UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS LONDRINA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

PAOLA MANTOANI RIGO

DESENVOLVIMENTO DE UM ALGORITMO GENÉTICO PARA RESOLUÇÃO DO PROBLEMA DE SEQUENCIAMENTO DE PRODUÇÃO EM JOB SHOPS FLEXÍVEIS

LONDRINA
2018

PAOLA MANTOANI RIGO

DESENVOLVIMENTO DE UM ALGORITMO GENÉTICO PARA RESOLUÇÃO DO PROBLEMA DE SEQUENCIAMENTO DE PRODUÇÃO EM JOB SHOPS FLEXÍVEIS

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Produção da Universidade Tecnológica Federal do Paraná, Câmpus Londrina, como requisito parcial para obtenção do título de “Engenheira de Produção”.

Orientador: Prof. Dr. Rafael Henrique Palma Lima

LONDRINA
2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Londrina
Coordenação de Engenharia de Produção
Engenharia de Produção

TERMO DE APROVAÇÃO

DESENVOLVIMENTO DE UM ALGORITMO GENÉTICO PARA RESOLUÇÃO DO PROBLEMA DE SEQUENCIAMENTO DE PRODUÇÃO EM JOB SHOPS FLEXÍVEIS

POR

PAOLA MANTOANI RIGO

Esta Monografia foi apresentada às 16 horas do dia 26 de novembro de 2018 como requisito parcial para obtenção do título de bacharel em ENGENHARIA DE PRODUÇÃO, Universidade Tecnológica Federal do Paraná – Campus Londrina. O candidato foi arguido pela Banca Examinadora composta pelos professores relacionados abaixo. Após deliberação, a Banca Examinadora considerou o trabalho: **APROVADO**.

Prof. Dr. Rogerio Tondato (UTFPR)
Banca Examidadora

Prof. Dr. Wellington Donizeti Previero (UTFPR)
Banca Examidadora

Prof. Dr. Rafael Henrique Palma Lima (UTFPR)
Presidente da Banca Examinadora
Orientador

RESUMO

O sequenciamento de produção é uma atividade essencial ao processo produtivo, pois é responsável por indicar a sequência de pedidos a serem produzidos, e as máquinas às quais os mesmos são designados. Um dos tipos de problemas de sequenciamento mais comum é chamado problema do sequenciamento job shop flexível (FJSP), no qual existem m máquinas disponíveis para sequenciar n ordens de produção, em que o problema é dividido em duas partes: indicar a sequência de produção dos itens, e eleger qual máquina, dentre as disponíveis, será responsável por realizar cada uma das operações. Uma das formas de encontrar soluções para o problema, é utilizando uma metaheurística chamada Algoritmos Genéticos (AG), a qual se baseia no cruzamento genético dos indivíduos, para gerar filhos com modificações genéticas. Dessa forma, foi desenvolvido um algoritmo para solução do FJSP, utilizando os princípios de AG, testando sua eficácia com a solução de diversas instâncias encontradas na literatura. Após o teste do algoritmo, constatou-se que este é capaz de encontrar soluções competitivas em pouco tempo computacional, e para instâncias pequenas e pouco complexas, encontra soluções ótimas; porém, encontra resultados distantes do ótimo global para problemas mais complexos, indicando que necessita de algumas modificações para melhorar seu desempenho. Diante disso, foi feita uma modificação para melhoria das soluções no algoritmo, a qual apresentou melhorias satisfatórias nos testes das instâncias, e foram propostas diversas melhorias que podem ser aplicadas futuramente.

Palavras-chave: *flexible job shop problem*; FJSP; algoritmos genéticos; sequenciamento de produção.

ABSTRACT

Production scheduling is an essential activity in the production process, as it is responsible for indicating the sequence of orders to be produced, and the machines to which they will be assigned. One of the most common types of scheduling problems is called flexible job shop problem (FJSP), in which there are m machines available to sequence n production orders, where the problem is divided into two parts: indicate the production sequence of the items, and select which machine, among those available, will be responsible for performing each one of the operations. One of the ways to find solutions to this problem is to use a metaheuristic called Genetic Algorithms (GA), which is based on the genetic crossing of individuals, to generate children with genetic modifications. For this purpose, an algorithm for FJSP solution was developed, using GA principles, testing its effectiveness with the solution of several instances found in the literature. After testing the algorithm, it was found that it is able to find competitive solutions in a short computational time, and for small and no complex instances, it finds optimal solutions; however, it finds results far from the global optimum for more complex problems, indicating that it needs some modifications to improve its performance. Therefore, a modification was made to improve the solutions in the algorithm, which presented satisfactory improvements during the tests of the instances, and several improvements were proposed that can be applied in the future.

Keywords: *flexible job shop problem*; FJSP; genetic algorithms; production scheduling.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AG	Algoritmos Genéticos
EDD	*Earliest Due Date*
FIFO	*First in, First out*
FJSP	*Flexible Job Shop Problem*
FOPNR	*Fewest Number of Operations Remaining*
HGST	*Highest Setup Time*
MOPNR	*Most Operations Remaining*
OX1	*Order Crossover Operator*
PMP	Planejamento Mestre de Produção
SPRT	Menor tempo restante de processamento
SPT	Menor tempo de processamento
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1. INTRODUÇÃO

O sequenciamento de produção é uma das escolhas mais importantes a serem realizadas durante o processo produtivo, pois suas decisões definirão quais pedidos serão produzidos, em quais máquinas e a sequência dessa produção, influenciando no tempo e recursos utilizados, além de definir o momento em que o produto final será entregue ao cliente. Para auxiliar nessa decisão, existem métodos para sequenciamento de ordens de produção, que podem ser utilizados para modelar o problema existente e definir qual a melhor sequência de produção, respeitando as restrições existentes, dentro do objetivo proposto (PINEDO, 2016).

Para resolver o problema da melhor maneira possível, é necessário que se conheça suas características, para que se encontre o modelo ideal e se possa aplicar métodos para sua resolução, de acordo com as restrições. Uma das classes de problema de sequenciamento que vem ganhando atenção na literatura especializada é o *Flexible Job Shop Problem* (FJSP), devido a sua alta complexidade computacional. Tal problema consiste no sequenciamento de *Jobs* ou tarefas, desmembradas em suas diversas operações, em uma ou mais máquinas que podem realizar a mesma função; as operações de cada *job* devem respeitar as regras de precedência e cada uma deve ser realizada em uma máquina disponível, sem interrupções (CHEN *et al*., 1999).

Como esse problema necessita que seja definida a alocação das operações nas máquinas e seu sequenciamento, é de difícil resolução, por isso, requer métodos possivelmente menos complexos, porém mais rápidos, que alcançam boas soluções, chamados métodos heurísticos (GAREY; JOHNSON; SETHI, 1976). Por conta da necessidade de uma rápida resposta de solução em um ambiente fabril, além requerer certa variedade na solução, um dos métodos indicados para a solução do FJSP são os Algoritmos Genéticos (AG), que se baseiam na teoria evolutiva, com o auxílio de diversas técnicas para definir a melhor solução possível para o problema.

Dessa forma, o objetivo geral deste trabalho é desenvolver um algoritmo genético para solucionar o FJSP e testar seu desempenho utilizando instâncias de *benchmarking* encontradas na literatura. A partir disso, os objetivos específicos deste trabalho são:

- Realizar uma revisão da literatura a respeito do FJSP, buscando esclarecimentos sobre o problema e seus tipos de solução;
- Desenvolver um algoritmo baseado em AG para resolver o FJSP, com base em programação computacional;
- Testar instâncias encontradas na literatura, comparando os resultados obtidos com o algoritmo desenvolvido, e os resultados apresentados na literatura;

O presente estudo se justifica por conta da relevância do problema para o desenvolvimento científico e empresarial. No âmbito científico, estudos recentes, como Chaudhry e Khan (2015) demonstram que o interesse por encontrar algoritmos para resoluções mais adequadas e com melhores soluções para o problema FJSP tem crescido nos últimos anos. Nesse estudo, foram analisados 191 artigos, sendo que 93,72% apresentam técnicas de resolução do problema, fator que demonstra grande interesse científico, reforçado por o problema apresentar difícil resolução matemática. Esse fator está aliado à perspectiva empresarial do problema, pois diversas empresas possuem processos com mais de uma máquina disponível para realizar uma função, e precisam de uma forma de sequenciar rapidamente suas tarefas, para que possam continuar a produção. Dessa forma, as heurísticas desenvolvidas apresentam uma solução bastante satisfatória para as empresas, por se tratarem de algoritmos relativamente simples, com soluções boas e rápido processamento, indispensável para o ambiente empresarial.

Com relação ao método de pesquisa do trabalho, a mesma tem por característica ser quantitativa, ou seja, trabalha com dados numéricos e históricos, buscando simplificar situações complexas, para que a discussão sobre os resultados seja simplificada (GARCIA, 2006). Com relação ao método, modelagem e simulação foram escolhidos para serem utilizados, pois a modelagem é um método utilizado para solucionar problemas encontrados em processos, através de um modelo que descreve um processo empresarial, sendo estudado e implementado, para que o processo possa ser continuamente melhorado (DAMIJ, 2007).

Assim, o restante do trabalho será dividido da seguinte forma: no Capítulo 2 será apresentada uma revisão da literatura sobre o problema, suas características e possibilidades de solução. No Capítulo 3 será apresentado o algoritmo para a solução do problema. Os resultados obtidos a partir do algoritmo serão apresentados no Capítulo 4, e no Capítulo 5 será apresentada uma conclusão para o estudo.

2. REFERENCIAL TEÓRICO

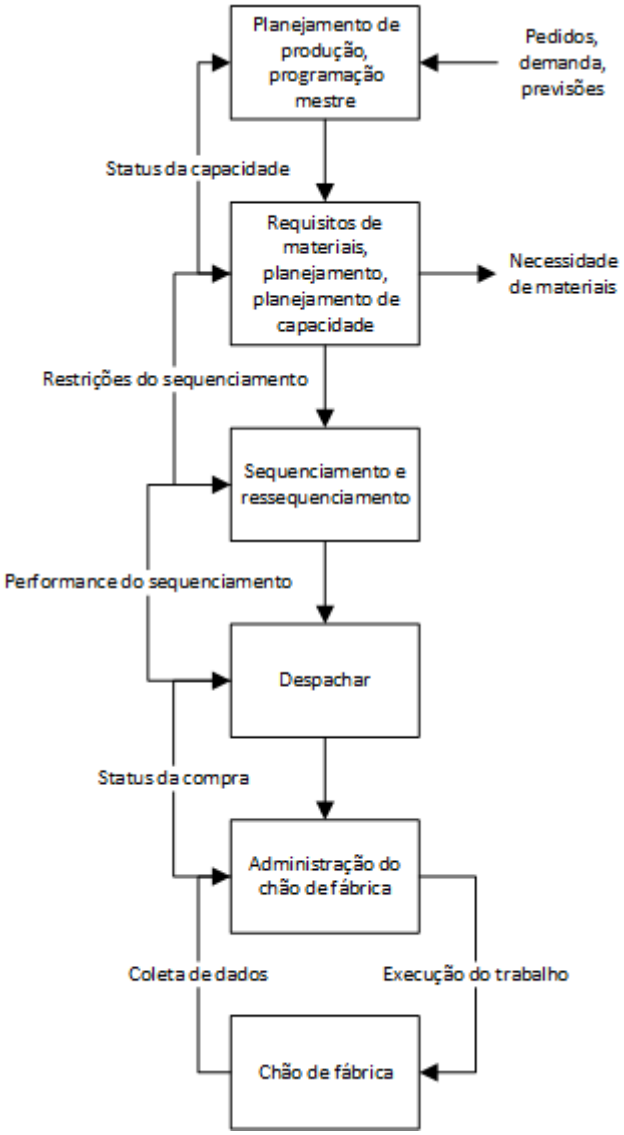
2.1 SEQUENCIAMENTO DA PRODUÇÃO

O sequenciamento da produção, segundo Pinedo (2016), é um processo decisório cotidiano em empresas e indústrias, que lida com a alocação de recursos a determinadas tarefas em um horizonte de tempo estabelecido. A principal meta ligada ao processo de sequenciamento é a otimização de objetivos, ou seja, maximizar ou minimizar certos critérios essenciais, tais como atraso total, tempo total de processamento das tarefas, ocupação total da máquina, entre outros (DUGARDIN *et al*, 2007).

Para que se possa realizar o sequenciamento, é necessário conhecer o tipo e a quantidade de recursos a serem alocados, além das tarefas às quais serão alocados, para que assim seja possível definir os limites do problema de sequenciamento. Os recursos podem ser máquinas em um ambiente de trabalho, sequências de voos em aeroportos, equipes de trabalho em construções, memória em um sistema computacional, entre outros. Já as tarefas podem ser as operações do processo produtivo, partidas e chegadas em aeroportos, execuções de programas em computadores, entre outros (LEUNG, 2004).

Em um sistema produtivo fabril, o sequenciamento se dá pela geração de ordens de produção que traduzem a tarefa que deve ser feita em determinado período de tempo, e a definição de uma sequência na qual as tarefas devem ser processadas, para que possam ser levadas ao chão de fábrica e a produção ocorra normalmente. O processo de sequenciamento envolve diferentes áreas relacionadas ao processo produtivo, e por isso demanda um sistema de interação eficiente, para que não haja perda de informação entre as fases, e não cause atrasos nos sequenciamentos criados. O fluxo de informações que passam por um processo de sequenciamento é descrito na Figura 1.

Figura 1: Fluxo de informações em um sistema de produção



Fonte: adaptado de Pinedo, 2016

A partir da Figura 1, pode-se observar que o processo de sequenciamento tem início na fase mais abrangente, chamada Planejamento Mestre de Produção (PMP), na qual são definidas as famílias de produto e suas quantidades a serem produzidas em um período geralmente longo, para que se possa atingir uma meta de faturamento e suprir a demanda pelo produto. Na sequência, é definida a necessidade de materiais (MRP), para que se possa atingir a meta de produção estabelecida no PMP. Nessa fase, é feita uma projeção do que será necessário, baseando-se na árvore de cada produto, e são geradas ordens de compra dos materiais necessários para a produção. Nesta fase também são definidas as capacidades produtivas, ou seja, o quanto será produzido em cada etapa do

processo em determinado tempo. Essas capacidades se tornam restrições para o problema de sequenciamento, pois definirão quais máquinas deverão ser utilizadas em determinada ordem, para produzir tudo o que for sequenciado (CORRÊA; CORRÊA, 2000).

Somente na etapa seguinte é realmente realizado sequenciamento de produção a partir dos materiais disponíveis ou da programação para recebimento, juntamente com o que deve ser produzido no período e a capacidade produtiva para tal. Assim, após determinadas as restrições, é possível desenvolver um sequenciamento que busque otimizar a relação entre quantidade produzida, restrições empregadas e prazo de entrega. Pode ocorrer a necessidade de um ressequenciamento, para se adequar a necessidades momentâneas ou imprevistos, fator que alterará a programação prevista, sendo necessária uma rápida resposta de um novo sequenciamento para que não haja parada de linha ou maiores atrasos na produção. Após o sequenciamento, as informações sobre o que deve ser produzido são enviadas para o chão de fábrica, para que o trabalho seja de fato executado, e o sequenciamento seja cumprido (CORRÊA; GIANESI; CAON, 2007).

Para que seja possível otimizar o sequenciamento utilizando as informações disponíveis, é utilizada uma abordagem quantitativa, com formulações matemáticas para traduzir as informações em uma função objetivo, a qual norteará as decisões a serem tomadas. A função objetivo deve abranger os objetos dependentes do sequenciamento, podendo ser o tempo requerido para realizar uma tarefa, o cumprimento do prazo estabelecido para entrega e a quantidade de trabalho empregada em um período determinado. Após a determinação da função objetivo e das restrições, é possível identificar o modelo de problema no qual este se encaixa, para que seja possível utilizar os melhores métodos para resolvê-lo (BAKER; TRIETSCH, 2009).

Para problemas de sequenciamento, segundo Abdolrazzagh-Nezhad e Abdullah (2017), os principais objetivos buscados são:

- Minimizar o tempo total de processamento (*Makespan*): diz respeito a minimizar o tempo de conclusão da última operação do sequenciamento a ser processada;
- Minimizar o tempo ponderado de processamento: diz respeito a minimizar também o tempo de processamento, porém, leva em conta

ponderações para o fluxo de trabalho, priorizando certas tarefas, se necessário;

- Minimizar o atraso máximo: leva em consideração a data em que cada tarefa deve ser entregue; caso uma tarefa não tenha sido ou esteja sendo processada no momento em que deveria ser entregue, é considerado atraso, o qual pode ser cumulativo, por isso deve ser minimizado;
- Minimizar o atraso máximo ponderado: também leva em conta o momento em que a tarefa deveria ser entregue e o momento que é concluída, porém, leva em conta tarefas com ponderações prévias;
- Minimizar o atraso médio: leva em consideração a média dos atrasos gerados pelo sequenciamento;
- Minimizar o número total de tarefas atrasadas: busca reduzir o número total de tarefas entregues após o prazo acordado.

Com relação à categorização dos problemas, estes podem ser categorizados por configuração dos recursos disponíveis e pela natureza das tarefas. Sendo assim, os problemas podem abranger uma única máquina - chamada *single machine* - ou múltiplas máquinas - chamada *multiple machine* -, e em ambos os casos as máquinas podem estar dispostas em grupos ou paralelas (máquinas que realizam a mesma função). Da mesma forma, os problemas podem ser estáticos, se não há mudança no sequenciamento no decorrer do tempo, ou dinâmicos, caso contrário. Finalmente, se as condições do problema são conhecidas, este é chamado determinístico, e quando há incertezas que necessitam de distribuições de probabilidade, o problema é chamado estocástico.

Por conta da complexidade do problema, muitas vezes não é possível encontrar uma solução ótima em curto espaço de tempo, fator necessário no ambiente empresarial, pois o sequenciamento é realizado em curto prazo, e tarefas precisam ser ressequenciadas quando necessário. Por isso, são empregadas soluções heurísticas, ou seja, técnicas que garantam soluções de boa qualidade em pouco tempo. Atualmente, pesquisas estão voltadas principalmente na utilização de metaheurísticas para a resolução desses problemas, pois com essas é possível obter soluções com qualidade, com menor tempo de processamento. Além disso,

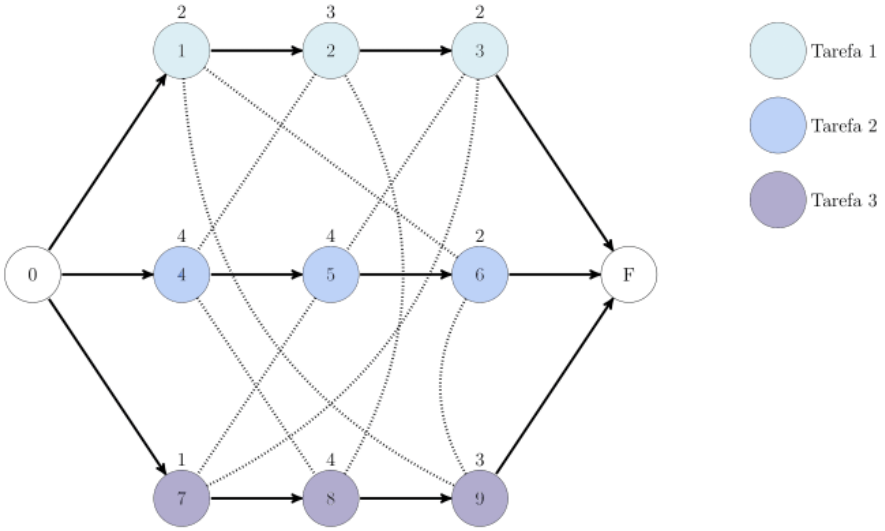
também é possível utilizar simulações para encontrar uma solução que se aplique ao problema, mesmo sendo uma técnica imperfeita (BAKER; TRIETSCH, 2009).

2.2 SEQUENCIAMENTO *JOB SHOP*

Uma produção com fluxo *job shop* é aquela em que existe uma quantidade de tarefas a serem realizadas em uma ordem específica, em que as operações e seu tempo de execução são determinadas pela máquina necessária para sua realização e o tempo de processamento na mesma, ou seja, cada uma das n operações são alocadas em m máquinas para serem sequenciadas. Outra característica do problema é que o fluxo de tarefas não é unidirecional, pois uma tarefa pode passar pela mesma máquina mais de uma vez, ou mesmo não passar por uma máquina. Entretanto, havendo n operações, cada uma delas deve ser designada a uma máquina, e o fluxo de entrada e saída das máquinas pode ser descrito como na Figura 2, onde não há uma máquina que realiza uma atividade final ou inicial, mas são todas partes do processo, que dependem do sequenciamento (BAKER; TRIETSCH, 2009).

Na Figura 2, existem o nó inicial (0), e o nó final (F), que dizem respeito ao momento em que, respectivamente, se inicia e finaliza a primeira operação a ser sequenciada e roteada, e, em cada um dos vértices, o número interno diz respeito ao índice da operação, e o externo ao tempo de processamento a ele associado.

Figura 2: Fluxo de tarefas em um sequenciamento *job shop*



Fonte: Previero, 2016

Assim, para esse tipo de problema existem algumas restrições, tais como (BŁAŻEWICZ; DOMSCHKE; PESCH, 1996):

- Não há restrições de precedência com relação às operações de diferentes tarefas;
- As operações não podem ser interrompidas, e cada máquina realiza somente uma tarefa por vez;
- Cada operação pode ser realizada em uma máquina por vez.

Para se iniciar a resolução de problemas de sequenciamento *job shop*, existem algumas regras gerais implementadas para que se possa rapidamente designar operações a máquinas disponíveis. Para diversas metaheurísticas, há a necessidade de se gerar uma solução inicial para o problema, para que possa ser melhorado; a partir disso, essas regras gerais geralmente são utilizadas também para definir esse sequenciamento de partida, ou uma solução inicial, pois assim a solução parte de um resultado normalmente melhor do que uma solução aleatória apresentaria. Dentre as regras mais utilizadas estão (GUPTA; GUPTA; BECTOR, 1989):

- Menor tempo de processamento (SPT): a tarefa com menor tempo de processamento iminente é selecionada;
- Menor tempo restante de processamento (SRPT): a tarefa com menor tempo restante a ser processado é selecionada;
- Menor data de entrega (*Earliest Due Date*) (EDD): a tarefa com data de entrega mais eminente é selecionada;
- Menor número de operações remanescente (*Fewest Number of Operations Remaining*) (FOPNR): a tarefa com menor número de operações remanescentes a serem processadas é selecionada;
- Maior número de operações remanescente (*Most Operations Remaining*) (MOPNR): a tarefa com maior número de operações remanescentes a serem processadas é selecionada;
- Maior tempo de setup (*Highest Setup Time*) (HGST): a tarefa cujo setup é mais demorado é selecionada;
- Primeiro que entra, primeiro que sai (*First in, First out*) (FIFO): a primeira tarefa recebida para ser sequenciada é processada;

2.3 *JOB SHOP* FLEXÍVEL

Existem casos em que é necessário uma maior adaptabilidade da planta produtiva, por isso, surge uma variação do modelo *job shop*, chamada *flexible job shop problem* (FJSP), em que a mesma tarefa pode ser realizada em diferentes máquinas, levando a uma maior flexibilidade no processo (CHEN *et al,* 2008). Assim, o problema FJSP pode ser decomposto em dois subproblemas: um deles consiste na alocação de cada operação a uma máquina dentro de todas as capazes de realizá-la, chamado problema de roteamento; já o problema de sequenciamento se trata de realmente sequenciar as operações designadas em todas as máquinas, buscando a melhor solução possível, capaz de minimizar a função objetivo predefinida. Por conta da dificuldade de solução do problema, já que trata de ambos o problema de roteamento e sequenciamento, este é chamado de *NP-hard*, classificação que se dá pela complexa combinação de problemas de otimização (ZHANG; GAO; SHI, 2011).

2.3.1 Formulação do problema

A partir das informações obtidas, o problema FJSP pode ser formulado, utilizando modelos de programação linear inteira mista, a qual leva em conta alguns parâmetros padrões para ser descrito. Existe um conjunto de N tarefas, dispostas em $J = \{J_1, J_2, \dots J_i, \dots, J_n\}$, e um conjunto de M máquinas tais que $M = \{M_1, M_2, \dots, M_k, \dots, M_m\}$. Para cada tarefa J_i há uma sequência pré-determinada de operações, que requer a seleção de uma das máquinas disponíveis e aptas a realizar a tarefa em questão; essa é a formulação do primeiro subproblema, o de roteamento. Já o segundo, o de sequenciamento, se baseia em determinar o tempo de início e término de cada atividade designada a cada máquina. Assim é definida a sequência de atividades em cada máquina específica, com os tempos definidos (ZHANG; GAO; SHI, 2011).

A notação utilizada para o problema, segundo Zhang, Gao e Shi (2011) e Dugardin *et al*. (2007), é a descrita a seguir. Na Tabela 1 são descritos os parâmetros do processo, enquanto na Tabela 2, as variáveis de decisão.

Tabela 1: Parâmetros de processo para solução de problemas FJSP

N	Quantidade total de *jobs*
M	Quantidade total de máquinas
O	Conjunto de todas as operações de todos os *jobs*
k, x	Índice do conjunto das máquinas alternativas
i	Índice do i-*ésimo job*
j	Índice da j-*ésima* tarefa do *job* J_i
x_{ik}	Variável binária que assume o valor 1 se a operação i é processada na máquina k e zero, caso contrário
y_{ij}	Variável binária que assume o valor 1 se a operação i precede j e ambas são processadas na mesma máquina
O_{ij}	J-*ésima* tarefa do *job* J_i
J_{io}	Quantidade total de tarefas do *job* J_i
X_{ij}	Conjunto de máquinas disponíveis para O_{ij}
L	Constante inteira suficientemente grande
P	Conjunto de pares ordenados correspondente à restrição de precedência entre duas operações

Tabela 2: Variáveis de decisão para solução de problemas FJSP

p_i	Tempo de processamento do *job* i
P_{ijk}	Tempo de processamento da tarefa O_{ij} na máquina k
S_{ijk}	Momento de início da tarefa O_{ij} na máquina k
E_{ijk}	Momento e término da tarefa O_{ij} na máquina k
O_k	Conjunto de operações que podem ser processadas na máquina k
E_k	Conjunto de pares ordenados distintos de O_k
T_i	Atraso do *job* i no sequenciamento parcial
d_i	Data de entrega do *job* i
C_{max}	Momento de conclusão da última tarefa (*makespan*)
$L = \sum_{i=1}^{N} J_{io}$	Soma de todas as tarefas de todos os *jobs*

Para o problema em questão, sua função objetivo consiste em minimizar o *makespan,* ou seja, minimizar o tempo de conclusão da última tarefa, e a formulação geral, utilizando a notação apresentada, pode ser representada por (BIRGIN *et al.*; 2014):

$$Minimizar \quad C_{max} \quad \forall_i \in O \quad (1)$$

$$Sujeito\ a \quad \sum_{k \in M} x_{ik} = 1 \quad \forall_i \in O \quad (2)$$

$$p_i = \sum_{k \in M} x_{ij}.p_{ij} \qquad \forall_i \in O \qquad (3)$$

$$C_{max} \geq s_i + p_i \qquad \forall_i \in O \qquad (4)$$

$$y_{ij} + y_{ij} \geq x_{ik} + x_{jk} - 1 \qquad \forall k \in M, \forall (i,j) \in E_k \qquad (5)$$

$$s_i + p_i - L.(1 - y_{ij}) \leq s_j \qquad \forall (i,j) \in E \qquad (6)$$

$$s_i + p_i \leq s_j \qquad \forall (i,j) \in P \qquad (7)$$

$$s_i \geq 0 \qquad \forall_i \in O \qquad (8)$$

$$x_{ik} \in \{0,1\} \qquad \forall k \in M, \forall (i,j) \in O_k \qquad (9)$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \qquad (10)$$

A Equação (1) diz respeito à função objetivo do problema, sendo, nesse caso, a minimização do *makespan*. Como dito anteriormente, existem outras funções objetivo possíveis, porém, esta é a mais utilizada. A restrição (2) garante que cada operação será executada em somente uma máquina, e a (3) determina o tempo de processamento de cada operação na máquina selecionada. A restrição (4) garante que o valor do *makespan* é maior ou igual ao maior tempo de término de uma operação. A restrição (5) define qual será a ordem de procedência de duas operações, caso ambas sejam atribuídas à mesma máquina, e a (6) define que estas operações não podem ser processadas ao mesmo tempo. Para garantir que a restrição (6) seja válida, L deve ser suficientemente grande. A restrição (7) assegura que a restrição de precedência das operações não seja violada. Já as restrições (8), (9) e (10) representam as variáveis do problema.

2.3.2 Abordagens para resolução do FJSP

O problema FJSP na área industrial vem sendo amplamente estudado por diversos autores, e, por ser um problema *NP-hard*, não pode ser resolvido em sua otimalidade em tempo polinomial. Por conta da dificuldade na resolução, é comum encontrar heurísticas e metaheurísticas utilizadas para resolver o problema, por trazerem sequenciamentos relativamente bons dentro de um tempo de processamento razoável, além de se aproximar muito de soluções ótimas. Dessa forma, é possível encontrar na literatura diversas soluções para o problema, utilizando métodos como Busca Tabu, *Simulated Annealing*, Algoritmos Genéticos, entre outros (ZHANG; GAO; SHI, 2011).

Pode-se então, encontrar na literatura, diversos autores que analisam o problema FJSP, e utilizam diversos métodos para resolvê-lo, tais como Xia e Wu (2005), que utilizam uma combinação de métodos heurísticos (busca local, busca global) para encontrar a solução ótima para o problema. Já Paulli (1995), assim como Hurink, Jurisch e Thole (1994) e Dauzère-Pérès (1997) optaram por utilizar a busca tabu para a solução do problema. Já Chen *et al.* (1999) utiliza algoritmo genético para encontrar a melhor solução para o problema FJSP, com o intuito de minimizar o *makespan*. O autor divide a solução do problema em duas partes, sendo a primeira parte de roteamento e a segunda de sequenciamento. Chan, Wong e Chan (2006) trazem também a solução do problema sob a perspectiva do algoritmo genético, iterativamente, passando pelas duas fases do problema, considerando a disponibilidade de recursos. Além destes, outros autores optaram por utilizar o algoritmo genético para resolver o problema, como trabalhos extremamente relevantes, tais como Jia *et al.* (2003), Ho e Tay (2004), Kacem, Hammadi e Borne (2002) e Pezzela, Morganti e Ciaschetti (2007). Existem ainda alguns autores que optam pela utilização de métodos exatos para a resolução do problema, como Fattahi *et al.* (2007) e Özgüven, Yavuz e Özbakir (2012), os quais são extremamente relevantes para determinar qual seria a solução ótima de cada problema analisado, mas, devido ao seu tempo de processamento, são, em sua maioria, inviáveis de serem aplicados em ambientes fabris.

Com relação ao objetivo do problema, Yazdani, Amiri e Zandieh (2010) focam na minimização do *makespan*, assim como Zhang, Gao e Shi (2011) e Chen *et al*., (1999). Já a minimização do atraso total é tratada por Loukil *et al*., (2007), Scrich *et al*., (2004) e Alvarez-Valdez *et al*., (2005), assim como Dugardin *et al.* (2007).

Em um estudo recente realizado por Chaudhry e Khan (2016) sobre o FJSP, algumas informações foram levantadas com relação a cada um dos pontos do problema, baseando-se na seleção de 191 artigos publicados entre janeiro de 1990 a fevereiro de 2014. Segundo os autores, com relação à função objetivo, em 44,67% dos artigos, somente o *makespan* é utilizado, enquanto em 39,59%, o mesmo é utilizado em conjunto com outras funções objetivo. O foco no atraso total é apresentado em 1,52%, enquanto o atraso médio aparece em 1,02%. O restante dos artigos apresenta combinações entre dois ou mais objetivos.

Ainda segundo os autores, na Tabela 3 são mostradas as porcentagens de cada técnica utilizada para a resolução do problema.

Tabela 3: Técnicas utilizadas para solução de problemas FJSP

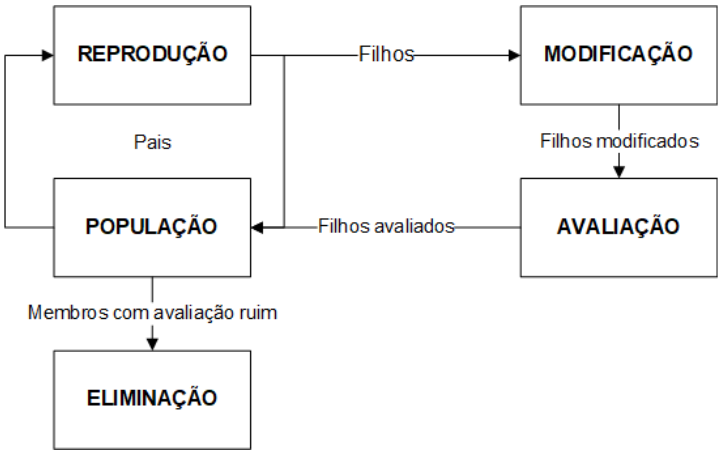
Técnica Utilizada	Número de Artigos	Porcentagem (%)
Técnicas Híbridas	69	35,03
Algoritmos Evolutivos (Genéticos)	47	23,86
Heurísticas Determinísticas	19	9,64
Busca Tabu	12	6,09
Programação Linear E Inteira	10	5,08
Nuvem De Partículas	8	4,06
Miscelânea De Técnicas	7	3,55
Busca Em Vizinhança	6	3,05
Sistemas Imunológicos Artificiais	5	2,54
Programação Matemática	4	2,03
Recozimento Simulado	4	2,03
Colônia Formigas	3	1,52
Greedy Randomized Adaptative Search Procedure (GRASP)	2	1,02
Colônia De Abelhas	1	0,51

Fonte: Chaudhry e Khan (2015)

2.4 ALGORITMOS GENÉTICOS

O Algoritmo Genético (AG) é uma metaheurística que se baseia no paradigma da evolução genética. A partir de uma solução inicial, ou seja, uma população inicial, o algoritmo explora a vizinhança da solução, utilizando operadores genéticos para produzir soluções “filhas”, as quais se presumem melhores que as soluções antecessoras, e podem ser modificadas para atender restrições. A cada geração (iteração) cada novo cromossomo corresponde a uma nova solução, a qual é avaliada para decidir se é ou não aceita, e caso seja melhor que uma solução anterior, substitui o cromossomo com a pior solução presente na população (CHAUDHRY; KHAN; KHAN, 2013). O ciclo de resolução de problemas utilizando AG é descrito na Figura 3.

Figura 3: Fluxo de um algoritmo genético genérico



Fonte: adaptado de Chaudhry, Khan e Khan, 2013

Para determinar a solução inicial, deve-se focar em dois aspectos: o tamanho da população e o método a ser utilizado para escolher os indivíduos. Segundo Revees e Rowe (2002), uma população com, no mínimo 30 cromossomos é bastante adequada na maioria dos casos, mas é adequado que a população aumente linearmente com o aumento do tamanho do vetor cromossomo. Já com relação à escolha dos indivíduos, muitas vezes considera-se uma população com soluções totalmente aleatórias, entretanto, pode-se não cobrir todas as possibilidades de solução uniformemente. Por isso, considera-se a utilização de heurísticas ou outros métodos conhecidos para definir soluções iniciais com qualidade mais alta, pois estas podem levar a melhores soluções mais rapidamente. Porém, apesar disso, é possível que isso limite a área de análise, e conduza a solução prematuramente para uma solução de baixa qualidade. Por isso, autores como Kacem, Hammadi e Borne (2002) optam por utilizar populações iniciais mistas, com parte formada por soluções aleatórias, e parte formada por soluções de derivam de outras heurísticas, para heterogeneizar a solução.

Para que haja a reprodução genética da solução, ou seja, as soluções iniciais gerem novas soluções, há uma troca de informações genéticas entre os cromossomos, realizada por *crossover*, com a possibilidade de utilizar operadores de mutação, quando se julga necessário, para manter a diversidade da solução. O objetivo dos operadores *crossover* é conseguir melhores soluções ao intercambiar os melhores genes de duas soluções “pais”, encontrando melhores soluções locais. Já os operadores de mutação buscam melhores soluções globais, alterando randomicamente um ou mais genes de uma solução, buscando garantir a

diversidade genética da solução. É definido um percentual de soluções da população para sofrer mutações de cada operador, buscando garantir que sejam avaliadas não somente soluções vizinhas às já existentes, mas também soluções longe da vizinhança da solução inicial, aumentando a chance de se encontrar solução ótimas globais (REVEES; ROWE, 2002) (GONG *et al.,* 2018).

Os *crossovers* mais utilizados na literatura são (REVEES; ROWE, 2002):

- *Crossover* de um ponto: é escolhido um ponto aleatório de cruzamento no cromossomo, e a partir deste as informações genéticas são trocadas entre os pais;
- *Crossover* de dois pontos: dois pontos aleatórios são escolhidos, e ocorre o cruzamento genético entre esses pontos;
- *Crossover* uniforme: através de um parâmetro global previamente definido, determina a probabilidade de cada uma das variáveis ser trocada entre os pais;
- *Crossover* parcialmente combinado: dois pontos de cruzamento são escolhidos dentro do vetor, e a seção entre os pontos determina o local de troca de genes;

Para se aplicar a mutação, deve-se determinar uma taxa de mutação, que representa a probabilidade de mutação de um cromossomo. A taxa deve ser suficiente para garantir a diversidade da população, porém não tão alta a ponto de eliminar ótimas soluções locais. Através da taxa, define-se um gene que sofrerá mutação, e aleatoriamente o mesmo tem seu valor alterando, levando a soluções fora dos ótimos locais (SIVANANDAM; DEEPA, 2007).

Após a geração de soluções filhas, a partir de *crossover* e mutação, os cromossomos gerados são avaliados, para determinar seu valor de aptidão com relação ao objetivo do problema. Após a determinação do valor para cada cromossomo, é necessário selecionar as soluções que irão compor a nova população. O objetivo da seleção é direcionar a busca por melhores cromossomos para regiões promissoras no espaço de busca. Para isso, foram criados métodos de seleção, buscando manter a diversidade necessária, e não limitando a população filha. Alguns métodos de seleção mais comuns são descritos a seguir (GEN; CHENG, 2000):

- Método da Roleta: o método se baseia na adequação do cromossomo à população e sua probabilidade de sobrevivência, ou seja, cada cromossomo é testado e sua probabilidade de sobrevivência com relação a sua aptidão, e os mais aptos possuem um peso maior para que sejam selecionados; a partir da definição das probabilidades e pesos, um número aleatório é sorteado, e o valor sorteado corresponde à posição do cromossomo que será selecionado;
- Método Determinístico (Elitista): os melhores cromossomos com relação a sua aptidão são selecionados, buscando garantir que os melhores cromossomos permaneçam na população; geralmente utilizado em conjunto com outros métodos, para permitir a diversidade da população;
- Método do Torneio: neste método, são formados m subgrupos de n cromossomos escolhidos aleatoriamente, e dentro dos subgrupos os melhores cromossomos são escolhidos, baseando-se em sua aptidão; o número de cromossomos a serem escolhidos em cada subgrupo é determinado previamente;
- Técnicas de compartilhamento: utilizadas para manter a diversidade da população, através da determinação da degradação da aptidão de um cromossomo com relação a de um vizinho; de acordo com a degradação de sua aptidão, a probabilidade de reprodução de um cromossomo na população é contida, enquanto outros são encorajados a se reproduzir.

Após a escolha dos cromossomos, é feita uma análise para determinar se algum ajuste nas novas soluções é necessário para que estas estejam dentro das restrições do problema. Após os ajustes, a nova população é definida, e os cromossomos não selecionados são eliminados. O ciclo é refeito até atingir o critério de parada definido previamente (REVEES; ROWE, 2002).

2.5 ALGORITMOS GENÉTICOS PARA O PROBLEMA FJSP

Para a solução de problemas de FJSP, o AG se mostra uma ferramenta extremamente útil, pois cada cromossomo carrega consigo o sequenciamento das tarefas a serem realizadas, cujo tempo total de processamento pode ser avaliado,

demonstrando sua aptidão, para testar se a nova solução é melhor que seus predecessores. Além disso, mais estratégias podem ser utilizadas para encontrar novos indivíduos para serem incluídos na população existente, incrementando a força do AG (DE GIOVANNI; PEZZELLA, 2010).

A estrutura geral do AG aplicado ao problema de FJSP é descrita por (PEZZELA; MORGANTI; CIASCHETTI, 2007):

- Codificação: cada gene do cromossomo representa a designação das operações às máquinas disponíveis, e a ordem dos genes representa o sequenciamento das operações;
- População inicial: através de uma escolha de método para geração de soluções, diversas são geradas, gerando uma população inicial a ser avaliada; podem ser utilizadas diversas regras de designação e sequenciamento inicialmente, para criar uma população heterogênea, tais como escolha aleatória, SRPT, FOPNR, entre outros;
- Avaliação da solução (fitness): é realizado o cálculo da função objetivo para cada um dos cromossomos da solução;
- Seleção: os melhores cromossomos são selecionados a cada iteração, de acordo com os métodos estabelecidos, utilizando métodos como método do Torneio, método da Roleta e método determinístico;
- Geração de descendentes: é realizado um cruzamento entre as soluções da população, para gerar cromossomos filhos, até que um número máximo de soluções previamente definido é atingido;
- Critério de parada: ao atingir o número máximo de gerações de soluções, é interrompida a geração de novas soluções, e as novas soluções são avaliadas.

Como mostrado na Tabela 3, AG é a técnica mais utilizada sozinha para resolver problemas FJSP, por conta de sua versatilidade ao permitir que diversas estratégias sejam empregadas para encontrar bons indivíduos para serem integrados à população. Além disso, por abranger buscas por soluções ótimas globais e locais, o algoritmo garante uma maior diversidade nas soluções, permitindo uma maior probabilidade de encontrar um ótimo global, juntamente com a vantagem de ter um curto tempo de processamento, fator necessário para o ambiente empresarial em que esse tipo de problema é explorado (ZHANG; GAO; SHI, 2011).

3. PROPOSIÇÃO DO ALGORITMO

Conforme descrito no Capítulo 2, sobre a relevância dos AG para a resolução do FJSP, essa metaheurística foi definida para modelar o problema, e a estrutura proposta para o AG pode ser descrita por:

- Representação da solução (codificação): a forma pela qual os genes vão representar a designação das máquinas e sequenciamento das tarefas;
- Avaliação da solução (*fitness*): o *makespan* de cada solução é calculado, para avaliar seu resultado;
- População inicial: seleção aleatória do sequenciamento das tarefas, e designação da máquina correspondente através da escolha daquela com menor tempo acumulado até então;
- Seleção: a cada iteração, utilizar o método do Torneio juntamente com elitismo, para identificar os melhores cromossomos;
- Geração de descendentes: utilização de um operador de *crossover* seguido de um operador de mutação das soluções, para garantir a heterogeneidade da solução; os novos indivíduos serão gerados até atingir o critério de quantidade máxima de soluções geradas;
- Critério de parada: o algoritmo é executado por um tempo pré-determinado pelo interlocutor; se este é atingido, o algoritmo apresenta a melhor solução até o momento como a final, caso contrário, recomeça a avaliação de soluções.

Dessa forma, os capítulos seguintes descrevem detalhadamente cada um dos passos do método de pesquisa.

3.1 REPRESENTAÇÃO DA SOLUÇÃO (CODIFICAÇÃO)

Para que a representação da solução seja completa, esta deve especificar a sequência de operações a serem realizadas, segundo sua ordem de prioridade, e a alocação de cada uma das tarefas em uma das máquinas disponíveis. Por conta da menor necessidade de decodificação, e menor complexidade computacional, a representação da solução deste trabalho se baseou na sistemática de representação desenvolvida por Zhang, Gao e Shi (2011), a qual utiliza somente um vetor com as informações, para representar a solução. Entretanto, por conta da representação

visual da solução não ser tão simples, e necessitar de tratamento para o interlocutor, decidiu-se por utilizar dois vetores para a representação, um contendo o sequenciamento das atividades, e um contendo a máquina responsável por realiza-la, baseando-se no estudo de Li, Pan e Liang (2010).

Dessa forma, a representação da solução deste estudo divide-se em dois vetores com uma única informação em cada um dos genes dos mesmos, sendo divididos em: um vetor com a informação sobre o sequenciamento de cada tarefa de cada *job*, e outro vetor, dependente do primeiro, contendo a informação sobre as máquinas utilizadas para realizá-las. Partindo desse princípio, o tamanho do vetor deve ser igual à quantidade de operações que precisam ser sequenciadas, e ambos os vetores possuem o mesmo comprimento, já que são dependentes. Cada item será explicado com detalhes a seguir.

3.1.1 Sequenciamento das operações

Considere um problema constituído por 3 *jobs*, com 2 tarefas cada, e duas máquinas disponíveis, como apresentado na Tabela 4, e uma possível solução apresentada na Figura 4.

Tabela 4: Exemplo de um problema FJSP

Job (J_i)	**Operação**(O_{ij})	**Tempo de realização da Operação na Máquina 1** (M_1)	**Tempo de realização da Operação na Máquina 2** (M_2)
1	1	43	-
1	2	87	95
2	1	63	53
2	2	-	73
3	1	125	135
3	2	43	61

Figura 4: Representação da solução

Job	1	3	1	2	2	3
Máquina	1	1	2	1	1	2

Fonte: Do Autor, 2018

Como existem 6 operações a serem sequenciadas, esse também deve ser o comprimento total do vetor. Para iniciar o sequenciamento, a primeira operação de um dos *Jobs* deve ser selecionada, respeitando a restrição de precedência. No caso

da Figura 4, a primeira operação a ser sequenciada será a O_{11}, representada no vetor pelo índice do *job* correspondente, ou seja, o J_1. A próxima operação a ser sequenciada deverá ser a segunda do J_1, ou a primeira dos *Jobs* J_2 ou J_3.

Dessa forma, a representação do primeiro vetor traz apenas o índice do *job*, e o índice da operação será sempre crescente, partindo da primeira operação correspondente ao *job* i, passando por todas as operações, até sequenciar a última operação existente desse *job*, respeitando a ordem de precedência. Para o exemplo da Figura 4, o sequenciamento das operações será:

$$S = (O_{11}; O_{31}; O_{12}; O_{21}; O_{22}; O_{32})$$

3.1.2 Designação das máquinas

O segundo vetor diz respeito à designação das máquinas para realizar a operação sequenciada. Como apresentado no exemplo da Tabela 4, uma operação pode ser realizada por uma ou mais máquinas, mas pode ter tempos de processamento diferentes em cada uma delas. Para a representação da solução, utiliza-se o índice da máquina disponível, ou seja, para a operação O_{21}, por exemplo, a máquina 1 tem o índice 1, e a máquina 2 tem o índice 2. Já a operação O_{22}, que possui somente uma máquina, a máquina 2, tem seu índice como 1, pois é a primeira máquina disponível para realizar a operação.

Para que seja possível determinar a máquina a ser utilizada, primeiramente é necessário sequenciar a operação, para que seja possível identificar as máquinas disponíveis. A partir do sequenciamento, levanta-se a quantidade de máquinas disponíveis para cada operação e aleatoriamente determina-se o índice da máquina que a processará.

Assim sendo, para a solução apresentada na Figura 4, a sequência de máquinas relacionadas às operações será:

$$S = (O_{11}, M_1)\ ; (O_{31}, M_1); (O_{12}, M_2); (O_{21}, M_1); (O_{22}, M_2); (O_{32}, M_2)$$

Ou seja, a interpretação final da solução será, na ordem de sequenciamento: a operação O_{11} será realizada pela primeira máquina disponível, a Máquina 1; a operação O_{31} será realizada pela primeira máquina disponível, a Máquina 1; a operação O_{12} será realizada pela segunda máquina disponível, a Máquina 2; a operação O_{21} será realizada pela primeira máquina disponível, a Máquina 1; a

operação O_{22} será realizada pela primeira máquina disponível, a Máquina 2; e a operação O_{32} será realizada pela segunda máquina disponível, a Máquina 2.

3.2 AVALIAÇÃO DA SOLUÇÃO (*FITNESS*)

A função objetivo escolhida para este estudo foi a de minimização do *makespan;* dessa forma, para avaliar as soluções encontradas, é necessário calcular o *makespan* de cada uma delas. Para o cálculo, são somados os tempos de conclusão de cada uma das operações, e o momento de conclusão da última operação do último *job* a ser processada é considerado o *makespan*, ou, caso a última operação não tenha o maior tempo de processamento, considera-se o maior tempo dentre todas as operações executadas. Como a função objetivo foca na minimização deste, as soluções com menor *makespan* são consideradas mais adequadas para o problema, ou seja, são as melhores soluções.

3.3 POPULAÇÃO INICIAL

Para obter uma população heterogênea, mas que aumente a probabilidade de encontrar uma melhor solução mais rapidamente, optou-se por realizar uma definição inicial aleatória do sequenciamento das operações, e, posteriormente a seleção de máquinas de acordo com o menor acumulado em cada máquina até o momento, utilizando como base a geração de população inicial definida por Zhang, Gao e Shi (2011).

Para dar início à geração de soluções, é gerado um vetor contendo todas as operações de todos os *Jobs*, o qual é aleatorizado para garantir a heterogeneidade da solução, poia, dessa forma, cada indivíduo gerado possui uma sequência diferente de operações, e, portanto, um tempo de processamento final diferente.

Após a definição do vetor sequenciamento, começa-se a alocar as operações às máquinas. Como no começo não há acumulado de tempo de processamento em máquinas, a primeira operação alocada o será sempre na máquina com menor tempo de processamento, para que se possa começar a contar o tempo de processamento acumulado em cada máquina. A partir disso, a cada iteração, analisa-se todas as máquinas disponíveis para alocar a operação, e a mesma é alocada temporariamente em todas essas máquinas, para que o tempo de processamento total até o momento possa ser analisado. A máquina que apresentar menor tempo de processamento após a alocação, dentre as disponíveis na iteração,

será aquela em que a operação será alocada, independentemente do seu tempo de processamento individual.

Dessa forma, é possível gerar uma população heterogênea, com uma maior chance de obter melhores soluções locais, pois a formação do vetor sequenciamento é aleatória, mas a escolha das máquinas segue o princípio de menor tempo de processamento, fator que contribui para a busca de um menor tempo de processamento final, ou seja, um menor *makespan*.

O tamanho da população é um parâmetro do algoritmo e será determinado com base no tamanho do problema, e ao determiná-la para cada problema, deve-se atentar, pois o tamanho da mesma influencia diretamente no desempenho e eficiência da resolução do mesmo. Caso a população seja muito pequena, pode significar uma diminuição no desempenho do problema, pois o espaço de busca é menor. Já uma população grande previne o problema destacado, além de prevenir a convergência a uma solução local precoce; entretanto, um problema com uma grande população inicial demanda um maior tempo de processamento e recursos computacionais para resolvê-lo (SCOFIELD, 2002).

3.4 SELEÇÃO

Para a seleção dos indivíduos que passarão pela reprodução, serão utilizados dois métodos distintos combinados: método do Torneio e método Determinístico. A combinação dos métodos foi escolhida para manter a melhor solução global do problema, para que não se perca entre as iterações, mas garantindo a heterogeneidade do problema, por conta da aleatoriedade utilizada no método do Torneio.

Começando com o método Elitismo Determinístico, primeiramente avalia-se todas as soluções presentes na população com relação a sua aptidão de solução, e os vetores são ranqueados, começando pelo menor *makespan*, até o maior, para que se possa analisar quais são as soluções mais aptas da iteração. Para populações de até 100 indivíduos, dois serão selecionados por esse método; para populações maiores, o número de escolhido é quatro, para preservar os melhores de cada iteração. Após a seleção desses cromossomos, os mesmos são excluídos da população atual, e inseridos sem modificações na população da próxima iteração, para que não sejam escolhidos para passar pela mudança genética.

Posteriormente, com os indivíduos restantes da população, começa-se a seleção pelo método do Torneio. O tamanho (n) dos subgrupos é de dois cromossomos, os quais são eleitos aleatoriamente para análise de aptidão. Dentre ambos, o com menor *makespan* é selecionado, para que passe pela modificação genética.

Os subgrupos aleatórios são analisados até que seja selecionada a quantidade predefinida de cromossomos para passarem pela reprodução genética. A quantidade é definida de acordo com o tamanho do problema, considerando limitação de soluções, para poucos cromossomos, e tempo de processamento, para maior quantidade de cromossomos.

3.5 GERAÇÃO DE DESCENDENTES

Para a geração de descendentes, deve ser definida antes da programação, a proporção de soluções filhas que passarão por *crossover*, fator necessário para manter a heterogeneidade da população filha. Para a reprodução de cromossomos, foi escolhido o operador de *crossover* permutacional chamado *Order Crossover Operator* (OX1). Para a implementação desse operador, foi usado como base o descrito pelos autores Kumar, Gopal e Kumar (2013) em sua pesquisa.

Para executar esse operador, deve-se primeiramente eleger dois cromossomos da população para se reproduzir. A partir disso, aleatoriamente é definido um intervalo de posições nos pais, cujo tamanho é definido previamente pelo interlocutor, em função de porcentagem, ou seja, o tamanho do *crossover* será a porcentagem correspondente ao tamanho total do indivíduo. Por exemplo, se o indivíduo tiver vinte operações totais a serem sequenciadas, se o interlocutor definir um tamanho de *crossover* de 10%, dois genes passarão pelo procedimento.

Definido o intervalo nos pais 1 e 2, os valores correspondentes são passados para os filhos de índice invertido, na mesma posição, ou seja, o intervalo no pai 1 passará para o filho 2, e vice-versa. Depois disso, deve-se avaliar a troca genética em cada um dos filhos separadamente. Para cada um dos filhos, avalia-se o outro pai em busca dos genes que ainda não estão presentes no filho; começando pela esquerda do cromossomo, reproduzindo-os no filho na ordem em que aparecem no pai, de forma a garantir que todas as operações presentes nos pais estejam passadas para o filho, sem duplicações ou faltas. O operador OX1 é exemplificado na Figura 5.

Figura 5: Exemplo de *Order Crossover Operator* (OX1)

Pai 1	8	4	7	3	6	2	5	1	9	0
Pai 2	0	1	2	3	4	5	6	7	8	9
Filho 2	***0***	***4***	***7***	3	6	2	5	1	***8***	***9***
Pai 1	8	4	7	3	6	2	5	1	9	0
Pai 2	0	1	2	3	4	5	6	7	8	9
Filho 1	***8***	***2***	***1***	3	4	5	6	7	***9***	***0***

Fonte: Do Autor, 2018

Dessa forma, os indivíduos mais aptos de cada torneio serão colocados em dupla para serem os pais que passarão pela troca genética, seguindo os procedimentos descritos acima.

3.6 OPERADOR DE MUTAÇÃO

Para a realização da mutação na iteração, previamente é definida, pelo interlocutor, a porcentagem de indivíduos da população que passará pelo procedimento, ou seja, caso seja definida uma mutação de 20%, em uma população de 100 indivíduos, 20 deles passarão por mutação a cada iteração.

A escolha dos indivíduos é aleatória, dentre os novos indivíduos da população, gerados após o *crossover*. Após escolhido, aleatoriamente é selecionada uma posição do indivíduo, cuja máquina será analisada. Caso haja mais de uma máquina disponível para realizar a operação selecionada, é sorteada uma máquina dentre as disponíveis, para substituir a atual. Dessa forma, há uma alteração no tempo de processamento do indivíduo, pois passou pela mutação genética em seu roteamento.

4. RESULTADOS E DISCUSSÃO

4.1 RESULTADOS OBTIDOS

O algoritmo proposto no Capítulo 3 deste trabalho foi programado segundo as diretrizes apresentadas, utilizando a linguagem Visual Basic for Applications (VBA), utilizando um processador Intel® Core™ i7, com 8,00 GB de memória RAM, e testado através de diversas instâncias de problemas da literatura (MASTROLILLI, 2018), para validar seu funcionamento.

Para que o problema pudesse ser executado, alguns parâmetros deveriam ser previamente definidos – tamanho da população, tempo de processamento, porcentagem de *crossover* e porcentagem de mutação –, como já citado anteriormente. Dessa forma, foram testadas diversas combinações, considerando tamanho do problema, complexidade, tempo de processamento, entre outros, até que se chegasse em uma que fosse satisfatória para a resolução do problema. Assim, os parâmetros finais definidos foram:

- Tamanho da população: 100 indivíduos para problemas com até 30 *jobs*, 200 indivíduos para problemas com até 60 *jobs*, 300 indivíduos para problemas com até 90 *jobs*, e assim sucessivamente;
- Tempo de processamento: como o critério de parada do problema foi definido por tempo, o tempo estabelecido para cada problema foi de dez minutos, visando permitir diversas iterações, para melhoria das soluções, porém levando em consideração um ambiente empresarial, em que não se dispõe de muito tempo para chegar a uma solução;
- Porcentagem de *crossover* e porcentagem de mutação: definidos a partir de quatro configurações levantadas com base no tamanho do problema e extensão de modificação das soluções; as configurações estão dispostas na Tabela 5.

Tabela 5: Configurações para operadores de *crossover* e mutação

Configurações	**1**	**2**	**3**	**4**
Crossover	5%	5%	10%	10%
Mutação	10%	20%	10%	20%

A partir dessas suposições, foram utilizadas as instâncias do autor Fattahi *et al.* (2007), que apresenta vinte pequenas e médias instâncias geradas

randomicamente, divididas entre dez pequenas e dez médias, para avaliar qual das quatro combinações de configuração geraria os melhores resultados. Os resultados para cada uma das configurações em cada uma das vinte instâncias são apresentados na Tabela 6, e um resumo dos melhores resultados por configuração é mostrado na Tabela 7. A última coluna da tabela representa um indicador binário que indica qual configuração utilizada foi a melhor dentre as 4.

Tabela 6: Comparação dos resultados obtidos com cada configuração com as instâncias Fattahi et al. (2007)

Código Instância	Melhor *Makespan* Conhecido	Resultado Obtido	*Gap*	Quantidade iterações executadas	Tempo de execução (segundos)	Tempo por iteração (segundos)	Configuração	Melhor Configuração
SFJS1	66	66	0,00%	2471	601	0,24	1	1
	66	66	0,00%	2485	601	0,24	2	1
	66	66	0,00%	2365	601	0,25	3	1
	66	66	0,00%	2362	601	0,25	4	1
SFJS2	107	107	0,00%	2518	601	0,24	1	1
	107	107	0,00%	2495	601	0,24	2	1
	107	107	0,00%	2501	601	0,24	3	1
	107	107	0,00%	2456	601	0,24	4	1
SFJS3	221	221	0,00%	3956	601	0,15	1	1
	221	221	0,00%	4001	601	0,15	2	1
	221	221	0,00%	4410	601	0,14	3	1
	221	221	0,00%	3600	601	0,17	4	1
SFJS4	355	355	0,00%	2569	601	0,23	1	1
	355	355	0,00%	4859	601	0,12	2	1
	355	355	0,00%	3698	601	0,16	3	1
	355	355	0,00%	4582	601	0,13	4	1
SFJS5	119	119	0,00%	4587	601	0,13	1	1
	119	119	0,00%	5214	601	0,12	2	1
	119	119	0,00%	4178	601	0,14	3	1
	119	119	0,00%	6028	601	0,10	4	1
SFJS6	320	347	8,44%	1540	601	0,39	1	0
	320	337	5,31%	1531	601	0,39	2	0
	320	330	3,13%	3359	601	0,18	3	0
	320	327	2,19%	978	601	0,61	4	1
SFJS7	397	407	2,52%	4743	601	0,13	1	1
	397	407	2,52%	4693	601	0,13	2	1
	397	407	2,52%	978	602	0,62	3	1
	397	407	2,52%	958	601	0,63	4	1
SFJS8	253	253	0,00%	1606	601	0,37	1	1
	253	253	0,00%	1509	601	0,40	2	1
	253	264	4,35%	3374	601	0,18	3	0

	253	256	1,19%	898	602	0,67	4	0
SFJS9	210	230	9,52%	1505	601	0,40	1	0
	210	227	8,10%	1480	601	0,41	2	1
	210	227	8,10%	907	601	0,66	3	1
	210	237	12,86%	927	602	0,65	4	0
SFJS10	516	650	25,97%	722	602	0,83	1	0
	516	540	4,65%	722	603	0,84	2	0
	516	643	24,61%	369	603	1,63	3	0
	516	524	1,55%	421	602	1,43	4	1
MFJS1	469	539	14,93%	83	612	7,37	1	0
	469	502	7,04%	296	605	2,04	2	0
	469	539	14,93%	314	605	1,93	3	0
	469	482	2,77%	2252	601	0,27	4	1
MFJS2	482	506	4,98%	2244	601	0,27	1	1
	482	550	14,11%	615	602	0,98	2	0
	482	545	13,07%	613	602	0,98	3	0
	482	511	6,02%	608	602	0,99	4	0
MFJS3	553	583	5,42%	506	603	1,19	1	0
	553	594	7,41%	515	602	1,17	2	0
	553	601	8,68%	492	603	1,23	3	0
	553	570	3,07%	480	603	1,26	4	1
MFJS4	634	731	15,30%	410	603	1,47	1	0
	634	653	3,00%	1667	601	0,36	2	1
	634	690	8,83%	403	603	1,50	3	0
	634	670	5,68%	393	603	1,53	4	0
MFJS5	625	655	4,80%	428	602	1,41	1	0
	625	698	11,68%	418	603	1,44	2	0
	625	698	11,68%	412	603	1,46	3	0
	625	628	0,48%	399	603	1,51	4	1
MFJS6	762	790	3,67%	394	604	1,53	1	0
	762	809	6,17%	401	603	1,50	2	0
	762	857	12,47%	407	602	1,48	3	0
	762	773	1,44%	1488	601	0,40	4	1
MFJS7	964	1298	34,65%	2927	601	0,21	1	0
	964	1032	7,05%	1279	602	0,47	2	1
	964	1067	10,68%	1167	601	0,51	3	0
	964	1086	12,66%	322	604	1,88	4	0
MFJS8	970	1283	32,27%	278	604	2,17	1	0
	970	1263	30,21%	275	604	2,20	2	0
	970	1148	18,35%	284	604	2,13	3	1
	970	1253	29,18%	91	604	6,64	4	0
MFJS9	1105	1815	64,25%	217	605	2,79	1	0
	1105	1463	32,40%	216	604	2,80	2	1

	1105	1698	53,67%	215	604	2,81	3	0
	1105	1554	40,63%	2162	601	0,28	4	0
MFJS10	1404	2159	53,77%	213	605	2,84	1	0
	1404	1877	33,69%	823	603	0,73	2	0
	1404	1657	18,02%	205	607	2,96	3	1
	1404	1837	30,84%	207	605	2,92	4	0

Tabela 7: Resumo dos resultados obtidos por configuração

Configurações	1	2	3	4
Melhores resultados	9	10	10	11

As cinco primeiras instâncias apresentaram o resultado ótimo para as quatro configurações, somando cinco pontos para cada uma delas; as demais foram distribuídas conforme a maior proximidade do ótimo global, resultando em um melhor desempenho da configuração de número 4. Dessa forma, a porcentagem de crossover aplicada a cada problema foi definida como 10%, com 20% dos indivíduos sofrendo mutações a cada iteração.

Após a definição dos parâmetros foi elencado um conjunto de instâncias para avaliar o desempenho geral do programa. O grupo de instâncias escolhidas foi do autor Hurink *et al*. (1994), o qual adaptou algumas instâncias da literatura para se encaixarem no problema *job shop* flexível, e desenvolveu três categorias de instâncias, com probabilidades de roteamento de máquinas diferentes:

- edata: algumas operações do problema podem ser designadas para mais de uma máquina;
- rdata: a maioria das operações podem ser designadas para mais de uma máquina;
- vdata: todas as operações podem ser designadas para mais de uma máquina.

Sendo assim, o programa foi testado em cada um dos grupos de instâncias, que contém, cada um, 66 instâncias com diferentes tamanhos. Os resultados obtidos pelo algoritmo proposto estão descritos nas Tabelas 8, 9 e 10.

As tabelas possuem nove colunas que representam, respectivamente, o código da instância dentro do grupo, o tamanho do problema, sendo n igual à quantidade de *Jobs*, e m igual à quantidade de máquinas, a quantidade total de operações do problema, tal que, para esse problema em particular, o número de

operações por job é igual ao número de máquinas totais disponíveis, o melhor *makespan* conhecido por instância, retirado do trabalho de Behnke e Geiger (2012), o resultado obtido pelo algoritmo proposto, sua porcentagem de diferença com relação ao melhor *makespan* conhecido, a quantidade de iterações executadas no tempo proposto, o tempo real de execução do problema, e o tempo médio por iteração. A Equação que descreve o *gap* com relação ao melhor *makespan* é definida pela Equação (11).

$$GAP = \frac{(resultado\ obtido - melhor\ makespan)}{melhor\ makespan} * 100 \quad (11)$$

Tabela 8: Resultados obtidos com o grupo de instâncias edata do autor Hurink *et al.* (1994)

Código Instância	nxm	Quantidade total de operações	Melhor Makespan Conhecido	Resultado Obtido	GAP	Quantidade iterações executadas	Tempo de execução (segundos)	Tempo médio por iteração (segundos)
mt06†	6x6	36	55	59	7,27%	342	604	1,77
mt10†	10x10	100	871	2949	238,58%	119	610	5,13
mt20†	20x5	100	1088	1840	69,12%	116	610	5,26
la01‡	10x5	50	609	826	35,63%	746	603	0,81
la02‡	10x5	50	655	1081	65,04%	638	602	0,94
la03‡	10x5	50	550	835	51,82%	608	602	0,99
la04‡	10x5	50	568	846	48,94%	640	602	0,94
la05‡	10x5	50	503	724	43,94%	632	603	0,95
la06‡	15x5	75	833	1469	76,35%	428	603	1,41
la07‡	15x5	75	762	1270	66,67%	410	602	1,47
la08‡	15x5	75	845	1986	135,03%	427	602	1,41
la09‡	15x5	75	878	1562	77,90%	413	604	1,46
la10‡	15x5	75	866	1415	63,39%	429	604	1,41
la11‡	20x5	100	1103	2397	117,32%	326	604	1,85
la12‡	20x5	100	960	2098	118,54%	321	605	1,88
la13‡	20x5	100	1053	2321	120,42%	323	604	1,87
la14‡	20x5	100	1123	2447	117,90%	323	603	1,87
la15‡	20x5	100	1111	2290	106,12%	324	604	1,86
la16‡	10x10	100	892	3027	239,35%	309	605	1,96
la17‡	10x10	100	707	2192	210,04%	302	605	2,00
la18‡	10x10	100	842	4049	380,88%	310	604	1,95
la19‡	10x10	100	796	2767	247,61%	308	603	1,96
la20‡	10x10	100	857	3597	319,72%	301	603	2,00
la21‡	15x10	150	1009	6578	551,93%	205	604	2,95
la22‡	15x10	150	880	5555	531,25%	202	607	3,00
la23‡	15x10	150	950	6296	562,74%	204	607	2,98
la24‡	15x10	150	908	7685	746,37%	63	613	9,73
la25‡	15x10	150	936	5426	479,70%	206	607	2,95
la26‡	20x10	200	1106	12408	1021,88%	155	609	3,93

la27‡	20x10	200	1181	11232	851,06%	156	608	3,90
la28‡	20x10	200	1142	12220	970,05%	127	609	4,80
la29‡	20x10	200	1107	11526	941,19%	154	608	3,95
la30‡	20x10	200	1188	11731	887,46%	153	606	3,96
la31‡	30x10	300	1532	20979	1269,39%	103	609	5,91
la32‡	30x10	300	1698	18316	978,68%	104	609	5,86
la33‡	30x10	300	1547	22190	1334,39%	100	609	6,09
la34‡	30x10	300	1599	23047	1341,34%	104	610	5,87
la35‡	30x10	300	1736	29564	1603,00%	104	614	5,90
la36‡	15x15	225	1162	18500	1492,08%	133	608	4,57
la37‡	15x15	225	1397	23051	1550,04%	128	609	4,76
la38‡	15x15	225	1141	29564	2491,06%	104	614	5,90
la39‡	15x15	225	1184	19786	1571,11%	130	609	4,68
la40‡	15x15	225	1144	19369	1593,09%	133	609	4,58
abz5†	10x10	100	1167	3422	193,23%	95	612	6,44
abz6†	10x10	100	925	3312	258,05%	312	604	1,94
abz7†	20x15	300	604	18469	2957,78%	99	609	6,15
abz8†	20x15	300	625	17011	2621,76%	98	609	6,21
abz9†	20x15	300	644	19775	2970,65%	99	612	6,18
car1‡	11x5	55	6176	14866	140,71%	581	602	1,04
car2‡	13x4	52	6327	9288	46,80%	623	603	0,97
car3‡	12x5	60	6856	14165	106,61%	534	603	1,13
car4‡	14x4	56	7789	14481	85,92%	570	604	1,06
car5‡	10x6	60	7729	21150	173,64%	531	603	1,14
car6‡	8x9	72	7990	36451	356,21%	431	602	1,40
car7‡	7x7	63	6123	13793	125,27%	641	603	0,94
car8‡	8x8	64	7689	23128	200,79%	490	603	1,23
orb1††	10x10	100	977	4911	402,66%	311	604	1,94
orb2††	10x10	100	865	2774	220,69%	313	604	1,93
orb3††	10x10	100	951	8843	829,86%	311	604	1,94
orb4††	10x10	100	984	3566	262,40%	310	603	1,95
orb5††	10x10	100	842	4280	408,31%	312	604	1,94
orb6††	10x10	100	958	5478	471,82%	312	605	1,94
orb7††	10x10	100	389	1174	201,80%	313	605	1,93
orb8††	10x10	100	894	4605	415,10%	309	604	1,95
orb9††	10x10	100	933	4227	353,05%	309	603	1,95
orb10††	10x10	100	933	4203	350,48%	307	603	1,96

Tabela 9: Resultados obtidos com o grupo de instâncias rdata do autor Hurink *et al.* (1994)

Código Instância	nxm	Quantidade total de operações	Melhor Makespan Conhecido	Resultado Obtido	GAP	Quantidade iterações executadas	Tempo de execução (segundos)	Tempo médio por iteração (segundos)
mt06†	6x6	36	47	63	34,04%	869	601	0,69
mt10†	10x10	100	686	2015	193,73%	310	605	1,95

mt20†	20x5	100	1022	1693	65,66%	325	604	1,86
la01‡	10x5	50	570	774	35,79%	638	602	0,94
la02‡	10x5	50	529	699	32,14%	614	603	0,98
la03‡	10x5	50	477	607	27,25%	634	602	0,95
la04‡	10x5	50	502	640	27,49%	646	603	0,93
la05‡	10x5	50	457	635	38,95%	632	602	0,95
la06‡	15x5	75	799	1199	50,06%	429	604	1,41
la07‡	15x5	75	749	1160	54,87%	129	609	4,72
la08‡	15x5	75	765	1189	55,42%	427	602	1,41
la09‡	15x5	75	853	1207	41,50%	429	603	1,41
la10‡	15x5	75	804	1286	59,95%	428	602	1,41
la11‡	20x5	100	1071	1676	56,49%	323	605	1,87
la12‡	20x5	100	936	1446	54,49%	326	604	1,85
la13‡	20x5	100	1038	1576	51,83%	318	604	1,90
la14‡	20x5	100	1070	1831	71,12%	326	604	1,85
la15‡	20x5	100	1089	1689	55,10%	323	604	1,87
la16‡	10x10	100	717	2324	224,13%	309	604	1,95
la17‡	10x10	100	646	2310	257,59%	310	603	1,95
la18‡	10x10	100	666	1990	198,80%	307	604	1,97
la19‡	10x10	100	700	1925	175,00%	310	605	1,95
la20‡	10x10	100	756	2533	235,05%	307	604	1,97
la21‡	15x10	150	808	5291	554,83%	208	607	2,92
la22‡	15x10	150	741	3963	434,82%	207	608	2,94
la23‡	15x10	150	816	4023	393,01%	206	605	2,94
la24‡	15x10	150	775	5936	665,94%	207	605	2,92
la25‡	15x10	150	768	5125	567,32%	203	605	2,98
la26‡	20x10	200	1056	9328	783,33%	155	607	3,92
la27‡	20x10	200	1085	8071	643,87%	158	607	3,84
la28‡	20x10	200	1075	11691	987,53%	154	606	3,94
la29‡	20x10	200	933	7295	681,89%	157	609	3,88
la30‡	20x10	200	1068	10002	836,52%	48	621	12,94
la31‡	30x10	300	1520	20204	1229,21%	104	614	5,90
la32‡	30x10	300	1657	20299	1125,05%	104	611	5,88
la33‡	30x10	300	1497	15715	949,77%	105	612	5,83
la34‡	30x10	300	1535	24504	1496,35%	105	613	5,84
la35‡	30x10	300	1549	23490	1416,46%	104	610	5,87
la36‡	15x15	225	1023	18919	1749,36%	131	608	4,64
la37‡	15x15	225	1062	19848	1768,93%	134	607	4,53
la38‡	15x15	225	954	13038	1266,67%	133	611	4,59
la39‡	15x15	225	1011	17143	1595,65%	133	610	4,59
la40‡	15x15	225	955	15203	1491,94%	133	610	4,59
abz5†	10x10	100	954	3033	217,92%	309	604	1,95
abz6†	10x10	100	807	2491	208,67%	311	603	1,94
abz7†	20x15	300	493	22008	4364,10%	98	612	6,24
abz8†	20x15	300	507	15320	2921,70%	99	609	6,15

abz9†	20x15	300	517	13503	2511,80%	99	613	6,19
car1‡	11x5	55	5034	6112	21,41%	576	603	1,05
car2‡	13x4	52	5985	6446	7,70%	612	602	0,98
car3‡	12x5	60	5622	7434	32,23%	531	602	1,13
car4‡	14x4	56	6514	9019	38,46%	567	602	1,06
car5‡	10x6	60	5615	10855	93,32%	532	602	1,13
car6‡	8x9	72	6147	11822	92,32%	435	604	1,39
car7‡	7x7	63	4425	7282	64,56%	635	603	0,95
car8‡	8x8	64	5692	13886	143,96%	491	602	1,23
orb1††	10x10	100	746	2599	248,39%	91	610	6,70
orb2††	10x10	100	696	1574	126,15%	307	604	1,97
orb3††	10x10	100	712	1879	163,90%	308	604	1,96
orb4††	10x10	100	753	2592	244,22%	310	605	1,95
orb5††	10x10	100	639	2086	226,45%	312	604	1,94
orb6††	10x10	100	754	2476	228,38%	311	605	1,95
orb7††	10x10	100	302	985	226,16%	309	604	1,95
orb8††	10x10	100	639	2547	298,59%	310	605	1,95
orb9††	10x10	100	694	2425	249,42%	312	604	1,94
orb10††	10x10	100	742	3160	325,88%	309	605	1,96

Tabela 10: Resultados obtidos com o grupo de instâncias vdata do autor Hurink *et al.* (1994)

Código Instância	nxm	Quantidade total de operações	Melhor Makespan Conhecido	Resultado Obtido	GAP	Quantidade iterações executadas	Tempo de execução (segundos)	Tempo médio por iteração (segundos)
mt06†	6x6	36	47	73	55,32%	862	602	0,70
mt10†	10x10	100	655	2503	282,14%	310	604	1,95
mt20†	20x5	100	1022	2097	105,19%	325	604	1,86
la01‡	10x5	50	570	931	63,33%	634	602	0,95
la02‡	10x5	50	529	901	70,32%	637	602	0,95
la03‡	10x5	50	477	751	57,44%	639	603	0,94
la04‡	10x5	50	502	796	58,57%	639	603	0,94
la05‡	10x5	50	457	735	60,83%	630	603	0,96
la06‡	15x5	75	799	1284	60,70%	425	603	1,42
la07‡	15x5	75	749	1310	74,90%	425	604	1,42
la08‡	15x5	75	765	1419	85,49%	431	604	1,40
la09‡	15x5	75	853	1413	65,65%	433	603	1,39
la10‡	15x5	75	804	1638	103,73%	425	603	1,42
la11‡	20x5	100	1071	2448	128,57%	323	604	1,87
la12‡	20x5	100	936	1758	87,82%	98	613	6,26
la13‡	20x5	100	1038	1974	90,17%	322	604	1,88
la14‡	20x5	100	1070	2195	105,14%	321	603	1,88
la15‡	20x5	100	1089	2252	106,80%	322	604	1,88
la16‡	10x10	100	717	1956	172,80%	301	604	2,01
la17‡	10x10	100	646	2085	222,76%	314	605	1,93
la18‡	10x10	100	663	2184	229,41%	313	604	1,93

la19‡	10x10	100	617	2646	328,85%	312	604	1,94
la20‡	10x10	100	756	2358	211,90%	310	603	1,95
la21‡	15x10	150	800	5073	534,13%	205	605	2,95
la22‡	15x10	150	733	4292	485,54%	207	604	2,92
la23‡	15x10	150	809	4550	462,42%	207	607	2,93
la24‡	15x10	150	773	5652	631,18%	209	606	2,90
la25‡	15x10	150	751	4579	509,72%	202	607	3,00
la26‡	20x10	200	1052	8687	725,76%	156	609	3,90
la27‡	20x10	200	1084	7987	636,81%	156	607	3,89
la28‡	20x10	200	1069	8913	733,77%	155	610	3,94
la29‡	20x10	200	993	8403	746,22%	154	605	3,93
la30‡	20x10	200	1068	8224	670,04%	155	609	3,93
la31‡	30x10	300	1520	24152	1488,95%	100	612	6,12
la32‡	30x10	300	1657	24369	1370,67%	105	614	5,85
la33‡	30x10	300	1497	18191	1115,16%	106	614	5,79
la34‡	30x10	300	1535	21613	1308,01%	104	613	5,89
la35‡	30x10	300	1549	18683	1106,13%	104	611	5,88
la36‡	15x15	225	948	19805	1989,14%	41	620	15,12
la37‡	15x15	225	986	14308	1351,12%	131	608	4,64
la38‡	15x15	225	943	10094	970,41%	134	612	4,57
la39‡	15x15	225	922	12627	1269,52%	132	608	4,61
la40‡	15x15	225	955	14579	1426,60%	132	609	4,61
abz5†	10x10	100	859	3479	305,01%	94	611	6,50
abz6†	10x10	100	742	2301	210,11%	308	605	1,96
abz7†	20x15	300	492	20081	3981,50%	99	615	6,21
abz8†	20x15	300	506	14201	2706,52%	96	610	6,35
abz9†	20x15	300	497	12713	2457,95%	99	609	6,15
car1‡	11x5	55	5005	8659	73,01%	575	603	1,05
car2‡	13x4	52	5929	8260	39,32%	616	603	0,98
car3‡	12x5	60	5597	11158	99,36%	535	602	1,13
car4‡	14x4	56	6514	8606	32,12%	570	602	1,06
car5‡	10x6	60	4909	10728	118,54%	530	603	1,14
car6‡	8x9	72	5486	14911	171,80%	428	602	1,41
car7‡	7x7	63	4281	9501	121,93%	640	603	0,94
car8‡	8x8	64	4613	12609	173,34%	470	603	1,28
orb1††	10x10	100	695	2224	220,00%	311	604	1,94
orb2††	10x10	100	620	2213	256,94%	312	605	1,94
orb3††	10x10	100	648	2440	276,54%	308	604	1,96
orb4††	10x10	100	753	2349	211,95%	306	604	1,97
orb5††	10x10	100	584	2362	304,45%	305	604	1,98
orb6††	10x10	100	715	2117	196,08%	305	604	1,98
orb7††	10x10	100	275	1105	301,82%	315	605	1,92
orb8††	10x10	100	573	2261	294,59%	312	604	1,94
orb9††	10x10	100	659	2181	230,96%	297	603	2,03

orb10††	10x10	100	681	2436	257,71%	309	604	1,95

A partir da análise das Tabelas de resultados, pode-se observar que o algoritmo proposto encontra resultados factíveis para o problema, porém, em várias instâncias, os resultados encontrados estão muito distantes da melhor solução conhecida. O *gap* médio de cada grupo de instâncias foi de 604,23% para o grupo edata, 542,28% para o grupo rdata, e 536,37% para o grupo vdata. A Tabela 11 apresenta uma média do *gap* entre a melhor solução global e a solução encontrada pelo algoritmo, em cada um dos grupos de instâncias.

Tabela 11: Comparação do *gap* médio por tamanho de instância em cada problema

Tamanho	Quantidade de operações	Gap médio por tamanho		
		edata	rdata	vdata
6x6	36	7,27%	34,04%	55,32%
10x10	100	333,54%	224,91%	250,78%
20x5	100	108,24%	59,11%	103,95%
10x5	50	49,07%	32,32%	62,10%
15x5	75	83,87%	52,36%	78,09%
15x10	150	574,40%	523,18%	524,60%
20x10	200	934,33%	786,63%	702,52%
30x10	300	1305,36%	1243,37%	1277,79%
15x15	225	1739,48%	1574,51%	1401,36%
20x15	300	2850,06%	3265,86%	3048,66%
11x5	55	140,71%	21,41%	73,01%
13x4	52	46,80%	7,70%	39,32%
12x5	60	106,61%	32,23%	99,36%
14x4	56	85,92%	38,46%	32,12%
10x6	60	173,64%	93,32%	118,54%
8x9	72	356,21%	92,32%	171,80%
7x7	63	125,27%	64,56%	121,93%
8x8	64	200,79%	143,96%	173,34%

Com a análise da Tabela, é possível perceber que o algoritmo atinge melhores resultados com problemas menores, contendo cerca de 50 operações totais, fator que pode ser confirmado pelos resultados obtidos pela Tabela 12, que contém os valores de uma segunda rodada realizada com a instâncias do autor Fattahi *et al.* (2007), com as configurações definidas previamente. Comparando as Tabelas, pode-se observar que esses problemas atingem um *gap* médio de até 50%,

com algumas variações, e para problemas pequenos, com cerca de 10 operações totais, atinge o melhor resultado global.

Tabela 12: Resultados obtidos com o grupo de instâncias do autor Fattahi *et al.* (2007)

Código Instância	nxm	Quantidade total de operações	Melhor *Makespan* Conhecido	Resultado Obtido	Gap	Quantidade iterações executadas	Tempo de execução (segundos)	Tempo médio por iteração (segundos)
SFJS1	2x2	4	66	66	0,00%	6681	601	0,09
SFJS2	2x2	4	107	107	0,00%	6910	601	0,09
SFJS3	3x2	6	221	221	0,00%	4102	601	0,15
SFJS4	3x2	6	355	355	0,00%	3969	601	0,15
SFJS5	3x2	6	119	119	0,00%	4065	601	0,15
SFJS6	3x2	9	320	337	5,31%	944	602	0,64
SFJS7	3x5	9	397	407	2,52%	2966	602	0,20
SFJS8	3x4	9	253	253	0,00%	2999	601	0,20
SFJS9	3x3	9	210	230	9,52%	2990	601	0,20
SFJS10	4x5	12	516	516	0,00%	2394	601	0,25
MFJS1	5x6	15	469	482	2,77%	1962	602	0,31
MFJS2	5x7	15	482	537	11,41%	1594	602	0,38
MFJS3	6x7	18	553	706	27,67%	1668	602	0,36
MFJS4	7x7	21	634	670	5,68%	1457	602	0,41
MFJS5	7x7	21	625	743	18,88%	1452	601	0,41
MFJS6	8x7	24	762	929	21,92%	1254	601	0,48
MFJS7	8x7	32	964	1298	34,65%	977	601	0,62
MFJS8	9x8	36	970	1148	18,35%	871	602	0,69
MFJS9	11x8	44	1105	1463	32,40%	717	602	0,84
MFJS10	12x8	48	1404	1657	18,02%	663	603	0,91

Isso mostra que o algoritmo proposto é capaz de atingir o resultado ótimo dos problemas, não somente resultados factíveis, porém, necessita de melhorias para atingir melhores resultados para problemas com maior número de operações e complexidade.

Apesar de o *gap* médio ser alto para instâncias maiores, o algoritmo proposto tem alto poder de evolução das soluções. Para demonstrar isso, foram geradas 1.000 soluções aleatórias para cada instância e selecionada a melhor delas. As Tabelas 13, 14 e 15 apresentam o resultado obtido para cada instância dos três grupos do autor Hurink *et al.* (1994), comparadas com o melhor resultado obtido em uma geração de 1000 indivíduos randomicamente, sendo elencado o menor *makespan* dentro de cada geração, e comparado com o resultado obtido para cada uma das instâncias, utilizando o algoritmo. Também na Tabela 16, a qual mostra a

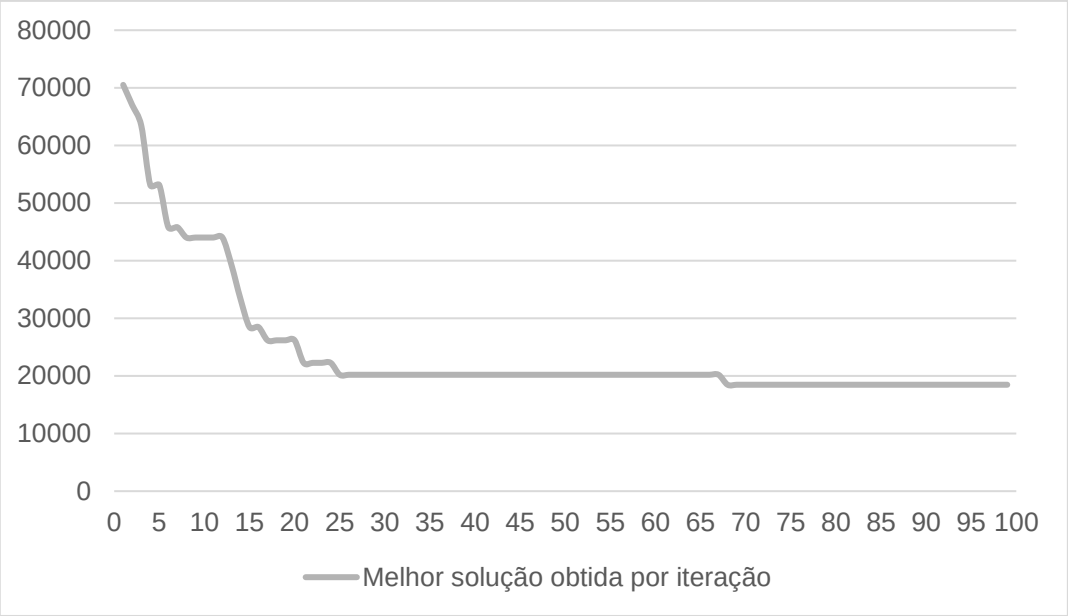
melhoria obtida pelo algoritmo com relação à melhor solução obtida na primeira iteração, com relação à melhor solução obtida pela última iteração realizada pelo algoritmo. Para a exemplificação dessa melhoria, foi elencada a instância abz7† de cada um dos três grupos de instâncias do autor, por ser a instância com maior número de operações, e, portanto, mais complexa.

O cálculo para determinação da porcentagem de melhoria com relação ao resultado obtido é dado pela Equação (12).

$$\%\, Melhoria = \frac{(melhor\ resultado\ aleatório - resultado\ obtido)}{resultado\ obtido} * 100 \qquad (12)$$

Além disso, para a instância com maior percentual de melhoria obtida dentre as analisadas na Tabela 16, a instância abz7† do grupo edata, foi plotado um gráfico (Figura 6) para ilustrar o quanto o algoritmo é capaz de melhor a solução a cada iteração. O gráfico mostra a melhor solução conhecida pelo algoritmo ao longo das interações. Com a análise do gráfico, é possível perceber que o algoritmo é capaz de melhorar rapidamente a solução nas primeiras iterações, encontrando ótimos locais no qual pode realizar melhorias. Após certo número de iterações, o algoritmo passa várias iterações realizando modificações nas soluções em busca de um novo ótimo local, que é alcançado, mas passa novamente pelo mesmo processo, necessitando de mais iterações para buscar uma nova melhoria.

Figura 6: Gráfico comparativo das melhores soluções obtidas por iteração da instância abz7†, do grupo edata, do autor Hurink *et al*. (1994)



Fonte: Do Autor, 2018

Tabela 13: Melhoria obtida com o algoritmo em comparação com resultados aleatórios com o grupo de instâncias edata do autor Hurink *et al.* (1994)

Código Instância	Melhor Resultado aleatório	Melhor Resultado Obtido pelo algoritmo	Redução do *Makespan* utilizando o algoritmo
mt06†	98	59	39,80%
mt10†	8939	2949	67,01%
mt20†	5012	1840	63,29%
la01‡	1353	826	38,95%
la02‡	1304	1081	17,10%
la03‡	1356	835	38,42%
la04‡	1401	846	39,61%
la05‡	1168	724	38,01%
la06‡	2102	1469	30,11%
la07‡	2310	1270	45,02%
la08‡	2335	1986	14,95%
la09‡	2467	1562	36,68%
la10‡	2597	1415	45,51%
la11‡	3619	2397	33,77%
la12‡	3880	2098	45,93%
la13‡	3318	2321	30,05%
la14‡	4430	2447	44,76%
la15‡	4298	2290	46,72%
la16‡	4898	3027	38,20%
la17‡	4074	2192	46,20%
la18‡	4749	4049	14,74%
la19‡	4503	2767	38,55%
la20‡	6544	3597	45,03%
la21‡	11824	6578	44,37%
la22‡	12095	5555	54,07%

la23‡	11470	6296	45,11%
la24‡	12851	7685	40,20%
la25‡	11773	5426	53,91%
la26‡	22756	12408	45,47%
la27‡	17932	11232	37,36%
la28‡	24018	12220	49,12%
la29‡	17082	11526	32,53%
la30‡	26785	11731	56,20%
la31‡	69883	20979	69,98%
la32‡	64353	18316	71,54%
la33‡	63820	22190	65,23%
la34‡	76326	23047	69,80%
la35‡	56447	29564	47,63%
la36‡	39226	18500	52,84%
la37‡	54859	23051	57,98%
la38‡	34287	29564	13,77%
la39‡	33552	19786	41,03%
la40‡	43125	19369	55,09%
abz5†	5646	3422	39,39%
abz6†	6061	3312	45,36%
abz7†	44076	18469	58,10%
abz8†	46625	17011	63,52%
abz9†	*52224*	*19775*	62,13%
car1‡	21735	14866	31,60%
car2‡	15547	9288	40,26%
car3‡	28014	14165	49,44%
car4‡	20876	14481	30,63%
car5‡	32705	21150	35,33%
car6‡	67757	36451	46,20%
car7‡	22976	13793	39,97%
car8‡	32295	23128	28,39%
orb1††	7680	4911	36,05%
orb2††	5903	2774	53,01%
orb3††	10103	8843	12,47%
orb4††	7243	3566	50,77%
orb5††	6883	4280	37,82%
orb6††	7152	5478	23,41%
orb7††	2160	1174	45,65%
orb8††	8310	4605	44,58%
orb9††	5433	4227	22,20%
orb10††	7177	4203	41,44%

Tabela 14: Melhoria obtida com o algoritmo em comparação com resultados aleatórios com o grupo de instâncias rdata do autor Hurink *et al.* (1994)

Código Instância	Melhor Resultado aleatório	Melhor Resultado Obtido pelo algoritmo	Redução do *Makespan* utilizando o algoritmo
mt06†	91	63	30,77%
mt10†	5510	2015	63,43%
mt20†	3870	1693	56,25%
la01‡	1424	774	45,65%
la02‡	1193	699	41,41%
la03‡	1217	607	50,12%

la04‡	1263	640	49,33%
la05‡	1158	635	45,16%
la06‡	2478	1199	51,61%
la07‡	2307	1160	49,72%
la08‡	2395	1189	50,35%
la09‡	2256	1207	46,50%
la10‡	2495	1286	48,46%
la11‡	3984	1676	57,93%
la12‡	3522	1446	58,94%
la13‡	3174	1576	50,35%
la14‡	3377	1831	45,78%
la15‡	4102	1689	58,82%
la16‡	5110	2324	54,52%
la17‡	4497	2310	48,63%
la18‡	5059	1990	60,66%
la19‡	5620	1925	65,75%
la20‡	6036	2533	58,04%
la21‡	10194	5291	48,10%
la22‡	13047	3963	69,63%
la23‡	11254	4023	64,25%
la24‡	13199	5936	55,03%
la25‡	11216	5125	54,31%
la26‡	26994	9328	65,44%
la27‡	18619	8071	56,65%
la28‡	24440	11691	52,16%
la29‡	25123	7295	70,96%
la30‡	25235	10002	60,36%
la31‡	83622	20204	75,84%
la32‡	90999	20299	77,69%
la33‡	58188	15715	72,99%
la34‡	80255	24504	69,47%
la35‡	85337	23490	72,47%
la36‡	42588	18919	55,58%
la37‡	46718	19848	57,52%
la38‡	44831	13038	70,92%
la39‡	39843	17143	56,97%
la40‡	38667	15203	60,68%
abz5†	7124	3033	57,43%
abz6†	5086	2491	51,02%
abz7†	54685	22008	59,75%
abz8†	55205	15320	72,25%
abz9†	*57413*	*13503*	76,48%
car1‡	14068	6112	56,55%
car2‡	12667	6446	49,11%
car3‡	16408	7434	54,69%
car4‡	13780	9019	34,55%
car5‡	22763	10855	52,31%
car6‡	24912	11822	52,54%
car7‡	14012	7282	48,03%
car8‡	26241	13886	47,08%
orb1††	5894	2599	55,90%
orb2††	5332	1574	70,48%
orb3††	7154	1879	73,73%
orb4††	4758	2592	45,52%

orb5††	6269	2086	66,73%
orb6††	6140	2476	59,67%
orb7††	2067	985	52,35%
orb8††	5137	2547	50,42%
orb9††	5266	2425	53,95%
orb10††	5353	3160	40,97%

Tabela 15: Melhoria obtida com o algoritmo em comparação com resultados aleatórios com o grupo de instâncias vdata do autor Hurink *et al.* (1994)

Código Instância	Melhor Resultado aleatório	Melhor Resultado Obtido pelo algoritmo	Redução do *Makespan* utilizando o algoritmo
mt06†	100	73	27,00%
mt10†	3453	2503	27,51%
mt20†	3876	2097	45,90%
la01‡	1323	931	29,63%
la02‡	1163	901	22,53%
la03‡	1252	751	40,02%
la04‡	1078	796	26,16%
la05‡	1210	735	39,26%
la06‡	2399	1284	46,48%
la07‡	2260	1310	42,04%
la08‡	2354	1419	39,72%
la09‡	2532	1413	44,19%
la10‡	2674	1638	38,74%
la11‡	4230	2448	42,13%
la12‡	3547	1758	50,44%
la13‡	3727	1974	47,04%
la14‡	3764	2195	41,68%
la15‡	3933	2252	42,74%
la16‡	4665	1956	58,07%
la17‡	4529	2085	53,96%
la18‡	4909	2184	55,51%
la19‡	4847	2646	45,41%
la20‡	5876	2358	59,87%
la21‡	11765	5073	56,88%
la22‡	10689	4292	59,85%
la23‡	12680	4550	64,12%
la24‡	13770	5652	58,95%
la25‡	13031	4579	64,86%
la26‡	25825	8687	66,36%
la27‡	25866	7987	69,12%
la28‡	30277	8913	70,56%
la29‡	21761	8403	61,39%
la30‡	23610	8224	65,17%
la31‡	81392	24152	70,33%
la32‡	77098	24369	68,39%
la33‡	69650	18191	73,88%
la34‡	88677	21613	75,63%
la35‡	62674	18683	70,19%
la36‡	43566	19805	54,54%
la37‡	59623	14308	76,00%
la38‡	50913	10094	80,17%

la39‡	45834	12627	72,45%
la40‡	40825	14579	64,29%
abz5†	7210	3479	51,75%
abz6†	5789	2301	60,25%
abz7†	51873	20081	61,29%
abz8†	38414	14201	63,03%
abz9†	*57932*	12713	78,06%
car1‡	13884	8659	37,63%
car2‡	13147	8260	37,17%
car3‡	16007	11158	30,29%
car4‡	14834	8606	41,98%
car5‡	18129	10728	40,82%
car6‡	25671	14911	41,92%
car7‡	12607	9501	24,64%
car8‡	20494	12609	38,47%
orb1††	5394	2224	58,77%
orb2††	5062	2213	56,28%
orb3††	4742	2440	48,54%
orb4††	5148	2349	54,37%
orb5††	4860	2362	51,40%
orb6††	4489	2117	52,84%
orb7††	1980	1105	44,19%
orb8††	4988	2261	54,67%
orb9††	4442	2181	50,90%
orb10††	5617	2436	56,63%

Tabela 16: Melhoria das soluções durante a execução do algoritmo, usando como base a instância abz7† do autor Hurink *et al.* (1994)

Grupo de instâncias	Melhor solução obtida na primeira iteração	Melhor solução obtida na última iteração	Melhoria
edata	70507	18469	73,81%
vdata	56402	22008	60,98%
rdata	32451	20081	38,12%

Para o grupo de instâncias edata, a melhoria média dos resultados com o uso do algoritmo foi de 43,02%, para o grupo de instâncias rdata, de 56,62%, e para o grupo de instâncias vdata, de 52,20%. Esses resultados, juntamente com as porcentagens de melhoria das soluções no decorrer das iterações, mostrado na Tabela 16, mostram que o algoritmo é capaz de melhorar consideravelmente os resultados obtidos, com melhorias de até 80%, mesmo em problemas mais complexos, ou seja, o algoritmo tem potencial para melhorar ainda mais os resultados e atingir o ótimo global, caso sejam realizadas melhorias e incrementos no algoritmo, para melhorar a convergência das soluções em soluções ótimas.

4.2 MELHORIA NO ALGORITMO

Devido à grande diferença nos resultados encontrados pelo algoritmo, foi implementada uma melhoria das soluções após o *crossover*, para ordenar as máquinas usando estratégia semelhante à adotada na geração de soluções iniciais.

A melhoria se baseia no princípio de alocar as máquinas segundo o menor tempo de processamento, até o momento, em cada uma das máquinas disponíveis. Dessa forma, após a formação da nova população, que passa pelos operadores de *crossover* e mutação, cada indivíduo tem seu sequenciamento analisado, e para cada operação é avaliado se a máquina roteada representa aquela com menor tempo de processamento até o momento. Caso não seja, é substituída pela máquina disponível que possui menor tempo acumulado de processamento até o momento. Esse procedimento é realizado sequencialmente, em todas as operações de cada um dos indivíduos da nova população.

Essa melhoria pode ser justificada pelo fato de que, devido aos cruzamentos e mutações, é possível que a designação de máquinas estivesse longe da ótima, portanto esse pós-processamento da população poderia melhorar os indivíduos, levando em conta seu tempo de processamento final.

Para testar a melhoria realizada no algoritmo, elegeu-se o grupo de instâncias de maior complexidade do autor Hurink *et al.* (1994), o conjunto vdata. Foram testadas as 66 instâncias, e seu resultado foi comparado com o obtido anteriormente, utilizando o algoritmo sem modificações. A comparação de resultados é representada na Tabela 17.

Tabela 17: Diferenças nos resultados obtidos após a melhoria do algoritmo, utilizando o grupo de instâncias vdata do autor Hurink *et al.* (1994)

Código Instância	Melhor Makespan Conhecido	Melhor resultado obtido após a modificação	Melhor resultado obtido antes da modificação	Novo Gap	Redução do Makespan após a modificação
mt06†	47	56	73	19,15%	30,36%
mt10†	655	2267	2503	246,11%	10,41%
mt20†	1022	1898	2097	85,71%	10,48%
la01‡	570	781	931	37,02%	19,21%
la02‡	529	741	901	40,08%	21,59%
la03‡	477	592	751	24,11%	26,86%
la04‡	502	675	796	34,46%	17,93%
la05‡	457	543	735	18,82%	35,36%
la06‡	799	1244	1284	55,69%	3,22%

la07‡	749	954	1310	27,37%	37,32%
la08‡	765	1108	1419	44,84%	28,07%
la09‡	853	1166	1413	36,69%	21,18%
la10‡	804	1181	1638	46,89%	38,70%
la11‡	1071	1590	2448	48,46%	53,96%
la12‡	936	1474	1758	57,48%	19,27%
la13‡	1038	1781	1974	71,58%	10,84%
la14‡	1070	1660	2195	55,14%	32,23%
la15‡	1089	1654	2252	51,88%	36,15%
la16‡	717	1852	1956	158,30%	5,62%
la17‡	646	1778	2085	175,23%	17,27%
la18‡	663	2122	2184	220,06%	2,92%
la19‡	617	2421	2646	292,38%	9,29%
la20‡	756	1806	2358	138,89%	30,56%
la21‡	800	4023	5073	402,88%	26,10%
la22‡	733	3787	4292	416,64%	13,34%
la23‡	809	4521	4550	458,84%	0,64%
la24‡	773	4302	5652	456,53%	31,38%
la25‡	751	4520	4579	501,86%	1,31%
la26‡	1052	7711	8687	632,98%	12,66%
la27‡	1084	7980	7987	636,16%	0,09%
la28‡	1069	8340	8913	680,17%	6,87%
la29‡	993	7028	8403	607,75%	19,56%
la30‡	1068	7568	8224	608,61%	8,67%
la31‡	1520	21157	24152	1291,91%	14,16%
la32‡	1657	18185	24369	997,47%	34,01%
la33‡	1497	16247	18191	985,30%	11,97%
la34‡	1535	17821	21613	1060,98%	21,28%
la35‡	1549	17132	18683	1006,00%	9,05%
la36‡	948	12813	19805	1251,58%	54,57%
la37‡	986	13427	14308	1261,76%	6,56%
la38‡	943	9997	10094	960,13%	0,97%
la39‡	922	10044	12627	989,37%	25,72%
la40‡	955	10041	14579	951,41%	45,19%
abz5†	859	3205	3479	273,11%	8,55%
abz6†	742	1931	2301	160,24%	19,16%
abz7†	492	9946	20081	1921,54%	101,90%
abz8†	506	11828	14201	2237,55%	20,06%
abz9†	497	12347	12713	2384,31%	2,96%
car1‡	5005	5991	8659	19,70%	44,53%
car2‡	5929	8121	8260	36,97%	1,71%
car3‡	5597	7772	11158	38,86%	43,57%
car4‡	6514	7895	8606	21,20%	9,01%
car5‡	4909	10685	10728	117,66%	0,40%

car6‡	5486	11997	14911	118,68%	24,29%
car7‡	4281	5733	9501	33,92%	65,72%
car8‡	4613	8221	12609	78,21%	53,38%
orb1††	695	2082	2224	199,57%	6,82%
orb2††	620	1863	2213	200,48%	18,79%
orb3††	648	2023	2440	212,19%	20,61%
orb4††	753	1896	2349	151,79%	23,89%
orb5††	584	1927	2362	229,97%	22,57%
orb6††	715	2005	2117	180,42%	5,59%
orb7††	275	957	1105	248,00%	15,46%
orb8††	573	1558	2261	171,90%	45,12%
orb9††	659	2175	2181	230,05%	0,28%
orb10††	681	2370	2436	248,02%	2,78%

A modificação do algoritmo levou a uma melhoria média de 21,52% nos resultados, sendo que o *gap* médio inicial era de 536,37%, e com a modificação, o *gap* médio de resultados caiu para 419,08%, ou seja, houve uma diminuição significativa na diferença entre os resultados obtidos e o melhor resultado global, mostrando que, com a ajuda de melhorias, o algoritmo é capaz de melhorar significativamente os resultados obtidos.

4.3 PROPOSIÇÃO DE MELHORIAS

Como discutido anteriormente, o algoritmo é capaz de encontrar soluções com gaps baixos para instâncias pequenas para problemas menores, porém, não exibe o mesmo desempenho para problemas maiores e mais complexos; por isso, são necessárias algumas melhorias para garantir que o algoritmo melhore seu desempenho total, e encontre melhores soluções. Para tanto, algumas ações podem ser realizadas buscando esse objetivo:

- Realizar testes com outros tamanhos de população inicial: identificar um tamanho de população que possa se adequar melhor a cada problema analisado, para aumentar a heterogeneidade da população, e aumentar a chance de obter ótimos locais;
- Aumentar o tempo de execução do problema: o algoritmo teria mais tempo para buscar melhores soluções dentre suas iterações;
- Implementar outro método para geração da população inicial: inserir indivíduos através da geração randômica, regra SPT, FOPNR, entre outras, para gerar indivíduos mais aptos no início do problema e

implementar um método visando encontrar as melhores soluções locais, utilizando-o juntamente com o método proposto, para aumentar a variedade de soluções;

- Implementar outro operador *crossover*: dividir a proporção de indivíduos que passarão pelo *crossover* implementado e o novo operador, tais como *Crossover* de dois pontos, *Crossover* uniforme, entre outros permutacionais, já citados no texto, para garantir a maior variedade na população;
- Incorporar no algoritmo um operador de mutação que melhore as soluções após o *crossover*: o operador atuaria na modificação das máquinas das operações visando a diminuição do tempo de processamento total dentro do sequenciamento proposto, melhorando os indivíduos escolhidos;
- Implementar outras estratégias de melhoria após o cruzamento dos indivíduos, usando novos algoritmos de melhoria local.

A implementação dessas melhorias pode fazer com que o algoritmo encontre soluções melhores que as reportadas neste trabalho, estando mais próximas das melhores soluções conhecidas.

5. CONCLUSÕES

Neste trabalho foi proposto um algoritmo para resolução do problema de sequenciamento *job shop* flexível, baseado nos conceitos de Algoritmos Genéticos, utilizando, para teste, conjuntos de instâncias da literatura. No algoritmo, foi proposto um modelo para geração de população inicial, baseado em um sequenciamento de operações randômico e um roteamento de máquinas focando no menor tempo de processamento acumulado por máquina, um operador *crossover* permutacional chamado OX1, e um operador de mutação que modifica o roteamento das máquinas designadas para algumas operações.

Para o teste do algoritmo, foram utilizados três grupos de instâncias do autor Hurink *et al.* (1994), e o grupo de instâncias do autor Fattahi *et al.* (2007), totalizando 218 instâncias analisadas. Com a resolução de cada instância, o algoritmo obteve um desvio com relação ao melhor resultado conhecido (*gap*) de 604,23% no grupo de instâncias edata, 542,28% no grupo de instâncias rdata, e 536,37% no grupo de instâncias vdata, do autor Hurink *et al.* (1994), com grandes discrepâncias de desvio com relação ao tamanho total da instância; e um desvio de 10,45% no grupo de instâncias do autor Fattahi *et al.* (2007).

Foi constatado que o algoritmo se comporta melhor quando resolve problemas menores, com menor quantidade de operações totais, como é o caso das instâncias do segundo autor analisado. Já problemas maiores e mais complexos se apresentam como desafios para o algoritmo, que encontra resultados muito distantes dos ótimos de cada problema.

Para testar a capacidade do algoritmo de obter melhores resultados, foi inserida uma melhoria no mesmo, aplicada em cada um dos indivíduos da população após passarem pelos operadores de *crossover* e mutação, que se baseia em determinar as máquinas para cada operação sequenciada, com base na máquina disponível com menor tempo acumulado. A melhoria fez com que houvesse uma redução significativa no *gap* médio das instâncias analisadas, provando que o algoritmo é capaz de obter melhores resultados, caso haja outras melhorias implementadas.

Sendo assim, algoritmo cumpre com o objetivo proposto, sendo capaz de resolver problemas de sequenciamento *job shop*, e encontrar os resultados ótimos para alguns problemas. Entretanto, seu desempenho poderia ser melhorado com algumas modificações e adições no algoritmo, como incremento de um operador

crossover, modificação do método para geração da população inicial, entre outros, chegando a resultados mais próximos do ótimo global para problemas maiores.

6. REFERÊNCIAS

ABDOLRAZZAGH-NEZHAD, M.; ABDULLAH, S.. ***Job Shop Scheduling: Classification, Constraints and Objective Functions***. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, v. 11, n. 4, p. 423-428, 2017. Disponível em <http://waset.org/publications/10006691/job-shop-scheduling-classification-constraints-and-objective-functions>. Acesso em 15 abril 2018.

ALVAREZ-VALDES, R. *et al.*. ***A heuristic to schedule flexible job-shop in a glass factory***. *European Journal of Operational Research*, pp. 525 – 534, 2005. Disponível em <https://www.sciencedirect.com/science/article/pii/S0377221704002589>. Acesso em 17 maio 2018.

BAKER, K. R.; TRIETSCH, D.. ***Principles of sequencing and scheduling***. *John Wiley & Sons*, 2009.

BŁAŻEWICZ, J.; DOMSCHKE, W.; PESCH, E.. ***The job shop scheduling problem: Conventional and new solution techniques***. *European journal of operational research*, v. 93, n. 1, p. 1-33, 1996. Disponível em <https://www.sciencedirect.com/science/article/pii/0377221795003622>. Acesso em 16 abril 2018.

BEHNKE, D.; GEIGER, M. J.. ***Test instances for the flexible job shop scheduling problem with work centers.*** *Helmut-Schmidt-University, Logistics Management Department*, Hamburgo, Alemanha. 2012.

BIRGIN, E. G. *et al*. ***A MILP model for an extended version of the Flexible Job Shop Problem***. *Optimization Letters*, v. 8, n. 4, p. 1417-1431, 2014.

CHAN, F.; WONG, T. & CHAN, L.. ***Flexible job-shop scheduling problem under resource constraints***. *International Journal of Production Research*, pp. 2071 – 2089, 2006. Disponível em <https://www.tandfonline.com/doi/abs/10.1080/00207540500386012>. Acesso em 17 maio 2018.

CHAUDHRY, I. A.; KHAN, A. A.. **A *research survey: review of flexible job shop scheduling techniques.*** *International Transactions in Operational Research*, v. 23, n. 3, p. 551-591, 2016. Disponível em <https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12199>. Acesso em 20 maio 2018.

CHAUDHRY, I. A.; KHAN, A. M.; KHAN, A. A.. ***A genetic algorithm for flexible job shop scheduling***. *Proceedings of the world congress on engineering,* p. 1-6, 2013. Disponível em <https://s3.amazonaws.com/academia.edu.documents/34427963/WCE2013_pp703-708.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1528238809&Signature=1BLY%2BzOPW9KEOn5V88TxCHXy6zY%3D&response-content-disposition=inline%3B%20filename%3DJob_Shop_Shceduling.pdf>. Acesso em 16 abril 2018.

CHEN, H.; IHLOW, J.; LEHMANN, C.. ***A genetic algorithm for flexible job-shop scheduling***. *IEEE International Conference on Robotics and Automation*, 2, pp. 1120 – 1125, 1999. Disponível em <https://ieeexplore.ieee.org/abstract/document/772512/>. Acesso em 16 abril 2018.

CHEN, J. C. *et al*. ***A study of the flexible job shop scheduling problem with parallel machines and reentrant process***. *The International Journal of Advanced Manufacturing Technology*, v. 39, n. 3-4, p. 344-354, 2008. Disponível em <https://link.springer.com/article/10.1007/s00170-007-1227-1>. Acesso em 20 maio 2018.

CORRÊA, H. L.; CORRÊA, C. A.. **Administração de Produção E Operações: Manufatura E Serviços: Uma Abordagem Estratégica** . Editora Atlas SA, 2000.

CORRÊA, H.; GIANESI, I.; CAON, M.. **Planejamento, programação e controle da produção**. São Paulo: Atlas, 2007.

DAMIJ, N.. ***Business process modelling using diagrammatic and tabular techniques***. *Business process management journal,* v. 13, p. 70-90, 2007. Disponível em

<https://www.emeraldinsight.com/doi/full/10.1108/14637150710721131>. Acesso em 02 fevereiro 2018.

DAUZÈRE-PÉRÈS, S., PAULLI, J.. ***An integrated approach for modeling and solving the general multi-processor job-shop scheduling problem using tabu search****. Annals of Operations Research,* v. 70, p. 281–306, 1997. Disponível em <https://link.springer.com/article/10.1023/A:1018930406487>. Acesso em 17 abril 2018.

DE GIOVANNI, L.; PEZZELLA, F**.. *An improved genetic algorithm for the distributed and flexible job-shop scheduling problem.*** *European journal of operational research*, v. 200, n. 2, p. 395-408, 2010. Disponível em <https://www.sciencedirect.com/science/article/pii/S0377221709000113>. Acesso em 20 maio 2018.

DUGARDIN, F. *et al*. ***Hybrid Job Shop and parallel machine scheduling problems: minimization of total tardiness criterion****. Multiprocessor Scheduling, Theory and Applications,* v. 16 p. 273 – 291, 2007. Disponível em <https://www.intechopen.com/books/multiprocessor_scheduling_theory_and_applications/hybrid_job_shop_and_parallel_machine_scheduling_problems__minimization_of_total_tardiness_criterion>. Acesso em 05 junho 2018.

FATTAHI, P.; MEHRABAD, M. S.; JOLAI, F.. ***Mathematical modeling and heuristic approaches to flexible job shop scheduling problems****. Journal of intelligent manufacturing*, v. 18, p. 331-342, 2007. Disponível em <https://link.springer.com/article/10.1007/s10845-007-0026-8>. Acesso em 10 outubro de 2018.

GARCIA, A. *et al*.. ***Modularizing design patterns with aspects: a quantitative study****. Transactions on Aspect-Oriented Software Development I.* p. 36-74, 2006. Disponível em <https://link.springer.com/chapter/10.1007/11687061_2>. Acesso em 17 abril 2018.

GAREY M. R., JOHNSON D. S., SETHI R.. *The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research,* v. 1, p.117–29, 1976. Disponível em <https://pubsonline.informs.org/doi/abs/10.1287/moor.1.2.117>. Acesso em 02 fevereiro 2018.

GEN, M.; CHENG, R.. ***Genetic algorithms and engineering optimization***. John Wiley & Sons, 2000.

GONG, G. *et al*.. ***A new double flexible job-shop scheduling problem integrating processing time, green production, and human factor indicators****. Journal of Cleaner Production*, v. 174, p. 560-576, 2018. Disponível em <https://www.sciencedirect.com/science/article/pii/S0959652617324952>. Acesso em 18 maio 2018.

GUPTA, Y. P.; GUPTA, M. C.; BECTOR, C. R.. ***A review of scheduling rules in flexible manufacturing systems****. International Journal of Computer Integrated Manufacturing*, v. 2, n. 6, p. 356-377, 1989. Disponível em <https://www.researchgate.net/profile/Mahesh_Gupta8/publication/242928913_A_review_of_scheduling_rules_in_flexible_manufacturing_systems/links/573e233908aea45ee842e79d/A-review-of-scheduling-rules-in-flexible-manufacturing-systems.pdf>. Acesso em 20 maio 2018.

HO N. B., TAY J. C.. ***GENACE: an efficient cultural algorithm for solving the Flexible Job-Shop Problem****. international conference on robotics and automation,* p. 1759–66, 2004. Disponível em <https://ieeexplore.ieee.org/abstract/document/1331108/>. Acesso em 18 abril 2018.

HURINK, E., JURISCH, B., THOLE, M.. ***Tabu search for the job shop scheduling problem with multi-purpose machines.*** *Operations Research Spectrum*, 15, 205–215, 1994. Disponível em <https://link.springer.com/article/10.1007/BF01719451>. Acesso em 17 maio 2018.

JIA H.Z., *et al.*. A modified genetic algorithm for distributed scheduling problems. International Journal of Intelligent Manufacturing 2003;14:351–62. Disponível em <https://link.springer.com/article/10.1023/A:1024653810491>. Acesso em 18 abril 2018.

KACEM, I., HAMMADI S., BORNE P.. ***Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems.*** *Transactions on Systems, Man, and Cybernetics,* v. 32, p. 1–13. Disponível em <https://ieeexplore.ieee.org/abstract/document/1009117/>. Acesso em 20 maio 2018.

KUMAR, R.; GOPAL, G.; KUMAR, R.. ***Novel crossover operator for genetic algorithm for permutation problems****. International Journal of Soft Computing and Engineering* (IJSCE), v. 3, p. 252-258, 2013. Disponível em <https://pdfs.semanticscholar.org/a3cb/a4955512230e4a8314d4bbd8b3aa5bf85b53.pdf>. Acesso em 20 maio de 2018.

LEUNG, J. Y. T.. ***Handbook of scheduling: algorithms, models, and performance analysis***. CRC Press, 2004.

LI, J.; PAN, Q.; LIANG, Y.. ***An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems****. Computers & Industrial Engineering*, v. 59, p. 647-662, 2010. Disponível em <https://www.sciencedirect.com/science/article/pii/S0360835210002056>. Acesso em 20 abril 2018.

LOUKIL, T., TEGHEM, J., FORTEMPS, P.. ***A multi-objective production scheduling case study solved by simulated annealing****. European Journal of Operational Research,* v. 179, pp. 709 – 722, 2007. Disponível em <https://www.sciencedirect.com/science/article/pii/S0377221705007319>. Acesso em 17 abril 2018.

MASTROLILLI, M.. ***Flexible Job Shop Problem***. http://people.idsia.ch/~monaldo/fjsp.html: [s.n.], 2018. Acessado em 14 outubro de 2018.

ÖZGÜVEN, C.; YAVUZ, Y.; ÖZBAKIR, L.. ***Mixed integer goal programming models for the flexible job-shop scheduling problems with separable and non-separable sequence dependent setup times****. Applied Mathematical Modelling,* v. 36, p. 846-858, 2012.

PAULLI, J.. ***A hierarchical approach for the FMS scheduling problem****. European Journal of Operational Research,* v. 86, 32–42, 1995. Disponível em <https://www.sciencedirect.com/science/article/pii/037722179500059Y>. Acesso em 20 maio 2018.

PEZZELLA, F.; MORGANTI, G.; CIASCHETTI, G.. ***A genetic algorithm for the flexible job-shop scheduling problem****. Computers & Operations Research*, v. 35, n. 10, p. 3202-3212, 2008. Disponível em <https://www.sciencedirect.com/science/article/pii/S0305054807000524>. Acesso em 18 abril 2018.

PINEDO, M. L.. ***Scheduling: theory, algorithms, and systems.*** Springer, 2016.

PREVIERO, W. D.. **Estratégias de resolução para o problema de job-shop flexível**. Tese de Doutorado. Universidade de São Paulo. Disponível em <http://www.teses.usp.br/teses/disponiveis/45/45134/tde-20102016-123243/pt-br.php>. Acesso em 20 maio 2018.

REVEES, C. R.; ROWE, J. E. ***Genetic algorithms–principles and perspectives***. *Kluwer Academic Publishers*, 2002.

SCOFIELD, W. C. L.. **Aplicação de Algoritmos Genéticos ao Problema Job-Shop**. Universidade Federal de Ouro Preto. MG. Brasil, 2002.

SCRICH, C., ARMENTANO, V.; LAGUNA, M.. ***Tardiness minimization in a flexible job shop: A tabu search approach****. Journal of Intelligent Manufacturing*, v. 15, p. 103 – 115, 2004. Disponível em <https://link.springer.com/article/10.1023/B:JIMS.0000010078.30713.e9>. Acesso em 17 abril 2018.

SIVANANDAM, S. N.; DEEPA, S. N.. ***Introduction to genetic algorithms***. Springer Science & Business Media, 2007.

XIA, W.; WU, Z.. ***An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems***. *Computers & Industrial Engineering*, v. 48, p. 409 – 425, 2005. Disponível em <https://www.sciencedirect.com/science/article/abs/pii/S0360835205000197>. Acesso em 18 abril 2018.

YAZDANI, M., AMIRI, M., ZANDIEH, M.. ***Flexible job-shop scheduling with parallel variable neighborhood search algorithm***. *Expert Systems with Applications*, v. 37, p. 678–687, 2010. Disponível em <https://www.sciencedirect.com/science/article/pii/S0957417409005685>. Acesso em 18 maio 2018.

ZHANG, G.; GAO, L.; SHI, Y... ***An effective genetic algorithm for the flexible job-shop scheduling problem***. *Expert Systems with Applications*, v. 38, p. 3563-3573, 2011. Disponível em <https://www.sciencedirect.com/science/article/pii/S095741741000953X>. Acesso em 17 abril 2018.