Vol. 3, Edição 3, 2020, pp. 13-28
ISSN: 2620-1607
eISSN: 2620-1747
DOI: https://doi.org/10.31181/oresta20303013s

UMA APLICAÇÃO DE ALGORITMOS DE OTIMIZAÇÃO METAHEURÍSTICA PARA RESOLVER O PROBLEMA DE AGENDAMENTO FLEXÍVEL DE JOB-SHOP

Aleksandar Stanković*, Goran Petrović, Žarko Ćojbašić, Danijel Marković

Faculdade de Engenharia Mecânica, Universidade de Niš, Sérvia

Recebido: 11 de junho de 2020
Aceito: 10 de agosto de 2020
Primeira vez on-line: 24 de setembro de 2020

Artigo científico original

Resumo. *O problema FJSP (Flexible Job Shop Planning) é outro problema de planejamento e programação. É uma continuação do problema clássico de agendamento de trabalhos, em que cada operação pode ser realizada em máquinas diferentes, enquanto o tempo de processamento depende da máquina que está sendo usada. O FJSP é um problema NP difícil que consiste em dois subproblemas: problemas de programação e operações de programação. O artigo apresenta um modelo para resolver o FJSP com base em algoritmos meta-heurísticos: Algoritmo genético (GA), busca Tabu (TS) e otimização de colônia de formigas (ACO). A eficiência da abordagem na solução do problema mencionado acima se reflete na busca flexível do espaço e escolha de soluções dominantes. Os resultados do cálculo são representados graficamente no gráfico de Gantt.*

Palavras-chave: *Agendamento, job-shop flexível, algoritmo genético, busca Tabu, otimização de colônia de formigas, busca local.*

1. Introdução

O planejamento da produção e dos processos de produção tem uma função muito importante no funcionamento bem-sucedido da produção. Os problemas de planejamento e programação ocorrem em quase todos os campos da economia, da engenharia e da produção industrial. Um dos problemas mais importantes da produção é o planejamento e a programação das operações. Um dos principais motivos para programar e planejar operações é aumentar a produtividade da produção. A programação e o planejamento de operações podem ser muito fáceis, mas também podem ser um dos problemas de programação mais difíceis, dependendo do tipo de problema e das condições de planejamento. Um problema em que há mais de uma máquina disponível para cada operação e em que há flexibilidade na seleção de uma máquina em um conjunto de máquinas alternativas é chamado de FJSP (Flexible Job Shop Problem). De acordo com o

* Autor correspondente.
aleksandar.stankovic@masfak.ni.ac.rs (A. Stanković),goran.petrovic@masfak.ni.ac.rs (G. Petrović),zcojba@ni.ac.rs (Ž. Ćojbašić),danijel.markovic@masfak.ni.ac.rs (D. Marković)

Na rotina de ação JSP, cada trabalho é processado em uma máquina com um tempo de processamento definido, e cada máquina pode processar apenas uma operação. Na prática, uma máquina pode ter a capacidade flexível de executar mais de um tipo de operação, o que leva à modificação do JSP no FJSP. Como Pinedo (2008) afirmou em seu livro, a definição de FJSP pode ser expressa como uma generalização do local de trabalho e do ambiente paralelo das máquinas. Em vez de *m* máquinas em uma fileira, há *c* centros de trabalho com cada centro de trabalho em paralelo com o mesmo número de máquinas idênticas. Cada trabalho tem sua própria rota a ser seguida em toda a oficina; o trabalho *j* requer processamento em cada centro de trabalho em apenas uma máquina e cada máquina pode funcionar. Se uma empresa em seu caminho pela loja puder visitar o centro de trabalho mais de uma vez, o *campo* b conterá a entrada *rcrc* para recirculação.

O objetivo deste artigo é testar e comparar métodos de otimização meta-heurística em árvore para minimizar o tempo gasto no planejamento e na programação de operações no conjunto de máquinas disponíveis. Os resultados obtidos com o uso de diferentes abordagens devem ajudar os gerentes a identificar um método adequado para essa classe de problema.

Há diferentes abordagens para resolver o FJSP disponíveis na literatura, e isso será analisado na próxima seção. Nos primeiros anos de pesquisa sobre problemas de planejamento e programação, os métodos exatos eram usados na alocação de recursos. Atualmente, métodos como programação com restrições e métodos de simulação estão sendo cada vez mais usados no mundo do planejamento. O objetivo deste artigo se reflete na aplicação de vários métodos meta-heurísticos, a saber, o algoritmo de Busca Tabu (TS), a Otimização por Colônia de Formigas (ACO) e o Algoritmo Genético (GA) e sua comparação na velocidade de convergência e precisão das soluções. A ideia básica é avaliar qual dos algoritmos aplicados é mais aplicável para resolver o FJSP.

2. Revisão da literatura

O planejamento e a programação de recursos, bem como os métodos usados para resolver problemas de programação, estão ganhando terreno em muitas áreas de logística, planejamento, pesquisa e roteamento, nas quais essa metodologia é muito significativa e aplicável. Falando em programação (Brucker & Schile, 1990), eles são um dos primeiros cientistas a desenvolver um algoritmo gráfico para planejamento e programação. Os algoritmos mais comumente usados atualmente para resolver problemas de programação no FJSP são algoritmos meta-heurísticos: Algoritmo Genético (Fraser, 1957; Bremermann, 1958; Holland, 1975), Otimização por Colônia de Formigas (Dorigo et al., 2006), Recozimento Simulado (Kirkpatrick et al., 1983), Busca Tabu (Glover & Laguna, 1997), Otimização por Enxame de Partículas (Kennedy & Eberhart, 1995) e outros. Na continuação do trabalho na Figura 1, são apresentados métodos com base nos quais é possível resolver os problemas de planejamento e programação de recursos.

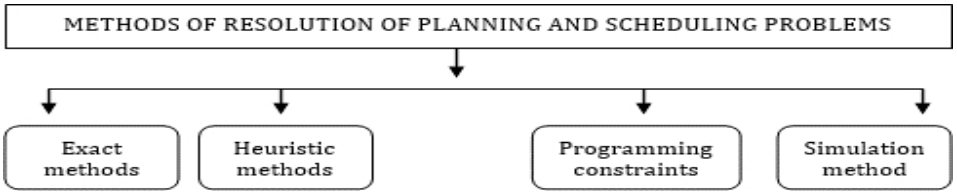

Figura 1. Métodos para resolver problemas de programação

Métodos exatos: O recurso básico dos métodos exatos é a precisão na definição do modelo matemático, bem a busca de soluções ideais, dependendo do tamanho dados testados. Uma das principais desvantagens dos métodos exatos é a solução de modelos robustos. Esse grupo de métodos inclui algumas das técnicas básicas usadas para resolver problemas de programação e planejamento: técnicas de programação não linear, linear, dinâmica, inteira e desconexa. Há muitos algoritmos exatos na literatura de pesquisa que são usados para resolver esses problemas de pesquisa, como os métodos Branch and Bound (Lomnicki, 1965), e eles podem ser definidos por meio de várias técnicas para determinar os limites inferior e superior.

Klein & Scholl (1996), bem como Blazewicz, et al. (1996) apresentam, em seus trabalhos, o método Branch e Baund no exemplo de planejamento de recursos, e o principal objetivo de seu trabalho é atribuir tarefas a um determinado número de máquinas para atingir a produtividade máxima. Liu, et al. (1996) e Thomalla (2001) apresentam, em seus trabalhos, o problema de agendamento pelo método de relaxamento Lagrangiano, onde provam que o problema de agendamento e planejamento pode ser resolvido com sucesso por esse método. Robson (1986) trabalhou para refinar um algoritmo que já havia desenvolvido para melhorar a complexidade temporal. Ostergard (2002) propôs um algoritmo Branch e Baund que define cada nó com uma cor diferente para distinguir os nós uns dos outros, o que, naquele momento, é uma nova metodologia de marcação. Vandaele (2000), Hasan & Arefin (2017) e Aslan et al. (2017) mostram, em seus artigos, o problema de planejamento e programação e o sucesso na solução desses problemas por meio de um método integrado de planejamento.

Métodos heurísticos: Alan Turing foi provavelmente o primeiro a usar algoritmos heurísticos quando decifrou o código alemão Enigma durante a Segunda Guerra Mundial. O próximo passo significativo na evolução dos algoritmos evolutivos foi dado por Holland (1975) e seus colegas da Universidade de Michigan nas décadas de 1960 e 1970. Esses métodos de busca não garantem que se encontre a solução ideal, mas que se encontre uma solução suficientemente boa. As heurísticas são divididas em: heurísticas que fornecem apenas uma solução dentro da busca e heurísticas que fornecem resultados durante a busca por meio de uma série de soluções iterativas. Nos trabalhos (Sentlelro, 1993), (Lagodimos & Leopoulos, 2000), (Spyropoulos, 2000), (Xing & Zhang, 2000), podemos ver uma abordagem heurística para resolver problemas de planejamento e programação, e também com base nos resultados obtidos, pode-se ver que essa abordagem fornece resultados ótimos. Kung e Chern (2009) mostram outra maneira de resolver problemas de planejamento. Nesse trabalho, podemos ver uma abordagem heurística de programação que se concentra na solução de operações de planejamento e programação de fábrica para diferentes estruturas de trabalho (produto). Para esse problema de planejamento, é proposto um algoritmo heurístico, denominado no documento como algoritmo heurístico de planejamento de fábrica, abreviado como HFPA. Xing e Zhang (2000) usam um método de abordagem heurística para resolver o problema de planejamento e programação problema de *M* máquinas paralelas com custo total mínimo. Sobeyko e Mönch (2016) apresentam uma abordagem heurística para resolver problemas de programação em operações flexíveis de larga escala. Com base nos resultados obtidos neste documento, podemos concluir que a heurística proposta chega rapidamente a soluções satisfatórias.

Restrições de programação: Essa abordagem de solução de problemas pertence ao grupo de problemas NP-difíceis, e a característica básica é a programação de restrições na solução de problemas em sistemas de planejamento e programação industrial. Há uma série de

de diferentes aplicativos de software usados nessa categoria de solução de problemas que podem ser usados para programar restrições de programação. Essa abordagem teve origem no campo da inteligência artificial. As linguagens de programação mais usadas atualmente para resolver problemas de planejamento e programação de inteligência artificial são: MatLab, Python, C
++, C #, Java e muito mais. Quando se trata de programação de tempo, planejamento de restrições e problemas de agendamento, os métodos meta-heurísticos são amplamente apresentados.

Um exemplo de solução de problemas de agendamento pode ser visto em (Stanković, et al., 2019). Deve-se observar que duas abordagens são apresentadas na literatura: a abordagem determinística e a abordagem estocástica (Pinedo, 2008). O tempo de conclusão da programação de operações (produtos) na literatura científica é indicado por C_{max}, que representa o valor total do critério da função. Exemplos de resolução de problemas de planejamento e programação podem ser vistos em (Jamili, 2018); (Fan, et al., 2019). A solução de FJSPs com base em algoritmos meta-heurísticos com restrições de programação na forma de restrições de tempo, períodos de indisponibilidade, pode ser vista nos artigos (Zhang, et al., 2011); (Stanković, et al., 2019).

Tamssaouet, et al. (2018) compara vários algoritmos meta-heurísticos com seus associados com períodos de indisponibilidade da máquina na forma de manutenção preventiva e corretiva da máquina. Liaw, (2000) apresenta a aplicação de um algoritmo híbrido com um algoritmo genético básico. Uma pesquisa mais aprofundada inclui um processo de aprimoramento de pesquisa baseado em guias locais. Os resultados obtidos mostram otimização em comparação com outros algoritmos de busca.

Métodos de simulação: A modelagem de simulação tem uma grande capacidade de apresentar sistemas complexos em uma infinidade de detalhes, o que é sua principal vantagem em relação a outros métodos. O planejamento baseado em simulação é usado para muitas operações e controles de sistema e, como resultado final, é obtido um plano de trabalho detalhado. Os modelos de planejamento baseados em simulação precisam ser mais detalhados do que outros modelos de simulação típicos. Na prática, muitos modelos com esse tipo de problema só podem ser resolvidos pelo método de otimização baseado em simulação, uma abordagem na qual o modelo de simulação é integrado a métodos de busca meta-heurísticos, como GA e TS (Laguna, et al., 2003).

O método Kanban é usado para aumentar a produtividade do fluxo de produtos em um único sistema de produção e eliminar possíveis erros no final de um ciclo. O Kanban é um sistema que controla o fluxo de material (recursos) por meio de vários processos de otimização múltipla. O sistema Kanban foi desenvolvido pelos engenheiros da Toyota - Taiichi Ohno (engenheiro industrial e empresário) - para otimizar seu processo de fabricação. A implementação e o sucesso da solução dos problemas de planejamento e programação de operações em pequenas e médias empresas podem ser vistos em muitos artigos profissionais. Schaefers et al., (2000) é um exemplo de solução de problemas de planejamento e fluxo de produtos, bem como de otimização de custos. Os problemas foram identificados, analisados e otimizados com base no método Kanban. A técnica de gerenciamento do setor japonês tem sido aplicada em muitos países ocidentais.

Graver & Price (1987) apresentam o método Kanban em seu trabalho e o utilizam para resolver problemas de planejamento e programação de JSP na forma de simulação, e os resultados finais da simulação mostram uma melhoria significativa do sistema em relação à prática de planejamento anterior. Kumar e Panneerselvam (2007) apresentam o sistema Kanban e 100 trabalhos de pesquisa de ponta, bem como sugestões para pesquisas futuras.

O método de controle de carga de trabalho (WLC) envolve três modelos: planejamento, controle e programação. A tarefa básica desse método é resolver o problema da carga de produção. Em muitos casos, ao planejar e selecionar um trabalho, são usadas as regras de prioridade de trabalho, dependendo do tempo de entrega de um determinado tipo de produto, que é um dos fatores mais importantes durante o processo de planejamento em pequenas e médias empresas. Esses métodos são de grande utilidade na forma de simulações e na solução de problemas de planejamento e programação, o que pode ser visto nos artigos (Thürer et al., 2012); (Thürer et al., 2017).

3. Metodologia

Métodos precisos e heurísticos são usados para resolver problemas de planejamento e programação. A aplicação de métodos exatos é limitada a problemas simples, enquanto métodos meta-heurísticos mais complexos são usados para sistemas reais complexos. O termo meta-heurística foi proposto pela primeira vez por Fred Glover em 1986, enquanto o mesmo autor definiu meta-heurística muitos anos depois como uma estrutura algorítmica de alto nível independente do problema que fornece um conjunto de diretrizes ou estratégias para desenvolver algoritmos de otimização heurística (Sörensen & Glover, 2013). As meta-heurísticas são projetadas para resolver problemas de otimização complexos quando outros métodos de otimização não conseguem resolver o problema de otimização de forma eficaz.

Atualmente, esses métodos são reconhecidos como uma das abordagens práticas mais importantes para a solução de muitos problemas complexos, e isso é especialmente importante para a solução de muitos problemas reais de otimização combinatória, daí a aplicação para o problema FJSP. Em geral, pode-se dizer que as meta-heurísticas são heurísticas de nível superior. A seguir, apresentamos três métodos meta-heurísticos que foram aplicados na solução dos problemas de alocação e programação das operações do FJSP.

3.1. Pesquisa Tabu

O algoritmo Tabu Search (TS) foi mencionado pela primeira vez por um cientista famoso, Glover (1986). O algoritmo TS é um método de pesquisa meta-heurística que usa métodos de pesquisa local. A implementação da pesquisa usa modelos estruturais que descrevem locais médios, ou seja, soluções possíveis, ou usam conjuntos que o usuário define como parâmetros iniciais do problema em consideração. Isso significa que, se uma solução potencial tiver sido visitada anteriormente em algum ponto da pesquisa ou se as regras de pesquisa definidas tiverem sido excedidas, ela será marcada na lista de guias. Assim, o algoritmo TS não considera a mesma solução várias vezes como soluções possíveis durante a pesquisa.

Em pesquisas anteriores, as buscas Tabu provaram ser o método de busca ideal em uma ampla gama de problemas de planejamento clássicos e práticos, e até mesmo no campo das redes neurais, como pode ser visto nos artigos (Nowicki & Smutnicki, 2005); (Zhang et al. 2007). Para evitar problemas durante a busca, o tamanho da lista tabu durante a busca precisa ser modificado (Talbi, 2009). O tamanho da lista tabu é crucial durante esse tipo de busca. Se a lista tabu for muito pequena, a busca tenderá a percorrer várias vezes as mesmas soluções possíveis, ao passo que, se a lista tabu for muito grande, a busca será mais rápida.

grande, a falta de movimentos disponíveis pode levar a possíveis erros durante a busca. O algoritmo TS é representado por um pseudocódigo na Tabela 1 (Glover, 1989).

Tabela 1. Pseudocódigo da Busca Tabu

Pseudocódigo da Busca Tabu

sBest ← s0
bestCandidate← s0
tabuList ← []
tabuList.push(s0)
While (not stoppingCondition())
sNeighborhood← getNeighbors(bestCandidate)
para (sCandidate em sNeighborhood)
se ((não tabuList.contains(sCandidate)) e (fitness(sCandidate) > fitness(bestCandidate)))
melhorCandidato← sCandidato
end for
end if
se (fitness(bestCandidate)> fitness(sBest))
sBest ← bestCandidate
fim se
tabuList.push(bestCandidate)
Se (tabuList.size> maxTabuSize)
tabuList.removeFirst()
fim se
end While

3.2. Otimização de colônia de formigas

O método de otimização por colônia de formigas (ACO) foi proposto pela primeira vez por Dorigo (1992). A otimização de colônia de formigas é uma meta-heurística baseada em população que pode ser usada para encontrar soluções ideais aproximadas para diferentes casos de teste. O algoritmo é inspirado no comportamento das formigas na natureza. A característica básica do comportamento coletivo das formigas é que todos os membros da colônia trocam informações sobre seu ambiente indireta ou diretamente, ou seja, o fenômeno da inteligência coletiva. Descobriu-se na natureza que cada formiga deixa um , liberando uma certa quantidade de uma substância química chamada feromônio. Quanto mais formigas percorrem um caminho, mais feromônios, ou seja, para cada formiga subsequente, informações positivas sobre a correção desse caminho. Dessa forma, as formigas se comunicam indiretamente umas com as outras por meio de feromônios. Todas as formigas começam com um valor de 0, o que significa que nenhuma operação é programada antes do início da pesquisa. Todos os nós têm um feromônio inicial 1. O feromônio diminuirá após cada rodada de pesquisa. A pesquisa local depende do número de feromônios e do tempo de pesquisa, e o tempo total é calculado com base no tempo extra necessário para ativar a próxima operação O_{ij} na máquina M_n disponível. Um valor aleatório é gerado para comparação com r_0. Se o valor rand for menor que r_0, a busca local selecionará a rota de planejamento com a quantidade máxima de feromônio. Somente as melhores formigas podem depositar o feromônio no caminho do ponto A ao ponto B. A quantidade total de feromônio depositado é calculada com base no valor de

expressão $\Delta\tau = 1 / (\ bestcost\ )$. O pseudocódigo do algoritmo ACO é apresentado na Tabela 2 (Yang, 2010).

Tabela 2. Pseudocódigo da otimização por colônia de formigas

Pseudocódigo da otimização por colônia de formigas
Função objetiva f (x), x= (x1,...xn)T
[ou f (xij) para o problema de roteamento em que (i, j)€ {1, 2, ... n}]
Defina a taxa de evaporação do feromônio ν
while (criterion)
loop **for** em todas as n dimensões (ou nós)
Gerar novas soluções
Avaliar as novas soluções
Marcar melhores locais/rotas com feromônio $\delta\Psi$ij Atualizar
feromônio: Ψij - (1-ν) Ψij + $\delta\Psi$ij
fim para
Ações do Daemon, como encontrar o melhor
finalizar
enquanto
Gerar os melhores resultados e distribuições de feromônios

3.3. Algoritmo genético

O Algoritmo Genético (AG) é uma técnica de otimização usada para resolver otimizações não lineares ou não diferenciais. O AG foi desenvolvido por Holland na década de 1970 do século passado (Holland, 1975). O algoritmo genético é caracterizado por vários estágios na solução de um problema definido, neste caso de planejamento e programação.

O algoritmo imita o processo de seleção natural, enquanto as alterações na estrutura genética são possíveis por meio da mutação do material genético, cuja essência é expandir a área de pesquisa e superar os extremos locais. O cruzamento na solução do AG é um processo de combinação de várias unidades para obter uma nova unidade selecionada, e esse tipo é comparado a um processo natural, como pais e filhos. Os novos indivíduos herdam os genes de seus pais. Quando se trata de resolver problemas de planejamento e programação, os exemplos mais comuns são baseados em um algoritmo genético. Uma das soluções mais comuns é baseada em uma matriz de programação de tarefas codificada usada para problemas de programação em mais de uma máquina. Um exemplo dessa matriz pode ser visto na Figura 2.

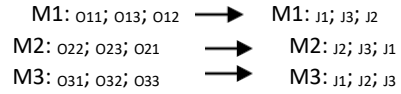

Figura 2. Matriz de programação de trabalhos

A mutação envolve uma mudança aleatória nos genes dos indivíduos. A mutação permite a alteração descontrolada do material genético. Ao alterar a estrutura genética, são obtidas soluções completamente novas. O objetivo básico é obter um indivíduo que não possa ser obtido em outros estágios.

Esse valor aleatório inicia uma pesquisa em todo o domínio permitido. A taxa de mutação deve ser pequena, de cerca de 0,001% a 0,01%, para evitar um procedimento aleatório, estocástico e sem controle. O pseudocódigo do GA é apresentado na Tabela 3 (Yang, 2010).

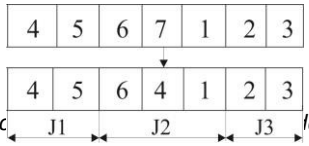

Figura 3. Mutação lamento Tabela

3. Pseudocódigo do GA

Pseudocódigo do GA
Função objetiva $f(x)$, $x= (x_1,....,x_n)^T$
Codificar a solução em cromossomos (cadeias binárias) Definir a aptidão F (por exemplo, F ∞ f(x*)* para maximização) Gerar a população inicial
Probabilidades iniciais de cruzamento (p_c) e mutação (p_m)
While (t< Número máximo de gerações)
Gerar uma nova solução por meio de cruzamento e mutação
Se $_{o\ pc}$> rand, Crossover;
fim se
Se $_{pm}$> rand, Mutate;
fim se
Aceitar as novas soluções se sua adequação aumentar
Selecionar a melhor solução atual para a nova geração
finalizar enquanto
Decodificar os resultados e a visualização

4. Estudo de caso

Esta seção apresenta um método para resolver FJSPs com base em três algoritmos meta-heurísticos propostos. A primeira parte dos estudos de caso apresenta um modelo para testar os algoritmos em um conjunto de dados clássico. A eficiência dos algoritmos é refletida na velocidade da solução de convergência por meio de uma série de iterações. Os parâmetros de entrada consistem em 25 trabalhos e 10 máquinas com um tempo de processamento definido para cada operação em uma máquina individual. O modelo matemático de otimização da solução com função objetiva e restrição de tempo é representado pela seguinte notação:

- *J* - número de trabalhos,
- *M* - número de máquinas idênticas,
- $_{pi,j,k}$ - o tempo de processamento de cada operação.

Restrições:

$$J> M> 1,\ J,\ M \in Z^{+} \tag{1}$$

$$\forall\ p \in T,\ p \in Z\ \&1^{+} \leq p \leq M \tag{2}$$

Função de custo:

$$f(x)_{=maxi} f(i), \quad i=1,2,..., M \tag{3}$$

Os parâmetros de entrada e os resultados do problema de otimização para todos os três algoritmos meta-heurísticos são apresentados na Tabela 4.

Tabela 4. Parâmetros de entrada e resultados: TS, ACO e GA

Parâmetros TS		
MaxIter	J	M
1000	25	10
Tempo para o algoritmo TS para o qual uma solução ótima é encontrada [s]		
Melhor local encontrado para 1000 iterações	187	
Parâmetros do algoritmo ACO		
MaxIter	J	M
1000	25	10
Tempo para o algoritmo ACO para o qual uma solução ótima é encontrada [s]		
Melhor local encontrado para 1000 iterações	182	
Parâmetros do GA		
MaxIter	J	M
1000	25	10
Tempo para o GA para o qual uma solução ótima é encontrada [s]		
Melhor local encontrado para 1000 iterações	176	

Com base nos resultados apresentados na Tabela 4, pode-se concluir que TS, ACO e GA apresentam resultados satisfatórios, mas que o GA apresenta os resultados mais favoráveis. Com base nos resultados obtidos, o GA foi usado no estudo de caso do FJSP de planejamento e programação de operações em um ambiente industrial.

O estudo de caso abrange o planejamento e a programação dos ciclos de produção relacionados à solução FJSP. A estrutura básica do problema que está sendo resolvido está relacionada a um conjunto de trabalhos que precisam ser realizados nas máquinas, e cada trabalho é alocado a uma lista de atividades que são processadas na ordem. A essência do problema e da atividade definida é manter o tempo total de conclusão o baixo possível, de acordo com as operações planejadas com o tempo definido de cada operação individualmente. O conjunto de operações realizadas para tornar um trabalho completo e, portanto, um produto acabado. O modelo matemático do FJSP pode ser representado da seguinte forma (Özgüven et al., 2010).

É necessário programar *n* produtos $J = \{J_1, J_2, ..., J_n\}$, com cada trabalho J_j (j = 1,2, ... *n*) tendo uma ordem predeterminada de operações *nj* (O_1, j, O_2, j , They, *j*), que deve ser realizadas na ordem dada em *m* máquinas $M = \{M_1, M_2, ..., M_m\}$. Em um problema totalmente flexível, cada máquina pode realizar apenas uma operação por vez, e o tempo de processamento de cada operação depende das máquinas disponíveis e é representado por *p, i, j, k* (tempo de processamento da operação O_i, *j* na máquina M_k). O objetivo do problema observado é atribuir cada operação a uma máquina correspondente e, em seguida, determinar a disposição de todas as máquinas atribuídas às operações

para reduzir a função-alvo, neste caso, minimizando o processo de fabricação com base no GA. Um exemplo do problema examinado é apresentado na Tabela 5.

Tabela 5. Matriz de entrada do FJSP parcial

Empregos	Operações	M1	M2	M3	M4	M5	M6
J1	O11	7	-	1	5	-	-
	O21	3	-	-	5	-	8
	O31	-	6	-	7	-	10
	O41	7	-	5	7	-	-
	O51	-	5	-	-	6	3
	O61	7	-	-	-	6	3
J2	O12	-	8	-	8	-	9
	O22	5	-	5	-	-	4
	O32	10	-	11	-	10	-
	O42	8	-	7	-	-	10
	O52	10	-	-	10	-	12
	O62	-	7	15	4	-	-
J3	O13	8	-	5	-	7	-
	O23	-	4	-	4	6	-
	O33	-	-	-	-	-	8
	O43	9	-	8	-	7	-
	O53	-	3	-	5	-	3
	O63	-	5	6	-	7	-
J4	O14	-	5	-	6	-	9
	O24	5	-	4	-	8	-
	O34	9	-	5	-	-	10
	O44	5	11	-	3	-	-
	O54	15	-	9	-	8	-
	O64	-	10	-	11	-	9
J5	O15	9	-	9	-	9	-
	O25	-	3	6	-	8	-
	O35	-	6	7	-	5	-
	O45	-	6	-	5	-	4
	O55	3	-	4	-	4	-
	O65	-	8	-	6	-	9
J6	O16	-	3	-	5	-	4
	O26	8	-	-	3	6	-
	O36	7	-	8	-	-	9
	O46	10	-	8	9	-	-
	O56 O66	-	9	-	10	4	-
		7	-	6	-	8	-

Deve-se observar que nem todas as máquinas precisam ser capazes de realizar todas as operações, como pode ser visto na Tabela 5 anexa. Essa abordagem de solução de problemas é chamada de solução parcial de problemas ou flexibilidade parcial, em que algumas operações só podem ser realizadas em máquinas específicas. Esses exemplos são muito mais comuns em casos do mundo real durante a otimização dos processos de produção. A Tabela 5 do problema descrito pode mostrar 6 trabalhos e 6 máquinas, o trabalho 1 tem 6 operações, a primeira operação só pode ser processada por uma máquina e essa máquina é a $_{M3}$ com um tempo de processamento de 1. Podemos ver os resultados gráficos do primeiro caso investigado de flexibilidade parcial de pesquisa na Figura 4 no gráfico de Gantt.

Um dos meios mais comuns de apresentar um resultado no caso do planejamento da produção são os gráficos de Gantt e, como são chamados na literatura de pesquisa, os gráficos de Gantt. O gráfico de Gantt é um diagrama em um sistema de coordenadas no qual o eixo horizontal é o tempo e o eixo vertical mostra as tarefas de planejamento para determinar: o início, a duração total e o fim do ciclo, que é o principal problema na determinação do planejamento e da programação nesse caso de máquina e trabalho. Outro caso examinado de planejamento e programação é chamado de FJSP total, com cada operação podendo ser implementada em qualquer uma das *m* máquinas disponíveis, já que todas as máquinas são capazes de executar cada operação em um horário especificado durante um tempo de processamento especificado de cada operação. Um exemplo do problema examinado é apresentado na Tabela 6.

Tabela 6. Matriz de entrada do FJSP total

Empregos	Operações	M1	M2	M3	M4	M5	M6
J1	O11	7	11	9	7	8	9
	O21	3	6	12	10	5	7
	O31	5	6	6	11	7	10
	O41	5	5	7	7	8	7
	O51	9	5	10	9	6	3
	O61	10	3	11	8	6	7
J2	O12 O22 O32 O42 O52 O62	7	8	7	5	7	7
		4	7	5	10	9	8
		5	7	9	10	10	9
		3	5	11	10	9	10
		10	7	9	9	6	6
		3	9	5	4	11	10
J3	O13 O23 O33 O43 O53 O63	5	6	5	7	8	9
		2	5	9	4	8	9
		8	5	9	10	10	8
		9	5	9	10	11	8
		7		7	10	7	9
		7	9	9	10	7	11
J4	O14 O24 O34 O44 O54 O64	7	5	7	8	9	10
		5	5	7	9	9	9
		5	9	5	10	6	10
		6	7	8	3	9	3
		2	5	9	9	8	10
		7	9	9	10	11	9
J5	O15 O25 O35 O45 O55 O65	9	5	9	8	10	11
		6	5	5	8	10	6
		9	5	9	10	5	9
		5	9	10	11	12	4
		3	5	5	6	9	11
		9	8	9	8	10	7
J6	O16 O26 O36 O46 O56 O66	2	3	9	8	10	7
		2	5	9	3	9	7
		2	5	9	9	10	9
		10	9	10	9	9	10
		9	5	10	6	4	9
		10	5	9	4	9	8

A Tabela 6 mostra os parâmetros de entrada do problema de teste em que temos 6 trabalhos (produtos) e cada um desses 6 trabalhos tem 6 operações, cada uma das quais pode ser executada em cada máquina com um tempo de processamento em minutos. Como podemos ver na parte anterior do documento, chamamos esse caso de flexibilidade completa. Os resultados gráficos da pesquisa podem ser vistos na Figura 5. Quanto à implementação do GA, os resultados experimentais foram testados em python e os valores numéricos foram derivados em uma plataforma de PC padrão baseada no MS Windows.

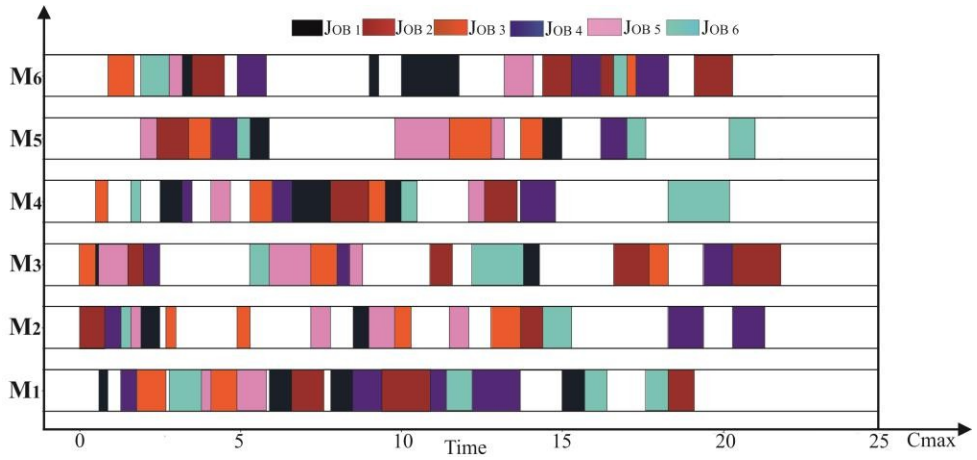

Figura 4. Representação gráfica dos resultados do primeiro caso da Tabela 5

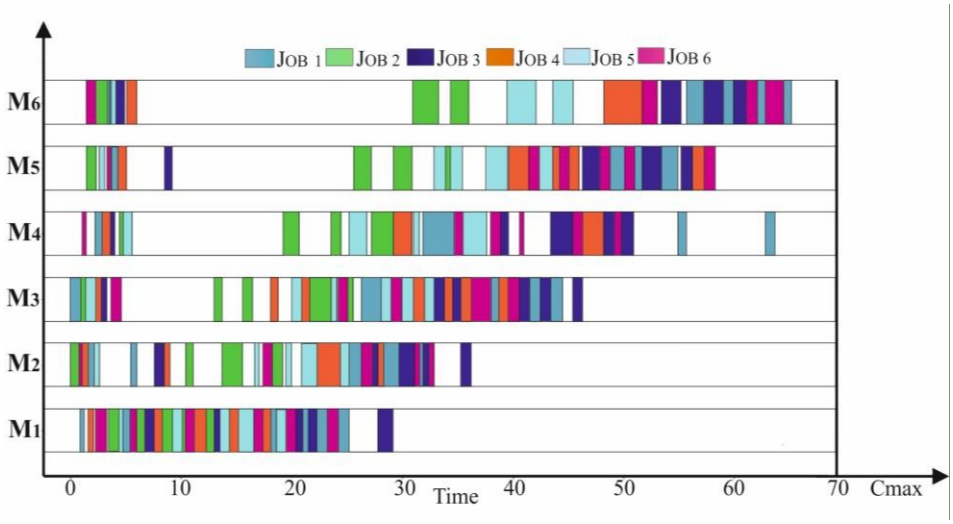

Figura 5. Representação gráfica dos resultados do segundo caso da Tabela 6

5. Resultados e discussão

Após discussões introdutórias e esclarecimentos sobre o FJSP, bem como uma revisão da literatura de pesquisa sobre o tópico de planejamento, uma configuração específica dos problemas e das

resultados do planejamento foi feito. É importante observar que os dados de teste testados são fornecidos aleatoriamente. Como pode ser visto na parte anterior do documento, foram testados dois casos de planejamento e programação, que era ao mesmo tempo o objetivo desse problema. A Tabela 5 mostra que nem todas as máquinas podem executar todas as operações durante o planejamento e a programação das operações. A flexibilidade parcial é uma das causas mais comuns que podem ser encontradas na manufatura; muitas vezes, nem todas as máquinas podem executar todas as operações para realizar o trabalho (produto). Os resultados dos dados de entrada da Tabela 5 são apresentados graficamente no gráfico da Figura 4 para o primeiro caso de flexibilidade parcial. Na Figura 4, podemos ver o layout das operações em máquinas individuais, dependendo das restrições iniciais. *J* define cada trabalho em uma cor diferente para diferenciar os trabalhos (produtos). No eixo horizontal, o tempo é apresentado com o valor da função de critério C_{max} = 218 unidades em minutos. Com relação aos testes, observou-se que a busca foi realizada por meio de um algoritmo genético com MaxIter = 500. A pesquisa baseada em GA em uma linguagem de programação python, ou seja, o tempo total de planejamento e programação é de 41,63 segundos para o primeiro caso testado. No eixo vertical, podemos ver todas as 6 máquinas que estão alinhadas linearmente. Com relação ao segundo caso examinado, ele é apresentado na Tabela 6 e é claramente diferente do primeiro caso examinado. Na Tabela 6, com base nos dados de entrada apresentados, é possível ver a flexibilidade total em que cada máquina pode executar cada operação com tempo definido.

Além disso, nesse caso, temos *J*= 6 trabalhos (produtos) e cada um dos trabalhos tem O_{ij}= 6 operações que podem ser executadas em cada máquina, dependendo da programação de operações com um tempo de processamento definido para cada operação individualmente. No eixo horizontal, podemos ver o tempo total representado pelo valor da função de critério C_{max} = 652 unidades em minutos. O valor da função de critério que representa o tempo total de conclusão de todas as operações e a conclusão do processo de planejamento é representado pela marca C_{max}, que pode ser vista na parte anterior do documento e é apresentada em minutos. O gráfico de Gantt apresenta um cronograma detalhado das operações que serão realizadas nas máquinas. As máquinas são representadas linearmente no eixo vertical. O tempo total de simulação com base no GA em uma linguagem de programação python é de 90,89 em segundos. O objetivo do problema examinado de planejamento e programação de operações foi alcançado com sucesso, como pode ser visto nos resultados em anexo. Os resultados obtidos confirmam o sucesso do algoritmo genético na solução do FJSP.

Com relação à proposta para pesquisas futuras, o problema FJS pode ser ampliado observando-se os trabalhadores disponíveis que participam da realização das atividades de produção. Nesse problema, os trabalhadores e as máquinas são limitados, e o problema é chamado de Dual Resource Constrained Flexible Job Shop.

Agradecimento

Esta pesquisa recebeu apoio financeiro do Ministério da Educação, Ciência e Desenvolvimento Tecnológico da República da Sérvia.

Referências

Aslan, E., Şimşek, T., & KarkacieR, A. (2017). A binary integer programming model for exam scheduling problem with sSeveral departments, 13th International Conference on Knowledge, Economoy and Management, Bilgi Ekonomisi ve Yönetimi Dergisi, 12 (2), 169-175.

Blazewicz, J., Domschke, W., & Pesch, E. (1996). The job shop scheduling problem: Conventional and new solution techniques. European Journal of Operational Research, 93, 1-33.

Bremermann, H. J. (1958). The evolution of intelligence. The nervous system as a model of its environment, Technical Report No. 1, Department of Mathematics, University of Washington, Seattle, WA.

Brucker, P., Schile, R. (1990). Job-shop scheduling with multi-purpose machines. Computing, 45(4), 369-375.

Dorigo, M. (1992). Optimization, learning and natural algorithms (otimização, aprendizado e algoritmos naturais) (em italiano), dissertação de doutorado, Dipartimento di Elettronica, Politecnico di Milano, Itália.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant Colony Optimization (Otimização por colônia de formigas). IEEE Computational Intelligence Magazine 1(4):28-39, DOI: 10.1109/MCI.2006.329691

Fan, K., Wang, M., Zhai, Y., & Li, X. (2019). Algoritmo de busca por dispersão para o problema de agendamento de tarefas de multiprocessador em um job-shop. Computers & Industrial Engineering, 127, 677-686.

Fraser, A. S., (1957). Simulation of genetic systems by automatic digital computers (Simulação de sistemas genéticos por computadores digitais automáticos). II: Effects of linkage on rates under selection, Austral. J. Biol. Sci. 10:492-499.

Glover, F. W. (1989). Tabu Search - Parte 1. ORSA Journal on Computing. 1 (2): 190-206. doi:10.1287/ijoc.1.3.190.

Glover, F. W., & Laguna, M. (1997). Tabu Search. Springer US.

Graver, M., Price, W. L. (1987). Using the Kanban in a job shop environment. International Journal of Production Research, 26(6):1105-1118, DOI: 10.1080/00207548808947921

Hasan, M., & Arefin, R. (2017). Application of Linear Programming in Scheduling Problem (Aplicação de programação linear em problemas de agendamento). Dhaka University Journal of Science, 65(2): 145-150.

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems (Adaptação em sistemas naturais e artificiais). University of Michigan Press, Ann Arbor.

Jamili, A., (2018). Programação de job shop com consideração de tempos de parada flutuantes sob incerteza. Aplicações de engenharia da inteligência artificial, Volume 78, 28-36.

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization (Otimização por enxame de partículas). Anais da Conferência Internacional do IEEE on Neural Networks. IV. pp. 1942- 1948. doi:10.1109/ICNN.1995.488968.

Kirkpatrick, S., Gelatt Jr., C. D., Vecchi, M. P. (1983). Optimization by Simulated Annealing (Otimização por recozimento simulado). Science. 220 (4598): 671-680.

Kumar, C. S., Panneerselvam, R., (2007). Revisão da literatura sobre o sistema JIT-KANBAN. International Journal of Advanced Manufacturing Technology 32(3):393-408, DOI: 10.1007/s00170-005-0340-2.

Kung, L. C., & Chern, C. C. (2009). Heuristic factory planning algorithm for advanced planning and scheduling (Algoritmo heurístico de planejamento de fábrica para planejamento e programação avançados). Computers&OperationsResearch, 36(9), 2513-2530.

Lagodimos, A. G., & Leopoulos, V. (2000). Greedy heuristic algorithms for manpower shift planning, International Journal of Production Economics, 68, 95-106.

Laguna, M., Marti, R., & Marti, R. C. (2003). Scatter Search: Methodology and Implementation in C. New York: Springer.

Liaw, C. F. (2000). A hybrid genetic algorithm for the open shop scheduling problem (Um algoritmo genético híbrido para o problema de programação de lojas abertas). European Journal of Operational Research, 124(1):28-42.

Liu, G., Luh, P.B., & Resch, R. (1996). Scheduling permutation flow shop using the lagrangian relaxation technique. 13º Congresso Mundial Trienal da IAFC. São Francisco. EUA, la-Ol 6.

Lomnicki, Z. A. (1965). Algoritmo "Branch-and-Bound" para a solução exata do problema de programação de três máquinas. J Oper Res Soc 16, 89-100. https://doi.org/10.1057/jors.1965.7

Nowicki, E., & Smutnicki, C. (2005). Um algoritmo avançado de busca tabu para o problema de job shop. Journal of Scheduling, 8(2), 145-159.

Ostergard, P. R. J. (2002). A fast algorithm for the maximum clique problem, Discrete Applied Mathematics, 120(1-3):197-207.

Özgüven, C., Özbakır, L., Yavuz, Y. (2010). Mathematical models for job-shop scheduling problems with routing and process plan flexibility, Applied Mathematical Modelling, 34, 1539-1548.

Robson, J.M., (1986). Algoritmos para conjuntos independentes máximos. Journal of Algorithms, 7(3):425-440.

Schaefers, J., Briquet, M., & Colin, J. (2000). A generic KANBAN based push-pull schedule in a job shop. IFAC Proceedings Volumes, 33(17):585-590

Sentlelro, J. J. S., (1933). Planning and scheduling: non-classic heuristic (Planejamento e programação: heurística não clássica), 12º Congresso Mundial Trienal. Sydney. Austrália.

Sobeyko, O., & Mönch, L. (2017). Planejamento e programação de processos integrados para job shops flexíveis de grande escala usando metaheurística. International Journal of Production Research, 55:2, 392-409, DOI: 10.1080/00207543.2016.1182227

Sörensen, K., & Glover, F. (2013). Metaheuristics (Metaheurística). Encyclopedia of Operations Research and Management Science. Springer, Nova York.

Spyropoulos, C. D., (2000). AI planning and scheduling in the medical hospital environment (Planejamento e programação de IA no ambiente médico hospitalar). Artificial Intelligence in Medicine, 20(2):101-111.

Stanković, A., Marković, D., Petrović, G., Ćojbašić, Ž. (2019). Simulated annealing and particle swarm optimization for the vehicle routing problem and communal waste collection in urban areas, 14ª Conferência Internacional sobre Realizações em Engenharia Mecânica e Industrial, DEMI 2019, 497-505, isbn: 978-99938-39-84-2.

Talbi, E. G. (2009). Metaheuristics: From design to implementation, John Wiley & Sons.

Tamssaouet, K., Dauzère-Pérès, S., & Yugma, C. (2018). Metaheurística para o problema de agendamento de job- shop com restrições de disponibilidade de máquina. Computers & Industrial Engineering, 125:1-8.

Thomalla, C. (2001). Job shop scheduling with alternative process plans. International Journal of Production Economics 74(1-3):125-134 DOI: 10.1016/S0925-5273(01)00119-0

Thürer, M., Huang, G., Stevenson, M., Silva, C., & Filho, M. (2012). O desempenho das regras de definição de datas de vencimento em job shops de montagem e de múltiplos estágios: uma avaliação por simulação. International Journal of Production Research, 50(20), 5949-5965.

Thürer, M., Land, M. J., Stevenson, M., & Fredendall, L. D. (2017). On the integration of due date setting and order release control (Sobre a integração da definição de data de vencimento e controle de liberação de pedidos). Production Planning & Control, 28(5), 420-430.

Vandaele, N. J. (2000). An integrated, hierarcmcal framework for planning, scheduling and controlling job shop. IFAC Proceedings Volumes, 30(14):121-126.

Xing, W., & Zhang, J. (2000). Parallel machine scheduling with splitting jobs (Programação de máquinas paralelas com divisão de tarefas). Discrete Applied Mathematics, 103(1-3):259-269.

Yang, X. S., (2010). Engineering Optimization: An introduction with metaheuristic applications. Universidade de Cambridge, Reino Unido.

Zhang, C. Y., Li, P. G., Guan, Z. L., & Rao, Y. Q. (2007). A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. Computers & Operations Research, 34(11):3229-3242

Zhang, G., Gao, L., & Shi, Y. (2011). An effective genetic algorithm for the flexible job- shop scheduling problem, Expert Systems with Applications, 38(4):3563-3573.