuções independentes para cada instância com cada configuração.

A Tabela 4 apresenta as melhores configurações considerando três quesitos em relação aos valores objetivos obtidos nas dez execuções. Os quesitos são: a menor média dos mínimos (BRKGA_1), a menor média das médias (BRKGA_2) e menor média dos máximos (BRKGA_3). Nota-se que o parâmetro mais significativo em todos os quesitos foi a população $p = 150$. No entanto, alguns valores de parâmetros foram superiores em dois dos três quesitos. Os valores são: $p_e = 15$, $p_m = 10$, $\rho_e = 0.6$, $n_{nimp} = 1000$. Nota-se, pois, que a configuração BRKGA_2 assume tais valores de parâmetros, além de também não ser a pior configuração entre as três nos demais quesitos. Dessa forma, escolhemos como melhor configuração a BRKGA_2.

Tabela 2: Tempo para o melhor valor objetivo e melhor limite do modelo PR

instância	obj	tempo _{obj} (s)	LB	tempo _{LB} (s)	LBc
5.2.S.gen	154	0.0	154	0.0	140
5.2.M.gen	302	0.0	302	0.0	275
5.2.L.gen	488	0.0	488	0.0	438
10.2.S.gen	217	0.2	217	1.2	208
10.2.M.gen	515	1.7	515	3.1	491
10.2.L.gen	1,004	1.0	1,004	2.8	944
15.2.S.gen	372	760.8	366	2,409.7	364
15.2.M.gen	965	3,024.7	939	2,535.5	936
15.2.L.gen	1,748	932.2	1,684	3,374.7	1,674
20.2.S.gen	419	1,489.4	412	0.0	412
20.2.M.gen	1,082	2,952.9	1,040	0.0	1,040
20.2.L.gen	1,949	251.2	1,868	0.0	1,868
25.2.S.gen	552	170.5	532	0.0	532
25.2.M.gen	1,524	959.3	1,455	0.0	1,455
25.2.L.gen	2,822	1,896.4	2,691	0.0	2,691
40.2.S.gen	757	2,790.7	730	0.0	730
40.2.M.gen	2,068	2,138.1	1,967	0.0	1,967
40.2.L.gen	4,202	1,018.2	3,968	0.0	3,968
50.2.S.gen	1,030	3,537.3	989	0.0	989
50.2.M.gen	2,568	1,257.0	2,457	0.0	2,457
50.2.L.gen	5,142	1,932.8	4,890	0.0	4,890
100.2.S.gen	2,266	2,003.4	2,162	0.0	2,162
100.2.M.gen	5,391	3,539.1	5,115	0.0	5,115
100.2.L.gen	10,425	715.8	9,803	0.1	9,803

Tabela 3: Valores dos parâmetros escolhidos no BRKGA

Parâmetro	Descrição	Valores testados
p	tamanho da população	100 e 150
p_e	tamanho da população de elite	15 e 25
p_m	tamanho da população de mutantes	10 e 15
ρ_e	probabilidade de herança do alelo de elite	0.6 e 0.75
n_g	número de gerações	20,000
n_{imp}	intervalo para reinício da população	1,000 e 1,500

A Tabela 5 apresenta os resultados da heurística BRKGA_2 em comparação com as heurísticas da literatura para as instâncias com $id = 2$. A heurística “MRFF” (*multistart randomized first fit*) se refere a uma heurística com múltiplos reinícios que a cada iteração seleciona aleatoriamente uma ordem para as tarefas e escalona as tarefas da mesma forma que o decodificador do BRKGA (ver Seção 3.2). Para ser comparável com o BRKGA_2, executamos o MRFF com $n_G \times p = 20,000 \times 150 = 3,000,000$ de iterações. Observe que dez execuções independentes foram executadas para cada instância tanto para o BRKGA_2 quanto para o MRFF. As heurísticas da literatura foram apresentadas em Khatami e Salehipour [2024], nas quais BS_P1 e BS_P2 se referem aos algoritmos de busca binária incorporados com os problemas P1 de Békési et al. [2014] e P2 de Khatami e Salehipour [2024], e R&S se refere à metaheurística *relax-and-solve* de Khatami e Salehipour [2024]. Os tempos dessas heurísticas foram omitidos, uma vez que o tempo limite de 3,600 segundos foi excedido em todas as instâncias com pelo menos 15 trabalhos. Observe que essas são as instâncias de interesse para as heurísticas, já que aquelas com dez trabalhos ou menos foram resolvidas pelas abordagens exatas em alguns segundos. A coluna “gap_H (%)” apresenta o *gap* entre o valor objetivo mínimo e o limite inferior obtido pelo método de programação por restrições (ver Tabela 2). Os valores em negrito indicam que o valor objetivo da solução é menor ou igual à melhor solução das heurísticas analisadas.

Tabela 4: Melhores configurações de parâmetros

	p	p_e	p_m	ρ_e	n_{nimp}	Média - mín.	Média - médias	Média - máx.
BRKGA_1	150	25	10	0.75	1,500	2,296.2	2,343.3	2,374.3
BRKGA_2	150	15	10	0.6	1,000	2,300.5	2,336.9	2,372.7
BRKGA_3	150	15	15	0.6	1,000	2,307.4	2,337.7	2,366.3

Como pode ser visto na Tabela 5, conforme se aumenta o tamanho da instância, torna-se mais evidente a influência do processo evolutivo do BRKGA_2 no *gap* em relação à heurística puramente aleatória MRFF, reduzindo o *gap* médio de 10.1% para 5.5%. Embora o tempo médio do BRKGA_2 tenha aumentado de 28.0 segundos para 40.4 segundos em relação ao MRFF, a partir das instâncias com 15 trabalhos, o tempo é significativamente menor que os métodos heurísticos da literatura, uma vez que excederam o tempo limite de uma hora. Além disso, a heurística BRKGA_2 obteve médias do valor objetivo mínimo e médio (2,060.9 e 2,090.4, respectivamente) menores que as médias das heurísticas BS.P1 e BS.P2 (2,714.1 e 2,178.7, respectivamente). No entanto, a menor média do valor objetivo foi obtida pela heurística R&S (2,029.0). A heurística BRKGA_2 obteve o menor valor objetivo mínimo para as seis instâncias com 5 e 25 trabalhos e para uma instância de 15 trabalhos, ou seja, 7 instâncias das 24 instâncias testadas (29.2%). Por outro lado, a heurística R&S obteve menor valor objetivo para as seis instâncias com até 10 trabalhos, uma instância com 20 trabalhos e nove instâncias maiores ou iguais a 40 trabalhos, ou seja, 16 instâncias das 24 instâncias testadas (66.7%). Vale ressaltar, entretanto, que os tempos de execução do BRKGA são substancialmente menores do que os da heurística R&S.

5. Considerações finais

Neste artigo, consideramos o problema de escalonamento de tarefas acopladas em uma única máquina minimizando o *makespan*. Propusemos um modelo de programação por restrições e um algoritmo genético de chaves aleatórias enviesadas (BRKGA). Salvo melhor juízo, esta é a primeira vez que tais abordagens são propostas no contexto de escalonamento de tarefas acopladas.

Experimentos computacionais preliminares indicam que as abordagens propostas são promissoras para lidar com o problema. O modelo de programação por restrições obteve a melhor solução conhecida em todas as instâncias preliminares testadas, exceto uma. Soluções estritamente melhores do que as disponíveis na literatura foram obtidas para 70.8% das instâncias testadas. Além disso, o modelo obteve *gaps* significativamente menores que as formulações de programação inteira da literatura, encontrando o limite inferior combinatório disponível para o problema instantaneamente. Com relação à metaheurística BRKGA, ela se mostrou eficaz para encontrar boas soluções em tempo computacional bastante reduzido quando comparada às heurísticas estado-da-arte da literatura. Além disso, os resultados computacionais indicaram a relevância do processo evolutivo da heurística BRKGA na construção de soluções melhores para o problema.

Tabela 5: Resultados do BRKGA_2 e do RFF com as heurísticas da literatura

instância	BRKGA_2				MRFF				Heurísticas - Literatura		
	obj _{min}	obj _{med}	tempo _{med} (s)	gap _H (%)	obj _{min}	obj _{med}	tempo _{med} (s)	gap _H (%)	BS.P1	BS.P2	R&S
5_2.S_gen	154	154.0	8.4	0.0	154	154.0	2.8	0.0	154	154	154
5_2.M_gen	302	302.0	8.5	0.0	302	302.0	3.0	0.0	302	302	302
5_2.L_gen	488	488.0	8.8	0.0	488	488.0	3.2	0.0	488	488	488
10_2.S_gen	224	226.0	14.0	3.1	224	225.1	6.8	3.1	217	217	217
10_2.M_gen	526	532.9	12.3	2.1	526	526.6	5.9	2.1	515	515	515
10_2.L_gen	1,008	1,020.9	13.8	0.4	1,008	1,011.8	6.2	0.4	1,004	1,004	1,004
15_2.S_gen	385	389.1	18.0	4.9	391	394.4	8.1	6.4	384	379	385
15_2.M_gen	985	1,006.4	18.2	4.7	1,000	1,019.1	9.9	6.1	980	980	1,005
15_2.L_gen	1,766	1,821.3	18.1	4.6	1,815	1,864.6	8.2	7.2	1,780	1,780	1,784
20_2.S_gen	435	444.4	23.2	5.3	454	461.1	13.7	9.3	441	434	436
20_2.M_gen	1,121	1,144.3	23.2	7.2	1,176	1,186.5	13.7	11.6	1,109	1,109	1,107
20_2.L_gen	2,020	2,065.0	22.6	7.5	2,114	2,140.7	16.1	11.6	1,983	1,983	2,021
25_2.S_gen	561	572.7	28.5	5.2	591	601.1	16.1	10.0	580	580	570
25_2.M_gen	1,564	1,591.9	28.6	7.0	1,629	1,672.6	15.8	10.7	1,596	1,596	1,573
25_2.L_gen	2,876	2,923.7	28.8	6.4	3,005	3,086.0	20.9	10.4	2,923	2,923	2,910
40_2.S_gen	789	800.2	45.0	7.5	858	861.8	37.5	14.9	854	800	763
40_2.M_gen	2,140	2,165.7	41.5	8.1	2,314	2,334.0	31.1	15.0	2,596	2,166	2,070
40_2.L_gen	4,317	4,383.3	45.3	8.1	4,747	4,785.8	31.1	16.4	5,054	4,426	4,160
50_2.S_gen	1,068	1,080.9	57.4	7.4	1,169	1,181.4	40.0	15.4	1,348	1,074	1,041
50_2.M_gen	2,674	2,707.2	55.9	8.1	2,931	2,956.0	40.3	16.2	3,068	2,714	2,593
50_2.L_gen	5,404	5,454.7	58.4	9.5	5,867	5,908.1	38.7	16.7	6,520	5,577	5,185
100_2.S_gen	2,339	2,367.7	132.4	7.6	2,647	2,689.7	96.5	18.3	3,704	2,528	2,306
100_2.M_gen	5,558	5,627.7	125.6	8.0	6,347	6,385.9	112.2	19.4	9,544	6,166	5,499
100_2.L_gen	10,757	10,900.5	134.2	8.9	12,405	12,484.7	94.5	21.0	17,994	12,394	10,607
Média	2,060.9	2,090.4	40.4	5.5	2,256.8	2,280.0	28.0	10.1	2,714.1	2,178.7	2,029.0

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001. Este trabalho foi parcialmente apoiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), processos 314662/2020-0 e 314718/2023-0. Este trabalho é parcialmente apoiado pelo projeto FAPESB INCITE PIE0002/2022.

Referências

- Ageev, A. A. e Baburin, A. E. (2007). Approximation algorithms for UET scheduling problems with exact delays. *Operations Research Letters*, 35(4):533–540.
- Andrade, C. E. (2019). BrkgMpIpr.jl. <https://github.com/ceandrade/BrkgMpIpr.jl>. Acessado em: 25/05/2024.
- Andrade, C. E., Toso, R. F., Gonçalves, J. F., e Resende, M. G. (2021). The multi-parent biased random-key genetic algorithm with implicit path-relinking and its real-world applications. *European Journal of Operational Research*, 289(1):17–30.
- Bautista, M. G. A., Suganob, N. J., Concepcion, R., Bandala, A., e Dadios, E. (2023). Automated cooking systems: Benefits, challenges, and future directions. In *2023 IEEE 15th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, p. 1–6.
- Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160.
- Békési, J., Dósa, G., e Galambos, G. (2022). A first fit type algorithm for the coupled task scheduling problem with unit execution time and two exact delays. *European Journal of Operational Research*, 297(3):844–852.
- Békési, J., Galambos, G., Jung, M. N., Oswald, M., e Reinelt, G. (2014). A branch-and-bound algorithm for the coupled task problem. *Mathematical Methods of Operations Research*, 80(1): 47–81.
- Bessy, S. e Giroudeau, R. (2019). Parameterized complexity of a coupled-task scheduling problem. *Journal of Scheduling*, 22(3):305–313.
- Condotta, A. e Shakhlevich, N. (2014). Scheduling patient appointments via multilevel template: A case study in chemotherapy. *Operations Research for Health Care*, 3(3):129–144.
- Condotta, A. e Shakhlevich, N. V. (2012). Scheduling coupled-operation jobs with exact time-lags. *Discrete Applied Mathematics*, 160(16-17):2370–2388.
- Gonçalves, J. F. e Resende, M. G. C. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525.
- Jiang, A. Z. e Zhou, M. (2022). Design of affordable self-learning home cooking robots. In *2022 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, p. 1–6.
- Khatami, M. e Salehipour, A. (2021). A binary search algorithm for the general coupled task scheduling problem. *4OR*, 19(4):593–611.

- Khatami, M. e Salehipour, A. (2024). The coupled task scheduling problem: an improved mathematical program and a new solution algorithm. *International Transactions in Operational Research*, 31(4):2399–2426.
- Khatami, M., Salehipour, A., e Cheng, T. (2019). Data for “Coupled task scheduling with exact delays: Literature review and models”. URL <http://dx.doi.org/10.17632/dd7ht5k5pn.1>. Acessado em: 07/06/2024.
- Khatami, M., Salehipour, A., e Cheng, T. (2020). Coupled task scheduling with exact delays: Literature review and models. *European Journal of Operational Research*, 282(1):19–39.
- Li, H. e Zhao, H. (2007). Scheduling coupled-tasks on a single machine. In *2007 IEEE Symposium on Computational Intelligence in Scheduling*, p. 137–142. IEEE.
- Liu, Z., Lu, J., Liu, Z., Liao, G., Zhang, H. H., e Dong, J. (2019). Patient scheduling in hemodialysis service. *Journal of Combinatorial Optimization*, 37(1):337–362.
- Mepani, M. M., Gala, K. B., Mishra, T. A., Suresh Bhole, K., Gholave, J., e Daingade, S. (2022). Design of robot arm for domestic culinary assistance. *Materials Today: Proceedings*, 68: 1930–1945. ISSN 2214-7853. URL <https://www.sciencedirect.com/science/article/pii/S221478532205307X>. 4th International Conference on Advances in Mechanical Engineering.
- Nethercote, N., Stuckey, P. J., Becket, R., Brand, S., Duck, G. J., e Tack, G. (2007). Minizinc: Towards a standard CP modelling language. In Bessière, C., editor, *Principles and Practice of Constraint Programming – CP 2007*, p. 529–543, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Orman, A. J. e Potts, C. N. (1997). On the complexity of coupled-task scheduling. *Discrete Applied Mathematics*, 72(1-2):141–154.
- Pérez, E., Ntamo, L., Wilhelm, W. E., Bailey, C., e McCormack, P. (2011). Patient and resource scheduling of multi-step medical procedures in nuclear medicine. *IIE Transactions on Healthcare Systems Engineering*, 1(3):168–184.
- Perron, L. e Didier, F. (2024). CP-SAT Solver. URL https://developers.google.com/optimization/cp/cp_solver/. Acessado em: 03/06/2024.
- Shapiro, R. D. (1980). Scheduling coupled tasks. *Naval Research Logistics Quarterly*, 27(3):489–498.
- Sharma, K. e Reddy, S. R. N. (2024). Design of smart automated cooker_a survey and feasibility study. *Multimedia Tools and Applications*. ISSN 1573-7721. URL <http://dx.doi.org/10.1007/s11042-024-19764-x>.
- Sherali, H. D. e Smith, J. C. (2005). Interleaving two-phased jobs on a single machine. *Discrete Optimization*, 2(4):348–361.
- Spears, W. e De Jong, K. A. (1991). On the virtues of parameterized uniform crossover. In Belew, R. e Booker, L., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, p. 230–236, San Mateo. Morgan Kaufman.
- Stuckey, P. J., Feydy, T., Schutt, A., Tack, G., e Fischer, J. (2014). The Minizinc challenge 2008–2013. *AI Magazine*, 35(2):55–60.

- Yi, J.-s., Luong, T. A., Chae, H., Ahn, M. S., Noh, D., Tran, H. N., Doh, M., Auh, E., Pico, N., Yumbla, F., Hong, D., e Moon, H. (2022). An online task-planning framework using mixed integer programming for multiple cooking tasks using a dual-arm robot. *Applied Sciences*, 12(8): 4018. ISSN 2076-3417. URL <http://dx.doi.org/10.3390/app12084018>.
- Zhai, J., Pan, G., Yan, W., Fu, Z., e Zhao, Y. (2015). Dynamic analysis of a dual-arm humanoid cooking robot. In *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*, p. 835–838.