LVI SBPO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
DE 04 ATÉ 07 NOVEMBRO 2024
FORTALEZA | CEARÁ

Algoritmo GRASP aplicado ao problema de sequenciamento de tarefas com taxa de deterioração dependente da posição

Luan Millard Corrêa Brasil
Universidade Federal de Viçosa
Av. P H Rolfs, s/n - Campus Universitário, Viçosa - MG
luan.brasil@ufv.br

Mateus Guimarães dos Santos
Universidade Federal de Viçosa
Av. P H Rolfs, s/n - Campus Universitário, Viçosa - MG
mateus.g.santos@ufv.br

José Elias Claudio Arroyo
Departamento de Informática - Universidade Federal de Viçosa
Av. P H Rolfs, s/n - Campus Universitário, Viçosa - MG
jarroyo@ufv.br

RESUMO

O sequenciamento de tarefas em linhas de produção em paralelo faz parte da problemática diária de empresas de médio e grande porte, considerando seus recursos escassos e a elevada demanda. Este artigo aborda um problema de sequenciamento de tarefas em trabalhadores paralelos com deterioração dos tempos dependentes da posição, minimizando o atraso máximo com relação às datas de entrega das tarefas. Este problema é caracterizado como NP-Difícil devido à alta complexidade computacional para encontrar soluções ótimas através de métodos exatos. Sendo assim, propõe-se uma heurística GRASP simples e eficiente. O objetivo é comparar o desempenho da heurística GRASP em relação a uma heurística VND da literatura e ao modelo matemático de programação linear inteira do problema.

PALAVRAS CHAVE. Sequenciamento de tarefas, Máquinas Paralelas, GRASP, Otimização.

Tópicos: Meta-heurísticas, Otimização combinatória

ABSTRACT

Job sequencing in parallel production lines is part of the daily problem of medium and large companies, considering scarce resources and high demand. This paper addresses a job sequencing problem on parallel workers with position-dependent deterioration of times, minimizing the maximum tardiness with respect to job due dates. This problem is characterized as being NP-Hard due to the high computational complexity to find optimal solutions through exact methods. Therefore, a simple and efficient GRASP heuristic is proposed. The objective is to compare the performance of the GRASP heuristic in relation to a VND heuristic from the literature and the integer linear programming mathematical model of the problem.

KEYWORDS. Job sequencing. Parallel machines. GRASP. Otimization

Paper topics: Metaheuristics, Combinatorial optimization

DOI: 10.59254/sbpo-2024-193465
https://proceedings.science/p/193465?lang=pt-br

1. Introdução

Problemas de sequenciamento e programação de tarefas são questões globais que impactam diversos setores, como plantas industriais de médio e grande porte, hospitais e o setor automobilístico. A otimização dos processos é crucial para aumentar a eficiência e reduzir tempos de execução, seja na coordenação de operadores e trabalhadores administrativos em indústrias, na gestão de atendimentos hospitalares com recursos limitados e prioridades diversas, ou na operação de linhas de máquinas paralelas na construção de veículos. A pesquisa e desenvolvimento de soluções para esses problemas têm um impacto significativo na melhoria de processos e serviços na sociedade. Problemas similares ao problema abordado nesse estudo englobam o foco de alguns trabalhos desenvolvidos há mais de 35 anos, como em Gupta e Gupta [1988] e estão presentes em artigos atuais como Atsmony e Mosheiov [2022].

Este trabalho aborda o sequenciamento de tarefas (atividades como soldagem, atendimento médico, etc.), considerando a execução por pessoas ou máquinas. Variáveis importantes do problema incluem os operadores/máquinas, que podem trabalhar em paralelo, e o desempenho de cada trabalhador. Este desempenho é afetado por fatores como bem-estar social, saúde física e mental, que influenciam a qualidade e a velocidade do trabalho. O trabalho desenvolvido por Thompson e Goodale [2006] discute sobre a importância da busca por um melhor sequenciamento de tarefas em relação à diferença de produtividade existente entre os trabalhadores devido a condições adversas de trabalho como fadiga e a imperícia para o desenvolvimento das atividades. Máquinas também perdem eficiência com o tempo devido ao aquecimento ou desgaste dos componentes. Diferentemente de outros estudos que assumem um comportamento ideal constante como, por exemplo, em Boudreau et al. [2003], este artigo considera a perda de eficiência ao longo do tempo, devido ao cansaço ou aquecimento, e assume que cada tarefa em uma certa posição da sequência é concluída por qualquer trabalhador em um mesmo tempo.

No contexto de muitas tarefas e trabalhadores, considerando a deterioração do rendimento de cada operador, o problema de sequenciamento de tarefas se torna mais complexo em termos de modelagem matemática e computacional, resultando em tempos de processamento elevados para encontrar soluções viáveis. O problema de pesquisa deste artigo é um subproblema mais complexo do sequenciamento de tarefas em máquinas paralelas idênticas, cujo objetivo principal é encontrar o menor makespan global. Este problema é considerado NP-Difícil, pois requere-se de um elevado tempo computacional para determinar soluções ótimas, mesmo para instâncias com poucas máquinas e poucas tarefas. Ao incluir a queda de rendimento de máquinas e/ou pessoas, a complexidade aumenta, mantendo-se na classe de problemas NP-Difícil (Ji e Cheng [2009]).

Atualmente, métodos baseados em programação linear inteira mista são aplicados em muitas indústrias para encontrar soluções factíveis. Entretanto, com o aumento dos valores dos parâmetros do problema, como número de tarefas, número de trabalhadores disponíveis e taxas de perda de produtividade, o tempo de execução do método aumenta drasticamente. Devido à baixa eficiência computacional desses métodos para sistemas de produção extensos, estudos sobre algoritmos heurísticos e metaheurísticos têm ganhado destaque. Esses algoritmos visam encontrar soluções ótimas em tempo aceitável, com foco em otimizar processos já existentes dentro de grandes empresas. Eles permitem simular diversos comportamentos de programação de máquinas com suas restrições de recursos, preenchendo assim uma lacuna deixada por algoritmos menos eficientes. Essa abordagem tem ganhado espaço no campo da otimização combinatória e da pesquisa operacional.

Este artigo se baseia no trabalho de Montoya-Torres et al. [2023], que aborda o problema de escalonamento de tarefas para máquinas paralelas, considerando a taxa de deterioração de rendimento dependente da posição ao longo do tempo. Montoya-Torres et al. [2023] também apresenta

DOI: 10.59254/sbpo-2024-193465
https://proceedings.science/p/193465?lang=pt-br

um algoritmo heurístico de pesquisa de vizinhança variável (VND) para encontrar soluções eficientes em comparação com um modelo matemático de programação linear inteira mista (PLIM) construído por Moreno et al. [2018].

Este trabalho propõe um algoritmo metaheurístico GRASP (*Greedy Randomized Adaptive Search Procedure*), visando encontrar soluções mais eficientes do que a heurística VND proposta por Montoya-Torres et al. [2023] para o problema de sequenciamento de tarefas em máquinas paralelas idênticas considerando a taxa de deterioração dependente da posição.

2. Formulação do Problema

O problema investigado nesse artigo abrange o sequenciamento de um conjunto de n tarefas independentes (j = 1, 2, ..., n) que devem ser realizadas por um grupo de m trabalhadores ou máquinas (i = 1, 2, ..., m) que ao princípio apresentam plena capacidade física e técnica de executarem qualquer tarefa j. Cada trabalhador pode executar apenas uma tarefa j por vez, não sendo possível a interrupção da realização da mesma caso algum trabalhador já tenha começado a executá-la. Outra condição atrelada ao problema é que todas as tarefas e trabalhadores estão disponíveis no início do expediente, ou seja, todos estão aptos a realizar qualquer tarefa no início do período de planejamento (t = 0).

Cada tarefa j apresenta um tempo inicial p_j que representa o tempo de execução caso seja colocada na primeira posição de qualquer trabalhador. Além disso, cada tarefa j está associada a uma constante d_j que representa o prazo de entrega da tarefa.

O tempo de execução da tarefa depende da posição em que ela vai ser executada no sequenciamento das tarefas. O tempo de execução de uma tarefa j é determinada por $[p_j \times r^{\alpha}]$, onde *r* representa a posição de execução da tarefa, e α (> 0) é à taxa de deterioração do operador. Esta taxa representa a queda de rendimento devido à fadiga, ao cansaço físico e mental acumulado durante a realização das primeiras tarefas. Note que quando a tarefa é executada na primeira posição ($r = 1$), o tempo inicial não é alterado. Com intuito de representar essa queda de rendimento do trabalhador de modo mais realista possível, utilizou-se a deterioração exponencial, levantada como uma maneira eficiente de simular essa perda de eficiência. No trabalho desenvolvido por Moreno et al. [2018] é simulada a perda de eficiência referente à fabricação têxtil manual intensiva.

O atraso de uma tarefa é calculado como a diferença entre o tempo de conclusão C_j da tarefa e o prazo de entrega d_j. Caso essa diferença seja negativa, devido ao tempo de execução ser menor que o prazo d_j, o atraso é considerado igual a zero. O objetivo do problema consiste em encontrar o melhor sequenciamento de tarefas a serem executadas para cada trabalhador, no qual o atraso máximo seja o menor possível. Esse atraso máximo é calculado por $T_{\max} = \max\{T_j\}$, sendo $T_j = \max\{0, (C_j - d_j)\}$ o atraso da tarefa j.

Como o problema é NP-Difícil, o tempo computacional para encontrar as soluções ótimas cresce na ordem exponencial e torna-se totalmente inviável encontrar essas soluções para instâncias que abrangem uma grande quantidade de tarefas e um elevado número de trabalhadores. Consequentemente, é essencial desenvolver e implementar algoritmos heurísticos ou metaheurísticos que possam encontrar soluções de boa qualidade em tempos de computação viáveis e para instâncias de grande porte.

3. Heurística Proposta - GRASP

A heurística proposta para a solução do problema em estudo tem como inspiração a metaheurística GRASP, que foi proposta por Feo e Resende [1995]. A estrutura da metaheurística GRASP pode ser dividida em duas partes: a fase de construção de uma solução, que consiste em construir a solução de forma gulosa e com seleção aleatória, e a fase de melhoria dessa solução mediante uma busca local, encontrando um ótimo local. O algoritmo utiliza-se da repetição dessas

DOI: 10.59254/sbpo-2024-193465
https://proceedings.science/p/193465?lang=pt-br

duas partes por um número finito de vezes. Como o algoritmo usa escolhas aleatórias, diferentes soluções otimizadas são encontradas em cada iteração. A melhor solução encontrada após a execução de todas as iterações é retornada.

3.1. Fase de Construção da Solução Inicial

A solução inicial é construída ordenando as tarefas de forma decrescente do tempo inicial de processamento p_j de cada tarefa (regra LPT – *Longest Processing Time*). Esta ordenação determina uma $Lista$ de tarefas. Deste modo, as tarefas que necessitam de maior tempo inicial de processamento têm as maiores prioridades de execução em detrimento das tarefas que apresentam menores tempos iniciais de execução. A atribuição das tarefas aos *m* trabalhadores é feita da seguinte maneira. Seguindo a ordenação das tarefas, a primeira tarefa da $Lista$ é atribuída ao primeiro trabalhador, a segunda tarefa é atribuída ao segundo trabalhador e assim por diante, até que a m-ésima tarefa seja atribuída ao m-ésimo trabalhador, de modo que inicialmente todos os trabalhadores tenham pelo menos uma tarefa na sua linha de execução. As (*n-m*) tarefas restantes são atribuídas aos trabalhadores utilizando o algoritmo de inserção de Nawaz et al. [1983]. A próxima tarefa da $Lista$ ordenada (ou seja, a tarefa com maior tempo de processamento p_j) é inserida temporariamente em todas as possíveis posições de todos os trabalhadores. Para todas essas possibilidades de inserção, calcula-se o atraso máximo. Assim, sabendo qual o trabalhador e qual a posição a atribuição desta tarefa acarreta em um menor atraso máximo para a construção da solução, então a tarefa é inserida de forma definitiva na sequência (solução em fase de construção). Esse procedimento é realizado até que não tenha mais nenhuma tarefa na $Lista$ ordenada e, consequentemente, todas as tarefas estejam atreladas a um trabalhador.

Para construir diferentes soluções, a cada iteração do algoritmo GRASP, considera-se uma Lista de tarefas Candidatas (LC) formadas pelas u primeiras tarefas da $Lista$ ordenada. A tarefa a ser atribuída a um trabalhador é selecionada aleatoriamente da lista LC.

3.2. Fase de Melhoria da Solução

Posteriormente à construção de uma solução válida na qual todos as tarefas estão vinculadas aos trabalhadores, inicia-se a segunda parte referente à melhoria da solução. Nessa parte, é determinada uma vizinhança de soluções a partir da solução S construída na primeira parte do algoritmo. As soluções vizinhas são geradas através de uma pequena alteração na solução S. Essa alteração consiste em retirar a última tarefa de um trabalhador, e inserir essa mesma tarefa em cada possível posição para cada trabalhador disponível, inclusive para o trabalhador em que a tarefa foi removida, gerando uma solução vizinha Sv diferente para cada inserção desta. Esse procedimento é realizado para todos os m trabalhadores. Para todas as soluções vizinhas geradas calcula-se os possíveis atrasos máximos ($Tmax$). Caso alguma solução vizinha apresente um menor atraso máximo quando comparado a solução referência S, a solução referência passa a ser a solução vizinha de menor atraso. Esse processo é repetido até que em uma vizinhança não tenha uma solução vizinha com a função objetivo melhor que a solução de referência. Neste trabalho, a busca local gera no máximo NI_BL vizinhanças, ou seja, o número máximo de iterações da busca local é NI_BL.

3.3. Estrutura da heurística GRASP

A fim de aumentar a eficiência da heurística GRASP, a construção da primeira solução é realizada de forma gulosa, ou seja, considera-se uma lista de candidatos (*LC*) de tamanho unitário ($u = 1$). A próxima tarefa a ser atribuída a um trabalhador é a tarefa de maior prioridade. As outras iterações da heurística GRASP constrói a solução de referência de maneira gulosa e aleatória. Utiliza-se uma (*LC*) de tamanho $u > 1$. Ou seja, a próxima tarefa a ser atribuída a um trabalhador é escolhida de maneira aleatória entre as u tarefas com maiores prioridades.

DOI: 10.59254/sbpo-2024-193465
https://proceedings.science/p/193465?lang=pt-br

O pseudocódigo da heurística GRASP é descrito no Algoritmo 1. O algoritmo possui 3 parâmetros de entrada: tamanho da Lista de tarefas Candidatas (u), número de iterações da busca local (NI_BL) e número de iterações do algoritmo GRASP (NI_G). O algoritmo completo, incluindo as duas partes, é executado no máximo (NI_G) vezes. Caso seja encontrada uma solução com atraso máximo igual a zero, o algoritmo finaliza. A solução com menor atraso máximo (S_g) encontrada em todo o processo de busca é retornada pelo algoritmo GRASP.

Algorithm 1 : GRASP (u, NI_BL, NI_G)

$It \leftarrow 0$
$Lista[] \leftarrow$ ordenar as tarefas em ordem decrescente dos tempo p_j
for $i \leftarrow 1$ até m **do**
    Alocar a tarefa $Lista[i]$ para o trabalhador i
end for
if $It == 0$ **then**
    Tamanho da Lista de Candidatas (LC): $|LC| \leftarrow 1$
else
    Tamanho da Lista de Candidatas (LC): $|LC| \leftarrow u$ (LC é formada pelas u primeiras tarefas de $Lista$)
end if
while $Lista \neq \emptyset$ **do** *//constrói uma solução S:*
    Escolher aleatoriamente uma tarefa j de LC
    Encontrar o melhor trabalhador i e a melhor posição k (i.e. que minimiza T_{max}) para inserir a tarefa j
    Remover a tarefa j da lista $Lista$
end while
$Tmax_g \leftarrow Tmax(S)$
$S_g \leftarrow S$
$Ib \leftarrow 0$
while $Ib < NI_BL$ **do**
    Construir uma vizinhança de soluções a partir da solução S_g
    Calcular $Tmax$ para cada solução vizinha
    for cada solução vizinha Sv **do**
        if $Tmax(Sv) < Tmax_g$ **then**
            $Tmax_g \leftarrow Tmax(Sv)$
            $S_g \leftarrow Sv$
        end if
    end for
    if $Tmax_g == 0$ **then**
        break **while**
    end if
    $Ib \leftarrow Ib + 1$
end while
if $Tmax_g == 0$ ou $It == NI_G$ **then**
    return $S_g, Tmax_g$
else
    $It \leftarrow It + 1$
    Repetir os passos 2 até 36.

3.4. Principais Parâmetros da Heurística GRASP

Entre os parâmetros do algoritmo heurístico GRASP, destacam-se três que, calibrados, permitem uma maior eficiência. O primeiro parâmetro é u, que corresponde ao tamanho da Lista de tarefas Candidatas, ou seja as terefas com maior prioridade. Uma destas tarefas é escolhida de forma aleatória para ser inserida na solução construída. Na primeira iteração do algoritmo GRASP utilizou-se uma lista de tamanho unitário, ou seja, sempre escolhe-se a primeira tarefa (a tarefa de maior prioridade). A partir da segunda iteração, utiliza-se uma Lista de tarefas Candidatas de tamanho $u = 4$. Este valor foi fixado após realizar uma calibração baseada em experimentos computacionais.

O segundo parâmetro fundamental para a heurística abrange o número máximo de vizinhanças que serão construídas na busca local (NI_BL). O valor para o parâmetro NI_BL foi

DOI: 10.59254/sbpo-2024-193465
https://proceedings.science/p/193465?lang=pt-br

fixado em 100, significando que, a cada iteração do algoritmo são geradas no máximo 100 vizinhanças determinadas pela inserção de tarefas.

O terceiro parâmetro é o número máximo de iterações do algoritmo GRASP (NI_G). Esse parâmetro está diretamente vinculado à eficiência e ao tempo total de execução do algoritmo heurístico apresentado. O valor utilizado para este parâmetro foi de 300.

Outro ponto relevante sobre o algoritmo que impacta densamente a eficiência das soluções encontradas é a maneira como é realizada a ordenação inicial de prioridade das tarefas para a construção de uma solução inicial viável, gulosa e aleatória. As duas variáveis conectadas diretamente a essa ordenação consistem no tempo de processamento de cada tarefa na posição inicial do trabalhador p_j e o prazo determinado para a realização de cada tarefa d_j. Neste artigo foi investigado quatro possibilidades de ordenação inicial de prioridade destas tarefas: por ordem crescente de prazo d_j, por ordem decrescente de tempo de processamento inicial p_j, por ordem crescente entre a razão do prazo e o tempo processamento inicial (d_j/p_j) e por ordem decrescente da razão entre o tempo de processamento inicial da tarefa pelo prazo de execução da tarefa (p_j/d_j). Os melhores resultados foram obtidos ordenando as tarefas de forma decrescente do tempo inicial de processamento p_j (regra LPT).

4. Experimentos Computacionais

Os dados que serviram de base para os experimentos computacionais foram gerados por Montoya-Torres et al. [2023] e armazenados em quatro arquivos de texto com diferentes tamanhos de problema. Cada arquivo de dados contém uma enumeração de tarefas, em que p corresponde a uma lista que representa os tempos iniciais de execução de cada tarefa e d consiste em uma lista correspondente aos prazos de entrega de cada respectiva tarefa.

O banco de dados utilizado para conduzir as simulações do modelo e aplicar a heurística é composto por um total de 240 instâncias. Dentre essas 240 instâncias, metade delas foi submetida à simulação com uma taxa de deterioração de 0,2, enquanto a outra metade foi simulada com uma taxa de 0,8.

Além disso, executou-se o conjunto total de 240 instâncias de simulações, divididas entre quatro cenários distintos com 40, 50, 100 e 200 tarefas, respectivamente. Dentro deste cenário, para cada 20 instâncias simuladas, ainda tinha o diferente contexto de 3, 7 e 10 trabalhadores. Os parâmetros que regem as instâncias do problema estão descritos na Tabela 1.

Tabela 1: Parâmetros das instância do problema.

Parâmetros	Fatores	Valores
Número de tarefas	4	40, 50, 100, 200
Número de trabalhadores	3	3, 7, 10
Taxa de deterioramento	2	0,2, 0,8
Número de instâncias simuladas combinação	10	-
Total de Instâncias	-	240

As variáveis destacadas representam o número de tarefas n, número de trabalhadores m, as taxas de deterioração α = 0,2, 0,8, e o número de 10 instâncias simuladas para cada combinação desses parâmetros independentes, gerando 240 instâncias totais.

4.1. Resultados Computacionais

Com intuito de validar e comprovar a eficiência do algoritmo heurístico GRASP introduzido em relação ao atraso máximo da tarefa e ao tempo de processamento, realizaram-se simulações

DOI: 10.59254/sbpo-2024-193465
https://proceedings.science/p/193465?lang=pt-br

utilizando o conjunto de 240 instâncias já descritas anteriormente. Foi utilizado um PC com processador Intel Core i7 7500 U, 2,70 GHz e 8 GB de RAM.

Com a intenção de comparar as soluções encontradas pelo GRASP proposto neste artigo com o modelo matemático proposto por Moreno et al. [2018] e com a heurística VND proposta por Montoya-Torres et al. [2023] em cada uma das 240 instâncias simuladas, utilizou-se o Índice de Desvio Relativo (IDR) que consiste em uma medida que associa a solução encontrada pelo método analisado com a melhor e pior solução obtida entre todos os métodos comparados. O IDR é calculada conforme a equação (1).

$$IDR(\%) = \frac{Sol_{\text{Método}} - Sol_{\text{Melhor}}}{Sol_{\text{Pior}} - Sol_{\text{Melhor}}} * 100, \quad (1)$$

onde $Sol_{\text{Método}}$, Sol_{Melhor} e Sol_{Pior} representam respectivamente a solução encontrada pelo método analisado, a melhor e pior solução encontrada entre os métodos, para uma determinada instância. O IDR foi proposta por Zemel [1981] e utilizado em trabalhos como Kim [1993] e Kim et al. [1996] com a intenção de comparar a qualidade das soluções encontradas para problemas de sequenciamento de tarefas, além de evitar o problema que alguns outros índices comparativos apresentam devido à indivisibilidade por zero, quando a solução de referência ou a melhor solução do problema possui atraso máximo igual a zero.

Deste modo, o IDR representa um índice que varia entre 0 e 100 para cada método, na qual uma boa solução apresenta um índice muito próximo de 0 e uma péssima solução obtém um índice próximo de 100. Vale ressaltar que, se as piores e melhores soluções tiverem o mesmo valor, todos os métodos fornecerão a mesma solução ótima, portanto, o índice será 0 para todos os métodos.

Os resultados encontrados para as instâncias do problema nos cenários de 40, 50, 100 e 200 tarefas são exibidos na Tabela 2. Destaca-se que a Tabela 2 não apresenta as instâncias onde os métodos GRASP, VND e PLIM comutativamente encontraram o atraso máximo igual a zero. Do total de 240 instâncias, encontrou-se o atraso máximo igual a zero em 148 instâncias. Assim, a Tabela 2 apresenta os resultados para 92 instâncias.

Conforme a Tabela 2, o algoritmo GRASP, para os problemas que envolvem 40, 50 e 100 tarefas, precisou na média de milissegundos para encontrar a melhor solução. Diferente do PLIM, que precisou na média de segundos para encontrar a melhor solução. O tempo computacional do GRASP foi competitivo quando comparado ao algoritmo VND.

Tabela 2: Resultados para as Instâncias com 40, 50, 100 e 200 tarefas

				PLIM			GRASP			VND		
ID	n	m	α	Tmax	IDR(%)	Tempo(s)	Tmax	IDR(%)	Tempo(s)	Tmax	IDR(%)	Tempo(s)
1	40	3	0,8	**731**	0	78,5	787	100	12,3	762	55,36	9,1
2	40	3	0,8	**620**	0	47,3	669	100	13,2	647	55,10	7,6
3	40	3	0,8	**475**	0	298,6	498	95,83	13,1	499	100	7,7
4	40	3	0,8	**538**	0	332,1	597	100	20,8	569	52,54	7,5
5	40	3	0,8	**740**	0	202,6	763	95,83	20,8	764	100	6,3
6	40	3	0,8	**1003**	0	165,5	1035	88,89	16,1	1039	100	8,5
7	40	3	0,8	**923**	0	355	970	100	17,4	950	57,45	5,1
8	40	3	0,8	**855**	0	172	873	90	19,2	875	100	8,8
9	40	3	0,8	**1095**	0	369,2	1152	100	18,9	1121	45,61	9,3
10	40	3	0,8	**1319**	0	125,5	1374	80,88	20,7	1387	100	8,8
1	50	3	0,8	**1605**	0	625,8	1642	78,72	69,5	1652	100	7,9
2	50	3	0,8	**1439**	0	411,4	1476	72,55	85,2	1490	100	11,0

DOI: 10.59254/sbpo-2024-193465
https://proceedings.science/p/193465?lang=pt-br

Tabela 2: Continuação.

				PLIM			GRASP			VND		
ID	n	m	α	Tmax	IDR(%)	Tempo(s)	Tmax	IDR(%)	Tempo(s)	Tmax	IDR(%)	Tempo(s)
3	50	3	0,8	**1107**	0	314,1	1147	100	83,3	1145	95	10,0
4	50	3	0,8	**1382**	0	138,3	1421	86,67	58,5	1427	100	10,4
5	50	3	0,8	**1394**	0	1000	1425	88,57	54,9	1429	100	8,7
6	50	3	0,8	**2180**	0	98,9	2244	100	43	2210	46,88	7,9
7	50	3	0,8	**2138**	0	138,4	2199	98,39	56,1	2200	100	6,4
8	50	3	0,8	**1493**	0	1000	1518	78,13	58,5	1525	100	9,1
9	50	3	0,8	**1166**	0	1000	1222	100	56,7	1192	46,43	5,5
10	50	3	0,8	**1762**	0	1000	1799	88,10	65,1	1804	100	6,1
6	100	3	0,2	41	100	1000	**0**	0	1	**0**	0	4,2
1	100	3	0,8	**7778**	0	1000	7820	100	287,0	7819	97,62	135,2
2	100	3	0,8	8120	100	1000	7621	12	283,4	**7547**	0	84,2
3	100	3	0,8	**6767**	0	1000	6939	100	281,0	6812	26,16	92,9
4	100	3	0,8	**6285**	0	1000	6435	100	276,1	6365	53,33	70,3
5	100	3	0,8	**6935**	0	1000	7047	100	275,6	7030	84,82	82,4
6	100	3	0,8	**8830**	0	1000	8882	100	272,1	8867	71,15	57,6
7	100	3	0,8	8796	100	1000	**8070**	0	278,4	8089	2,62	58,1
8	100	3	0,8	**8882**	0	1000	8964	100	277,1	8921	47,56	79,5
9	100	3	0,8	9028	100	1000	8974	69,32	277,7	**8852**	0	101,5
10	100	3	0,8	**7943**	0	1000	8075	100	278,6	8062	90,15	74,7
6	100	7	0,8	125	100	1000	**0**	0	1,7	**0**	0	7,5
1	200	3	0,2	18307	100	1000	**1580**	0	1009	1655	0,45	1000
2	200	3	0,2	12356	100	1000	**1437**	0	1009	1540	0,94	830,7
3	200	3	0,2	21230	100	1000	**1783**	0	1004	2119	1,73	1000
4	200	3	0,2	22057	100	1000	**2533**	0	1012	2577	0,23	1000
5	200	3	0,2	17199	100	1000	**2498**	0	1002	2599	0,69	1000
6	200	3	0,2	19317	100	1000	**3543**	0	1005	3607	0,41	1000
7	200	3	0,2	16355	100	1000	**3567**	0	1001	3674	0,84	916
8	200	3	0,2	24111	100	1000	**3701**	0	1008	3925	1,10	1000
9	200	3	0,2	24022	100	1000	**4600**	0	1010	4714	0,59	1000
10	200	3	0,2	5369	100	1000	**4539**	0	1003	4854	37,95	1000
1	200	3	0,8	249218	100	1000	**37349**	0	1009	37516	0,08	1000
2	200	3	0,8	355647	100	1000	32856	0,10	1007	**32526**	0	948,4
3	200	3	0,8	391699	100	1000	36057	0,03	1010	**35941**	0	1000
4	200	3	0,8	271351	100	1000	37197	0,01	1008	**37165**	0	620,3
5	200	3	0,8	411140	100	1000	37144	0,02	1009	**37053**	0	1000
6	200	3	0,8	363946	100	1000	37229	0,05	1015	**37078**	0	718,3
7	200	3	0,8	377179	100	1000	38661	0,03	1013	**38570**	0	1000
8	200	3	0,8	371798	100	1000	**37176**	0	1005	37227	0,02	1000
9	200	3	0,8	394027	100	1000	39093	0,07	1001	**38849**	0	1000
10	200	3	0,8	407790	100	1000	**37479**	0	1010	37619	0,04	1000
1	200	7	0,2	20655	100	1000	**0**	0	1002	**0**	0	0,0
2	200	7	0,2	12095	100	1000	**0**	0	1004	**0**	0	0,0
3	200	7	0,2	21038	100	1000	**0**	0	1002	**0**	0	0,1
4	200	7	0,2	10687	100	1000	**0**	0	1002	**0**	0	0,0

DOI: 10.59254/sbpo-2024-193465
https://proceedings.science/p/193465?lang=pt-br

Tabela 2: Continuação.

				PLIM			GRASP			VND		
ID	n	m	α	Tmax	IDR(%)	Tempo(s)	Tmax	IDR(%)	Tempo(s)	Tmax	IDR(%)	Tempo(s)
5	200	7	0,2	21670	100	1000	**0**	0	1007	**0**	0	0,0
6	200	7	0,2	22455	100	1000	66	0,29	1001	**0**	0	210,1
7	200	7	0,2	16144	100	1000	83	0,51	1003	**0**	0	257,1
8	200	7	0,2	23207	100	1000	300	0,92	1001	**88**	0	1000
9	200	7	0,2	23591	100	1000	**801**	0	1006	824	0,10	990,7
10	200	7	0,2	22813	100	1000	**777**	0	1008	845	0,31	1000
1	200	7	0,8	400544	100	1000	**4794**	0	1007	5093	0,08	1000
2	200	7	0,8	375152	100	1000	4105	0,01	1003	**4061**	0	926
3	200	7	0,8	392694	100	1000	**4898**	0	1006	5165	0,07	1000
4	200	7	0,8	394927	100	1000	**5641**	0	1003	5691	0,01	1000
5	200	7	0,8	394100	100	1000	5633	0,01	1007	**5599**	0	891,8
6	200	7	0,8	406731	100	1000	6544	0,03	1003	**6430**	0	835,8
7	200	7	0,8	394743	100	1000	6810	0,03	1001	**6712**	0	1000
8	200	7	0,8	396811	100	1000	**6769**	0	1005	6942	0,04	1000
9	200	7	0,8	401739	100	1000	7709	0	1004	**7695**	0	1000
10	200	7	0,8	386347	100	1000	**7480**	0	1006	7626	0,04	1000
1	200	10	0,2	20048	100	1000	**0**	0	1006	**0**	0	0
2	200	10	0,2	18825	100	1000	**0**	0	1002	**0**	0	0
3	200	10	0,2	20429	100	1000	**0**	0	1004	**0**	0	0
4	200	10	0,2	6086	100	1000	**0**	0	1001	**0**	0	0
5	200	10	0,2	20555	100	1000	**0**	0	1006	**0**	0	0
6	200	10	0,2	21773	100	1000	**0**	0	1002	**0**	0	0
7	200	10	0,2	21943	100	1000	**0**	0	1001	**0**	0	0
8	200	10	0,2	22129	100	1000	16	0,07	1000	**0**	0	634,4
9	200	10	0,2	23167	100	1000	176	0,40	1001	**84**	0	1000
10	200	10	0,2	22003	100	1000	**170**	0	1006	171	0	1000
1	200	10	0,8	389285	100	1000	**482**	0	1003	726	0,06	1000
2	200	10	0,8	361145	100	1000	**235**	0	1002	321	0,02	594,5
3	200	10	0,8	378846	100	1000	780	0	1007	**764**	0	1000
4	200	10	0,8	388766	100	1000	1481	0,01	1003	**1447**	0	817
5	200	10	0,8	378534	100	1000	1395	0,01	1012	**1374**	0	809
6	200	10	0,8	389487	100	1000	**2398**	0	1007	2505	0,03	375,8
7	200	10	0,8	383802	100	1000	**2528**	0	1004	2609	0,02	1000
8	200	10	0,8	367689	100	1000	**2714**	0	1007	2869	0,04	1000
9	200	10	0,8	390504	100	1000	**3558**	0	1000	3600	0,01	1000
10	200	10	0,8	368873	100	1000	3440	0,03	1004	**3332**	0	810

No cenário de 40 tarefas, a Tabela 2 exibe 10 instâncias nas quais todas possuem: m = 3, α = 0.8. O algoritmo GRASP supera o algoritmo VND em 5 destas instâncias, ou seja, em 50% das instâncias analisadas. O PLIM, para todas as instâncias, encontra a solução ótima exata. No referencial temporal, o GRASP supera o PLIM em todas as 10 instâncias, chegando a ser, na média, 10 vezes mais rápido. Entretanto, não é tão rápido quanto o VND, que chega a ser 20 vezes mais rápido.

Para o contexto de 50 tarefas, a Tabela 2 mostra 10 instâncias nas quais todas possuem:

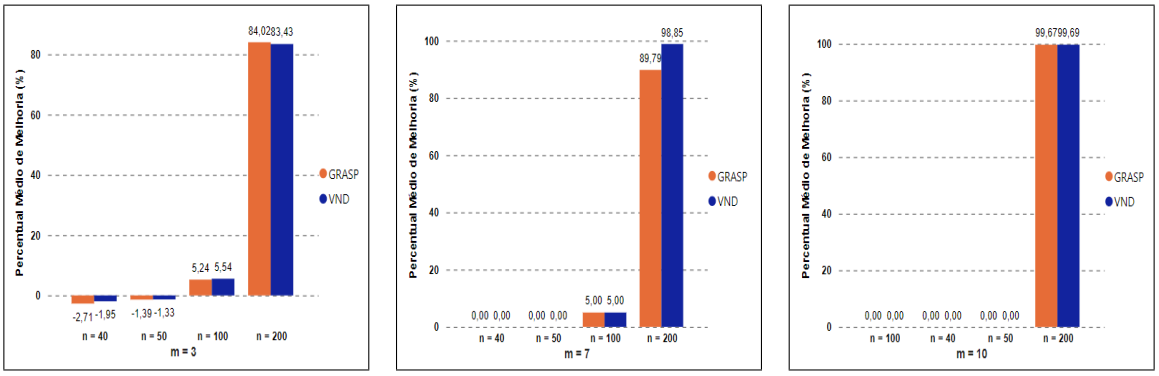
DOI: 10.59254/sbpo-2024-193465
https://proceedings.science/p/193465?lang=pt-br

m = 3, α = 0.8. A heurística GRASP supera a heurística VND em 7 instâncias, ou seja, em 70% das instâncias analisadas. O PLIM, para todas as instâncias, encontra a solução ótima exata. No referencial temporal, o GRASP supera o PLIM em todas as 10 instâncias, atingindo a eficiência de ser 6 vezes mais rápido, entretanto, não é tão rápido quanto o VND que atinge a média de 36 vezes mais rápido.

Para o cenário de 100 tarefas, são exibidas 12 instâncias na qual o GRASP supera o VND em uma instância e é competitivo nas outras 11 instâncias. Em relação ao PLIM, o GRASP apresenta resultados melhores em 5 instâncias, o que representa uma melhora de 41,67% nas instâncias em que foram encontrados atrasos. No quesito temporal, o GRASP é 4 vezes mais eficiente que o PLIM e o VND é 10 vezes mais eficiente que o PLIM.

No contexto de 200 tarefas, a Tabela 2 exibe as 60 instâncias. A metaheurística GRASP supera o PLIM em todas as instâncias. Em relação à heurística VND, o GRASP apresenta melhores resultados para o atraso máximo em 27 instâncias, representando uma melhora de 45% em todas as instâncias. Nas outras 33 instâncias, o algoritmo introduzido GRASP é bem competitivo. No tempo, o GRASP e o PLIM apresentam um tempo médio de execução de 1000 segundos e o VND o tempo médio de 609 segundos.

Com foco ainda em validar os resultados, calculou-se o percentual médio de melhoria de cada método (GRASP e VND) em relação ao PLIM, buscando analisar a eficiência dos algoritmos mediante uma visão mais ampla. Essa análise foi realizada separando em blocos as instâncias que apresentam o mesmo número de tarefas e o mesmo número de trabalhadores disponíveis. Os resultados são apresentados na Figura 1.



(a) 3 trabalhadores. (b) 7 trabalhadores. (c) 10 trabalhadores.

Figura 1: **Percentual médio de melhoria em relação ao modelo PLIM para os cenários com 3, 7 e 10 trabalhadores e 40, 50, 100 e 200 tarefas.**

Conforme apresentado na Figura 1.a), para o cenário de 3 trabalhadores, a heurística GRASP demonstrou uma melhora média de cerca de 84,82% em relação ao PLIM para o problema de 200 tarefas, tendo uma performance superior à heurística VND que apresentou uma melhora média de 83,43%. Para o problema de 100 tarefas, o GRASP teve um comportamento competitivo em relação ao VND, apresentando uma melhora média de 5,24% contra 5,54% respectivamente. No contexto envolvendo 40 e 50 tarefas, o GRASP obteve valores de atraso máximo na média de 2,71% e 1,39%, respectivamente, maiores para o problema, mostrando ser um método competitivo apesar do resultado médio ser levemente inferior ao PLIM. O VND também apresentou uma performance inferior ao PLIM de 1,95% e 1,32%, respectivamente, para o problema de 40 e 50 tarefas.

Na Figura 1.b), dentro do contexto de 7 trabalhadores, evidencia-se que o método GRASP

DOI: 10.59254/sbpo-2024-193465
https://proceedings.science/p/193465?lang=pt-br

obteve uma melhora média de 89,79% quando relacionado ao PLIM para o problema de 200 tarefas. O método VND para esse caso teve uma performance superior ao PLIM de 96,85%. Dentro do contexto de 40, 50 e 100 tarefas, o GRASP teve um desempenho igual ao VND. Para 100 tarefas, ambos tiveram uma melhora média de 5% dos resultados de atraso máximo em relação ao PLIM. Já para 40 e 50 tarefas, o GRASP e o VND atingiram os mesmos resultados de atraso máximo que o PLIM, não apresentando melhora.

Segundo a Figura 1.c), para o cenário de 10 trabalhadores, destaca-se que o GRASP teve um resultado igual ao método VND, quando comparados ao PLIM. O GRASP melhorou na média os atrasos em 99,67% e o VND melhorou em 99,69% para o problema de 200 tarefas. Para o problema envolvendo 40, 50 e 100 tarefas, o GRASP e o VND encontraram os mesmos atrasos que o PLIM, não apresentando melhora na média.

Destaca-se então que, para 3, 7 e 10 trabalhadores, a heurística GRASP apresenta um tempo médio de execução consideravelmente menor que o modelo matemático para os cenários de 40, 50 e 100 tarefas. Já em relação ao tempo do VND, o algoritmo introduzido apresenta tempos superiores, porém competitivos para cada instância. Para o cenário de 200 tarefas, os tempos de execução do GRASP e do PLIM são similares, e o tempo do VND é um pouco inferior.

5. Conclusões

Neste trabalho foi proposto uma heurística GRASP simples e eficiente para o problema de sequenciamento de tarefas com trabalhadores em paralelo, considerando taxas de deterioração e minimizando o atraso máximo das tarefas. Com base nos resultados apresentados, tanto no quesito do valor da função objetivo (atraso máximo) como no quesito de tempo de execução dos algoritmos, pode-se concluir que a heurística proposta GRASP é bastante eficiente e competitivo em relação à heurística VND proposta por Montoya-Torres et al. [2023] para todos os cenários apresentados do problema. Embora o PLIM, desenvolvido em Moreno et al. [2018], encontre as soluções ótimas na maior parte das instâncias para cenários com menores quantidades de tarefas, como 40 e 50, esse comportamento não acontece para contextos de 100 e 200 tarefas. Além disso, nos cenários de 100 e 200 tarefas, o modelo matemático é considerado lento quando comparado às heurísticas GRASP e VND, chegando a ter uma performance temporal até 60 vezes pior.

Os resultados em relação à qualidade do valor da função objetivo também indicam que o GRASP, quando comparado à heurística VND, possui maior impacto para o pior contexto possível do problema apresentado, ou seja, quando o número de tarefas é 200 e o número de trabalhadores disponíveis é 3. Para muitos cenários, como por exemplo para 50 e 100 tarefas abrangendo todas as possíveis situações de trabalhadores disponíveis, 3, 7 e 10, a heurística GRASP atingiu resultados similares ao algoritmo VND. Além disso, analisando o quesito temporal, o GRASP é eficiente quando comparado ao PLIM, e chega a atingir um comportamento próximo ao algoritmo VND, ambos para situações que envolvem grandes quantidades de tarefas. Entretanto, para atmosferas que abarcam menores quantidades de tarefas, quando comparado com o PLIM, a heurística GRASP continua com tempos menores ou próximos. Em relação ao VND, os períodos de tempo utilizados pelo GRASP para atingir as soluções ótimas por aproximação são levemente maiores.

Vistos esses pontos, enaltece que o problema de sequenciamento de tarefas com trabalhadores operando em paralelo às tarefas demandadas por si só já é considerado um problema NP-Difícil, devido à sua elevada complexidade requerida para a resolução de muitos cenários e instâncias distintos. Deste modo, acrescentando as condições de taxa de deterioração da eficiência de execução de cada tarefas ao decorrer da linha produtiva, aumenta ainda mais essa complexidade do problema exaltando a necessidade de estudos e pesquisas mais profundas direcionadas em algoritmos e modelagens matemáticas com intuito de encontrar melhores maneiras de resolver essa

DOI: 10.59254/sbpo-2024-193465
https://proceedings.science/p/193465?lang=pt-br

problemática. Assim, estudos com foco em heurísticas e metaheurísticas como o algoritmo proposto nesse artigo (GRASP), são maneiras que apresentam um grande potencial para otimizar tanto os valores das funções objetivos (como o atraso máximo das tarefas), como também os recursos escassos de grandes empresas.

Como trabalhos futuros pode ser realizada uma análise mais aprofundada dos parâmetros da heurística GRASP utilizando a ferramenta IRace [López-Ibáñez et al., 2016], e a realização de testes estatísticos para validar os resultados obtidos.

Agradecimentos

Os autores agradecem o apoio financeiro da CAPES.

Referências

Atsmony, M. e Mosheiov, G. (2022). Minimizing total completion time with linear deterioration: A new lower bound. *Computers & Industrial Engineering*, 163:107867.

Boudreau, J. W., Hopp, W., McClain, J. O., e Thomas, L. J. (2003). On the interface between operations and human resources management. *Manufacturing & Service Operations Management*, 5(3):179–202.

Feo, T. A. e Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, 6:109–133.

Gupta, J. N. D. e Gupta, S. K. (1988). Single facility scheduling with nonlinear processing times. *Computers & Industrial Engineering*, 14(4):387–393.

Ji, M. e Cheng, T. C. E. (2009). Parallel-machine scheduling of simple linear deteriorating jobs. *Theoretical Computer Science*, 410(38–40):3761–3768.

Kim, Y. (1993). Heuristics for flowshop scheduling problems minimizing mean tardiness. *Journal of Operational Research Society*, 44:19–28.

Kim, Y., Lim, H., e Park, M. (1996). Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process. *European Journal of Operational Research*, 91:124–143.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., e Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Montoya-Torres, J. R., Moreno-Camacho, C. A., e Velez-Gallego, M. C. (2023). Variable neighbourhood search for job scheduling with position-dependent deteriorating processing times. *Journal of the Operational Research Society*, 74(3):873–887.

Moreno, C. A., Montoya-Torres, J. R., e Vélez-Gallego, M. C. (2018). A comparison of mixed-integer linear programming models for workforce scheduling with position-dependent processing times. *Journal of the Operational Research Society*, 74(3):873–887.

Nawaz, M., Jr., E. E. E., e Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95. URL `https://doi.org/10.1016/0305-0483(83)90088-9`.

Thompson, G. M. e Goodale, J. C. (2006). Variable employee productivity in workforce scheduling. *European Journal of Operational Research*, 170(2):376–390.

Zemel, E. (1981). Measuring the quality of approximate solutions to zero–one programming problems. *Mathematics of Operations Research*, 6:319–332.

DOI: 10.59254/sbpo-2024-193465
https://proceedings.science/p/193465?lang=pt-br