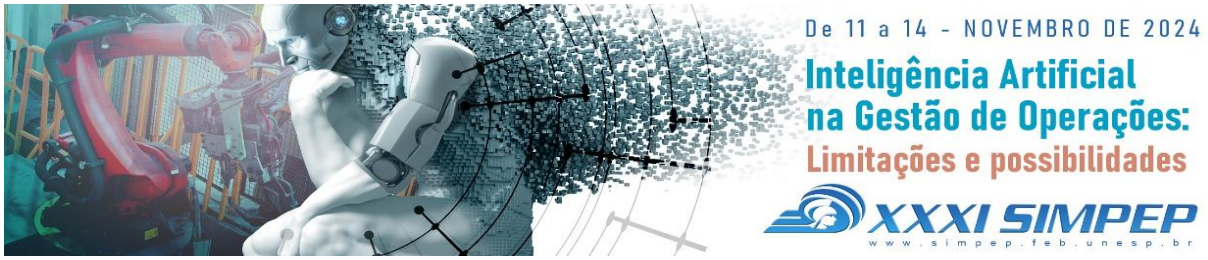

APLICAÇÃO DA METODOLOGIA GRASP NO PROBLEMA DE FLOW SHOP COM BLOQUEIO PARA MINIMIZAR O MAKESPAN

LAÍS GONÇALVES VIANA – lais.viana@ufv.br
UNIVERSIDADE FEDERAL DE VIÇOSA - UFV - RIO PARANAÍBA

HELEN VITALINE DE CASTRO SANTOS – helen.castro@ufv.br
UNIVERSIDADE FEDERAL DE VIÇOSA - UFV - RIO PARANAÍBA

THIAGO HENRIQUE NOGUEIRA – thiagoh.nogueira@ufv.br
UNIVERSIDADE FEDERAL DE VIÇOSA - UFV - RIO PARANAÍBA

RAIANE RIBEIRO MACHADO GOMES – raianemachado@ufv.br
UNIVERSIDADE FEDERAL DE VIÇOSA - UFV - RIO PARANAÍBA

LARISSA SOUSA CAMPOS – larissa.sousa@ufv.br
UNIVERSIDADE FEDERAL DE VIÇOSA - UFV - RIO PARANAÍBA

ÁREA: *3. PESQUISA OPERACIONAL*
SUBÁREA: *3.1 – MODELAGEM, SIMULAÇÃO E OTIMIZAÇÃO*

RESUMO: *ESTE ARTIGO APRESENTA UMA ANÁLISE SOBRE A APLICAÇÃO DA TÉCNICA DE SEQUENCIAMENTO DE TAREFAS EM AMBIENTES DE PRODUÇÃO, COM FOCO NO PROBLEMA DE ESCALONAR N TAREFAS EM M MÁQUINAS EM UM FLOWSHOP COM BLOQUEIO. O OBJETIVO PRINCIPAL É DETERMINAR A MELHOR ORDEM DE PROCESSAMENTO DAS TAREFAS PARA MINIMIZAR O TEMPO TOTAL DE PRODUÇÃO OU MAXIMIZAR O DESEMPENHO DO SISTEMA PRODUTIVO. PARA ISSO, É PROPOSTO O MÉTODO GRASP COMO UMA TÉCNICA PROMISSORA PARA RESOLVER ESSE PROBLEMA. OS RESULTADOS MOSTRAM QUE A APLICAÇÃO EFETIVA DESSA TÉCNICA PODE MELHORAR SIGNIFICATIVAMENTE OS PROCESSOS INDUSTRIAIS, GARANTINDO QUE AS ATIVIDADES SEJAM CONCLUÍDAS DENTRO DO PRAZO E MINIMIZANDO OS CUSTOS DE PRODUÇÃO.*

PALAVRAS-CHAVES: *GRASP; HEURÍSTICA COMBINATÓRIA; FLOWSHOP COM BLOQUEIO.*

1. INTRODUÇÃO

O sequenciamento de tarefas é uma técnica utilizada nos setores industriais que visa trazer melhorias para os processos, determinando a ordem em que atividades devem ser executadas de modo que maximize a eficiência e minimize os custos de produção. O objetivo desta técnica é determinar a melhor ordem de processamento de um conjunto de tarefas em um conjunto de máquinas, garantindo que cada atividade seja finalizada no momento certo, para que a conclusão de uma não dependa da conclusão de outra que ainda não foi iniciada ou concluída (RUIZ, VÁZQUEZ-RODRÍGUEZ; 2010). É de extrema importância aplicar de forma efetiva essa técnica, para que o tempo total de produção seja minimizado e as atividades sejam concluídas dentro do prazo estabelecido.

Um ambiente de produção comum em muitos setores é o *flow shop* com bloqueio. Nesse ambiente, um conjunto de máquinas está organizado em série, e todas as tarefas devem ser processadas na mesma sequência, seguindo a ordem das máquinas. Não há estoque intermediário entre as máquinas, o que pode causar bloqueios caso a próxima máquina não esteja disponível para receber a tarefa. Esse problema pode ser ainda mais desafiador em ambientes em que a tecnologia de produção exige que as tarefas permaneçam em uma determinada máquina até que a próxima esteja disponível, de modo a evitar deterioração na tarefa, retrabalho e custos adicionais.

O problema de *flow shop* com bloqueio acontece com muita frequência em diversos ramos industriais, fábricas de automóveis, por exemplo, muitos componentes precisam passar por várias etapas de produção diferentes antes de serem montados em um veículo, que são dependentes uma das outras o que pode acarretar no bloqueio e no atraso da produção (BAUTISTA-VALHONDO, ALFARO-POZO; 2020). Outros setores onde pode vir a acontecer o problema são na produção de chips, na fabricação de computadores integrados e de produtos tecnológicos em geral, já que os mesmos possuem uma série de etapas complexas e interdependente que são realizadas cada uma por equipamentos diferentes, quando um desses equipamentos é bloqueado todo o processo pode ser interrompido, causando atrasos e redução da produtividade (DONG , 2022).

O objetivo do sequenciamento de tarefas no problema de *flow shop* com bloqueio é a minimização do *makespan*, que nada mais é do que a duração total do do tempo da execução de conjunto de tarefas em um conjunto de máquinas. Este tipo de problema é considerado NP-difícil para mais de 3 máquinas (RUIZ, VÁZQUEZ-RODRÍGUEZ; 2010). Dentro da

literatura vários métodos são utilizados para resolver este problema. Segundo Ruiz e Vázquez-Rodríguez (2010) o *Branch and Bound* (B&B) é a técnica preferida para a resolução do problema de *flow shop* com bloqueio. Já segundo Ribas e *Companys* (2015) o método mais eficiente é a utilização da metodologia *Greedy Randomized Adaptive Search* (GRASP), onde é realizado busca de vizinhança na fase de melhoria para gerar soluções aleatórias gulosas

Dessa forma, este presente trabalho propõe uma heurística combinatória baseado na metodologia *Greedy Randomized Adaptive Search* (GRASP) para o problema de *flow shop* com bloqueio, visando a minimização do *makespan*. O objetivo é encontrar uma solução que represente a melhor ordem de produção possível para as tarefas, minimizando o tempo total de processamento, levando em consideração os dados coletados pelo artigo base e as restrições impostas pelo ambiente de produção. Este artigo foi organizado em referencial teórico, metodologia, experimentos computacionais e conclusão.

2. REFERENCIAL TEÓRICO

O problema do *flow shop* com bloqueio é um desafio complexo de otimização que envolve a programação de tarefas em máquinas de modo a minimizar o tempo total de processamento, também conhecido como *makespan* (BLAZEWCIZ et al., 1983). Dada a natureza NP-difícil deste problema, é comum a aplicação de metaheurísticas, que são métodos de otimização estocástica capazes de fornecer soluções de alta qualidade em tempo computacional razoável para problemas complexos (GLOVER; KOCHENBERGER, 2003). Uma dessas metaheurísticas é o procedimento de busca adaptativa randomizada gananciosa, conhecido como GRASP. O GRASP é um algoritmo iterativo que consiste em duas fases principais: construção de uma solução inicial e aprimoramento desta solução por meio de uma busca local (FEO; RESENDE, 1995).

Na fase de construção da solução GRASP, é muito comum a utilização da Lista de Candidatos Restrita (RCL) para selecionar os próximos elementos da solução de forma semi-aleatória (FEO; RESENDE, 1995). A RCL é composta pelos melhores candidatos não selecionados e cada elemento é selecionado aleatoriamente desta lista. Isso permite que a solução inicial seja construída adicionando sucessivamente elementos à solução, sendo aplicada na pesquisa de Aqil e Allali (2021) como uma das heurísticas para resolver o problema de programação de *flow shop* híbrido, obtendo resultados satisfatórios.

Já na pesquisa de Alekseeva et al. (2017), a RLC foi aplicada e definida com um parâmetro que controla a aleatoriedade do processo, sendo destacado como uma técnica para manter a aleatoriedade requerida no método GRASP. Nesta fase a heurística NEH também vai

ser aplicada em conjunto com a RCL, como visto na pesquisa de Lahby et al. (2022), onde a heurística obtém os melhores resultados quando comparado com as outras heurísticas aplicadas no trabalho. Essa heurística foi proposta por Nawas, Encore e Ram e tem se mostrado muito eficiente para encontrar soluções próximas do ótimo em problemas de *flow shop* com bloqueio (UTAMA, 2019).

Após a geração da solução inicial, a fase de busca local do GRASP é aplicada para explorar a vizinhança da solução construída e encontrar melhorias (FEO; RESENDE, 1995). Neste trabalho, a busca local é realizada usando a técnica SWAP, onde duas tarefas são trocadas em uma permutação para criar uma nova solução. A troca é feita em todas as combinações possíveis de duas tarefas e a melhor solução é escolhida para a próxima iteração. A vizinhança é definida pelo conjunto de soluções que podem ser obtidas por meio da troca de duas tarefas adjacentes na permutação.

Apesar de a técnica GRASP já ter sido aplicada a problemas de otimização similares de acordo com Pratta et al. (2018), o uso de uma geração totalmente aleatória da solução inicial e a busca local com o método SWAP para o problema *do flow shop* com bloqueio apresenta uma nova abordagem (MALEKI-DAROUNKOLAEI et al., 2012). O algoritmo proposto visa fornecer soluções de alta qualidade, mantendo uma execução eficiente, contribuindo assim para a área de pesquisa da otimização de *flow shop.*

Alguns autores que contribuíram para o tema incluem Blazewicz et al., Glover e Kochenberger, Feo e Resende. Além disso, existem trabalhos recentes que investigam variantes do problema do *flow shop* híbrido considerando bloqueio de máquina e tempos de configuração independentes e dependentes da sequência (PRATA et al., 2018). Moccellin et al. propuseram algoritmos heurísticos baseados em regras de prioridade tradicionais para resolver essa variante do problema. Também há pesquisas sobre abordagens baseadas em aprendizado de máquina para problemas do *flow shop* com recursos alternativos, tempos de configuração dependentes da sequência e bloqueio (HARTL et al., 2019). Benda et al. propuseram um método baseado em árvore de decisão para gerar uma regra de prioridade para despacho de trabalhos.

3. DESCRIÇÃO DO PROBLEMA

Neste estudo, abordamos um cenário complexo de sequenciamento em ambiente produtivo caracterizado por um conjunto de máquinas em linha e um conjunto de tarefas que devem ser processadas em uma sequência específica. O sistema inclui um total de n tarefas

representadas pela sequência $L=\{L_1,L_2,...,L_k,...,L_n\}$, onde L_k é a k-ésima tarefa na sequência, e um total de m máquinas representadas pela sequência $M = \{M_1,M_2,...,M_i,...,M_m\}$.

Este ambiente particular não contém espaço de armazenamento entre as máquinas, eliminando a possibilidade de filas de tarefas esperando processamento, o que pode resultar em bloqueio entre as tarefas. Além disso, cada tarefa L_k é processada uma vez em cada máquina M_i, e o tempo de processamento da tarefa L_k na máquina M_i é denotado por $p_{Lk,i}$,. A conclusão da tarefa L_k na máquina M_i é denotada por D_{Lk}, enquanto $C_{Lk,i,}$ denota o instante em que a tarefa deixa a máquina.

Agora, para representar esses detalhes como restrições matemáticas, vamos introduzir três equações fundamentais que explicam o tempo de saída da primeira tarefa e das subsequentes em todas as máquinas.

$$(1) \qquad D_{L1,i} = \sum_{l=1}^{i} p_{L1,l} \qquad i = 1, \dots, m$$

$$(2) \quad D_{Lk,i} = \max\left(D_{Lk,i-1} + p_{Lk,i}, D_{Lk-1,i+1}\right) \qquad k = 2, \dots, n\,; i = 1, \dots, m-1$$

$$(3) \quad D_{Lk,i} = D_{Lk,m-1} + p_{Lk,m} \qquad k = 2, \dots, n$$

A equação (1), tem o objetivo de calcular os tempos de saída da primeira tarefa da permutação L em cada máquina, onde não ocorre bloqueio entre as permutações. Em seguida, a equação (2), é usada para verificar se a tarefa de permutação *L* pode sair de sua máquina ou se deve esperar a tarefa *k-1* ser processada na máquina *m+1*. Essa equação calcula o tempo de saída de todas as tarefas subsequentes em cada máquina, com exceção da última máquina m. Finalmente, a equação (3), calcula os tempos de saída das tarefas na última máquina *m*.

O objetivo final deste problema é encontrar a permutação da sequência de tarefas L que minimize o *makespan* (C_{max}). Este *makespan*, dado por $C_{max}=D_{Ln,m}$, é o momento em que a última tarefa deixa a última máquina, representando o tempo total de produção. O cálculo realizado para encontrar o *makespan* é:

Através da resolução deste problema, esperamos fornecer insights valiosos para o campo de sequenciamento de tarefas em ambientes de produção em linha.

4. DESCRIÇÃO DO ALGORITMO PROPOSTO

Para este trabalho a heurística construtiva definida foi a *Nawas,Encore,HAM* (NEH) e para a estruturação dessa heurística, foi escolhida a metaheurística GRASP (Greedy Randomized Adaptive Search Procedures) com múltiplas inicializações de Feo e Resende (1989), com o intuito de aprimorar as soluções obtidas. A solução inicial será obtida por meio da heurística NEH. O algoritmo vai ser reinicializado sempre visando obter uma melhor solução para o problema, para garantir que diferentes soluções sejam geradas a cada reinicialização, a primeira tarefa *K* que será inserida na permutação *L* deve ser escolhida de uma forma completamente aleatória, onde as demais serão inseridas por meio da heurística NEH.

O GRASP é um processo iterativo, que consiste em duas fases em cada iteração: a fase de construção e a fase de busca local. A fase de construção é onde a solução viável para o problema é construída de maneira iterativa, adicionando um elemento de cada vez. Este processo de adição é orientado por um procedimento guloso, que ordena os candidatos potenciais com base no benefício que cada adição traria à solução já em construção de acordo com Silva (2011), esta fase continua até que uma solução completa seja encontrada.

No contexto do problema de *flow shop* com bloqueio, a solução inicial é construída selecionando, em cada iteração, uma tarefa que minimize o *makespan* (tempo total de conclusão) através da heurística gulosa, levando em consideração as tarefas já alocadas nas máquinas (MIYATA, 2019). Uma técnica muito comum para gerar solução inicial é chamada de Lista Restrita de Candidatos (RCL - *Restricted Candidate List*). A RCL é uma lista que contém um subconjunto dos candidatos disponíveis para serem selecionados em cada passo da construção da solução (PRABHAKARAN; RAKESH, 2006).

Na fase de construção do GRASP para obtenção da solução inicial, a heurística selecionada foi a NEH. Temos uma heurística de inserção baseada em ordenação, ou seja, as tarefas são inicialmente ordenadas de acordo com algum critério para em seguida serem inseridas uma a uma com o intuito de minimizar o tempo total de produção (RIBAS; MATEO, 2010). O NEH é um algoritmo heurístico construtivo geralmente aplicado de forma determinística, isto é, sempre escolhendo a tarefa com o maior tempo total de processamento. Ele insere essa tarefa em uma permutação parcial e, em seguida, para cada uma das tarefas restantes, seleciona a posição que proporciona o melhor *makespan* e insere a tarefa nessa posição. Dessa forma, é gerada uma permutação *L*.

Para combinar a estratégia gulosa do NEH com a aleatoriedade requerida no GRASP, a Lista Restrita de candidatos será utilizada fundamentando no tempo total de processamento das tarefas e então selecionando uma tarefa desta lista de maneira aleatória por meio de um

parâmetro de aleatoriedade. Esse parâmetro será definido como α, onde o valor sofrerá variações de $0<=\alpha<=1$, sendo usado para regular o equilíbrio entre a exploração (escolha totalmente aleatória) e a exploração (escolha gulosa). Um valor de alfa mais próximo de 0 resultará em uma RCL mais restrita e mais orientada para a exploração, enquanto um valor de alfa mais próximo de 1 resultará em uma RCL mais diversificada e mais orientada para a exploração.

A fase seguinte da metodologia é a fase de busca local, onde nesta etapa o algoritmo recebe uma matriz de tempos de processamento e uma permutação de tarefas, gerados na etapa anterior, como entrada. O algoritmo inicia a melhor permutação e o melhor *makespan* com os valores iniciais. Em seguida, itera sobre todas as posições da permutação, realizando a troca com as tarefas subsequentes. Para cada troca, é calculado o *makespan* da nova permutação. Se o novo *makespan* for menor que o melhor *makespan* atual, a nova permutação e o novo *makespan* são atualizados.

Para a isso, a estrutura de vizinhança definida foi a SWAP. Esta vizinhança é o conjunto de todas as soluções que podem ser alcançadas a partir da solução atual através de um único "movimento". No caso do problema de *flowshop* com bloqueio, um movimento comum é a troca (SWAP) de duas tarefas na sequência. Matematicamente, se $L=\{L_1,L_2,...,L_i,...,L_j,...,L_n\}$, é a sequência atual de tarefas, um movimento de troca entre as L_i tarefas e L_j resulta na nova sequência $L'=\{L_1,L_2,...,L_i,...,L_j,...,L_n\}$. Este conceito foi sugerido por Taillard (1993).

Para a solução atual, todas as possíveis soluções vizinhas que podem ser alcançadas por meio de um único movimento de troca (swap) entre duas tarefas são geradas. Este movimento é aplicado considerando as restrições do problema, como a ausência de espaço de armazenamento entre as máquinas e a sequência específica de processamento das tarefas.
O *makespan* para cada solução vizinha é então calculado. Seja $L=\{L_1,L_2,...,L_n\}$ na sequência de tarefas e $M = \{M_1,M_2,...,M_m\}$ a sequência de máquinas. Além disso, seja $p_{i,j}$. O tempo de processamento da tarefa L_i na máquina M_j. O tempo de conclusão da tarefa L_i. na máquina M_j, denotado por $C_{i,j}$, pode ser calculado como:

- Para a primeira tarefa na primeira máquina: $C_{1,1}=p_{1,1}$.
- Para as tarefas subsequentes na primeira máquina: $C_{i,1}=C_{i-1,1} + p_{1,1}$. para $i=2,...,n$.
- Para as tarefas subsequentes nas máquinas subsequentes: $C_{i,j}= max(C_{i,j-1} + C_{i,1j})+p_{i,j}$ para i=2,...,n e j=2,...,m.

O *makespan*, denotado por C_{max}, é então o tempo de conclusão da última tarefa na última máquina, ou seja, $C_{max} = C_{n,m}$. A solução atual se torna a solução vizinha que tem o menor *makespan* e é permitida. Se essa solução é melhor do que a melhor solução encontrada até o momento, a melhor solução também é atualizada. Matematicamente, se s^* é a melhor solução encontrada até agora, então $s^*=s'$ se $C_{max}(s') < C_{max}(s^*)$. Como parâmetro para o número máximo de repetições da busca local, foi definido como *IGrasp* onde vai ser mostrado como se calcula na etapa seguinte.

Finalmente, é verificado se o critério de parada foi atingido. Se um determinado número de iterações foi realizado ou se o tempo de execução exceder um limite, o algoritmo é parado. Caso contrário, o processo retorna para a geração do conjunto de soluções vizinhas. Matematicamente, se I é o número atual de iterações I_{max} é o número máximo de iterações permitido, t é o tempo de execução atual, e t_{max} é o tempo máximo de execução permitido, então o algoritmo para se $I \geq I_{max}$ ou $t \geq t_{max}$.

Já como técnica de perturbação empregada envolve a destruição e reconstrução de uma permutação L. Após remover aleatoriamente d tarefas de L, é formada uma permutação parcial LD e uma lista (LC) com as tarefas removidas. Enquanto LC não estiver vazia, uma tarefa j é selecionada aleatoriamente dentre as tarefas de LC. Em seguida, utiliza-se a fase de inserção do algoritmo NEH para reintroduzir a tarefa j em LD na posição que resulta no menor *makespan*. Ao final do procedimento, uma permutação L completa é retornada. O critério de permutação é definido como d.

5. EXPERIMENTOS COMPUTACIONAIS

Para realização dos experimentos foram utilizadas as instâncias propostas por Taillard (1993), porém reduzimos o conjunto de tarefas, onde o conjunto de diferentes tarefas selecionadas contém quatro tarefas em que $n= \{20,50\}$. Já o conjunto de máquinas onde essas tarefas serão executadas possui um total de três diferentes níveis de máquinas e é dado por m = {5,10,20}. As combinações de ambos os conjuntos totalizam 40 instâncias para serem analisadas. Os resultados obtidos pela metodologia *GRASP* em conjunto com a heurística construtiva NEH e na busca local sendo utilizada a técnica *SWAP*, foram comparados com resultados encontrados na literatura por Shao et al. (2019), onde os autores utilizaram um método que chamaram de *Discrete Evasive Weed Optimization* (DIWO) para obter resultados e compararam com outros na literatura.

O algoritmo proposto para esse trabalho foi implementado na linguagem *Python* e os experimentos foram realizados em uma máquina com processador *Intel® Core™ I7-3537U*

CPU @2.00 GHz, 8,00 GB de memória RAM, com sistema operacional Windows 10. Foram utilizados como número máximo de iterações da busca local como *IGrasp* = 3x[tamanho da primeira matriz], já como o parâmetro de permutação foi definido com *d* = 8 e como critério de parada foi utilizado o tempo limite de execução que pode ser calculado com *T* = 100 x *m* x *n* milissegundos.

Os resultados obtidos são apresentados na Tabela 1, onde cada uma das 40 instâncias foram executadas dez vezes e a tabela foi organizada da seguinte forma: a primeira coluna representa o nome das instâncias, a segunda coluna indica a *best know solution* (BKS), a terceira coluna representa o DIWO, a coluna seguinte indica o resultados óbitos pelo artigo base chamado de ILSBFSP e a última coluna foi o resultado obtido pela metodologia desenvolvida neste artigo.

TABELA 1 - Comparação dos Resultados

Instâncias	BKS	DIWO	ILSBFSP	GRASPNEH
J20M5N1	1374	1374	1374	1376
J20M5N2	1408	1408	1408	1434
J20M5N3	1280	1280	1280	1273
J20M5N4	1448	1448	1448	1311
J20M5N5	1341	1341	1341	1325
J20M5N6	1363	1363	1363	1325
J20M5N7	1381	1381	1381	1389
J20M5N8	1379	1379	1379	1396
J20M5N9	1373	1373	1373	1385
J20M5N10	1283	1283	1283	1253
J20M10N1	1698	1698	1698	1709
J20M10N2	1833	1833	1833	1800
J20M10N3	1659	1659	1659	1674
J20M10N4	1535	1535	1535	1520
J20M10N5	1617	1617	1617	1585
J20M10N6	1590	1590	1590	1625
J20M10N7	1622	1622	1622	1626
J20M10N8	1731	1731	1731	1678
J20M10N9	1747	1747	1747	1774
J20M10N10	1782	1782	1782	1789
J20M20N1	2436	2436	2436	2378
J20M20N2	2234	2234	2234	2293
J20M20N3	2479	2479	2479	2486
J20M20N4	2348	2348	2348	2339
J20M20N5	2435	2435	2435	2413
J20M20N6	2383	2383	2383	2331
J20M20N7	2390	2390	2390	2374
J20M20N8	2328	2328	2328	2317
J20M20N9	2363	2363	2363	2419

J20M20N10	2323	2323	2323	2357
J50M5N1	2980	2983	2974	2982
J50M5N2	3180	3180	3177	3180
J50M5N3	2995	3000	2989	2987
J50M5N4	3115	3115	3114	3029
J50M5N5	3139	3145	3138	3004
J50M5N6	3158	3161	3158	3158
J50M5N7	3005	3008	3004	3002
J50M5N8	3042	3044	3039	3047
J50M5N9	2889	2891	2889	2980
J50M5N10	3097	3102	3096	3103
J50M10N1	3611	3614	3605	3696
J50M10N2	3470	3473	3475	3473
J50M10N3	3465	3465	3466	3416
J50M10N4	3650	3654	3645	3654
J50M10N5	3582	3619	3620	3678
J50M10N6	3571	3575	3579	3414
J50M10N7	3657	3669	3669	3654
J50M10N8	3549	3549	3553	3612
J50M10N9	3508	3518	3510	3454
J50M10N10	3608	3610	3606	3645
J50M20N1	4479	4479	4490	4461
J50M20N2	4262	4262	4262	4333
J50M20N3	4261	4261	4261	4289
J50M20N4	4339	4342	4347	4251
J50M20N5	4249	4250	4257	4260
J50M20N6	4271	4271	4285	4360
J50M20N7	4291	4296	4299	4293
J50M20N8	4298	4298	4311	4290
J50M20N9	4304	4305	4304	4343
J50MN10	4398	4398	4398	4315

Fonte: Elaborado pelos autores (2022)

6. CONCLUSÃO

Neste estudo, o objetivo foi desenvolver uma metodologia embasada no procedimento *GRASP* para resolver o problema de *Flow Shop* com bloqueio, visando a minimização do *makespan*. Utilizamos a heurística NEH, conhecida por sua eficácia em situações de *Flow Shop* sem bloqueio, para a construção da solução inicial, e incorporamos aleatoriedade com a Lista Restrita de Candidatos (RCL). Na fase de busca local, aplicamos a estratégia de geração de vizinhança baseada em trocas *(*SWAP).

Os experimentos conduzidos demonstraram que a metodologia proposta é promissora, produzindo soluções de alta qualidade. A metodologia mostrou-se capaz de lidar eficientemente com uma variedade de atividades e complexidades crescentes do problema, incluindo as instâncias mais desafiadoras derivadas do artigo de Taillard (1993).

Em resumo, a combinação do GRASP, heurística NEH e estratégia SWAP se mostrou um caminho promissor para que sejam solucionados problemas de *Flow Shop* com bloqueio. Este trabalho inicia uma importante discussão sobre estes desafios e serve como um ponto de partida para futuras pesquisas nesta área. Sugere-se, para estudos futuros, a exploração de mais possibilidades do GRASP e a combinação dele com outras metaheurísticas em cenários variados de *Flow Shop* com bloqueio. Sugere-se também incluir no método a exploração de novas vizinhanças para comparação de resultados com os já obtidos.

REFERÊNCIAS

ALEKSEEVA, Ekaterina et al. Parallel multi-core hyper-heuristic GRASP to solve permutation flow-shop problem. **Concurrency and Computation: Practice and Experience**, v. 29, n. 9, p. e3835, 2017.

AQIL, Said; ALLALI, Karam. Two efficient nature inspired meta-heuristics solving blocking hybrid flow shop manufacturing problem. **Engineering Applications of Artificial Intelligence**, v. 100, p. 104196, 2021.

BAUTISTA-VALHONDO, Joaquín; ALFARO-POZO, Rocío. Mixed integer linear programming models for Flow Shop Scheduling with a demand plan of job types. **Central european journal of operations research**, v. 28, n. 1, p. 5-23, 2020.

BENDA F.; BRAUNE R.; DOERNER K.F.; HARTL R.F… **A machine learning approach for flow shop scheduling problems with alternative resources, sequence-dependent setup times and blocking**. OR Spectrum v.41 p.871–893 2019.

BLAZEWCIZ, J. et al. **Scheduling in computer and manufacturing systems. Berlin: Springer-Verlag**, 1983.

BONFIGLIO, Juan José et al. Serine ADP-ribosylation depends on HPF1. **Molecular cell**, v. 65, n. 5, p. 932-940. e6, 2017.

CHEN, Huaping et al. A hybrid differential evolution algorithm for a two-stage flow shop on batch processing machines with arbitrary release times and blocking. **International Journal of Production Research**, v. 52, n. 19, p. 5714-5734, 2014.

DE MARCHI, Mônica Maria et al. Aplicação do Método Grasp no Problema de Posicionamento de Radares de Vigilância. **XXXVII Simpósio Brasileiro de Pesquisa Operacional (SBPO), Gramado, RS**, p. 1295-1304, 2005.

DONG, Zhuoran et al. Minimizing the Late Work of the Flow Shop Scheduling Problem with a Deep Reinforcement Learning Based Approach. **Applied Sciences**, v. 12, n. 5, p. 2366, 2022.

FEO, T.; RESENDE, M. **Greedy randomized adaptive search procedures. Journal of Global Optimization**, v. 6, n. 2, p. 109-133, 1995.

FEO, Thomas A.; RESENDE, Mauricio GC. Greedy randomized adaptive search procedures. **Journal of global optimization**, v. 6, p. 109-133, 1995.

GLOVER, F.; KOCHENBERGER, G. **Handbook of metaheuristics. Boston: Kluwer Academic Publishers**, 2003.

HALL, Nicholas G.; SRISKANDARAJAH, Chelliah. A survey of machine scheduling problems with blocking and no-wait in process. **Operations research**, v. 44, n. 3, p. 510-525, 1996.

KOMESU, Adriano Seiko. **Heurística evolutiva para a minimização do atraso total em ambiente de produção Flow Shop com buffer zero**. 2015. Tese de Doutorado. Universidade de São Paulo.

LIU, Shi Qiang; KOZAN, Erhan. Scheduling a flow shop with combined buffer conditions. **International Journal of Production Economics**, v. 117, n. 2, p. 371-380, 2009.

MALEKI-DAROUNKOLAEI A.; MODIRI M.; TAVAKKOLI-MOGHADDAM R.; SEYYEDI I… **A three-stage assembly flow shop scheduling problem with blocking and sequence-dependent set up times**. Journal of Industrial Engineering International v.8 p.26 2012.

MIYATA, Hugo Hissashi. **Contribuições para o problema flow shop com bloqueio, tempos de setup dependentes da sequência e funções-objetivo hierárquicas sujeitas ao custo total de manutenção preventiva**. 2020. Tese de Doutorado. Universidade de São Paulo.

MIYATA, Hugo Hissashi; NAGANO, Marcelo Seido. The blocking flow shop scheduling problem: A comprehensive and conceptual review. **Expert Systems with Applications**, v. 137, p. 130-156, 2019.

MOCCELLIN, J.V.; NAGANO, M.S.; PITOMBEIRA NETO, A.R.; PRATA B.A. **Heuristic algorithms for scheduling hybrid flow shops with machine blocking and setup times**. Journal of the Brazilian Society of Mechanical Sciences and Engineering, v. 40, n. 40, p. 1-14, 2018.

MONTEMANNI, Roberto; MOON, Jim NJ; SMITH, Derek H. An improved tabu search algorithm for the fixed-spectrum frequency-assignment problem. **IEEE transactions on vehicular technology**, v. 52, n. 4, p. 891-901, 2003.

NEJJAROU, Omar; AQIL, Said; LAHBY, Mohamed. Constructive heuristics and mathematical formulation for solving the permutation flow shop scheduling problem with setup time. In: **2022 IEEE 3rd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)**. IEEE, 2022. p. 1-5.

PRABHAHARAN, G.; KHAN, B. Shahul Hamid; RAKESH, L. Implementation of grasp in flow shop scheduling. **The International Journal of Advanced Manufacturing Technology**, v. 30, p. 1126-1131, 2006.

RIBAS, Imma; COMPANYS, Ramon. Efficient heuristic algorithms for the blocking flow shop scheduling problem with total flow time minimization. **Computers & Industrial Engineering**, v. 87, p. 30-39, 2015.

RIBAS, Imma; COMPANYS, Ramon; TORT-MARTORELL, Xavier. Efficient heuristics for the parallel blocking flow shop scheduling problem. **Expert Systems with Applications**, v. 74, p. 41-54, 2017.

RIBAS, Imma; MATEO, Manel. Improvement tools for NEH based heuristics on permutation and blocking flow shop scheduling problems. In: **Advances in Production Management Systems. New Challenges, New Approaches: IFIP WG 5.7 International Conference, APMS 2009, Bordeaux, France, September 21-23, 2009, Revised Selected Papers**. Springer Berlin Heidelberg, 2010. p. 33-40.

RONCONI, Débora P. A note on constructive heuristics for the flowshop problem with blocking. **International Journal of Production Economics**, v. 87, n. 1, p. 39-48, 2004.

RUIZ, Rubén; VÁZQUEZ-RODRÍGUEZ, José Antonio. The hybrid flow shop scheduling problem. **European journal of operational research**, v. 205, n. 1, p. 1-18, 2010.

SHAO, Zhongshi et al. An efficient discrete invasive weed optimization for blocking flow-shop scheduling problem. **Engineering Applications of Artificial Intelligence**, v. 78, p. 124-141, 2019.
SILVA, Paulo Henrique Asconavieta da. O problema do caixeiro viajante alugador: um estudo algorítmico. 2011.

TAILLARD, Eric. Benchmarks for basic scheduling problems. **european journal of operational research**, v. 64, n. 2, p. 278-285, 1993.

TAKANO, Mauricio Iwama. **Uma contribuição para o problema de programação de operações flow shop com buffer zero e tempos de setup dependente da sequência e da máquina**. 2016. Tese de Doutorado. Universidade de São Paulo.

UTAMA, Dana Marsetiya. An effective hybrid sine cosine algorithm to minimize carbon emission on flow-shop scheduling sequence dependent setup. **Jurnal Teknik Industri**, v. 20, n. 1, p. 62-72, 2019.