Algoritmo de colônia artificial de abelha e da têmpera simulada para resolução do problema da programação da produção *job-shop* flexível

Maurício Luiz Sobrinho[1] e Jacinto José Franco[2]

Departamento de Computação da Universidade Federal de São Carlos (DC-UFSCar)

São Carlos - Brasil

m.luizsobrinho@gmail.com, jacinto.franco@bag.ifmt.edu.br

Abstract. *The modeling and control of product manufacturing systems or efficient, optimized and sustainable production increase in the largest industrial sectors. Techniques that aim to optimize as stages of a productive chain to boost a company's profitability. In this context, the problem of flexible job-shop scheduling (JSP) arises, which consists in searching for an arrangement of tasks and tasks performed in a given set of machines. This work proposes an implementation that combines the bee colony algorithm as the main heuristic and the simulated temperature algorithm as an auxiliary heuristic in the hierarchical approach to the JSP problem. The results obtained with our methodology are shown to be optimal or close to the level according to a specialized literature.*

Resumo. *A modelagem e o controle de sistemas de manufatura propiciam o aumento eficaz, otimizado e sustentável da produtividade na grande maioria dos setores industriais. Técnicas que visem otimizar as etapas de uma cadeia produtiva podem potencializar a lucratividade de uma empresa. Nesse contexto surge o problema do escalonamento job-shop flexível - JSP, que consiste na busca por um arranjo de jobs (tarefas) e operações executadas em um determinado conjunto de máquinas no menor tempo possível (makespan). Este trabalho propõe uma implementação que combina o algoritmo de colônia artificial de abelha como heurística principal e o algoritmo da têmpera simulada como heurística auxiliar na abordagem hierárquica do problema JSP. Os resultados obtidos com nossa metodologia se mostraram ótimos ou próximos do ótimo de acordo com a literatura especializada.*

1. INTRODUÇÃO

Neste trabalho é proposto um modelo para a programação da produção aplicada ao problema de escalonamento do *Job-Shop* flexível em um sistema de manufatura. Essa problemática modelada é considerada NP-hard (Muller et al. 2003) e se consiste na otimização do arranjo das partes em um conjunto de máquinas, através de uma sequência de operação em cada parte, de modo a minimizar o tempo total de produção (*Makespan*). A problemática descrita foi atacada através da combinação de duas heurísticas de busca probabilística aproximada distintas, o algoritmo de colônia de abelhas artificial e o algoritmo da têmpora simulada. Nessa abordagem

consideramos o problema no contexto Hierárquico, ou seja, em que o problema da programação e do roteamento são tratados (Oliveira et al. 2004) de forma separada por cada uma dessas heurísticas. Este artigo está organizado da seguinte maneira:

Na seção 2 são apresentados os conceitos e definições acerca do problema da programação *job-shop flexible*, a seção 3 trata do algoritmo de colônia artificial de abelha, a seção 4 aborda o algoritmo da têmpera simulada (*simulated annealing*, do inglês), na seção 5 é descrita toda a metodologia abordada, na seção 6 é realizada a análise detalhada do desempenho do algoritmo implementado nesse trabalho. Finalmente na seção 8 são debatidas as conclusões finais do artigo e as projeções para trabalhos futuros que poderão dar continuidade ao que foi proposto até aqui.

2. O PROBLEMA DA PROGRAMAÇÃO DA PRODUÇÃO *JOB-SHOP* FLEXÍVEL

O problema da programação da produção job-shop, considerado um problema NP-Hard, se consiste na programação de tarefas (jobs) que chegam ao sistema para serem processadas através de uma sequência de operações em um conjunto de máquinas, sendo que esse processo deve ser otimizado no sentido de alguma medida de desempenho (Xia et al. 2005) nesse presente trabalho otimizado em termos do tempo total de produção (Makespan).

Existem várias abordagens para esse problema (Oliveira et al. 2004) no entanto nossa pesquisa está pautada na abordagem *flex* (Xia et al. 2005) que se refere ao fato de que as máquinas podem realizar mais de um conjunto de operações de forma variável

Dois conceitos essenciais nesse problema são o conceito de programação e roteamento, em nosso trabalho modelamos o problema através da abordagem hierárquica, sendo que para tal implementamos o ACO como heurística principal para o tratamento do problema da produção e o SA como heurística auxiliar para o tratamento do problema do roteamento.

3. ALGORITMO DE COLÔNIA ARTIFICIAL DE ABELHA

Muitos são os comportamentos biológicos, evolutivos e sociais (cooperativos, competitivos etc) que inspiram a matemática e a computação na busca por soluções ótimas dos mais diversos problemas (Karaboga et al. 2009). Essa inteligência coletiva apresentada por diversas espécies de seres vivos, traz consigo aspectos de sobrevivência na busca por alimentos, parceiros de acasalamento, abrigo, dentre outros objetivos, que serviram de inspiração (bio-inspiração) para o desenvolvimento de conceitos e técnicas capazes de encontrar soluções aproximadas de interesse para problemáticas dos mais diversos campos de pesquisa. Nesse contexto, o Algoritmo de Colônia de Abelhas Artificiais (ABC - do inglês *Artificial Bee Colony*) é um algoritmo bio-inspirado proposto em 2005 por Karaboga para a resolução de problemas de otimização multimodais e multidimensionais. Ele foi inspirado no comportamento coletivo das abelhas na busca por fontes de alimento, com o objetivo de encontrar soluções aproximadas em problemas de otimização discretos, sendo inicialmente desenvolvido para ser aplicado à problemas sem restrições.

Esse algoritmo possui a representação de três grupos de abelhas (Karaboga et al. 2009). As empregadas determinam aleatoriamente as fontes iniciais de alimento em um dado espaço de busca; essas informações iniciais são passadas para as abelhas assistentes que

as memorizam e assim passam a procurar fontes melhores de alimento através de buscas locais por novas fontes de alimentos. Se essa busca local não levar a melhoria da solução, tal fonte de alimento é abandonada sendo assim acionadas as batedoras, que são um tipo de abelhas empregadas que partem em busca de novas fontes de alimento.

4. ALGORITMO DA TÊMPERA, RECOZIMENTO OU ARREFECIMENTO SIMULADO (*SIMULATED ANNEALING*)

Proposto em 1983 por Scoot Kirkpatrick et al., foi inspirada na tentativa de simular o processo de recozimento de metais MMM 8/. Nesse processo o resfriamento rápido de um metal conduz a produtos meta-estáveis, de maior energia interna, porém o seu resfriamento lento conduz a produtos mais estáveis, estruturalmente fortes e de menor energia.

Durante o recozimento o material passa por vários estados possíveis sendo que em um tempo suficientemente longo um elemento qualquer do ensemble passa por todos os seus estados acessíveis.

A fundamentação desse método está baseada no Algoritmo de Metropolis (Gibbs, 1953), que se consiste na tomada de uma sequência de temperaturas decrescentes para gerar soluções de um problema de otimização. Esse algoritmo começa com um valor T elevado e a cada T geram-se soluções até que o equilíbrio daquela temperatura seja alcançado (Scoot Kirkpatrick et al 1983)

A temperatura é então rebaixada e o processo prossegue até o congelamento (ou seja, não se obtêm mais uma diminuição de custo). A sequência de temperaturas empregadas, juntamente com o número de iterações a cada temperatura, constitui uma prescrição de *anneling* que deve ser definida empiricamente e os estados possíveis de um metal durante o processo de arrefecimento correspondem a soluções do espaço de busca.

A energia em cada estado corresponde ao valor da função objetivo sendo que a energia mínima (se o problema for de minimização ou máxima, se de maximização) corresponde ao valor de uma solução ótima local, possivelmente global para o problema (Scoot Kirkpatrick et al. 1983)

A cada iteração do método, um novo estado é gerado a partir do estado corrente por uma modificação aleatória neste, se o novo estado é de energia menor que o estado corrente, esse novo estado passa a ser o estado corrente, se o novo estado tem uma energia maior que o estado corrente um dado intervalo de tempo, calcula-se a probabilidade de se mudar do estado corrente para o novo.

5. GERENCIAMENTO DE ESPAÇOS LIVRES

Para o desenvolvimento deste trabalho, múltiplas abordagens inspiradas no gerenciamento de espaços livres de um sistema operacional foram utilizadas para a escolha das máquinas onde as operações das tarefas iriam executar, o que possibilitou chegar em resultados próximos do ótimo para os arquivos 8x8, 10x10 e no ótimo para 3x3 e 4x5.

No trabalho, a abelha assistente fez uso de métodos puramente aleatório e, para a empregada, buscou-se adaptar os algoritmos ***Worst Fit*** (pior encaixe), ***Best Fit***(melhor

encaixe), ***First Fit***(primeiro encaixe) do gerenciamento de memória dos sistemas operacionais Tanenbaum(1995) para o problema *Flexible Job Shopt* e mais o algoritmo ***Pior Permitido***. Para que fosse possível a utilização de tais abordagens foi necessário manter uma lista de espaços livres e, para cada espaço, avaliou-se a possibilidade de alocar uma operação para a porção de tempo disponível da máquina.

Quando o gerador de números aleatórios seleciona o identificador de uma das abordagens acima listadas, percorre-se, por meio de um laço de repetição a listagem de espaços livres com o intuito de encontrar uma máquina que tenha tempo disponível e que possibilite a execução da referida operação. A partir deste ponto, segue-se as seguintes regras:

1 *Worst Fit* – tem a intenção de verificar qual é o espaço que permite a execução da operação e que deixe a maior quantidade de espaços livres.
2 *Best Fit* – escolhe o espaço para alocar a operação que deixe o menos ou nenhum espaço livre.
3 *First Fit* –a alocação ocorrerá quando for encontrado um espaço livre que permita executar a operação e quando encontra o laço é finalizado.
4 *Pior Permitido* – escolhe o espaço a ser alocado com base no espaço entre duas operações. O critério estabelecido é que o espaço deve ser superior a 1 unidade de tempo.

5 DESCOBERTA DE NOVAS FONTES DE ALIMENTO

Para a descoberta de novas fontes de alimento no *Flexible Job Shop* realiza de duas randomizações. A primeira define uma sequência de tarefas (por exemplo, 3 1 2 4 5 para 5 tarefas) e a segunda que determina uma sequência de atribuição de máquinas para as operações (por exemplo, 4, 1 e 2, onde a quarta máquina executa a primeira operação).

Para gerar e filtrar quais fontes de alimento irão fazer parte da listagem global utilizou-se a têmpera simulada com um filtro baseado no *fitness*, onde para cada fonte de alimento gerado o algoritmo da têmpera simulada toma a decisão de agregar ou não na lista de aceitos de acordo com a temperatura atual da índice do laço. Após conclusão do procedimento da têmpera, calcula-se o *fitness* das fontes geradas e considera somente as melhores fontes na listagem global utilizada por todas as abelhas. O cálculo do *fitness* das fontes é necessário, pois notou-se que a têmpera simulada demora um certo tempo para explorar a vizinhança, mas como a quantidade de fontes de alimento necessário é de somente 50 na global e 30 quando executado a têmpera.

6 DETALHES GERAIS DA IMPLEMENTAÇÃO

A Figura 1 mostra o fluxograma simplificado dos algoritmos desenvolvidos no presente trabalho. Como pode ser averiguado, o algoritmo é subdividido em três blocos de acordo com o tipo de abelha, onde pode ser uma empregada, assistente ou batedora.

A abelha empregada fica encarregada de executar a maior parte dos algoritmos propostos neste trabalho, onde executa-se funções que visam otimizar o uso do tempo de livre das máquinas e, após este processo, uma nova fonte de alimento é gerada e só será considerada na lista global de fontes de alimento caso seja notado alguma melhora. A assistente faz uso somente de métodos aleatórios, tornando a execução mais rápida,

mas menos eficaz que o trabalho executado pela empregada. Para toda fonte de alimento gerada nos laços de repetição da empregada e da assistente são efetuadas checagens para averiguar se a fonte de alimento melhorou ou não.

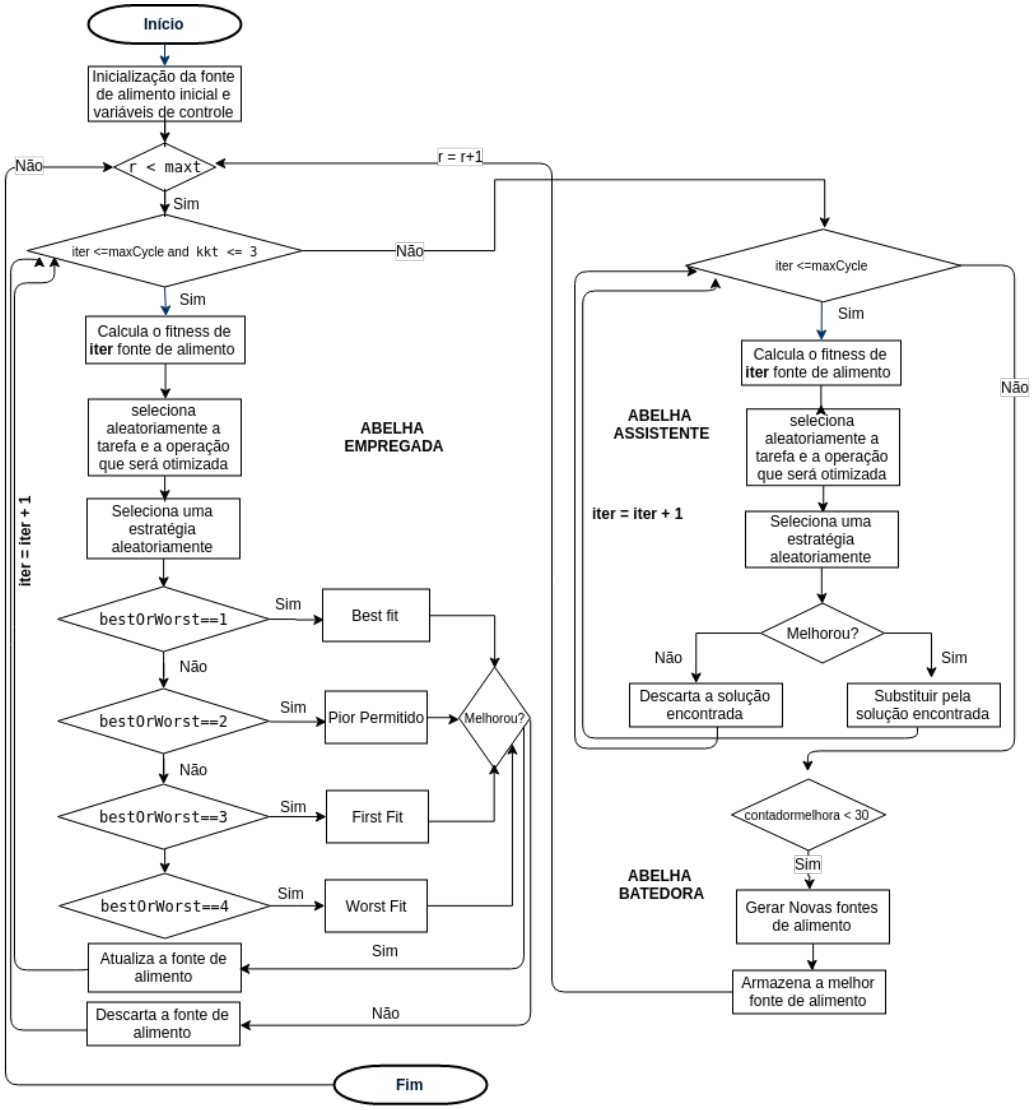

Figura 1: Fluxograma simplificado do algoritmo desenvolvido

A última abelha, a batedora é regida pelo contador de melhora e caso este contador contenha valores maiores que 29, novas fontes de alimento serão geradas com a têmpera simulada. O código do algoritmo desenvolvido se encontra hospedado no https://github.com/geekaia/flexiblejobshopscilab.git, o que permite ao leitor etudar e analisar a maneira com que o problema do Flexible Job Shop foi resolvido com ABC e têmpera simulada.

7 RESULTADOS

Para elucidar a efetividade do algoritmo desenvolvido no presente trabalho utilizou-se 4 instâncias do problema *Flexible Job Shop* de Kacem et al. (2002), o 3x3,

4x5, 8x8 e 10x10. Para cada instância foram executadas o mínimo de 10 repetições. A seguir será esboçado os resultados comentados sobre cada instância.

7.1 ARQUIVO 3X3 E 4X5

Estas instâncias não devem ser consideradas como um desafio computacional, pois o tempo requerido para encontrar o ótimo é de poucos segundos e na maior parte dos testes realizados para o 3x3 o ótimo é encontrado na primeira iteração do laço principal. Contudo, estes arquivos serviram para validar a abordagem inicialmente desenvolvida neste trabalho e os resultados estão ilustrados na figura abaixo.

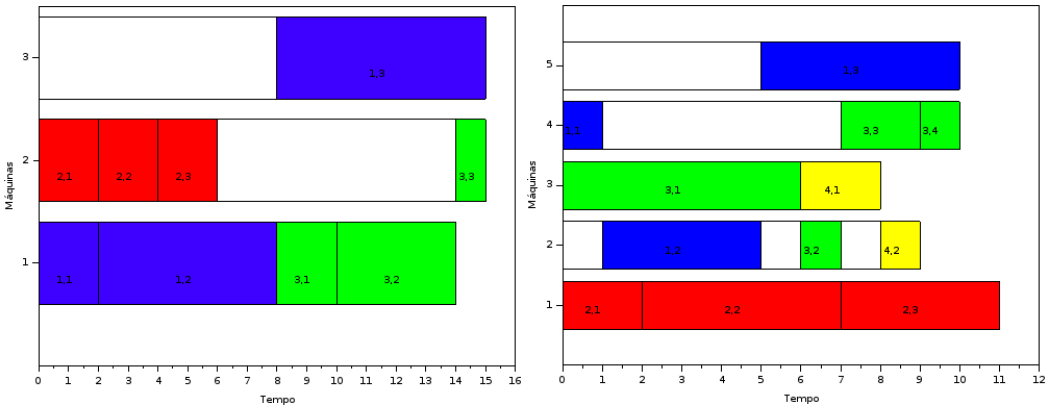

Figura 2: Gráfico de Gantt 3x3 (esquerda) e 4x5 (direita)

7.2 ARQUIVO 8X8

Esta instância é considerada semi flexível, o que implica que algumas máquinas não podem executar determinadas operações. Das instâncias testadas, foi esta que demandou o maior esforço computacional para chegar próximo do valor ótimo e embora o ótimo do algoritmo seja de apenas 17. É importante ressaltar que nem todas as execuções conseguiram chegar no valor ótimo deste algoritmo, o que pode ser visualizado na Tabela 1 e ilustrado na Figura 6 das iterações requeridas e do tempo para a conclusão total e do tempo de cada repetição (Figura 6). O tempo estimado para a finalização de cada execução foi calculado de acordo com o tempo médico do Figura 4.

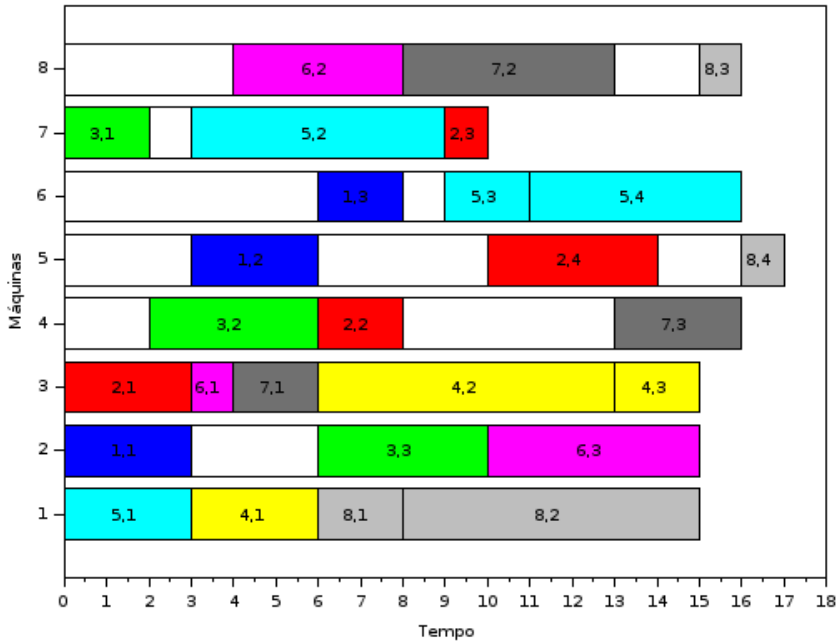

Figura 3: Gráfico de G*antt* para a instância 8x8

Tabela 1: Dados das repetições da instância 8x8 e 10x10

Arquivo	**Repetição**	**Iteração**	***Fitness***	**Tempo em minutos**
8x8	1	2061	17	106
8x8	2	799	17	41
8x8	3	255	18	13
8x8	4	48	19	2
8x8	5	1913	17	99
8x8	6	331	19	17
8x8	7	96	19	5
8x8	8	331	19	17
8x8	9	2000	17	103
8x8	10	2628	17	135
8x8	11	1952	17	101
8x8	12	3891	17	200
8x8	13	598	17	31
8x8	14	351	17	18
8x8	15	3961	17	204
8x8	16	2143	18	110
8x8	17	879	18	45

10x10	1	14	9	1
10x10	2	72	8	6
10x10	3	374	8	33
10x10	4	249	8	22
10x10	5	69	8	6
10x10	6	27	9	2
10x10	7	77	8	7
10x10	8	74	8	7
10x10	9	58	9	5
10x10	10	23	8	2

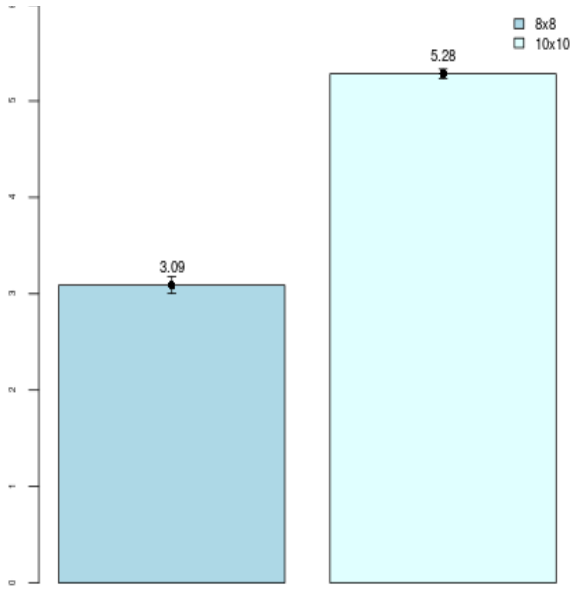

Figura 4: Tempo (em segundos) requerido para completar uma iteração do laço principal para os arquivos 8x8 e 10x10

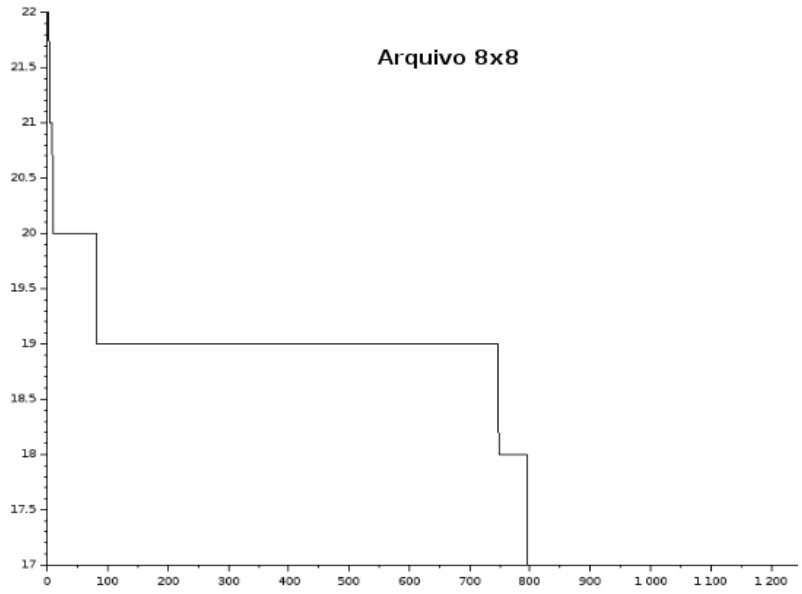

Figura 5: Mínimos de cada iteração do laço principal para o arquivo 8x8

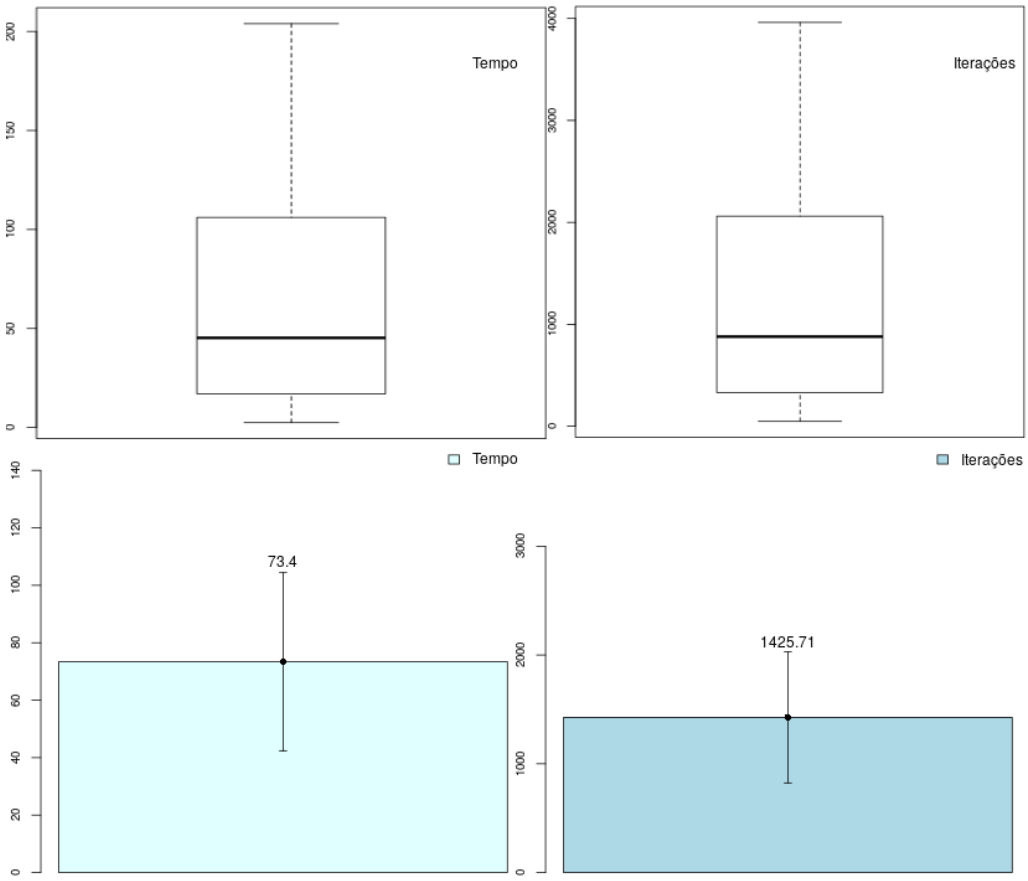

Figura 6: Tempo (esquerda) e Iterações (direita) requeridas até o algoritmo pare de melhorar

7.3 ARQUIVO 10X10

O algoritmo conseguiu chegar rapidamente perto do ótimo conhecido porque este problema é totalmente flexível e mais homogênio. Se comparado com o ótimo conhecido, a diferença foi de apenas 14,29% ou 1 unidade de tempo, o que pode ser verificado na Tabela 1, onde há o melhor valor obtido em cada repetição do laço mais externo do algoritmo. A configuração das máquinas, tarefas e operações podem ser visualizadas pelo gráfico de Gantt abaixo.

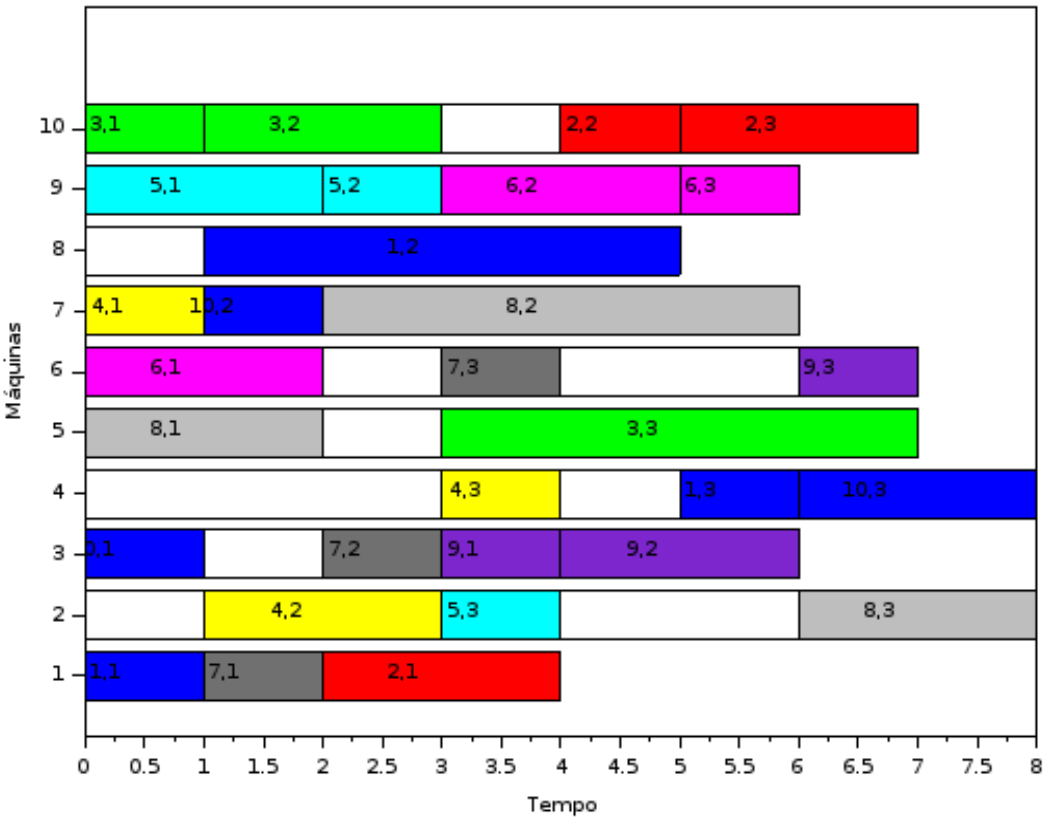

Figura 7: Gráfico de Gantt para o arquivo 10x10

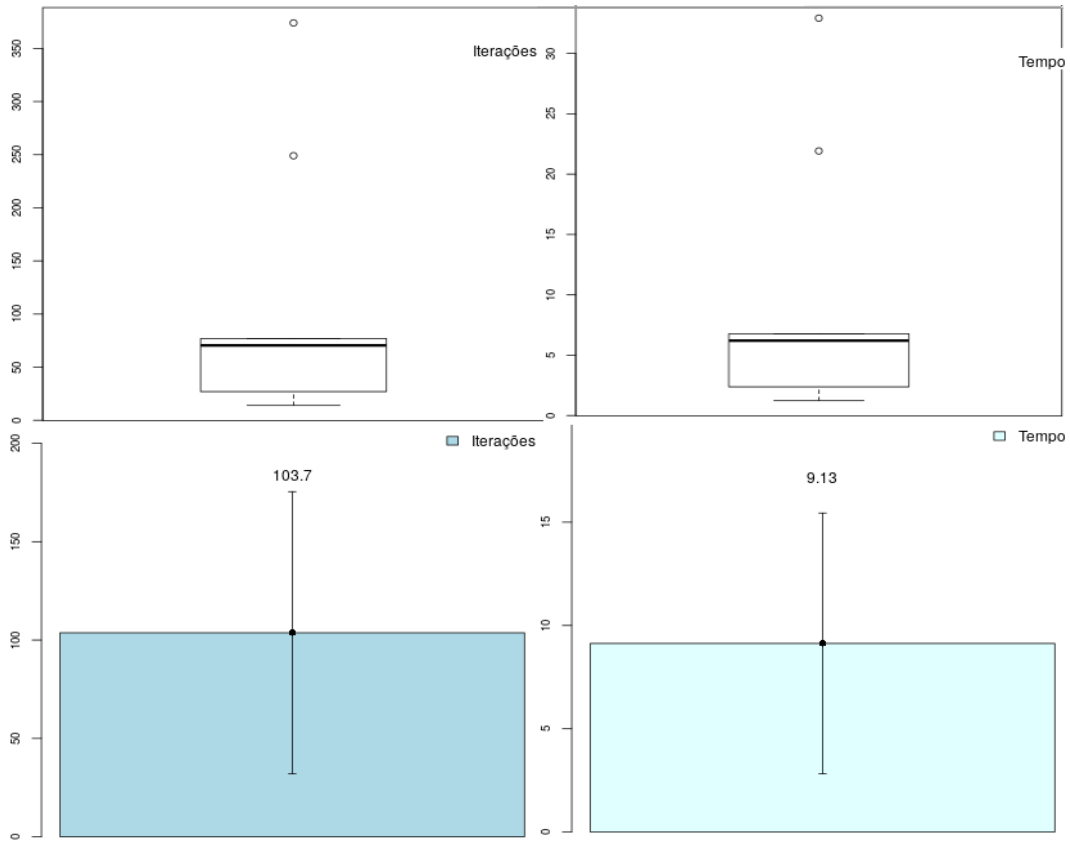

Figura 8: Iterações (esquerda) e Tempo (direita) até que o algoritmo pare de melhorar

Nos gráficos das iterações e tempo da Figura 9 verifica-se que o ótimo do algoritmo é alcançado com muito menos iterações se comparado com 8x8, o que implica uma economia em termos de processamento.

No caso da Figura 8, onde encontra-se o os boxplots e gráfico de barras com o intervalo de confiança onde podemos concluir que devido a variabilidade das iterações, outliers o intervalo de confiança ficou grande e para melhorar o intervalo de confiança seria necessário a replicação do experimento para obter mais dados. No boxplot pode-se notar que os outliers acabaram modificando a média do gráfico de barra das iterações e do tempo. Ainda no boxplot, nota-se também que a maior parte dos resultados das replicações estão contidas na parte inferior do gráfico.

Por fim, o gráfico contigo da Figura 9 corrobora com o que já foi dito acerca do resultado do 10x10, pois ilusta a rápida obtenção do ótimo do algoritmo.

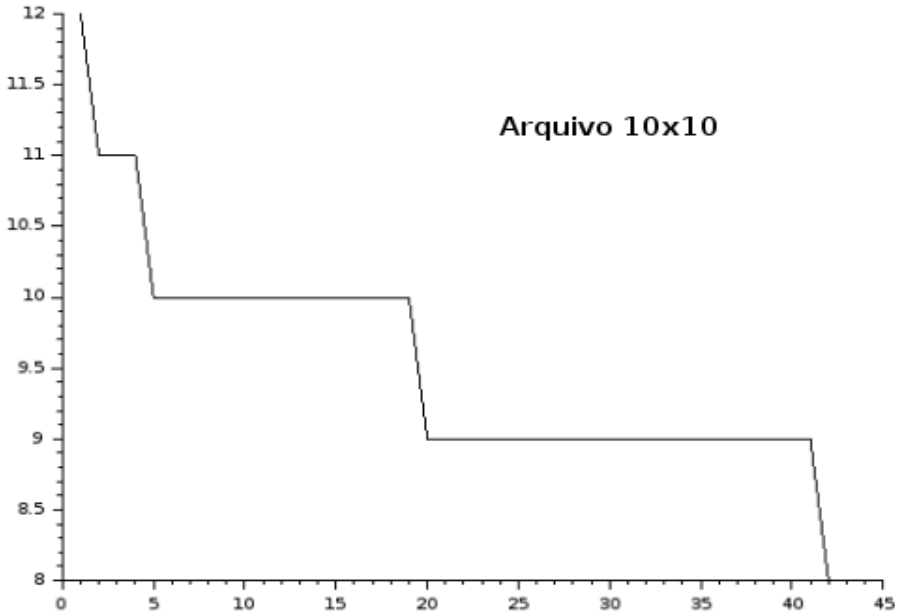

Figura 9: Mínimos de cada iteração do laço principal para o arquivo 10x10

7. CONCLUSÕES

Nesse trabalho foi apresentada uma implementação utilizando duas heurísiticas distintas para a resolução do problema da produção JSP. Os resultados obtidos se mostraram ótimos ou próximos do ótimo de acordo com a literatura da área. Toda a metodologia adota foi apresentada assim como a descrição aprofundada das análises de performance para o algoritmo proposto. Por fim, é possível concluir que esse desenvolvimento se mostrou adequado ao propósito inicial dessa pesquisa.

Como sugestões para trabalhos futuros é possível pontuar a implementação de outras heurísticas principais e auxiliares com o intuito de comparar performances. Também é sugestiva a ideia de utilizar outros benchmarks para serem testado com o código que foi desenvolvido nesse projeto. Por fim, é interessante a pretensão de aumentar a complexidade do modelo proposto abordando variáveis que tornem o problema o mais próximo possível das necessidades reais de uma indústria tais como setup das máquinas, tempo de entrega etc, ou ainda tornar o problema multiobjectivo, acrescentando ainda mais complexidade ao que foi apresentado nesse trabalho.

8 REFERÊNCIAS

Kacem, I., Hammadi, S., & Borne, P. (2002). Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 32(1), 1-13.

Karaboga, D. & Akay, B. Artif Intell Rev (2009) 31: 61. doi:10.1007/s10462-009-9127-4

Muller, Gilberto Irajá, and Gómez Arthur Tórgo (2003). Um modelo para Programaçao de Produção nos Ambientes de Job-shop e Manufatura Flexível: Abordagem a partir de um Estudo de Caso. *XXXIX SBPO*

Oliveira, A. C., & Shimizu, T. (2004). Um algoritmo de busca genética híbrida para o problema de programação de tarefas em job shop dinâmico. *SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL*, *36*, 1239-1247.

Tanenbaum, Andrew S.; Machado Filho, Nery. Sistemas operacionais modernos. Prentice-Hall, 1995.

Xia, W., & Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, *48*(2), 409-425.

Yen, Gary G., and Brian Ivers. (2009). "Job shop scheduling optimization through multiple independent particle swarms." *International Journal of Intelligent Computing and Cybernetics* 2.1: 5-33.