Dayan de Castro Bissoli

Meta-heurísticas para resolução de alguns problemas de planejamento e controle da produção

Vitória, ES

2018

Dayan de Castro Bissoli

Meta-heurísticas para resolução de alguns problemas de planejamento e controle da produção

Tese apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Doutor em Ciência da Computação.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Orientador: Ph.D. André Renato Sales Amaral

Vitória, ES

2018

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Setorial Tecnológica,
Universidade Federal do Espírito Santo, ES, Brasil)

Bissoli, Dayan de Castro, 1987-

B623m Meta-heurísticas para resolução de alguns problemas de planejamento e controle da produção / Dayan de Castro Bissoli. – 2018.

144 f. : il.

Orientador: André Renato Sales Amaral.

Tese (Doutorado em Ciência da Computação) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Controle de produção. 2. Planejamento da produção. 3. Otimização. 4. Meta-heurísticas. 5. Problema Job shop scheduling (JSP). 6. Problema Job shop scheduling flexível (JSPF). 7. Problema de balanceamento de linhas de montagem (SALBP-2). I. Amaral, André Renato Sales. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

Elaborada por Sandra Mara Borges Campos – CRB-6 ES-000593/O

Meta-heurísticas para resolução de alguns problemas de planejamento e controle da produção

Dayan de Castro Bissoli

Tese submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Aprovada em 03 de agosto de 2018 por:

Prof. Dr. André Renato Sales Amaral (Orientador)
UFES/ES

Prof. Dr. Renato Elias Nunes de Moraes (Examinador Interno)
UFES/ES

Prof. Dr. Geraldo Regis Mauri (Examinador Interno)
UFES/ES

Prof. Dr. Luciano Lessa Lorenzoni (Examinador Externo)
IFES/ES

Prof. Dr. Jorge Pinho de Sousa (Examinador Externo)
UNIVERSIDADE DO PORTO/ PORTUGAL

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Vitória-ES, 03 de agosto de 2018.

A minha família...

Agradecimentos

Agradeço à minha família, pelo amor e apoio incondicional.

À Katia, pelo amor e força nas horas de dificuldade. Pelo seu companheirismo e compreensão em todos os momentos. E pela grande contribuição no desenvolvimento das figuras deste trabalho.

Às tias Áurea e Teresa, pela recepção e estadias em Vitória.

Ao Professor e orientador André, pelas oportunidades cedidas, pelos conselhos e rumos ministrados.

Aos Membros da Banca Examinadora pela participação na defesa deste trabalho e também pelas valiosas contribuições cedidas.

Aos professores e colaboradores do Programa de Pós-Graduação em Informática (PPGI), da Universidade Federal do Espírito Santo (UFES).

Ao Departamento de Computação (DCOMP) do Centro de Ciências Exatas, Naturais e da Saúde (CCENS) da UFES, pela receptividade e pelo grande apoio prestado na realização deste trabalho.

Aos colegas da Coordenadoria de Informática (COINFO), do Istituto Federal do Espírito Santo (IFES), Campus Colatina, que num curto período como professor substituto, realizaram um grande incentivo para cursar o Doutorado.

Aos colegas do PPGI, que me acompanharam nesta jornada.

A todos meus amigos, pelo incentivo e força.

À UFES, pela oportunidade de crescimento pessoal e profissional.

À Fundação de Amparo à Pesquisa do Espírito Santo (FAPES), pelo auxílio financeiro para realização deste trabalho.

Resumo

Este estudo aborda a resolução de três diferentes problemas, amplamente encontrados no real contexto de planejamento e controle da produção. Inicialmente, é proposta uma meta-heurística GRASP para solucionar um problema de balanceamento de linhas de montagem (SALBP-2). O método proposto apresentou resultados competitivos em relação à literatura, também focando numa simplicidade de operação para ser aplicada em casos reais. Na sequência, utilizou-se o mesmo método para solucionar o problema *Job Shop Scheduling* (JSP). O GRASP desenvolvido para o JSP também apresentou bons resultados, com baixo desvio relativo médio em relação às melhores soluções conhecidas da literatura. Em seguida, abordou-se uma extensão do JSP, o problema *Job Shop Scheduling* Flexível (FJSP). O JSP limita-se ao sequenciamento de operações em máquinas fixas, enquanto que no FJSP a atribuição de uma operação não é pré-fixada e pode assim ser processada num conjunto de máquinas alternativas. Portanto, o FJSP não se restringe ao sequenciamento, estendendo-se na atribuição de operações para as máquinas adequadas (roteamento). O FJSP é, portanto, mais complexo do que o JSP, pois considera a determinação da atribuição da máquina para cada operação. Para solucionar o FJSP, propôs-se quatro meta-heurísticas: GRASP, *Simulated Annealing* (SA), *Iterated Local Search* (ILS) e *Clustering Search* (CS). O SA apresentou resultados inferiores, porém, ao incorporá-lo numa versão híbrida do ILS, que o utiliza como busca local, os resultados melhoraram, principalmente em instâncias mais complexas. Considerando a característica híbrida do CS, utilizou-se também o SA, nesse caso como meta-heurística geradora de soluções. Essa abordagem também apresentou resultados superiores ao SA. Tanto o ILS quanto o CS geraram resultados com valores iguais ou próximos àqueles das melhores soluções conhecidas para um extenso conjunto de instâncias para o FJSP, assim como também proveram alguns novos melhores valores conhecidos.

Palavras-chaves: SALBP-2, JSP, FJSP, meta-heurística, Planejamento e Controle da Produção, Otimização.

Abstract

This study addresses the resolution of three different problems, widely encountered in the real context of production planning and control. Initially, a GRASP metaheuristic is proposed to solve an assembly-line balancing problem (SALBP-2). The proposed method presented competitive results in relation to the literature, also focusing on a simplicity of operation to be applied in real cases. Subsequently, the same method was used to solve the Job Shop Scheduling Problem (JSP). The GRASP developed for the JSP also presented good results, with low average relative deviation in relation to the best solutions known in the literature. Next, we approached an extension of the JSP, the Flexible Job Shop Scheduling Problem (FJSP). The JSP is limited to the sequencing of operations on fixed machines, whereas in the FJSP the assignment of an operation is not preset and can thus be processed on a set of alternative machines. Therefore, the FJSP is not restricted to sequencing, extending in the assignment of operations to the appropriate machines (routing). The FJSP is more complex than the JSP because it considers the determination of the assignment of the machine for each operation. In order to solve the FJSP, we proposed four meta-heuristics: GRASP, Simulated Annealing (SA), Iterated Local Search (ILS) and Clustering Search (CS). SA presented lower results, however, incorporating it into a hybrid version of ILS, which uses it as a local search, the results improved, especially in more complex instances. Considering the hybrid characteristic of CS, the SA was also used, in this case as a solution-generating metaheuristic. This approach also presented superior results to SA. Both ILS and CS generated results with values equal to or close to those of the best known solutions for an extensive set of instances for the FJSP, as well as providing some new best known values.

Keywords: SALBP-2, JSP, FJSP, metaheuristic, Production Planning and Control, Optimization.

Lista de figuras

Lista de tabelas

Lista de algoritmos

Sumário

1 Introdução

Ambientes industriais possuem como característica a necessidade de utilizar eficientemente recursos disponíveis como estações de trabalho, funcionários, máquinas, etc. Assim, para maximizar o lucro necessitam produzir com o menor custo possível, minimizando o desperdício de tempo e respeitando determinadas restrições.

Inúmeros são os problemas encontrados nesses ambientes industriais, um deles seria no setor de Planejamento e Controle da Produção (PCP). Na área de produção, vários problemas são caracterizados como problemas de Otimização Combinatória (OC).

Na indústria capixaba, especificamente em uma indústria de confecções, foram identificados alguns problemas de OC, conforme ilustrado na Figura 1 onde apresenta-se um exemplo simplificado de um ambiente produtivo de uma indústria de confecção.

Na *área de produção* há um conjunto de produtos a serem produzidos simultaneamente. Tais produtos são subdivididos em um conjunto de operações que devem ser processadas nos setores (que podem ser máquinas, estações de trabalho, operadores, etc.) identificados de 01 a 08, em uma sequência pré-definida, caracterizando um problema *job shop scheduling* (JSP), abordado no Capítulo 4.

Quando considera-se a possibilidade de terceirização de algum setor, conforme indicado nos setores 03 e 04, agrega-se complexidade ao JSP, possibilitando que uma determinada operação possa vir a possuir rotas alternativas de produção, o que caracteriza um problema *job shop scheduling* flexível (FJSP), explanado no Capítulo 5.

Considera-se que algum setores da área de produção são arranjados de forma celular (04 a 06), no qual um uma operação consiste em um conjunto de sub-operações, que são agrupadas para serem processadas em uma linha de produção (TUBINO, 1999). Nesse contexto identifica-se um problema de balanceamento de linhas, que é abordado no Capítulo 3.

Problemas de OC podem ser representados por um modelo matemático que visa maximizar ou minimizar uma função objetivo, cujas variáveis são sujeitas a determinadas restrições. Considerando a complexidade desses problemas, que geralmente são caracterizados como *NP-Hard* (KNUTH, 1974), isto é, que até o momento não se conhece um algoritmo capaz de solucioná-lo em tempo polinomial, torna-se interessante a aplicação de métodos de otimização baseados em heurísticas e meta-heurísticas para solucioná-los (ARENALES et al., 2007).

Meta-heurísticas visam explorar de forma inteligente o espaço de soluções, de modo a proporcionar boas soluções (próximas da otimalidade) a um razoável custo computacional.

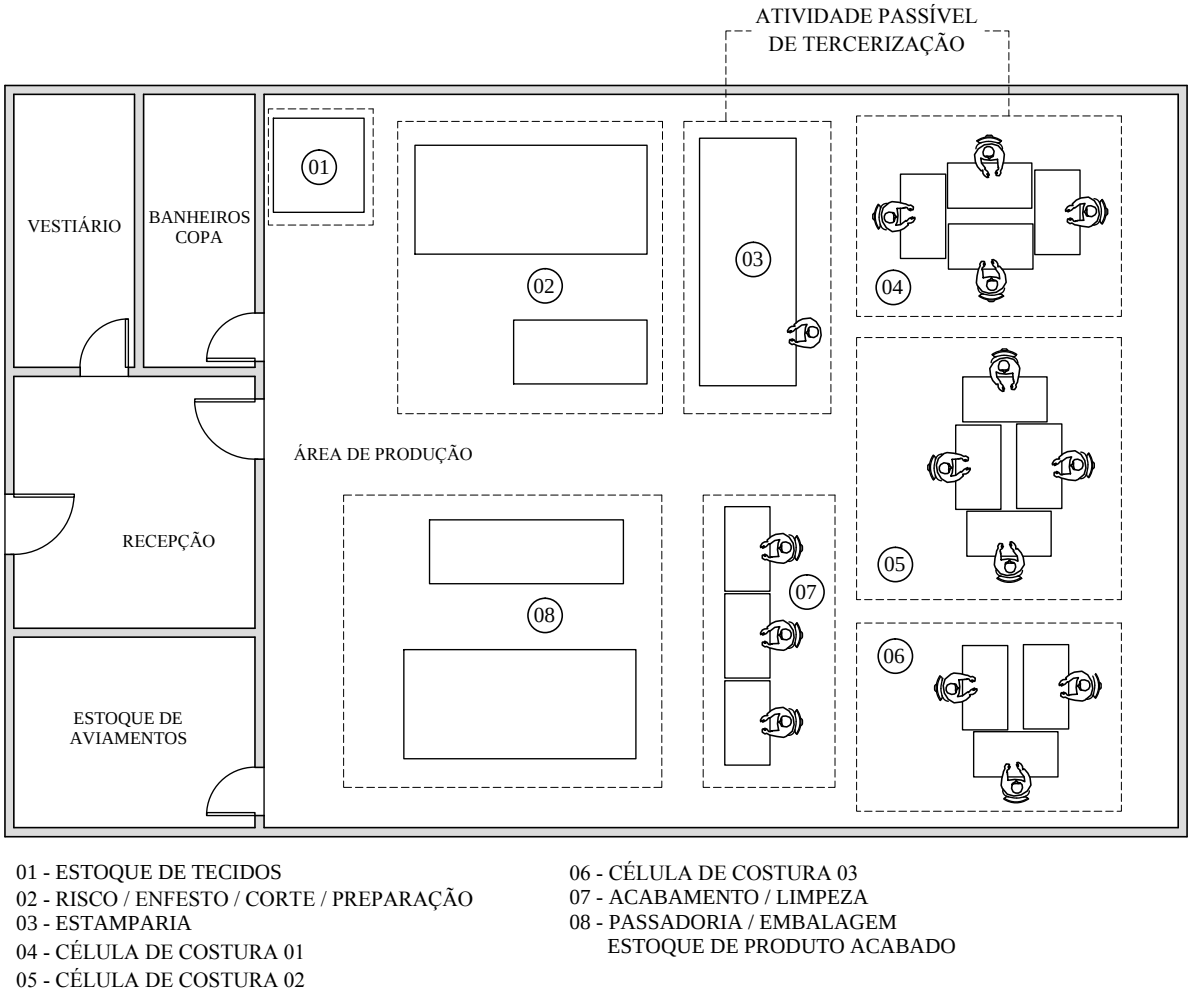

Figura 1 – Exemplo de arranjo físico de uma indústria de confecção.

Seu funcionamento baseia-se na combinação de heurísticas que agregam características específicas do problema, aplicando-as um ciclo de repetições, fazendo uso da aleatoriedade. Tais heurísticas visam escapar de ótimos locais, explorando a vizinhança por meio de pertubações e/ou busca locais (GENDREAU; POTVIN, 2010).

1.1 Objetivos e Contribuições

Esta tese se propõe a contribuir com o desenvolvimento de algoritmos baseados em meta-heurísticas para resolver diferentes problemas de otimização combinatória no contexto de PCP. Entretanto, considerando o grande número de variações existentes, torna-se praticamente impossível tratar todas nesse estudo. Logo, buscou-se chamar a atenção de um subconjunto de problemas, listados a seguir:

(i) Problema de Balanceamento de Linhas de Montagem Simples do tipo dois (SALBP-2) (REITER, 1969).

(ii) Problema *Job Shop Scheduling* (JSP) (JOHNSON, 1954).

(iii) Problema *Job Shop Scheduling* Flexível (FJSP) (BRUCKER; SCHLIE, 1990).

Os principais objetivos deste trabalho são, a partir da identificação de problemas de otimização combinatória, os quais ocorrem na indústria de confecção capixaba, desenvolver algoritmos eficientes para solucionar os problemas. Para avaliar a eficiência dos métodos desenvolvidos, realiza-se análises considerando o estado da arte da literatura.

Para tanto, desenvolveu-se algoritmos baseados na meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) (FEO; RESENDE, 1989) para os problemas SALBP-2, JSP e FJSP. Considerando a relevância do FJSP, e que o GRASP não apresentou resultados satisfatórios para o problema, buscou-se desenvolver outras meta-heurísticas, com o objetivo de obter melhores resultados para o respectivo problema. Dessa forma, foram propostos mais três algoritmos baseados nas seguintes meta-heurísticas: *Simulated Annealing* (KIRKPATRICK; GELATT; VECCHI, 1983), *Iterated Local Search* (LOURENÇO; MARTIN; STÜTZLE, 2002) e *Clustering Search* (OLIVEIRA, 2004).

1.2 Organização do Trabalho

Esse trabalho está organizado em seis capítulos, sendo este o primeiro. No Capítulo 2 são introduzidos os conceitos básicos sobre a área de pesquisa e, de forma resumida, as principais características das meta-heurísticas utilizadas.

No Capítulo 3 aborda-se o simples problema de balanceamento de linhas de montagem do tipo 2 (SALBP-2). Uma descrição do problema e uma breve revisão bibliográfica referente ao SALBP-2 são realizadas. Em seguida, apresenta-se a meta-heurística proposta para resolução do problema, os resultados computacionais e algumas considerações.

Os Capítulos 4 e 5 apresentam, respectivamente, os problemas *Job Shop Scheduling* (JSP) e, sua extensão, *Job Shop Scheduling* Flexível (FJSP). Nesses capítulos também são apresentadas as definições, formulações matemáticas e breve revisões bibliográficas sobre os problemas abordados, assim como as meta-heurísticas propostas para solucioná-los, os resultados computacionais e algumas considerações.

O Capítulo 6 sumariza as conclusões e contribuições deste estudo, seguidas por algumas sugestões para sua continuação. Por conseguinte, as referências são listadas, asssim como os Apêncices A e B.

2 Fundamentos Teóricos

Neste capítulo aborda-se os fundamentos teóricos que embasam todas as abordagens realizadas durante a tese.

2.1 Otimização Combinatória

A otimização combinatória é um subconjunto de otimização matemática e está relacionada à pesquisa operacional e à teoria da computação. Possui aplicações importantes em vários campos, incluindo inteligência artificial, aprendizado de máquina e engenharia de *software*. De acordo com Wolsey (1998), nessa categoria de problema, geralmente é dado um conjunto finito $N = \{1, ..., n\}$, pesos c_j para cada $j \in N$, e um conjunto F de subconjuntos viáveis de N. O problema de encontrar um subconjunto viável de peso mínimo é caracterizado como um Problema de Otimização Combinatória (Equação 2.1).

$$min_{S \subseteq N}\{\sum_{j \in S} c_j : S \in F\} \tag{2.1}$$

Problemas de otimização geralmente envolvem um grande número de alternativas, sendo encontrados nos setores público e privado da economia. Nesses problemas, dado um conjunto de soluções finitas X e uma função real $f : x \to \mathbb{R}$, busca-se encontrar uma solução x^* tal que $f(x^*) \leq f(x), \forall x \in X$. Alguns problemas podem possuir características específicas que podem alterar seu domínio, podendo ser pertencente ao conjunto dos números inteiros (chamados de problemas de programação inteira - PI) ou mesmo possuir variáveis inteiras e reais, chamados de problemas de programação inteira mista (PIM). São exemplos de problemas de otimização o projeto de redes de telecomunicações, a construção de horários rentáveis de equipes de trabalho, ou mesmo problemas de planejamento e controle de produção. Para encontrar uma solução ótima em um problema de otimização combinatória é, teoricamente, possível enumerar as soluções e avaliar cada uma com respeito ao objetivo definido. No entanto, na prática, muitas vezes é impossível seguir essa estratégia de enumeração completa porque o número de combinações geralmente cresce exponencialmente com o tamanho do problema e, nesse contexto, surge a necessidade de aplicação de métodos de otimização (WOLSEY, 1998; BAZARAA; JARVIS; SHERALI, 2009).

Verifica-se um grande volume de trabalho nos últimos anos para desenvolver métodos de otimização que não exigem explicitamente uma análise de cada alternativa. Essa característica de pesquisa deu origem ao campo de otimização combinatória, que vem proporcionando uma capacidade crescente de solucionar problemas cada vez mais

complexos do mundo real (PAPADIMITRIOU; STEIGLITZ, 1982). Ainda assim, grande parte dos problemas encontrados na indústria são computacionalmente intratáveis pela natureza ou suficientemente grandes para impedir o uso de algoritmos exatos. Nesses casos, métodos heurísticos geralmente são empregados para encontrar boas soluções, mas não necessariamente garantidas ótimas. A eficácia desses métodos depende da sua capacidade de se adaptar a um contexto específico, evitando o aprisionamento em ótimos locais, explorando a estrutura básica do problema.

2.2 Métodos heurísticos

Dado que muitos, se não a maioria, dos problemas práticos que desejamos resolver são *NP*-hard, não é de surpreender que algoritmos heurísticos ou de aproximação desempenhem um papel importante em problemas de otimização combinatória. Segundo Wolsey (1998), métodos heurísticos possuem algumas características:

- Uma heurística pode ser vista como uma técnica inspirada em processos intuitivos que procura uma boa solução a um custo computacional aceitável;
- Em geral, não garantem a otimalidade, nem permitem verificar o quão próximo uma solução está do ótimo global;
- O desafio é produzir, em tempo reduzido, soluções tão próximas quanto possível da solução ótima.
- A maioria das heurísticas é muito específica para um problema particular.

Colin (2007) faz uma analogia entre os métodos heurísticos e o problema de localizar, empiricamente, o ponto mais alto da Terra. Para resolver este problema, partiria-se de um ponto viável, ou seja, de qualquer lugar na superfície terrestre, em busca das montanhas mais altas. Neste processo, várias montanhas seriam escaladas e suas alturas comparadas. O ponto mais alto iria progressivamente aumentando com as novas descobertas (os chamados ótimos locais). Até que em determinado momento as buscas se dessem por encerradas por algum critério (por exemplo, a não descoberta de pontos mais altos por um longo período, falta de segurança, restrições de tempo ou financeiras, tempo, etc.) e o ponto mais alto fosse definido, mesmo sem uma comprovação científica, mesmo existindo a possibilidade de existir outro ponto mais alto, ainda inexplorado.

Assim são os métodos heurísticos, uma busca contínua e empírica, que pode-se agregar conhecimento específico do problema (inteligência), que tende a encontrar vários ótimos locais, cujo resultado é o melhor que se pode encontrar sob determinadas condições. Em relação a métodos heurísticos, há duas abordagens básicas: algoritmos construtivos e busca local.

Algoritmos construtivos consistem em construir uma solução de um problema de forma incremental. A cada passo, um componente é escolhido e depois inserido na solução até gerar uma solução completa. O componente escolhido em cada passo é um candidato definido de acordo com algum critério.

Partindo de uma solução inicial, a busca local consiste em "navegar" iterativamente pelo espaço de busca movendo-se, em cada passo, de uma solução para uma solução vizinha (adjacente). Dessa forma, define-se os seguintes conceitos:

- Vizinhança: seja S o espaço de busca do problema e s a solução de um problema. A função vizinhança é uma função $N(s)$ que mapeia cada solução $s \in S$ para um subconjunto $N(s) \subseteq S$. Um elemento qualquer de $N(s)$ é denominado vizinho de s.

- Movimento ou Perturbação: uma operação para um vizinho $s' \in N(s)$ ser alcançado pela solução s.

Dessa forma, métodos de solução que coordenam procedimentos de busca local com estratégias de mais alto nível (podendo ter inspiração na natureza, ou no problema em específico), de modo a criar um processo capaz explorar a vizinhança de uma solução, visando escapar de ótimos locais, são denominados meta-heurísticas (GLOVER; KOCHENBERGER, 2003).

2.3 Meta-heurísticas utilizadas na tese

A seguir descreve-se genericamente as meta-heurísticas que são utilizadas durante o desenvolvimento da tese.

2.3.1 *Greedy Randomized Adaptive Search Procedure*

A meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) foi proposta inicialmente por Feo e Resende (1989) e desde então vem sendo aplicada em vários problemas de otimização combinatória (RESENDE; RIBEIRO, 2003; FESTA; RESENDE, 2009b). O GRASP é uma meta-heurística *multi-start*, sendo que cada iteração é constituída por uma fase de construção, na qual é gerada uma solução viável, e uma fase de busca local, que se aplica uma intensificação na solução construída. A solução é construída a partir de um algoritmo que é parte guloso e parte aleatório. Na busca local, aplica-se um processo iterativo que visa encontrar uma solução localmente ótima. As aplicações repetidas de um procedimento de construção produzem diferentes soluções iniciais para a busca local, diversificando a busca (FESTA; RESENDE, 2009a; FESTA; RESENDE, 2011).

O Algoritmo 1 ilustra o pseudocódigo de implementação genérica do GRASP para um problema de minimização. A entrada para o GRASP inclui parâmetros para definir o tamanho da lista de candidatos (*ListSize*), o número máximo de iterações (*MaxIter*), e a semente para o gerador de números aleatórios (*RandomSeed*). As iterações do GRASP são realizadas nas linhas 3-7. Cada iteração do GRASP consiste na fase de construção (linha 3), fase de busca local (linha 4) e, se necessário, atualização da melhor solução encontrada (linhas 5-7). Dessa forma, após chegar ao limite de número de iterações definido, o algoritmo retorna a melhor solução encontrada (linha 8).

Algoritmo 1: Pseudocódigo genérico do GRASP para um problema de minimização

entrada : $ListSize$, $MaxIter$, $RandomSeed$
saída : A melhor solução encontrada x^*

```
1 MIN ← ∞
2 para iter ← 1 até MaxIter faça
3     x ← Constroi_Solucao(ListSize, RandomSeed)
4     x ← Busca_Local(x)
5     se x é viável e f(x) < MIN então
6         x* ← x
7         MIN ← f(x)
8 retorna x*
```

Na fase de construção, a solução é construída elemento a elemento. Inicialmente, o elemento está em uma lista de candidatos (LC). Através de um parâmetro é criada uma lista restrita de candidatos (LRC) que possui os melhores elementos de LC. A LRC pode ser determinada de diferentes formas:

- Baseada em cardinalidade: seleciona-se os p elementos com o menor custo. Para isso, é necessário definir, geralmente de forma empírica, o valor do parâmetro p.
- Baseada em qualidade: considerando c_{min} e c_{max} como o menor e o maior custo dos elementos, insere-se na LRC os elementos e cujo custo seja: $c_{min} \leq c(e) \leq +\alpha \times (c_{max} - c_{min})$. É necessário a definição do parâmetro α, que regula o quão aleatório ou guloso o algoritmo se comportará.
- Baseada em probabilidade: seleciona-se os elementos aleatoriamente. Em geral, usa-se uma distribuição uniforme.

A segunda fase do GRASP, a fase de busca local, consiste em refinar a solução gerada na fase de construção, aplicando um método de busca local. Isso corresponde a uma intensificação na solução encontrada. explorando regiões vizinhas através da busca local para encontrar um ótimo local.

Algumas aplicações desse método são apresentadas em Festa e Resende (2009b), Bissoli e Amaral (2015) e Bissoli e Amaral (2016).

2.3.2 *Simulated Annealing*

A meta-heurística *Simulated Annealing* (SA) é um método de busca local que aceita movimentos de piora para escapar de ótimos locais. O SA foi proposto originalmente por Kirkpatrick, Gelatt e Vecchi (1983) e baseia-se em uma analogia com a termodinâmica ao simular o resfriamento de um conjunto de átomos aquecidos. Essa técnica inicia sua busca a partir de uma dada solução inicial. O procedimento principal consiste em um laço que gera aleatoriamente, em cada iteração, um único vizinho s' da solução atual s.

Apresenta-se no Algoritmo 2 o pseudocódigo genérico do SA. Os seguintes parâmetros são recebidos pelo algoritmo SA: uma solução inicial s construída, a temperatura inicial T_0, a temperatura de congelamento T_c, o número máximo de iterações SA_{max}, uma taxa de resfriamento β e um tempo limite de execução T_l. Estes parâmetros geralmente são calibrados empiricamente, sendo que aplicar uma temperatura inicial muito alta faz com que um espaço muito grande de soluções seja visitado, aumentando o tempo computacional e dificultando a busca. Por outro lado, uma temperatura inicial muito baixa tende a uma convergência prematura.

Algoritmo 2: Pseudocódigo genérico *Simulated Annealing*.

entrada : T_0, T_f, β, SA_{max}, Solução s
saída : Solução s^*

```
1  s* ← s;
2  T ← T0;
3  enquanto T > Tf faça
4      iter ← 0 ;
5      enquanto iter < SAmax faça
6          iter ← iter + 1 ;
7          s' ← N(s);
8          se f(s') < f(s) então
9              s ← s';
10             se f(s') < f(s*) então
11                 s* ← s';
12         senão
13             s ← s', com probabilidade e^(-(f(s')-f(s))/T);
14     T ← T × β;
15 retorna s*
```

Aplica-se SA_{max} iterações em s (linhas 6-13) para cada temperatura, que é decrementada de acordo com o valor β (linha 14). A cada iteração, realiza-se uma perturbação

na solução atual, gerando s' (linha 7). Em seguida, a solução atual é avaliada (linha 8). Caso s' seja melhor que a solução global, atualiza-se s^* (linhas 9-11). Caso contrário, avalia-se, considerando uma dada probabilidade, se a busca continua a partir de s' ou em s (linhas 12-13).

A temperatura T inicialmente possui um valor elevado T_0. Conforme a temperatura decrementa-se, aproximando do valor T_f, a probabilidade de aceitação de soluções piores pelo SA também é diminuída, intensificando a busca. Algumas aplicações desse método são apresentadas em Bissoli, Altoé e Amaral (2017), Ahonen, Alvarenga e Amaral (2014), Ribeiro, Mauri e Lorena (2011), Mauri (2008), Koulamas, Antony e Jaen (1994), Laarhoven e Aarts (1987) e Kirkpatrick, Gelatt e Vecchi (1983).

2.3.3 *Iterated Local Search*

A meta-heurística *Iterated Local Search* (ILS) foi proposta por Lourenço, Martin e Stützle (2002). O ILS considera que gerar novas soluções iniciais por meio de aplicação de perturbações na solução ótima local possibilita uma melhor exploração da vizinhança pela busca local.

O Algoritmo 3 apresenta o pseudocódigo genérico do ILS, composto por quatro módulos: o procedimento *SoluçãoInicial* (linha 1), que objetiva gerar uma solução inicial qualquer; a *BuscaLocal* (linhas 2 e 5), que visa explorar a solução em busca de um ótimo local; *Pertubação* (linha 4), que consiste em pertubar a solução atual s, diversificando a busca, e o critério de *Aceitação* (linha 6), que decide em qual solução a próxima pertubação será aplicada.

Algoritmo 3: Pseudocódigo genérico do ILS.

entrada : ILS_{max}
saída : Solução s

```
1 s ← SolucaoInicial();
2 s ← BuscaLocal(s);
3 para i ← 1 até ILS_max faça
4 |   s_i ← Pertubacao(s);
5 |   s_i ← BuscaLocal(s_i);
6 |   s ← Aceitacao (s, s_i);
7 fim
```

Como parâmetro a ser calibrado, o ILS apresenta o número máximo de iterações ILS_{max}. Algumas aplicações do ILS podem ser encontradas em Subramanian e Battarra (2013), Penna, Subramanian e Ochi (2013), Subramanian (2012), Gendreau e Potvin (2010), Lourenço, Martin e Stützle (2002) e Besten, Stützle e Dorigo (2001).

2.3.4 *Clustering Search*

A primeira versão da meta-heurística *Clustering Search* (CS) surgiu em Oliveira (2004), que desenvolveu o *Evolutionary Clustering Search* (ECS) para solucionar problemas de minimização de funções numéricas sem restrições e ao *Gate Matrix Layout Problem*.

O ECS tem como base um algoritmo evolutivo, no qual em seu processo de busca, verifica se há uma maior concentração de indivíduos nas quais localizam-se os indivíduos de melhor qualidade. Dessa forma, aplica-se uma etapa de agrupamento iterativamente com um algoritmo evolutivo com o objetivo de identificar grupos de indivíduos com uma certa similaridade. O objetivo consiste em detectar regiões promissoras, baseado na quantidade de soluções geradas em cada região da busca. Nessas regiões promissoras, aplica-se um processo de intensificação, isto é, uma busca local (OLIVEIRA, 2004; CHAVES, 2009).

Posteriormente, o *Clustering Search* passou por uma generalização, isto é, foi definido como uma meta-heurística híbrida que une uma meta-heurística geradora de soluções (e não especificamente um algoritmo evolutivo, como o ECS) e um processo de agrupamento (*clustering*), que pode ser aplicada a qualquer problema de otimização combinatória (CHAVES, 2009; OLIVEIRA; CHAVES; LORENA, 2013).

O CS visa dividir o espaço de busca e identificar regiões promissoras, subdividindo o espaço de busca em *clusters*. Um *cluster* geralmente é caracterizado como $C = (c, v, r)$. O centro c_i é uma solução que representa o *cluster* C_i. O volume v_i consiste na quantidade de soluções contidas no *cluster* C_i. Um parâmetro λ define um número limitante de volume para que, ao ser atingindo, um *cluster* seja considerado promissor. O atributo r_i representa a ineficiência do *cluster*, armazenando o número de vezes consecutivas que a busca local foi aplicada em C_i sem melhora na solução, sendo limitado pelo parâmetro r_{max}.

A etapa de agrupamento de soluções (*clustering*) necessita, de alguma forma, avaliar a similaridade de duas soluções, tais como distância de Manhattan, distância de Hamming, distância de Levenshtein, entre outras (CHAVES, 2009).

O CS é uma meta-heurística híbrida que une três componentes principais: uma meta-heurística geradora de soluções, um processo de agrupamento e um método de busca local. A cada iteração do CS, uma solução s_k é gerada pela meta-heurística e, em seguida, inicia-se o processo de agrupamento. A partir de um método previamente definido, avalia-se qual *cluster* possui seu centro mais similar à s_k. Ao inserir a solução s_k em um determinado *cluster*, este deve ter seu centro atualizado. Se o volume do *cluster* atingir o valor do parâmetro λ, este é definido como promissor e então aplica-se o processo de busca local. Caso a busca local não proporcione melhora na solução, incrementa-se o valor r daquele *cluster*, até atingir o limitante r_{max} ou obter melhora. Ao r_{max} ser alcançado, uma perturbação é aplicada no centro deste *cluster*, com o objetivo de diversificar a busca. Caso contrário, isto é, a busca local encontre uma solução melhor, o respectivo

valor de r é zerado, e atualiza-se o centro do *cluster*. Assim, ao encerrar a etapa de agrupamento, o processo retorna para a meta-heurística, que irá gerar uma nova solução. Tal procedimento é repetido até que um critério de parada seja atingido. Apresenta-se na Figura 2 o fluxograma da meta-heurística CS.

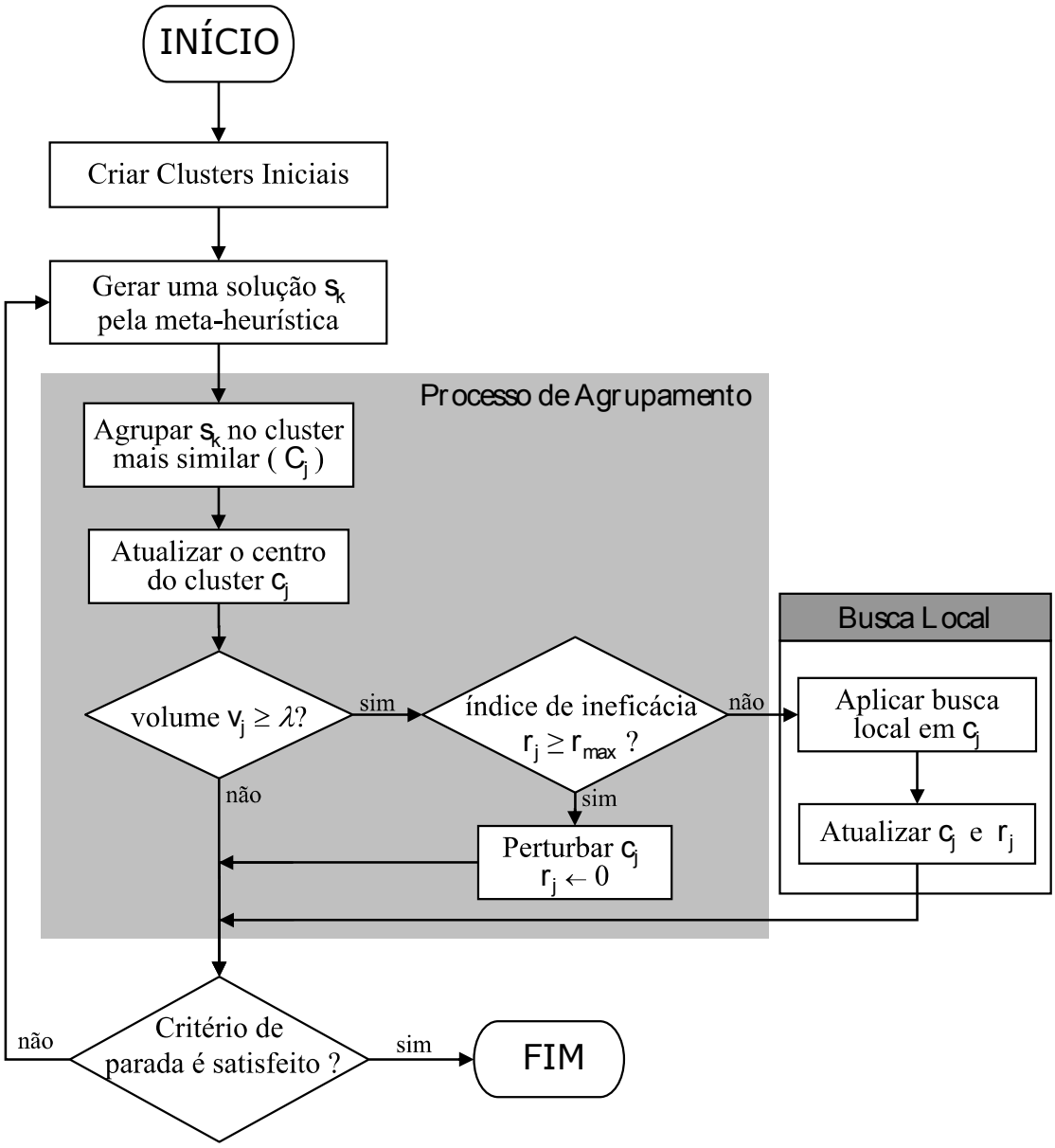

Figura 2 – Fluxograma da meta-heurística CS.
Fonte: Chaves (2009).

3 Problema de Balanceamento de Linhas de Montagem

Na produção industrial de grandes volumes de produtos padronizados, ou mesmo na produção de pequenos volumes personalizados, os sistemas de produção baseados em um fluxo direcionado são chamados *linhas de montagem*.

Uma linha de montagem consiste em uma determinada quantidade de estações de trabalho dispostas ao longo do equipamento de manuseio de materiais (por exemplo, uma esteira rolante) onde as peças são movidas consecutivamente de uma estação de trabalho para a próxima, até chegarem ao fim da linha (ver Figura 3). Na linha de montagem, um número específico de tarefas é executado para obter um produto acabado, com cada estação de trabalho executando um subconjunto dessas tarefas. Uma tarefa requer uma estação de trabalho específica e um certo tempo de processamento. Devido às condições tecnológicas e organizacionais, as restrições de precedência entre as tarefas devem ser respeitadas. Como várias tarefas são executadas simultaneamente, é necessário distribuir a carga entre as estações de trabalho de forma mais equânime possível, de modo que as estações possuam aproximadamente a mesma quantidade de tempo para a execução da tarefa atribuída. Desta forma, os problemas que visam otimizar processos relacionados à fabricação de produtos através de linhas de montagem são conhecidos como *problemas de balanceamento de linha de montagem* (*Assembly Line Balancing Problems* - ALBP) (BECKER; SCHOLL, 2006).

A prioridade entre tarefas de montagem pode ser vista por meio de um *grafo de precedência*. Neste grafo, cada nó representa uma tarefa, cada peso do nó representa o tempo de processamento da tarefa, e cada arco representa uma restrição de precedência.

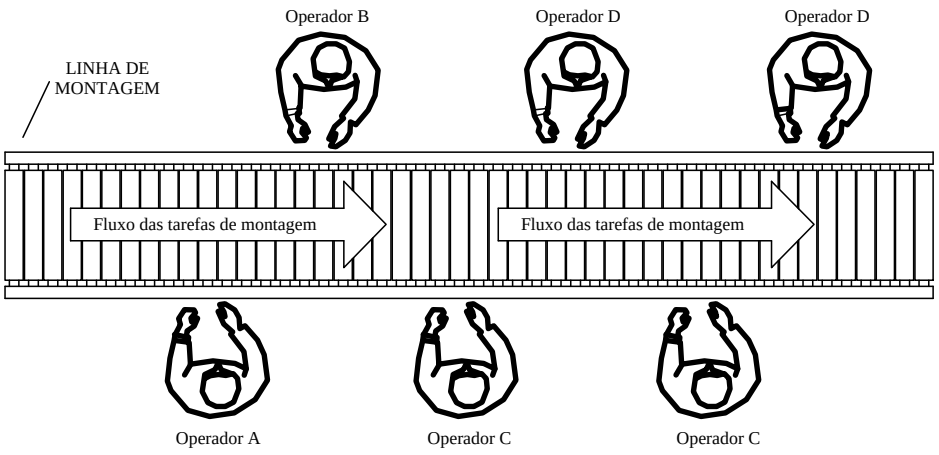

Figura 3 – Fluxo de operações em uma linha de montagem.

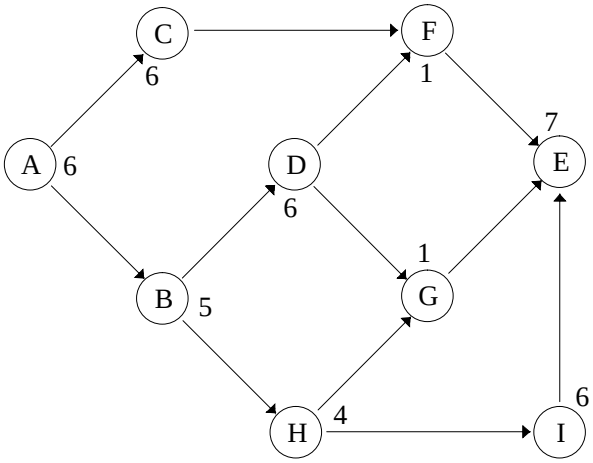

Figura 4 – Grafo de precedência.

A Figura 4 ilustra um grafo de precedência com 9 tarefas com tempos de processamento variando entre 1 e 7 unidades de tempo. Em relação às restrições de precedência, veja, por exemplo, que a Tarefa H, para ser processada, exige que Tarefas B (predecessor direto) e A (predecessor indireto) sejam processadas anteriormente. Por outro lado, a Tarefa H deve ser concluída antes que seus sucessores diretos (G e I) e sucessor indireto (E) possam ser iniciados.

O presente estudo considera o problema de balanceamento de linha de montagem simples na sua versão número dois (SALBP-2), no qual o objetivo é minimizar o tempo de ciclo, para um número fixo de estações de trabalho (BECKER; SCHOLL, 2006).

Este estudo está organizado da seguinte forma: Na seção 3.1 é dada uma visão geral do problema de balanceamento da linha de montagem. Posteriormente, aborda-se a literatura relacionada para o SALBP-2, e sua formulação matemática, respectivamente nas Seções 3.2 e 3.3. Em seguida (Seção 3.4), um algoritmo GRASP é proposto para o SALBP-2. A Seção 3.5 apresenta os resultados computacionais, seguidos pela Seção Considerações finais (3.6).

3.1 Características do problema de balanceamento da linha de montagem

Uma série de artigos de revisão sobre o problema de balanceamento de linha são encontrados na literatura (por exemplo, Baybars (1986), Ghosh e Gagnon (1989), Erel e Sarin (1998), Becker e Scholl (2006), Scholl e Becker (2006), Tasan e Tunali (2008), Saif et al. (2014), Sivasankaran e Shahabudeen (2014)). O problema de balanceamento da linha de montagem (ALBP) é frequentemente classificado de acordo com suas várias restrições e objetivos diferentes, como mostrado na Figura 5.

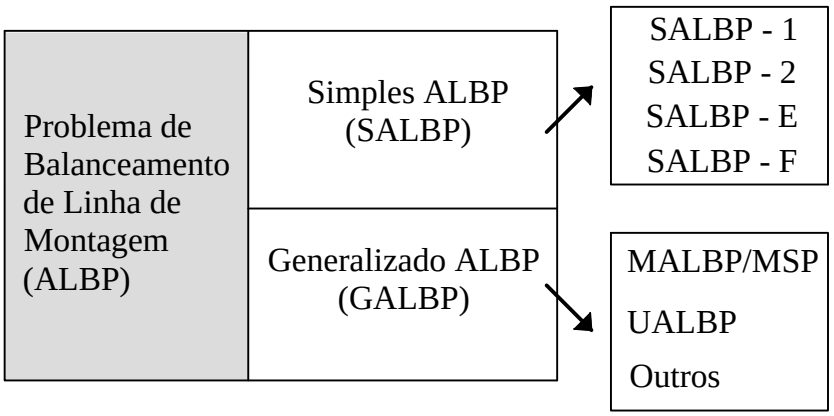

Figura 5 – Classificação dos problemas de balanceamento da linha de montagem.

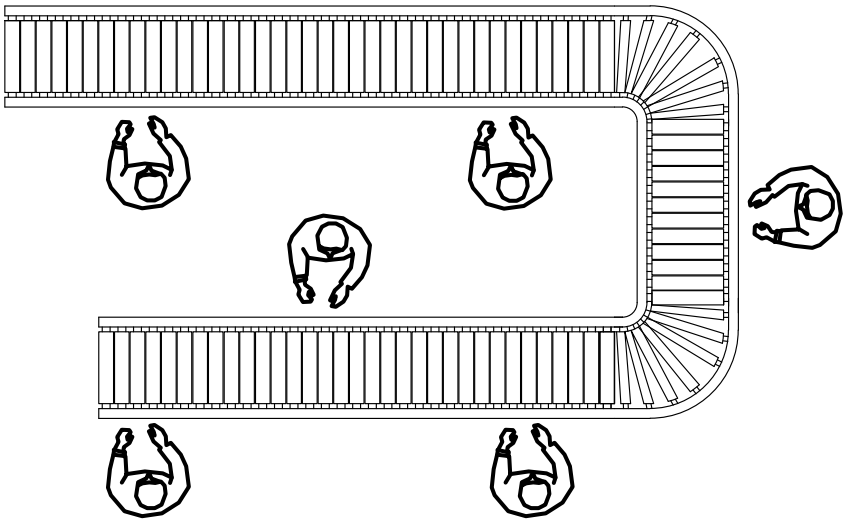

Figura 6 – Linha de montagem em formato U.

De acordo com a Figura 5, o ALBP é dividido em duas classes: Problema de balanceamento de linha de montagem simples (*Simple Assembly Line Balancing Problem* - SALBP) e problema de balanceamento de linha de montagem generalizado (*Generalized Assembly Line Balancing Problem* - GALBP). O GALBP inclui, por exemplo, linhas de montagem com layout em U (UALBP), que é ilustrado na Figura 6 (WEN-CHYUAN; KOUVELIS; URBAN, 2007; FATTAHI; TURKAY, 2015; OGAN; AZIZOGLU, 2015), ou com estações de trabalho paralelas (MALBP/MSP) (OZBAKIR et al., 2011; KUCUKKOC; ZHANG, 2015; ÇIL et al., 2017), ou outras variações (BAYKASOGLU, 2006; HWANG; KATAYAMA, 2010; OTTO; SCHOLL, 2011; FATTAHI; ROSHANI; ROSHANI, 2011; YAGMAHAN, 2011; ZACHARIA; NEARCHOU, 2012; HAMTA et al., 2013; BAUTISTA; BATALLA-GARCÍA; ALFARO-POZO, 2016; TRIKI; MELLOULI; MASMOUDI, 2017). O SALBP consiste em atribuir um conjunto de tarefas a um conjunto de estações, a fim de minimizar o número de estações ou o tempo de ciclo da linha.

Grande parte dos estudos sobre balanceamento de linha de montagem foca-se em modelar e resolver variações do SALBP (BOWMAN, 1960; EREL; SARIN, 1998; LAPI-

ERRE; RUIZ; SORIANO, 2006; PEETERS; DEGRAEVE, 2006; BAUTISTA; PEREIRA, 2009; YU; YIN, 2010; PETROPOULOS; NEARCHOU, 2011; NEARCHOU, 2011; PAPE, 2015; PITAKASO, 2015). O SALBP apresenta as seguintes características principais (BECKER; SCHOLL, 2006; SCHOLL; BECKER, 2006):

- produção em massa de um produto homogêneo;
- ritmo da linha com tempo de ciclo fixo c;
- tempos de operações t_j determinísticos (e integrais);
- não há restrições de atribuição além das restrições de precedência;
- *layout* em linha com m estações em apenas um lado;
- todas as estações são equipadas com a mesma quantidade de máquinas e trabalhadores;
- maximizar a eficiência da linha $E = t_{sum}/(m \times c)$ com tempo total de tarefa sendo $t_{sum} = \sum_{j=1}^{n} t_j$.

O SALBP é relevante para a simples montagem de produtos em linha reta no qual apenas as restrições de precedência entre as tarefas são consideradas. São definidos quatro tipos dessa classificação (REITER, 1969; BAYBARS, 1986; BECKER; SCHOLL, 2006):

- SALBP-1: consiste na atribuição de tarefas aos postos de trabalho de tal modo que o número de estações (m) é minimizado para uma dada taxa de produção (tempo de ciclo fixo, c), o que é equivalente a minimizar o tempo ocioso total;
- SALBP-2: busca minimizar o tempo de ciclo (maximizar a taxa de produção), sujeito a restrições de precedência para um determinado número (m) de estações disponíveis;
- SALBP-E: é a versão mais geral do problema, que visa maximizar a eficiência da linha (E), e, simultaneamente minimizar c e m, considerando suas inter-relações;
- SALBP-F é um problema de viabilidade que se baseia em estabelecer se há ou não um balanceamento de linha viável para uma dada combinação entre c e m.

3.2 Literatura relacionada para o SALBP-2

Para o SALBP-2 existem abordagens baseadas em métodos exatos e/ou heurísticos. No que diz respeito aos métodos exatos, até a presente data, o trabalho que obteve os melhores resultados para o SALBP-2 foi o de Klein e Scholl (1996). Os autores propuseram

um procedimento *Branch and Bound* denominado SALOME-2, que usa uma nova técnica de enumeração, que eles chamaram de método do limite inferior local, complementado por uma série de regras de limite e dominância.

Em relação aos métodos heurísticos, Scholl e Voß (1997) propuseram um algoritmo Busca Tabu (TS) para SALBP-1 e SALBP-2. Kim, Kim e Kim (1996) aplicaram um algoritmo genético chamado pGA para resolver cinco variações do SALBP, sendo uma delas, multi-objetivo. Uğurdağ, Rachamadugu e Papachristou (1997) desenvolveram um método heurístico em duas etapas baseado em uma formulação de programação inteira do SALBP-2. Gonçalves e Almeida (2002) implementaram duas versões de algoritmos genéticos, que eles definiram como rGA_prb e rGA_rks. Liu, Ong e Huang (2003) consideraram o SALBP-2 e implementaram algoritmos que primeiro geram uma solução inicial por um procedimento de atribuição bidirecional e então melhoram a solução inicial obtida realizando trocas de tarefas entre estações de trabalho.

Duas heurísticas baseadas no método de evolução diferencial (*Differential Evolution* - DE), nomeadas *random-keys* (DE_rks) e *codificação baseada em prioridade* (DE_prb) foram apresentadas por Nearchou (2007) para o SALBP-2. Os resultados expostos foram superiores aos de algoritmos evolutivos previamente publicados.

Boysen e Fliedner (2008) elaboraram uma heurística de dois estágios baseada em um algoritmo de gráfico e otimização de colônias de formigas para resolver todas as versões do SALBP. No entanto, esse método genérico apresenta desempenho inferior quando comparado a algoritmos especializados.

Kilincci (2010) realizou uma abordagem em que o SALBP-2 é modelado com uma heurística baseada em Petri net (PN) usando rede de precedência. Neste caso, as tarefas disponíveis são determinadas e, em seguida, uma tarefa é atribuída a uma estação de trabalho usando as prioridades do PN. Diferentes versões das heurísticas propostas são obtidas utilizando os procedimentos *forward* (PNA-for), *back* (PNA-back) e bidireccional (PNA-bid). As heurísticas propostas obtiveram bons resultados quando comparadas com a literatura, especialmente para grandes linhas de montagem.

Mais recentemente, Zhang et al. (2016) desenvolveram variações do método DE, denominando-o como *Integer Coded Differential Evolution Algorithm* (IDEA) para a solucionar o SALBP-2. O IDEA apresentado pode lidar diretamente com variáveis inteiras do SALBP-2 em um espaço discreto. Para melhorar o desempenho do IDEA, algumas mudanças são feitas no esquema de mutação e no operador de seleção DE básico. Dois esquemas de mutação diferentes para o IDEA, chamados de DE/rand/2/bin (IDEA_rd2) e *self-adaptive double scheme* (IDEA_apt) foram implementados. O segundo apresentou um desempenho superior ao resolver instâncias maiores do SALBP-2.

3.3 Formulação matemática do SALBP-2

O SALBP-2 pode ser formulado matematicamente como se segue. Uma instância (T, G, m) consiste em três componentes. $T = \{1, ..., n\}$ é um conjunto de n tarefas. Cada tarefa $i \in T$ tem o tempo de processamento pré-definido como $t_i > 0$. Sem perda de generalidade, os tempos de processamento são, daqui em diante, considerados valores inteiros. Além disso, é dado um grafo de precedência acíclico orientado $G = (T, A)$, sendo T seu conjunto de nós e A seu conjunto de arcos. Finalmente, m representa um número pré-definido de estações de trabalho que estão ordenadas de 1 a m. Um arco $a_{i,j} \in A$ indica que a tarefa i deve ser processada antes da tarefa j. Dada uma tarefa $j \in T$, $P_j \subset T$ denota o conjunto de tarefas que devem ser processadas antes de j. Uma solução viável é obtida através da atribuição de cada tarefa a exatamente uma estação de trabalho, de modo que as restrições de precedência entre as tarefas sejam satisfeitas.

Seja x_{is} uma variável binária definida como 1 se e somente se a tarefa $i \in T$ é atribuída à estação de trabalho $1 \leq s \leq m$. Dessa forma, o SALBP-2 pode ser expresso como um problema de programação inteira (PI) da seguinte maneira (BAYBARS, 1986):

$$\text{Minimizar } z \tag{3.1}$$

$$\text{Sujeito a: } \sum_{s=1}^{m} x_{is} = 1, \qquad \forall i \in T \tag{3.2}$$

$$x_{is} \leq \sum_{s'=1}^{s} x_{js'}, \qquad \forall i \in T, s = 1, ..., m, j \in P_i \tag{3.3}$$

$$\sum_{i \in T} t_i x_{is} \leq z, \qquad s = 1, ..., m \tag{3.4}$$

$$x_{is} \in \{0, 1\}, \qquad \forall i \in T, s = 1, ..., m \tag{3.5}$$

$$z > 0 \tag{3.6}$$

A função objetivo (3.1) minimiza o tempo de ciclo $z > 0$. A restrição (3.2) garante que cada tarefa $i \in T$ seja atribuída a uma única estação de trabalho $1 \leq s \leq m$. A restrição (3.3) reflete os relacionamentos de precedência entre as tarefas. Mais especificamente, se a tarefa $i \in T$ é atribuída à estação de trabalho $1 \leq s \leq m$, todas as tarefas $j \in P_i$ devem ser atribuídas às estações de trabalho $1 \leq s' \leq m$, sendo $s' \leq s$. A restrição (3.4) garante que a soma dos tempos de processamento das tarefas atribuídas às estações de trabalho $1 \leq s \leq m$ não excedam o tempo de ciclo z.

Considerando que as versões do SALBP são *NP-Hard* (SCHOLL, 1999), o presente trabalho realiza uma abordagem heurística para o problema. Tal abordagem consiste na implementação de um algoritmo baseado na metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP), que ainda não foi aplicada ao problema. O funcionamento genérico da GRASP é descrito na Seção 3.4.

3.4 Algoritmo GRASP para o SALBP-2

Conforme descrição genérica realizada na Seção 2.3.1, o GRASP possui característica de possuir duas fases: Construção e Busca Local. A seguir aborda-se a implementação dessas duas etapas especificamente para solucionar o SALBP-2.

3.4.1 Fase de contrução

A solução inicial é obtida a partir de uma heurística construtiva (veja Algoritmo 4) que, a cada iteração, insere um novo elemento (tarefa) na solução atual. A fase de construção da solução usa um parâmetro de entrada α, $(0.0 \leq \alpha \leq 1.0)$, que representa o percentual de aleatoriedade envolvido na escolha de elementos a serem inseridos na solução parcial em construção. Outras entradas são: tempos das tarefas $(t_i)_{i \in T}$, o dígrafo de precedência G e o número de estações de trabalho m. Na linha 1, o algoritmo de construção calcula um limite inferior LB do valor da solução como o máximo entre $\frac{\sum_{i \in T} t_i}{m}$ e $\max_{i \in T} t_i$. A primeira expressão representa um valor uniforme da distribuição das tarefas e a segunda representa o custo da tarefa que tem o valor mais alto. Na linha 2 inicializa-se a solução $(x_{is})_{i \in T,\, s \in S}$ para um vetor com valores iguais a zero. A carga L_s de cada estação de trabalho s consiste na soma dos horários das tarefas atribuídas a s. Inicializa-se na linha 3 a carga L_s de cada estação de trabalho s com o valor zero. Na linha 4 copia-se o grafo de precedência G para $\bar{G}$. Na linha 5, s é apontada para a primeira estação de trabalho. No laço *while* (linhas 6-14), a solução é construída. A linha 7 define CL como a lista de tarefas que estão em $\bar{G}$, mas não tem predecessores em $\bar{G}$ com CL ordenados de modo que as tarefas com tempos menores se apresentem primeiro. Assim, a lista CL contém os possíveis candidatos a serem inseridos, em uma determinada iteração, na solução. A aleatoriedade é usada se houver mais de uma tarefa disponível para ser selecionada na lista ordenada CL. A linha 8 define a chamada Lista Restrita de Candidatos (LRC), que contém as primeiras $\lceil(\alpha \times |CL|)\rceil$ tarefas de CL. Quanto mais perto de zero os valores de α, mais restrita é a LRC, o que leva a uma baixa diversidade de soluções construídas. No entanto, para a escolha de valores de α mais perto de 1, ocorre um comportamento mais aleatório, o que é importante para a diversificação de soluções construídas, mas, por outro lado, muitas soluções tendem a apresentar qualidade inferior, tornando lento o processo de busca local. Na linha 9, uma tarefa i é selecionada aleatoriamente na LRC. Nas linhas 10-13, a tarefa i é atribuída à estação de trabalho atual s $(s < m)$ se, com essa atribuição, a carga L_s da estação de trabalho não exceder o limite inferior LB. Caso contrário, a tarefa i é atribuída à próxima estação de trabalho $s + 1$. Quando a estação de trabalho atual é a última $(s = m)$, todas as tarefas restantes são atribuídas à esta estação de trabalho. Na linha 14, uma tarefa que foi atribuída a uma estação de trabalho é excluída de $\bar{G}$. Eventualmente, $\bar{G}$ estará vazio, o que significa que todas as tarefas foram atribuídas a alguma estação de trabalho. Este é o critério de parada na linha 6 para o laço

while. O algoritmo retorna uma solução $(x_{is})_{i\in T,\, s\in S}$.

Algoritmo 4: Pseudocódigo da construção do GRASP para o SALBP-2

entrada : tempos das tarefas $(t_i)_{i\in T}$, grafo de precedência G, número de estações de trabalho m, parâmetro α
saída : solução construída $(x_{is})_{i\in T,\, s\in S}$, carga da estação de trabalho $(L_s)_{s\in S}$

$LB \leftarrow max\ \{\ \frac{\sum_{i\in T} t_i}{m},\ \underset{i\in T}{max} t_i\}$
$(x_{is})_{i\in T,\, s\in S} \leftarrow (0,\ 0, ..., 0)$
$(L_s)_{s\in S} \leftarrow (0,\ 0, ..., 0)$
Grafo de precedência $\bar{G} \leftarrow G$
$s \leftarrow 1$
enquanto $\bar{G} \neq \emptyset$ **faça**
    $CL \leftarrow$ lista ordenada de tarefas que estão em $\bar{G}$ mas não tem predecessores em $\bar{G}$ (tarefas com tempos menores)
    $LRC \leftarrow$ lista das $\lceil(\alpha \times |CL|)\rceil$ melhores tarefas de CL
    Selecione aleatoriamente uma tarefa i da LRC
    se ($L_s + t_i > LB$) ***e*** $(s < m)$ **então**
        $s \leftarrow s + 1$
    $x_{is} \leftarrow 1$
    $L_s \leftarrow L_s + t_i$
    Exclua a tarefa i de $\bar{G}$
retorna $(x_{is})_{i\in T,\, s\in S}$, $(L_s)_{s\in S}$

3.4.2 Busca local

Depois de construída uma solução inicial viável $(x_{is})_{i\in T,\, s\in S}$, o conjunto de tarefas que foram atribuídas a cada estação de trabalho é conhecido. Então, o processo de exploração da vizinhança é iniciado pela busca local, como mostrado no Algoritmo 5. Seja s^* a estação de trabalho com a maior carga (isto significa que s^* é a estação de trabalho que determina o tempo de ciclo). Portanto, uma tarefa i é removida de s^* e realiza-se a tentativa de atribuí-la, respeitando a ordem de precedência, para cada estação de trabalho, até obter um menor tempo de ciclo. Nesta etapa, a primeira tentativa que melhora a solução é escolhida. Quando há mais de uma estação de trabalho definindo o tempo de ciclo, é escolhido aquele com o menor número de tarefas. Se o empate persistir, a estação de trabalho com o menor índice é escolhida. A Figura 7 ilustra o processo de exploração da vizinhança. Se a solução foi melhorada, uma nova estação de trabalho s^* é definida e o processo reinicia. Caso contrário, se nenhuma melhoria foi alcançada, o algoritmo é encerrado.

A Figura 8 (a) mostra uma solução viável para um problema com $m = 4$ e nove tarefas, ou seja, $|T| = 9$, enquanto a Figura 8 (b) ilustra a respectiva solução a partir de uma perspectiva real de operação. A solução construída inicialmente tem $z = 13$. A

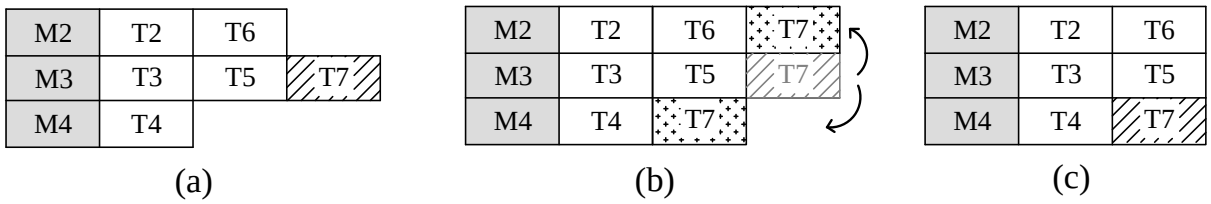

Figura 7 – Exemplo do processo de exploração da vizinhança.

Algoritmo 5: Pseudocódigo da busca local do GRASP para o SALBP-2

entrada : número de estações de trabalho m, número de tarefas n, tempos das tarefas $(t_i)_{i \in T}$, carga das estações de trabalho $(L_s)_{s \in S}$, solução $(x_{is})_{i \in T,\, s \in S}$

saída : solução possivelmente melhorada $(x_{is})_{i \in T,\, s \in S}$, $(L_s)_{s \in S}$

```
1  INÍCIO:
2  Determine a estação de trabalho s* que possua a maior carga, i.e. L_{s*} = max_{1≤s≤m} L_s ;
3  para i ← 1 até n faça
4      se (x_{is*} = 1) então
5          para s ← 1 até m faça
6              se (s ≠ s*) e (L_s + t_i < L_{s*}) e (todos os predecessores da tarefa i estão
                na estação de trabalho {1, ..., s}) então
7                  x_{is*} ← 0;
8                  L_{s*} ← L_{s*} − t_i;
9                  x_{is} ← 1;
10                 L_s ← L_s + t_i;
11                 goto INÍCIO;
12 retorna (x_{is})_{i∈T, s∈S}, (L_s)_{s∈S};
```

aplicação da busca local no grafo de precedência permitiu uma melhoria nesta solução, obtendo o valor de $z = 12$, como mostrado na Figura 9.

3.5 Experimentos computacionais

Os experimentos computacionais são apresentados da seguinte forma: Inicialmente apresenta-se a configuração dos experimentos, em seguida, os resultados dos algoritmos são exibidos que, por fim, são comparados com a literatura.

3.5.1 Configuração dos experimentos

O algoritmo GRASP descrito neste estudo foi implementado na linguagem C e compilado com o compilador GCC. Os testes foram executados em um computador com um processador Intel Core i5-750 com 2,67Ghz, 8 GB de RAM e sistema operacional Windows 7. Para verificar o desempenho do algoritmo, foi obtido um conjunto de instâncias SALBP-2 tradicionais, juntamente com informações sobre suas melhores soluções conhecidas, de

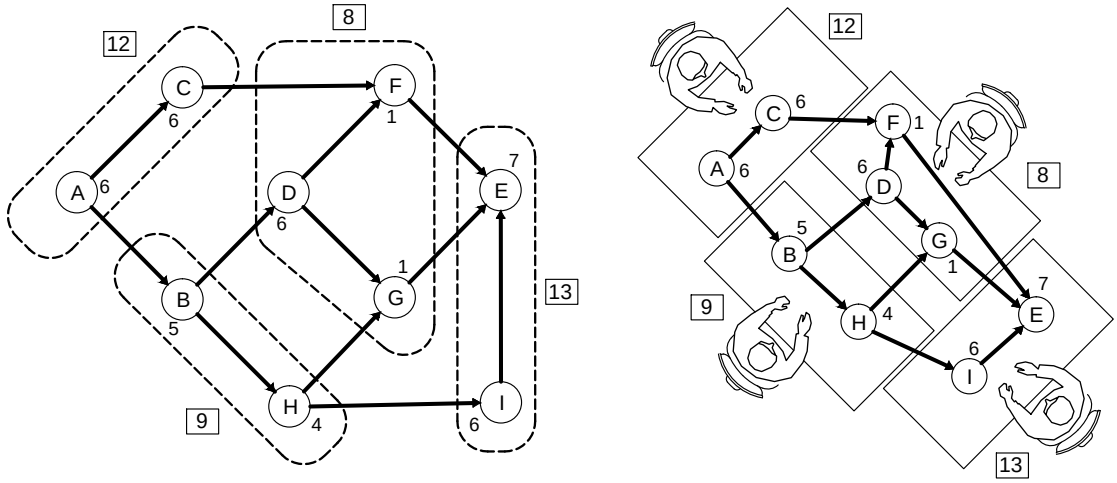

Figura 8 – Solução inicial para o grafo de precedência com $m = 4$ e $z = 13$.

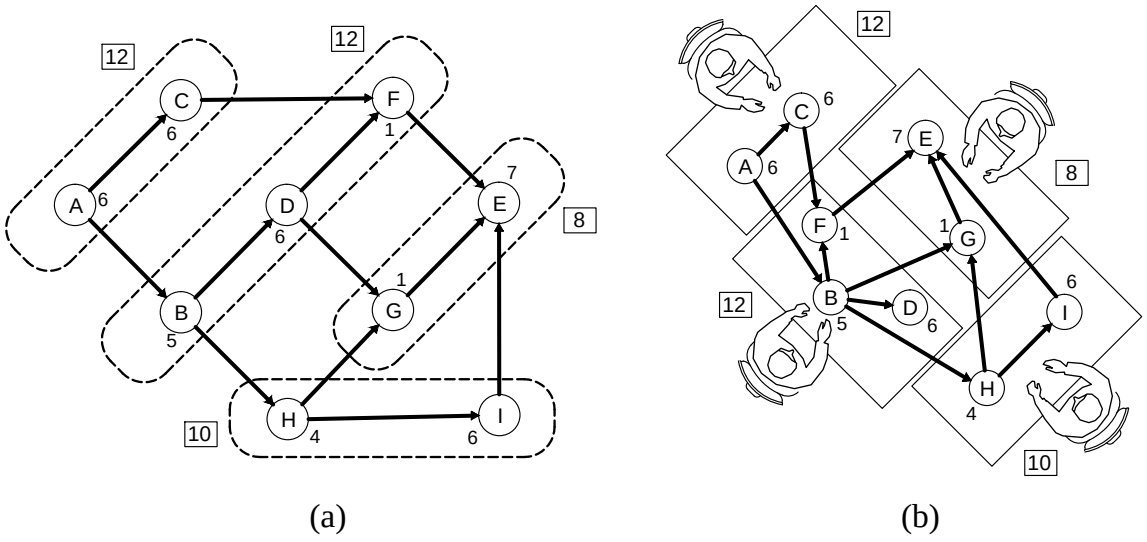

Figura 9 – Solução após aplicação da busca local. Tempo de ciclo passou de $z = 13$ para $z = 12$.

um *site* dedicado a todos os tipos de problemas de balanceamento de linha de montagem, mantidos por Armin Scholl (<http://alb.mansci.de/>). As instâncias são subdivididas em dois grupos chamados *Dataset*1 e *Dataset*2. *Dataset*1 tem 128 instâncias para 9 grafos de precedência, com o número de tarefas variando de 29 a 111, enquanto *Dataset*2 tem 174 instâncias para 8 grafos de precedência, com número de tarefas variando de 53 a 297.

O algoritmo GRASP foi executado 10 vezes para cada instância e armazenou-se o melhor e a média dos valores da função objetivo. Conforme Blum (2011), e dado 180 segundos para o GRASP encontrar uma solução viável. Em caso de sucesso, é dado ao algoritmo mais 180 segundos para buscar um menor tempo de ciclo, isto é, uma solução vizinha que melhore a função objetivo. Em relação à parametrização do algoritmo GRASP, o valor do parâmetro α não configurou-se de maneira fixa. Um intervalo de valores possíveis

para α foi determinado em $\{0,1; 0,2; ...; 0,9\}$ e, portanto, um valor diferente é selecionado com a mesma probabilidade em cada execução da fase de construção. Esta estratégia foi definida com base no estudo realizado em Murphey, Pardalos e Pitsoulis (1998).

Os resultados produzidos pelo algoritmo GRASP são comparados com os resultados apresentados por Zhang et al. (2016) para sete heurísticas: versões do DE's (DE_prb, DE_rks) (NEARCHOU, 2007), algoritmo genético pGA (KIM; KIM; KIM, 1996), algoritmos genéticos híbrido rGA_prb e rGA_rks (GONÇALVES; ALMEIDA, 2002) e com duas versões IDEA ($IDEA\_rd2$ e $IDEA\_apt$) propostas por Zhang et al. (2016).

Adotou-se a mesma análise de desempenho usada por Zhang et al. (2016), o percentual de tempo de ciclo excedente (*percent excess cycle time*): $PECT = ((CT/\overline{CT}) - 1) \times 100$, onde CT é o tempo de ciclo da melhor melhor solução gerada pela heurística e $\overline{CT}$ é o melhor tempo de ciclo conhecido na literatura.

3.5.2 Comparação com a literatura

A Tabela 1 mostra a média do $PECT$ calculado das instâncias de cada grafo de precedência. As duas primeiras colunas fornecem o nome do grafo de precedência e o número de tarefas (n). As próximas sete colunas fornecem o PECT médio apresentado por Zhang et al. (2016) para as sete heurísticas; e a última coluna indica o PECT médio do GRASP.

Para o $Dataset1$, a média geral PECT para os GAs (2,52, 2,51 e 2,99%) e para os DEAs (2,05, 1,85, 2,02 e 2,04%) não são maiores do que 2,02%. No entanto, para o GRASP, a média geral PECT é de 0,69%, o que se mostrou melhor do que os apresentados pelas heurísticas concorrentes. Em particular, é importante destacar o desempenho do GRASP para o conjunto mais complexo de instâncias, "Arcus2": O GRASP apresentou a média PECT em 0,84%, enquanto as outras heurísticas apresentaram, respectivamente, 5,07, 4,78, 6,12, 5,87, 6,15, 4,98 e 4,64%.

Para o $Dataset2$, no menor grafo "Hahn", o GRASP apresenta o PECT médio um pouco pior. No entanto, quando a complexidade das instâncias aumenta, o GRASP apresenta um PECT médio melhor em relação às outras heurísticas. O GRASP apresentou uma valor médio PECT igual a 1,69%, o que é melhor do que os resultados apresentados pelas outras heurísticas: 3,61, 3,31, 4,48, 4,44, 4,99, 3,41 e 2,97%, respectivamente.

Na Tabela 2 são referenciadas a média dos custos computacionais das heurísticas. Não é possível realizar uma comparação precisa em termos de tempos devido a se tratarem de execuções em diferentes máquinas. O GRASP foi executado em um computador de 2,67 GHz e as outras heurísticas em um computador de 2,29 GHz. Entrentanto, apresenta-se uma análise aproximada dos tempos apenas como uma referência. No $Dataset1$, as heurísticas DEA_rks, pGA e rGA_rks alcançaram seu critério de parada mais cedo (em

Tabela 1 – *PECT* (%) média das instâncias de cada grafo de precedência.

Grafo de Precedência	n	DEA_prb	DEA_rks	pGA	rGA_prb	rGA_rks	IDEA_rd2	IDEA_apt	GRASP
Dataset1									
Buxey	29	2,15	2,01	2,91	2,77	2,74	2,21	3,07	**0,27**
Sawyer	30	3,64	2,36	3,24	3,67	4,11	3,34	3,52	**0,67**
Lutz1	32	0,57	**0,32**	0,38	0,54	0,83	0,35	0,71	**0,32**
Gunther	35	1,02	0,21	0,81	1,34	1,61	1,02	1,27	**0**
Kilbridge	45	0,71	0,79	0,87	1,02	1,24	0,60	0,60	**0**
Tonge	70	2,35	2,08	3,51	3,24	3,85	2,07	1,78	**0,66**
Arcus1	83	**0,78**	1,12	1,88	1,10	2,31	0,98	**0,78**	1,08
Lutz2	89	2,15	2,94	2,99	3,01	4,08	2,64	**1,99**	2,38
Arcus2	111	5,07	4,78	6,12	5,87	6,15	4,98	4,64	**0,84**
Média		2,05	1,85	2,52	2,51	2,99	2,02	2,04	**0,69**
Dataset2									
Hahn	53	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0,43
Warnecke	58	4,12	3,52	5,06	4,51	4,97	3,98	3,51	**2,43**
Wee-Mag	75	2,01	1,23	2,98	2,17	2,89	1,63	1,39	**0,89**
Lutz3	89	1,68	1,64	2,77	2,01	3,57	2,88	1,57	**1,44**
Barthold1	148	0,42	0,81	1,02	0,94	1,01	0,46	0,26	**0,05**
Barthold2	148	6,87	5,89	9,08	10,21	10,16	4,79	4,79	**3,98**
Scholl	297	10,14	10,11	10,44	11,21	12,31	10,16	9,24	**2,62**
Média		3,61	3,31	4,48	4,44	4,99	3,41	2,97	**1,69**

média entre 3 e 4 segundos), enquanto o GRASP e as demais heurísticas terminam com um tempo maior, apresentando tempos semelhantes (média entre 23 e 28 segundos).

No *Dataset*2, as heurísticas DEA_rks, pGA e rGA_rks também atingem seu critério de parada mais prematuramente que as outras, mas os tempos de processamento aumentam em relação ao *Dataset*1 num fator de 3,9, 1,3 e 2, respectivamente. O tempo para o rGA_prb aumenta pelo fator de 5 e, para o DEA_prb, IDEA_rd2 e IDEA_apt, incrementa-se por um fator em torno de 11 em relação ao *Dataset*1. O GRASP tem o menor aumento (um fator de 1,2) e, portanto, gera a solução mais rapidamente do que a maioria das heurísticas (DEA_prb, rGA_prb, IDEA_rd2 e IDEA_apt).

Os resultados detalhados são apresentados no Apêndice A.

3.6 Considerações finais

Este estudo apresentou uma implementação do GRASP para o SALBP-2. A fase construtiva do GRASP usa uma Lista Restrita de Candidatos (LRC). O LRC é ordenado topologicamente, e todas as tarefas candidatas têm a mesma probabilidade de serem selecionadas. O tamanho do LRC é regulado pelo parâmetro α, que varia cada vez que uma solução é construída. Em seguida, um procedimento de busca local é aplicado, que consiste em reatribuir tarefas da estação de trabalho que define o tempo de ciclo para outras estações de trabalho de modo a buscar uma solução melhor. Esse processo é repetido até atingir o tempo limite.

A implementação proposta do GRASP foi testada em um grande conjunto de instâncias da literatura. Suas soluções foram comparadas com as melhores soluções conhecidas e seu desempenho com as de sete heurísticas anteriores. Os resultados computacionais mostram que o algoritmo GRASP obtém bons resultados quando comparado à literatura. Além disso, a característica de obter automaticamente valores para o parâmetro α, incorporada no GRASP clássico, tende a facilitar sua aplicação em casos reais do problema.

Tabela 2 – Média dos tempos de execução (em segundos) entre as instâncias de cada grafo de precedência.

Grafo de precedência	n	DEA_prb	DEA_rks	pGA	rGA_prb	rGA_rks	IDEA_rd2	IDEA_apt	GRASP
Dataset1									
Buxey	29	0,56	0,96	1,91	1,24	1,06	0,61	0,59	0,38
Sawyer	30	1,31	1,61	0,85	1,47	1,56	1,06	1,13	0,04
Lutz1	32	0,34	0,81	0,44	1,59	0,84	0,58	0,56	0,13
Gunther	35	0,72	1,08	0,84	1,63	1,24	1,15	1,45	3,98
Kilbridge	45	1,66	1,57	1,12	4,13	1,94	1,38	1,28	0,43
Tonge	70	27,56	3,53	3,84	16,44	6,02	32,87	32,12	25,49
Arcus1	83	46,54	4,84	7,22	36,11	8,45	43,12	44,89	100,36
Lutz2	89	16,75	1,21	6,84	12,13	3,84	20,54	25,12	10,68
Arcus2	111	156,83	18,12	4,82	140,14	11,12	140,12	148,45	102,37
Média 1		28,03	3,75	3,1	23,88	4,01	26,83	28,4	27,10
Dataset2									
Hahn	53	2,75	1,02	0,68	1,27	0,84	4,56	4,15	3,66
Warnecke	58	7,84	3,21	3,02	4,72	1,99	10,25	10,38	25,69
Wee-Mag	75	19,21	3,62	3,79	13,02	6,21	13,89	13,82	6,55
Lutz3	89	52,12	4,89	5,84	30,98	6,85	50,45	51,82	20,59
Barthold1	148	368,45	21,32	3,98	220,74	11,13	324,12	334,84	15,89
Barthold2	148	575,21	29,23	4,64	19,45	16,05	508,78	527,45	32,83
Scholl	297	986,54	38,54	6,08	421,34	14,01	1221,48	1345,12	117,73
Média 2		287,45	14,55	4,00	126,07	8,15	304,79	326,8	31,85
$\frac{Média\ 2}{Média\ 1}$		10,3	3,9	1,3	5,3	2,0	11,4	11,5	1,2

4 Problema *Job Shop Scheduling*

O escalonamento (*scheduling*) baseia-se num processo de tomada de decisão que desempenha um importante papel em indústrias manufatureiras e de serviços. Trata-se da alocação de operações em máquinas (isto é, realiza o sequenciamento de operações em máquinas ou estações de trabalho) de tal forma que alguns objetivos de desempenho, tais como o tempo de produção, o tempo de fluxo ou o atraso podem ser minimizados. Considerando o atual contexto competitivo do mercado, com produtos com ciclo de vida cada vez menores, produtos personalizados e padrões de demanda em constante mudança, o escalonamento eficiente de produção tornou-se uma questão importante para o crescimento e sobrevivência de empresas de manufatura. Para se sustentarem no atual ambiente competitivo, empresas de fabricação têm de focar na otimização de seus processos produtivos. Um desses processos caracteriza-se como problema *Job Shop Scheduling* (JSP) (SINGH et al., 2016).

Pode-se descrever o JSP clássico da seguinte forma: são dados n *jobs*, cada um composto de várias operações. Esses *jobs* devem ser processados em m máquinas. Cada operação utiliza uma das m máquinas por uma duração fixa de tempo. Cada máquina pode processar no máximo uma operação por vez; e uma vez que uma operação inicie o processamento em uma determinada máquina, esta operação deve ser concluída sem interrupção. As operações que compõe um determinado *job* devem ser processadas em uma determinada sequência, isto é, respeitando uma ordem de precedência. O problema consiste em encontrar um escalonamento das operações nas máquinas, tendo em vista as restrições de precedência, de modo a minimizar o $Cmax$ (*makespan*), isto é, o tempo de finalização da última operação completada no escalonamento.

A formulação matemática do JSP pode ser definida como segue. Dado um conjunto M de máquinas (considere o tamanho de M como $|M|$) e o conjunto J de *jobs* (considere o tamanho de J como $|J|$), seja $\sigma_1^j \prec \sigma_2^j \prec \sigma_{|M|}^j$ o conjunto ordenado das $|M|$ operações do *job* j, onde $\sigma_k^j \prec \sigma_{(k+1)}^j$ indica que a operação $\sigma_{(k+1)}^j$ pode iniciar somente depois de completar a operação σ_k^j. Seja O o conjunto de operações. Cada operação σ_k^j é definida por dois parâmetros: M_k^j é a máquina onde σ_k^j é processada e $p_k^j = p(\sigma_k^j)$ é o tempo de processamento da operação σ_k^j. Definindo $t(\sigma_k^j)$ como o instante de início da k-ésima operação $\sigma_k^j \in O$, uma formulação para o JSP é apresentada a seguir (AIEX; BINATO; RESENDE, 2003):

$$\text{Minimizar } C_{max} \tag{4.1}$$

$$\text{Sujeito a: } C_{max} \geq t(\sigma_k^j) + p(\sigma_k^j), \quad \forall \sigma_k^j \in O, \tag{4.2}$$

$$t(\sigma_k^j) \geq t(\sigma_l^j) + p(\sigma_l^j), \quad \forall \sigma_l^j \prec \sigma_k^j, \tag{4.3}$$

$$t(\sigma_k^j) \geq t(\sigma_l^i) + p(\sigma_l^i) \vee t(\sigma_l^i) \geq t(\sigma_k^j) + p(\sigma_k^j), \quad \forall i, j \in J : M_{\sigma_l^i} = M_{\sigma_k^j}, \tag{4.4}$$

$$t(\sigma_k^j) \geq 0, \quad \forall \sigma_k^j, \in O. \tag{4.5}$$

Uma solução viável para o JSP pode ser gerada a partir da permutação de J em cada uma das máquinas em M, observando as restrições de precedência, a restrição de que a máquina pode processar apenas uma operação por vez, e exigindo que uma vez iniciado, o processamento de uma operação deve ser ininterrupto até a sua conclusão. Cada conjunto de permutações tem um escalonamento correspondente. Assim, o objetivo do JSP é encontrar um conjunto de permutações com o menor valor de *makespan* (C_{max}) (AIEX; BINATO; RESENDE, 2003).

Um exemplo de instância é mostrado na Tabela 3, no qual relaciona-se para cada *job*, o roteiro que deve seguir nas máquinas (Roteiro/Máquina), com seus respectivos custos (Tempo). A Figura 10 ilustra o roteiro de uma instância com três *jobs* e quatro máquinas, conforme apresentado na Tabela 3. Percebe-se na referida figura que os três *jobs* possuem diferentes roteiros de processamento de suas operações nas máquinas.

Tabela 3 – Exemplo de instância JSP com três *jobs* e quatro máquinas.

Job	Roteiro/Máquinas	Tempo
J1	1,4,3,2	25,7,18,15
J2	2,3,1,4	10,30,7,15
J3	4,1,2,3	18,22,10,7

A solução do JSP pode ser representada por um gráfico de Gantt. Na Figura 11 ilustra-se um gráfico de Gantt referente a uma solução da instância descrita na Tabela 3. O gráfico consiste em dispor o conjunto de máquinas no eixo vertical, sendo indicado no eixo horizontal a escala de tempo, definindo uma barra horizontal para cada tempo de processamento de cada operação em cada máquina.

O JSP pode ser modelado graficamente por um grafo disjuntivo conforme ilustrado na Figura 12. Na citada figura, apresentam-se os vértices do grafo, representando cada tempo de processamento e dois tipos de arcos: Conjuntivo e Disjuntivo. O conjunto de arcos conjuntivos são relativos à sequência de operações de um *job*, ou seja, tais arcos representam as restrições de precedência entre as operações de um mesmo *job*. O conjunto de arcos disjuntivos são correspondentes às limitações dos recursos, não possuindo direção, representando o par de operações de diferentes *jobs* a serem executadas na mesma máquina,

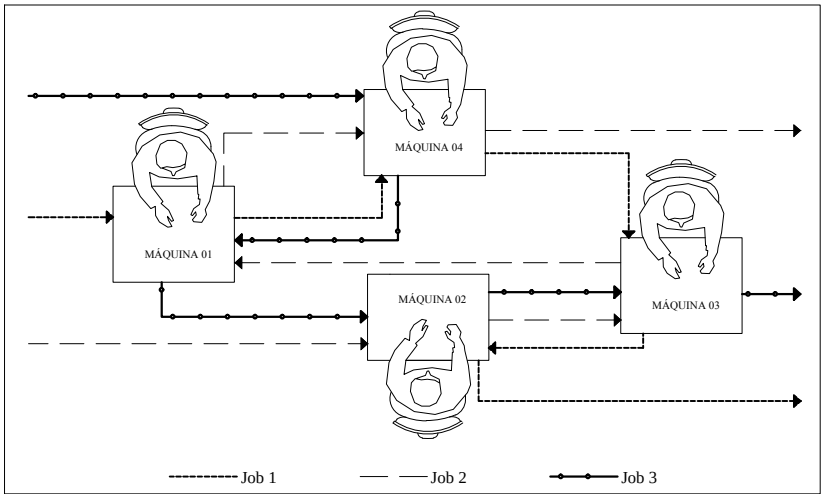

Figura 10 – Representação esquemática de uma instância com três *jobs* e quatro máquinas.

Fonte: Adaptado de Morales (2012).

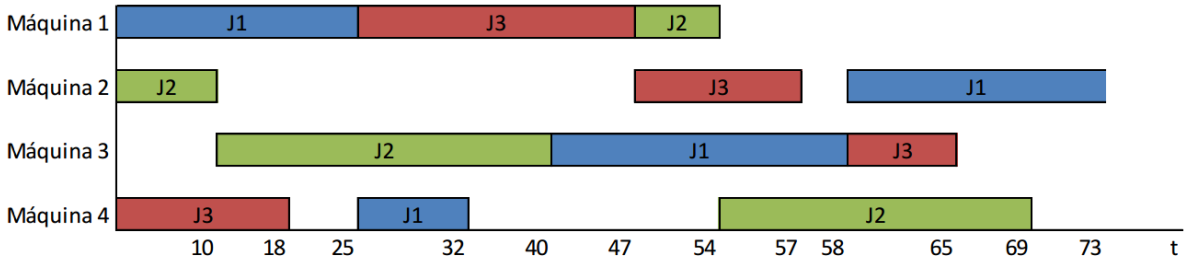

Figura 11 – Representação de uma solução para uma instância JSP com três *jobs* e quatro máquinas.

Fonte: Morales (2012).

sendo que a escolha de uma direção desses arcos estabelecerá a ordem de execução dos *jobs* na mesma máquina.

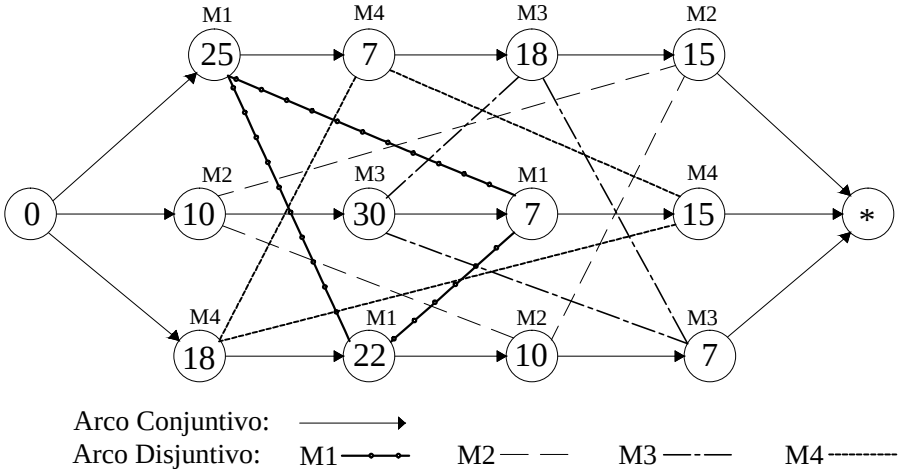

Figura 12 – Grafo disjuntivo: representação de um escalonamento. O objetivo é obter caminho hamiltoniano em cada clique de tarefas relacionada a cada máquina.
Fonte: Adaptado de Morales (2012).

O JSP é *NP-hard* (GAREY; JOHNSON; SETHI, 1976). Laguna e Marti (1999) provaram que, mesmo para instâncias com três máquinas e tempos unitários de processamento, assim como para instâncias com três *jobs*, o JSP já caracteriza-se como *NP-hard*.

Dessa forma, no presente trabalho, implementou-se um algoritmo baseado na metaheurística GRASP. Os experimentos computacionais com um conjunto padrão de instâncias para o problema indicaram que a implementação proposta do GRASP é competitiva para encontrar soluções aproximativas para o JSP.

Este estudo está organizado da seguinte forma. Na próxima seção, apresenta-se a aplicação do algoritmo GRASP para o JSP. Os experimentos computacionais são relatados na Seção 4.3 e na sequência, as considerações finais.

4.1 Literatura Relacionada

A primeira versão do JSP foi apresentada por Johnson (1954) e desde então o problema vem recebendo atenção de muitos pesquisadores. Várias abordagens já foram desenvolvidas para solucioná-lo, como pode ser observado nos trabalhos de revisão de literatura encontrados em Mellor (1966), Cheng, Gen e Tsujimura (1996), Pacheco e Santoro (1999), Jain e Meeran (1999), Cheng, Gen e Tsujimura (1999), Jones, Rabelo e Sharawi (2001), Xie e Allen (2015), Çaliş e Bulkan (2015) e Zhang et al. (2017).

Mais recentemente, Kurdi (2016) apresentou um método denominado *New Island Model Genetic Algorithm* (NIMGA) para solucionar o JSP. Foram propostos novos mecanismos de evolução e de critério de seleção de migração, que foram capazes de melhorar a diversificação da busca e adiar a convergência prematura quando comparado com a versão clássica do algoritmo, chamada de *Island Model Genetic Algorithm* (IMGA). O NIMGA foi submetido a testes em instâncias clássicas para o JSP, e seus resultados se mostraram competitivos quando comparado com a literatura.

4.2 Um algoritmo GRASP para o JSP

O funcionamento genérico do GRASP é descrito na Seção 2.3.1. Na fase de construção, considera-se que uma solução para um problema consista de vários elementos, que são adicionados um por vez. Dessa forma, a cada iteração da fase de construção, no máximo $|J|$ *jobs* podem ser escalonados e, este conjunto de *jobs* candidatos é indicado por O_c. Os elementos ainda não adicionados à solução são ordenados em uma lista de candidatos de acordo com uma função gulosa que mede a melhora que seria obtida no valor da solução pela adição de cada elemento. Considere que a função adaptativa gulosa dada por $h(\tau)$ denote o *makespan* resultante a partir da adição do *job* τ aos *jobs* já escalonados. A escolha do próximo *job* a ser escalonado é dado por:

$$\underline{\tau} = min\,(h(\tau)|\tau \in O_c).$$

Definindo,

$$\overline{\tau} = max\,(h(\tau)|\tau \in O_c),$$

$\underline{h} = h\,(\underline{\tau})$, e $\overline{h} = h\,(\overline{\tau})$, a Lista Restrita de Candidatos (LRC) é definida como:

$$LRC = \Big\{\tau \in O_c|\underline{h} \leq h(\tau) \leq \underline{h} + \alpha(\overline{h} - \underline{h})\Big\},$$

onde o parâmetro α restringe-se a $0.0 \leq \alpha \leq 1.0$.

Para $\alpha = 0$ são geradas soluções totalmente gulosas, e $\alpha = 1$, são geradas soluções totalmente aleatórias. O componente adaptativo da heurística decorre do fato de que os benefícios associados a cada elemento são atualizados a cada iteração da fase de construção para refletir as mudanças trazidas pela seleção dos elementos anteriores. O componente probabilístico do GRASP é caracterizado pela escolha aleatória de um dos melhores candidatos na LRC, mas geralmente não o melhor. Essa forma de escolha permite que diferentes soluções sejam obtidas ao final da fase de construção do GRASP, embora não necessariamente comprometa o componente adaptativo guloso.

Uma solução gerada pelo GRASP na fase construtiva não possui garantia de ser localmente ótima em relação às simples definições de vizinhança. Por este motivo, aplica-se uma busca local para tentar melhorar cada solução gerada na fase construtiva. Um algoritmo de busca local funciona de forma iterativa, substituindo sucessivamente a solução atual por uma solução melhor, em sua vizinhança. Define-se como seu critério de parada o fato de não encontrar nenhuma solução melhor na vizinhança, em relação a alguma função de custo.

4.2.1 Fase de construção

A fase de construção do GRASP é baseada em Binato et al. (2002), e gera uma solução viável conforme descrito na Seção 2.3.1. O detalhamento da implementação da construção do GRASP para solucionar o JSP apresenta-se no Algoritmo 6, que é descrito a seguir. É gerado um escalonamento viável alocando operações individuais, uma por vez, até que todas as operações sejam inseridas no escalonamento (linhas 2 a 8). Dessa forma, de acordo com as restrições de precedência, lista-se os O_c *jobs* que podem ser escalonados na iteração atual (linha 3). Em seguida, constrói-se a LRC (linha 4). A próxima operação a ser agendada é escolhida aleatoriamente da LRC, na qual todos os candidatos possuem a mesma probabilidade de serem selecionados (linha 5). Em seguida, verifica-se o menor tempo possível para inserir a operação selecionada na solução, isto é, o menor tempo, obedecendo as restrições de precedência (linha 6). Então a operação é adicionada em s

(linha 7), o valor do makespan é calculado (linha 8) e a respectiva operação adicionada em s é excluída do conjunto $\bar{O}$ de operações de *jobs* (linha 9).

Algoritmo 6: Pseudocódigo da construção do GRASP para o JSP

entrada : conjunto de máquinas M, conjunto de *jobs* J, conjunto de operações O, α
saída : solução construída s

1 Conjunto de operações $\bar{O} \leftarrow O$
2 **enquanto** $\bar{O} \neq \emptyset$ **faça**
3 $O_c \leftarrow$ lista dos *jobs* que podem ser escalonados
4 $LRC \leftarrow \{\ job\ \tau \in O_c | \underline{h} \leq h(\tau) \leq \underline{h} + \alpha(\overline{h} - \underline{h})\}$
5 Selecione aleatoriamente um *job* τ da LRC
6 Verifica o tempo disponível em s_i para adicionar o *job* τ
7 $s_i \leftarrow s_i$+*job* τ
8 Calcule o *makespan* da solução s
9 Exclua o *job* τ de $\bar{O}$
10 **retorna** s

4.2.2 Busca Local

O pseudocódigo da busca local é apresentado no Algoritmo 7. A geração da vizinhança de soluções e, consequentemente, a lista de possíveis movimentos a serem realizados, são determinadas a partir do caminho crítico (*critical path*), definido como o maior caminho entre o início de processamento dos *jobs* até a conclusão de todos os *jobs* em todas as máquinas. Ao calcular o *makespan*, a partir do caminho crítico, conforme descrito por Taillard (1994), guarda-se o caminho de vértices mais longo do grafo, conforme apresentado na Figura 13 (linha 3). Em todos os pares de operações consecutivas que compartilham a mesma máquina no caminho crítico são realizadas tentativas de trocas (linhas 6-7). Se a troca gerar uma melhora no *makespan*, ela é aceita (linhas 8-9). Caso contrário, a troca é desfeita. Uma vez que uma troca é aceita, o caminho crítico pode mudar e um novo caminho crítico deve ser identificado. Dessa forma, foi adotada a estratégia de escolher o melhor movimento dentre os movimentos possíveis do caminho crítico (melhor melhora), para obter a nova solução.

A solução apresentada no formato de gráfico de Gantt na Figura 11 é representada agora num grafo disjuntivo na Figura 13. Indica-se na Figura 13 as direções definidas de cada arco disjuntivo de cada máquina, assim como o caminho crítico da solução, assinalado com hachura.

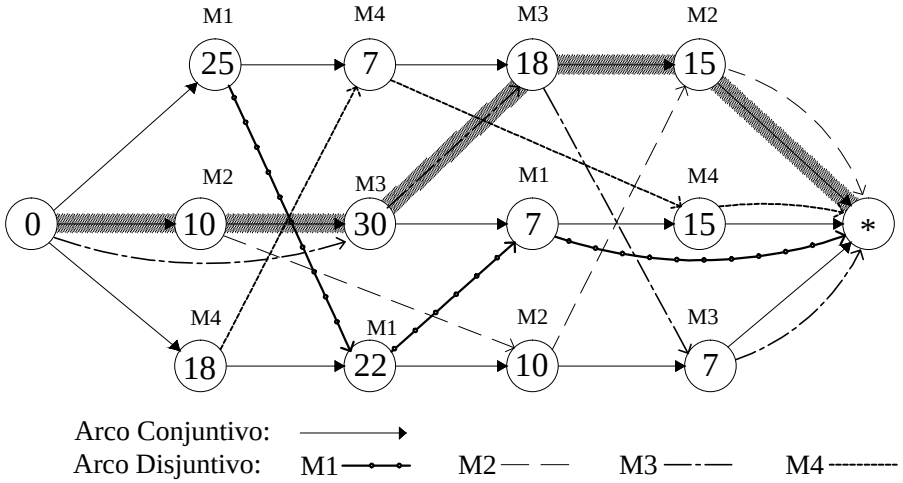

Figura 13 – Grafo disjuntivo: Solução da instância e definição do caminho crítico.
Fonte: Adaptado de Morales (2012).

Algoritmo 7: Pseudocódigo da Busca Local do GRASP para o JSP

entrada : solução s, conjunto de máquinas M, conjunto de *jobs* J, conjunto de operações O
saída : solução possivelmente melhorada s^*

```
1  s* ← s' ← s
2  INÍCIO:
3  Calcule o caminho crítico c ← Taillard(s')
4  Calcule a quantidade de operações o_c do caminho crítico c
5  para i ← 1 até o_c faça
6      Troque a operação o_c[i] em s' com a sua operação à direita, caso houver
7      Reconstrói a solução s' a partir da troca efetuada
8      se f(s') < f(s*) então
9          s* ← s'
10     s' ← s
11 retorna s*
```

4.3 Experimentos Computacionais

Para avaliar a eficácia do algoritmo GRASP descrito neste estudo, consideram-se 43 instâncias tradicionais do JSP: Instâncias FT06, FT10 e FT20 de Fisher e Thompson (1963) e instâncias LA01 a LA40 de Lawrencen (1984). Esse conjunto de instâncias apresenta-se em várias configurações desde 6 a 30 *jobs* e 5 a 15 máquinas.

O algoritmo GRASP foi implementado em linguagem C e compilado com o compilador GCC e os testes foram executados em um computador com processador Intel Core i5-2450M com 2.50Ghz e sistema operacional Windows 8.1 Pro.

Em relação à parametrização do algoritmo GRASP, definiu-se empiricamente o parâmetro α como 0,5 (por apresentar o melhor resultado médio quando considerado instâncias de diferentes tamanhos) e o número de iterações como 10000 (baseado em Aiex, Binato e Resende (2003)). O algoritmo GRASP foi executado dez vezes, para cada instância. Armazenou-se o melhor valor da função objetivo e o respectivo valor médio das execuções.

A parametrização informada foi escolhida por ter apresentado a melhor média no conjunto total de instâncias, sendo que em alguns casos, ao variar o parâmetro α dentro do intervalo $(0.1, 0.2, ..., 0.9)$, em algumas instâncias a melhor solução foi melhor do que a apresentada na Tabela 4, mas consequentemente piorando o desvio entre as execuções.

A Tabela 4 apresenta os resultados da seguinte forma: Coluna I apresenta o nome da instância utilizada; a quantidade de *jobs* e a quantidade de máquinas são representados por J e M, respectivamente; A coluna BKS (*Best Known Solution*) representa a melhor solução conhecida para cada instância, retirada do recente trabalho apresentado por Kurdi (2016); A coluna FO apresenta as melhores soluções encontradas pelo algoritmo GRASP proposto; A média das 10 soluções geradas para cada instância pelo algoritmo GRASP proposto é relacionada na coluna Média FO; O desvio médio entre as dez soluções geradas pelo algoritmo GRASP proposto apresenta-se na coluna D; O Desvio entre FO e BKS apresenta-se na colunas D BKS. O tempo total em segundos referente as dez execuções do GRASP proposto é apresentado na coluna T(s).

A partir da Tabela 4, percebe-se que o GRASP atinge o valor da melhor solução conhecida, ou próximo dela. Considerando a média geral dos resultados gerados nas instâncias consideradas, fica distante apenas $4,1\%$ do BKS. Porém, ressalta-se a simplicidade do algoritmo GRASP proposto nesse estudo e a robustez do mesmo, considerando que o desvio médio entre as soluções geradas a partir das dez execuções para cada instância ficou em menos de 1%, no geral.

A Tabela 5 aponta os resultados da comparação dos resultados do GRASP proposto com os apresentados pelo algoritmo NIMGA (KURDI, 2016). A primeira coluna designa o nome da instância utilizada (I). Em seguida, para cada algoritmo considerado na comparação, discrimina-se a melhor solução encontrada para cada instância (FO) e a média das soluções geradas para cada instância (Média FO). Em relação as médias, o GRASP foi executado 10 vezes para cada instância. O NIMGA foi submetido a diferentes experimentos computacionais, variando a quantidade de vezes que o mesmo foi executado. Para essa comparação, com o objetivo de se obter uma comparação mais igualitária, considerou-se os resultados apresentados a partir de 10 execuções independentes do NIMGA, porém, nesse experimento, os autores não apresentam os tempos computacionais.

Constata-se na Tabela 5 que os algoritmos apresentam, na média, resultados com uma diferença de aproximadamente 3,62% para as melhores soluções encontradas

Tabela 4 – Resultados Computacionais - GRASP para o JSP

I	J	M	BKS	FO	Média FO	D	D BKS	T(s)
ft06	6	6	55	55	55,0	0,00%	0,0%	9,3
ft10	10	10	930	973	982,7	1,00%	4,6%	79,1
ft20	20	5	1165	1196	1207,8	0,99%	2,7%	174,0
la01	10	5	666	666	666,0	0,00%	0,0%	25,6
la02	10	5	655	656	662,7	1,02%	0,2%	26,3
la03	10	5	597	628	633,7	0,91%	5,2%	24,9
la04	10	5	590	598	598,0	0,00%	1,4%	25,9
la05	10	5	593	593	593,0	0,00%	0,0%	26,8
la06	15	5	926	926	926,0	0,00%	0,0%	64,4
la07	15	5	890	890	894,0	0,45%	0,0%	49,4
la08	15	5	863	863	863,0	0,00%	0,0%	64,3
la09	15	5	951	951	951,0	0,00%	0,0%	60,1
la10	15	5	958	958	958,0	0,00%	0,0%	60,1
la11	20	5	1222	1222	1222,0	0,00%	0,0%	120,4
la12	20	5	1039	1039	1039,0	0,00%	0,0%	128,5
la13	20	5	1150	1150	1150,0	0,00%	0,0%	133,7
la14	20	5	1292	1292	1292,0	0,00%	0,0%	99,4
la15	20	5	1207	1210	1224,0	1,16%	0,2%	111,0
la16	10	10	945	979	979,8	0,08%	3,6%	57,1
la17	10	10	784	795	804,0	1,13%	1,4%	55,6
la18	10	10	848	861	874,2	1,53%	1,5%	54,9
la19	10	10	842	856	863,8	0,91%	1,7%	56,0
la20	10	10	902	914	919,9	0,65%	1,3%	60,7
la21	15	10	1046	1133	1154,1	1,86%	8,3%	172,0
la22	15	10	927	1020	1032,2	1,20%	10,0%	170,5
la23	15	10	1032	1057	1068,8	1,12%	2,4%	152,3
la24	15	10	935	1029	1040,3	1,10%	10,1%	152,6
la25	15	10	977	1064	1089,2	2,37%	8,9%	170,9
la26	20	10	1218	1348	1374,1	1,94%	10,7%	321,6
la27	20	10	1235	1375	1394,3	1,40%	11,3%	337,1
la28	20	10	1216	1330	1365,4	2,66%	9,4%	423,8
la29	20	10	1152	1366	1382,5	1,21%	18,6%	646,8
la30	20	10	1355	1417	1443,8	1,89%	4,6%	340,6
la31	30	10	1784	1822	1837,1	0,83%	2,1%	1004,5
la32	30	10	1850	1911	1944,7	1,76%	3,3%	999,8
la33	30	10	1719	1747	1786,7	2,27%	1,6%	828,7
la34	30	10	1721	1827	1841,3	0,78%	6,2%	947,9
la35	30	10	1888	1915	1951,2	1,89%	1,4%	782,0
la36	15	15	1268	1398	1412,3	1,02%	10,3%	261,0
la37	15	15	1397	1503	1517,2	0,94%	7,6%	266,6
la38	15	15	1196	1303	1325,3	1,71%	8,9%	273,4
la39	15	15	1233	1329	1349,8	1,57%	7,8%	242,0
la40	15	15	1222	1309	1319,1	0,77%	7,1%	289,1
Média			**1080,0**	**1127,3**	**1139,3**	**0,9%**	**4,1%**	**240,7**

pelos algoritmos, e aproximadamente 3,85% para as médias das 10 execuções realizadas pelos respectivos métodos. Em relação ao custo computacional, o NIMGA mostra uma convergência mais rápida, porém ressalta-se que ambos os algoritmos foram executados em computadores com velocidades diferentes de processamento, logo não pode-se efetuar uma comparação direta.

4.4 Considerações finais

Este estudo apresenta uma implementação do GRASP para o JSP clássico. A fase construtiva gera uma solução viável, um elemento por vez, sendo que a próxima operação a ser agendada é escolhida aleatoriamente da LRC, na qual todos os candidatos possuem a mesma probabilidade de serem selecionados. Por conseguinte, é aplicada uma busca local, que consiste em calcular o *makespan* e, ao obter o caminho mais longo do grafo disjuntivo, que representa o escalonamento de maior custo, escolhe o melhor movimento dentre os possíveis, para obter a nova solução.

O algoritmo GRASP proposto foi testado em um conjunto de 43 instâncias padrões e comparado com os melhores valores conhecidos (KURDI, 2016). Os resultados computacionais mostram que o algoritmo GRASP atinge os mesmos valores das melhores soluções conhecidas (BKS) ou próximos nas instâncias testadas. De maneira geral, o algoritmo GRASP produz resultados com desvio relativo médio de $4,1\%$ em relação ao BKS.

Tabela 5 – Comparação dos resultados computacionais do GRASP com o NIMGA.

	GRASP		**NIMGA**	
I	**FO**	**Média FO**	**FO**	**Média FO**
ft06	55	55,0	55	55,0
ft10	973	982,7	930	952,1
ft20	1196	1207,8	1173	1187,4
la01	666	666,0	666	666,0
la02	656	662,7	655	656,1
la03	628	633,7	597	598,2
la04	598	598,0	590	590,0
la05	593	593,0	593	593,0
la06	926	926,0	926	926,0
la07	890	894,0	890	890,0
la08	863	863,0	863	863,0
la09	951	951,0	951	951,0
la10	958	958,0	958	958,0
la11	1222	1222,0	1222	1222,0
la12	1039	1039,0	1039	1039,0
la13	1150	1150,0	1150	1150,0
la14	1292	1292,0	1292	1292,0
la15	1210	1224,0	1207	1207,0
la16	979	979,8	946	953,2
la17	795	804,0	784	784,5
la18	861	874,2	848	854,5
la19	856	863,8	842	845,9
la20	914	919,9	907	909,3
la21	1133	1154,1	1058	1089,7
la22	1020	1032,2	937	952,4
la23	1057	1068,8	1032	1032,0
la24	1029	1040,3	947	964,8
la25	1064	1089,2	992	1012,9
la26	1348	1374,1	1218	1241,1
la27	1375	1394,3	1269	1293,9
la28	1330	1365,4	1253	1262,1
la29	1366	1382,5	1247	1267,5
la30	1417	1443,8	1355	1358,5
la31	1822	1837,1	1784	1784,0
la32	1911	1944,7	1850	1850,0
la33	1747	1786,7	1719	1719,0
la34	1827	1841,3	1721	1723,9
la35	1915	1951,2	1888	1889,0
la36	1398	1412,3	1293	1314,1
la37	1503	1517,2	1439	1464,0
la38	1303	1325,3	1222	1261,7
la39	1329	1349,8	1259	1280,7
la40	1309	1319,1	1246	1266,5
Média	1127,30	1139,28	1088,67	1097,00

5 Problema *Job Shop Scheduling* Flexível

Quando consideram-se casos reais, o JSP possui a limitação de possuir uma sequência fixa de operações para cada *job*. Surge então uma extensão denominada problema *Flexible Job Shop Scheduling* (FJSP). Nesta perspectiva, é possível que uma operação de um dado *job* possa ser processada em máquinas alternativas, isto é, um mesmo *job* possa vir a ter variações no sequenciamento da sua produção (HO; TAY, 2004).

A Figura 14 ilustra um exemplo de ambiente produtivo baseado no FJSP, onde as setas, equivalentes aos arcos disjuntivos, indicam os caminhos possíveis dos *jobs* através das máquinas, ilustrando as alternativas de processamento das operações. A atribuição de cada operação a uma máquina alternativa implica em encontrar, para cada *job*, um caminho que inicie à esquerda da figura e termine à sua direita. Esses caminhos se traduzem em arcos direcionados de um grafo, que ao serem selecionados, de acordo com a flexibilidade, definem o percurso do *job* entre suas operações. Para se obter uma solução do FJSP, além de definir as atribuições, ainda é necessário determinar as sequências das operações a serem processadas em cada máquina. Segue a descrição do ambiente representado na Figura 14, que visa esclarecer como os *jobs* do FJSP podem percorrer caminhos alternativos através das máquinas:

- O job 1 tem 4 operações e sua primeira operação pode ser processada alternativamente nas máquinas 1 ou 2. A segunda operação do job 1 deve ser processada na máquina 1. Dessa forma, caso a primeira operação seja processada também na máquina 1, ocorre recirculação. A terceira operação do job 1 deve ser executada na máquina 3 e a última operação, na máquina 4.

- O job 2 possui 3 operações, sendo que somente a segunda delas tem alternativas de processamento, as máquinas 3 ou 4.

- O job 3 também possui 3 operações e sua segunda operação pode ser processada tanto na máquina 2 quanto na máquina 3.

Considerando que o FJSP é um dos problemas de escalonamento NP-completo mais difíceis de resolver, a aplicação de métodos exatos se torna limitada (KACEM; HAMMADI; BORNE, 2002). Dessa forma, no presente trabalho implementou-se quatro meta-heurísticas com a função objetivo de minimizar o *makespan*: um algoritmo GRASP; um algoritmo *Simulated Annealing* (SA); um algoritmo híbrido baseado na meta-heurística *Iterated Local Search* (ILS) que utiliza o SA como método de busca local; e, uma meta-heurística híbrida *Clustering Search* (CS). Os experimentos computacionais com um conjunto padrão de

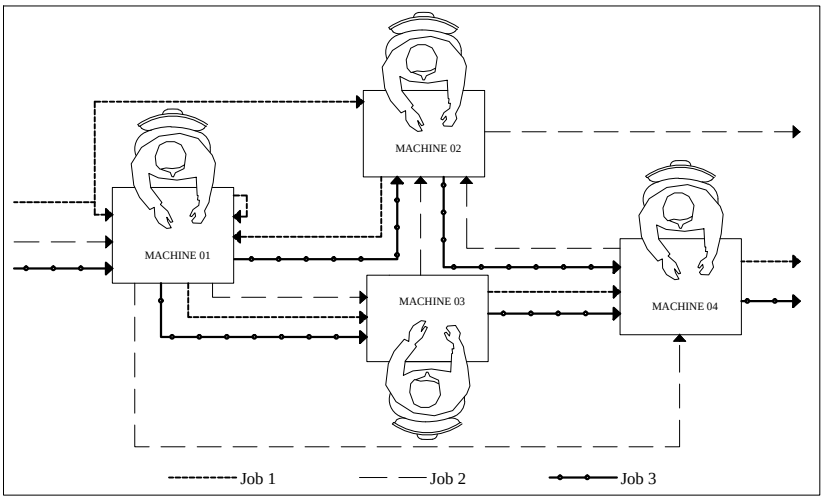

Figura 14 – Representação de um escalonamento FJSP. O objetivo é obter caminho hamiltoniano em cada clique de *jobs* relacionada a cada máquina.

instâncias para o problema indicaram que as implementações propostas são eficientes para fornecer soluções de qualidade para o FJSP.

Este estudo está organizado da seguinte forma. A próxima seção apresenta uma breve revisão a respeito do FJSP. Em seguida, aborda-se a metodologia utilizada, demonstrando a modelagem do problema e as meta-heurísticas propostas para solucioná-lo. Os experimentos computacionais são relatados na Seção 5.4 e na sequência, as considerações finais.

5.1 Literatura Relacionada

O FJSP foi proposto por Brucker e Schlie (1990), que desenvolveram um algoritmo que soluciona em tempo polinomial uma instância com duas máquinas e, desde então, várias abordagens foram realizadas para solucionar essa classe de problema (CINAR; TOPCU; OLIVEIRA, 2015; CHAUDHRY; KHAN, 2016). Diferentes formulações matemáticas são encontradas na literatura para o FJSP. Demir e İşleyen (2013) avaliaram modelos existentes e, considerando os resultados obtidos, os autores recomendam a utilização da formulação matemática baseada em programação inteira mista (PIM) proposta por Özgüven, Özbakir e Yavuz (2010), que é baseada no modelo introduzido por Manne (1960). Birgin et al. (2014) desenvolveram um novo modelo PIM para o problema, produzindo melhores resultados quando comparado a Özgüven, Özbakir e Yavuz (2010).

Em relação a métodos utilizados para solucionar o FJSP, grande parte dos trabalhos realizaram abordagens baseadas em heurísticas e/ou meta-heurísticas (CHAUDHRY; KHAN, 2016; CINAR; TOPCU; OLIVEIRA, 2015). Considerando métodos exatos, Fattahi, Mehrabad e Jolai (2007) aplicaram um algoritmo *branch-and-bound* para o FJSP e compararam com duas meta-heurísticas baseadas em TS e SA. Mais recentemente, Previero (2016) empregou duas estratégias para resolver o FJSP. A primeira consiste na aplicação

de um algoritmo exato *Branch-and-Cut* (B&C) e uma variação proposta que agrega inequações válidas baseada em Applegate e Cook (1991) a um modelo proposto por Birgin et al. (2014). A segunda abordagem baseia-se em duas meta-heurísticas: *Local Branching* (LB) e *Diversification, refining e tight-refining* (DRT). Os resultados foram comparados com Birgin et al. (2014) e com o resolvedor Gurobi (GUROBI, 2016). De uma forma geral, as abordagens baseadas em meta-heurísticas tiveram um melhor desempenho em relação aos métodos exatos, sendo que na análise realizada, apenas em 3 de um total de 59 instâncias testadas não obtiveram os melhores resultados.

Considerando métodos heurísticos, Brandimarte (1993) propôs abordagens hierárquicas, nas quais as atribuições das operações dos *jobs* às máquinas (roteamento) e a programação das operações (*scheduling*) foram estudadas separadamente. No entanto, a maioria dos trabalhos na literatura considera os dois subproblemas simultaneamente, como o de Hurink, Jurisch e Thole (1994) e Dauzère-Pérès e Paulli (1997), que aplicaram a meta-heurística *Tabu Search* (TS) para o FJSP, sendo que no segundo é proposto um novo procedimento de re-atribuições das operações.

Mastrolilli e Gambardella (2000) propuseram duas estruturas de vizinhança para melhorar o algoritmo TS proposto por Dauzère-Pérès e Paulli (1997) e desde então, vários métodos vem sendo aplicados ao FJSP (CINAR; TOPCU; OLIVEIRA, 2015; CHAUDHRY; KHAN, 2016). Mais recentemente, Cruz-Chávez, Martínez-Rangel e Cruz-Rosales (2015) apresentaram um algoritmo baseado na meta-heurística *Simulated Annealing* (SA), com um mecanismo de escalonamento parcial e outro de resfriamento, que é função do desvio padrão. Constatou-se que o SA com o mecanismo proposto, denominado *SA-Partial Controlled* (SA-PC), converge mais rapidamente para boas soluções do que o SA que não implementa o mecanismo proposto.

Uma meta-heurística que combina *Scatter Search* e *Path Relinking* com TS (SSPR) foi desenvolvida por González, Vela e Varela (2015) para o FJSP. Os autores buscaram combinar a característica de diversificação provida pelo *Scatter Search* e *Path Relinking*, com a natureza de intensificação do TS. Tal método foi comparado com a SA (MASTROLILLI; GAMBARDELLA, 2000); AG Híbrido (AGH) e *Variable Neighborhood Descent* (VND) (GAO; SUN; GEN, 2008); *Hybrid Harmony Search* (HHS) e *Large Neighborhood Search* (LNS)(YUAN; XU, 2013); *Discrepancy search* (HMIDA et al., 2010); Algorítmo Memético (GONZÁLEZ; VELA; VARELA, 2013); *Parallel Tabu Search Based Meta*^{2}*heuristics* (TSBM^{2}M) (BOZEJKO; UCHROŃSKI; WODECKI, 2010); e AGH (GUTIÉRREZ; GARCÍA-MAGARIÑO, 2011). O SSPR obteve os melhores valores conhecidos em 58 de 178 instâncias do FJSP consideradas.

Ishikawa, Kubota e Horio (2015) implementaram um método evolutivo híbrido denominado *Hierarchical Multi-Space Competitive Distributed Genetic Algorithm* (HmcDGA). O método foi comparado com Chen, Ihlow e Lehmann (1999), Pezzella, Morganti e Cias-

chetti (2008), Ida e Oka (2011), Al-Hinai e ElMekkawy (2011) e Teekeng e Thammano (2012) e seus resultados apresentaram-se competitivos, porém com um elevado custo computacional.

Gao et al. (2016) propuseram um algoritmo *Discrete Harmony Search* (DHS) para o FJSP com múltiplos objetivos. Para avaliar seus resultados, os autores consideraram o objetivo minimizar *makespan* e compararam com TS (BRANDIMARTE, 1993), Algoritmo Genético (AG) (PEZZELLA; MORGANTI; CIASCHETTI, 2008), *Efficient Search Method* (XING; CHEN; YANG, 2009a), *simulation modeling* (XING; CHEN; YANG, 2009b) e uma versão híbrida da TS (LI; PAN; LIANG, 2010).

Um algoritmo *Quantum Particle Swarm Optimization* (QPSO) foi proposto por Singh e Mahapatra (2016) para resolver o FJSP. O operador de mutação, uma técnica que se origina a partir do algoritmo genético, é usado como um recurso para ajudar a evitar a convergência prematura e diversificar a solução. Outra ferramenta usada pelo QPSO para fornecer maior diversidade e reduzir o custo computacional na busca é o uso de números caóticos (mapa logístico) em vez de números aleatórios. O QPSO se mostrou competitivo quando comparado com TS (MASTROLILLI; GAMBARDELLA, 2000), AG (PEZZELLA; MORGANTI; CIASCHETTI, 2008), AGH e VND (GAO; SUN; GEN, 2008), algoritmo *Variable Neighborhood Search* (VNS) paralelo (YAZDANI; AMIRI; ZANDIEH, 2010), *Knowledge-Based Ant Colony Optimization* (XING et al., 2010) e *effective* GA (eGA) (ZHANG; GAO; SHI, 2011).

Li et al. (2017) desenvolveram para o FJSP um algoritmo híbrido *Artificial Bee Colony* (HABC) com base nos algoritmos *Artificial Bee Colony* (ABC) e TS. Os recursos método de roleta e operador de cruzamento para abelhas empregadas foram implementados com o objetivo de melhorar a inicialização da população e exploração, respectivamente. Na média, os resultados apresentados pelo HABC foram superiores aos resultados apresentados pelo algoritmo *Artificial Immune System* (AIS) (BAGHERI et al., 2010), TS (SAIDI-MEHRABAD; FATTAHI, 2007), GA com TS (ZHANG; MANIER; MANIER, 2012), AIS e *Particle Swarm Optimization* (PSO)(SADRZADEH, 2013), *Biogeography-Based Optimization* (WANG et al., 2013) e algoritmo híbrido ABC (WANG; DUAN, 2014).

Uma variação do DHS, denominada *Effective Operations Permutation-Based Discrete Harmony Search* (EOPDHS) foi aplicada ao FJSP por Gaham, Bouzouia e Achour (2017). Os autores utilizaram um operador de busca complementar, chamado de *Modified Intelligent Mutation*, a fim de equilibrar probabilisticamente a carga de trabalho máxima das máquinas durante o processo geral de pesquisa. Os resultados foram comparados com os apresentados por Wang et al. (2012), Yazdani, Amiri e Zandieh (2010), N. e Kobti (2012), Pezzella, Morganti e Ciaschetti (2008), Li et al. (2011) e Rahmati e Zandieh (2012) e indicam que o algoritmo proposto é efetivo para a resolução do FJSP.

5.2 Modelagem do Problema

O FJSP é uma extensão do JSP, que permite que uma operação seja processada por qualquer máquina a partir de um determinado conjunto de máquinas alternativas (MASTROLILLI; GAMBARDELLA, 2000; KACEM; HAMMADI; BORNE, 2002; HO; TAY, 2004; CHAUDHRY; KHAN, 2016). Um FJSP geral pode ser formulado da seguinte forma (BIRGIN et al., 2014): Seja (V, A) um grafo orientado acíclico. Os vértices deste grafo representam as *operações* e os arcos representam as *restrições de precedência*. É dado um conjunto M de *máquinas* e uma função F que associa o conjunto não-vazio $F(v)$ de M a cada operação v. As máquinas em $F(v)$ são as únicas que podem processar a operação v. Adicionalmente, para cada operação v e cada máquina k em $F(v)$, há um número racional positivo $p_{v,k}$ representando o *tempo de processamento* da operação v na máquina k.

A *designação de máquinas* é uma função f que atribui uma máquina $f(v) \in F(v)$ para cada operação v. Dado a designação da máquina f, então $p_v^F := p_{v,f(v)}$.

Para cada máquina k, considere V_k como o conjunto de operações que podem ser processadas pela máquina k, isto é, $V_k = \{v \in V : k \in F(v)\}$. De modo a limitar que v e w sejam processados simultaneamente na máquina k, define-se B_k como o conjunto de todos os pares ordenados (v, w) dos elementos distintos de V_k. Considere B como a união de todos B_k. Logo, $(v, w) \in B$ se e somente se $v \neq w$ e $F(v) \cap F(w) \neq \emptyset$.

Dada uma designação de máquina f, considere B^f como o conjunto de todos os pares ordenados de operações distintas a serem processadas na mesma máquina, isto é, $B^f = \{(v, w) \in B : f(v) = f(w)\}$. Uma *seleção* consiste em qualquer subconjunto Y de B^f, de modo que, para cada (v, w) em B^f, exatamente um de (v, w) e (w, v) está em Y. Uma seleção corresponde a um pedido das operações para serem processadas na mesma máquina. Uma seleção Y é *admissível* se $(V, A \cup Y)$ for um grafo orientado acíclico, ou seja, se não conflitar com as restrições de precedência atribuídas, nem com ela própria.

Dada uma designação de máquinas f e uma seleção admissível Y, um *escalonamento* para $(V, A \cup Y, p^f)$ é uma função s de V para o conjunto de números racionais não-negativos tal que $s_v + p_v^f \leq s_w$ para cada (v, w) em $(A \cup Y)$, isto é, há um escalonamento desde que $(V, A \cup Y)$ seja um grafo orientado acíclico. O número s_v é o tempo de início da operação v. O *makespan* do escalonamento s é o número $mks(s) := max_{v \in V}(s_v + p_v^f)$.

O *comprimento* de um caminho $(v_1, v_2, ..., v_l, v_{l+1})$ no grafo orientado acíclico $(V, A \cup Y)$ é o número $p_{v1}^f + p_{v2}^f + ... + p_{vl}^f$. Para cada caminho P em $(V, A \cup Y)$ terminando em v e qualquer escalonamento s, o tamanho de P é no máximo s_v. Para cada v em V, seja s_v^* como o máximo de comprimento de todos os caminhos em $(V, A \cup Y)$ terminando em v. A função s^* pode ser definida como *escalonamento apertado* para $(V, A \cup Y, p^f)$. A partir do escalonamento apertado s^*, que é determinado por caminhos compridos, pode-se encontrar o *caminho crítico* $P = (v_1, v_2, ..., v_l, v_l + 1)$ em $(V, A \cup Y)$, tal que o comprimento

de P mais p_{v_l+1} é igual a $mks(s^*)$. Dessa forma, percebe-se que o escalonamento apertado possui o *makespan* mínimo dentre os demais escalonamentos para $(V, A \cup Y, p^f)$.

O *makespan* de uma seleção admissível Y para uma dada designação de máquina f, denotada por $mks(Y)$, é o *makespan* do escalonamento apertado para $(V, A \cup Y, p^f)$. O FJSP pode ser representado como: Encontre uma atribuição de máquina f e uma seleção admissível Y tal que $mks(Y)$ seja mínimo. As seguintes suposições são consideradas em um FJSP:

1. Todas as máquinas estão disponíveis no instante $t = 0$.
2. Todos os *jobs* estão disponíveis no instante $t = 0$.
3. Cada operação deve ser processada por apenas uma máquina de cada vez.
4. Não existem restrições de precedência entre as operações de diferentes *jobs*; Por isso os *jobs* são independentes uns dos outros.
5. Uma operação iniciada não pode ser interrompida.
6. Tempo de transporte entre *jobs* e as máquinas e tempo para configurar a máquina para o processamento de uma determinada operação estão incluídos no tempo de processamento.

O JSP limita-se ao sequenciamento de operações em máquinas fixas, enquanto que no FJSP a atribuição de uma operação não é pré-fixada e pode assim ser processada num conjunto de máquinas alternativas. Portanto, o FJSP não se restringe ao sequenciamento, extendendo-se na atribuição de operações para as máquinas adequadas (roteamento). O FJSP é, portanto, mais complexo do que o JSP, pois considera a determinação da atribuição da máquina para cada operação. Dessa forma, o escalonamento dos *jobs* no FJSP pode ser categorizado em dois subproblemas (CHAUDHRY; KHAN, 2016):

1. Um subproblema de roteamento no qual tem-se que selecionar uma máquina adequada entre as disponíveis para processar cada operação.
2. Um subproblema de escalonamento no qual as operações atribuídas são sequenciadas em todas as máquinas selecionadas para obter uma programação viável que minimiza um objetivo predefinido.

Baseado na flexibilidade, Kacem, Hammadi e Borne (2002) classificaram o FJSP nos seguintes subproblemas:

1. FJSP total: cada operação pode ser processada em qualquer uma das m máquinas disponíveis.

2. FJSP parcial: parte das operações só podem ser processadas em algumas das m máquinas disponíveis.

A Figura 15 ilustra um grafo disjuntivo de um FJSP com 3 *jobs* e 4 máquinas, sendo uma abstração do ambiente produtivo apresentado na Figura 14. Os vértices 1 a 10 representam as operações. Os vértices 0 e 11 correspondem, respectivamente, a operações artificiais de início e fim do escalonamento. Os vértices são conectados por arcos conjuntivos e disjuntivos. Arcos conjuntivos ligam consecutivamente as operações de cada *job*, indicando a ordem de precedência a ser respeitada. Os arcos que não possuem orientação definida são os chamados arcos disjuntivos, que ligam operações a serem executadas numa mesma máquina. Percebe-se que alguns vértices possuem incidência de arcos disjuntivos simultaneamente em diferentes máquinas, caracterizando flexibilidade.

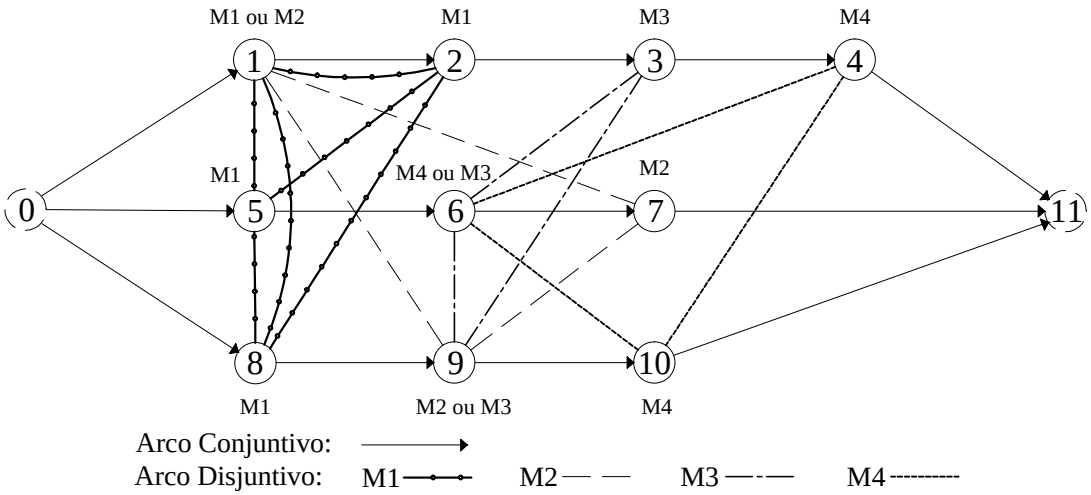

Figura 15 – Grafo disjuntivo: representação de um escalonamento.

O grafo do FSJP define uma solução viável se for acíclico (DAUZÈRE-PÉRÈS; PAULLI, 1997). A Figura 16 exemplifica uma solução obtida pela definição das orientações dos arcos disjuntivos do grafo da Figura 15, assim como a abstração em grafo disjuntivo da Figura 14.

5.2.1 Formulação matemática

De acordo com Birgin et al. (2014), o FJSP pode ser formulado matematicamente como um problema de Programação Linear Inteira Mista (PLIM) (CASTILLO et al., 2001). Para isso, utiliza-se uma matriz binária x para representar as atribuições da máquina e uma matriz binária y para representar seleções. A primeira matriz possui um compomente $x_{v,k}$ para cada v em V e cada k em $F(v)$. A segunda dispõe de um componente $y_{v,w}$ para cada (v, w) em B. Também utiliza-se duas matrizes racionais s e $p^{'}$, e um número racional z, onde s representa os tempos iniciais, $p^{'}$ indica os tempos de processamento

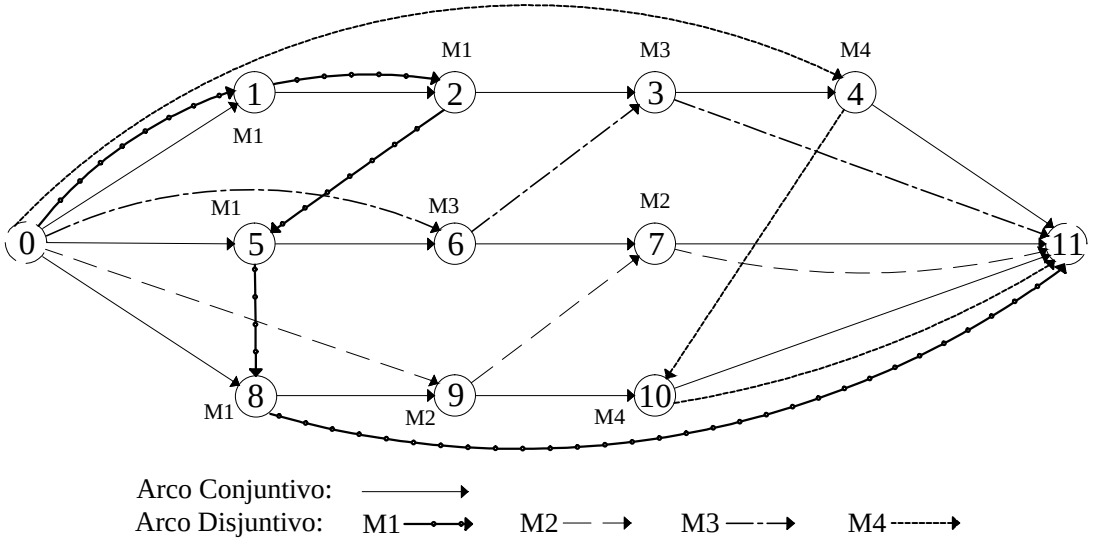

Figura 16 – Solução do FJSP.

correspondentes à atribuição da máquina dada por x, e z representa o valor do *makespan* do escalonamento s.

Considerando que há necessidade de um limitante superior L do *makespan* para o FJSP, esse valor pode ser obtido a partir da seleção de uma solução viável arbitrária ou, alternativamente, um limitante global $\sum_{v \in V} max_{k \in F(v)} p_{v,k}$. Dessa forma, uma formulação PLIM é proposta por Birgin et al. (2014) da seguinte forma: encontre um número racional z, matrizes racionais s e $p^{'}$, e matrizes binárias x e y tal que

$$\text{Minimizar } z \tag{5.1}$$

$$\text{Sujeito a: } s_v + p^{'}_v \leq z, \quad \forall v \in V, \tag{5.2}$$

$$\sum_{k \in F(v)} x_{v,k} = 1, \quad \forall v \in V, \tag{5.3}$$

$$p^{'}_v = \sum_{k \in F(v)} p_{v,k} x_{v,k}, \quad \forall v \in V, \tag{5.4}$$

$$y_{v,w} + y_{w,v} \geq x_{v,k} + x_{w,k} - 1, \quad \forall k \in M \text{ e } \forall (v,w) \in B_k, \tag{5.5}$$

$$s_v + p^{'}_v \leq s_w, \quad \forall (v,w) \in A, \tag{5.6}$$

$$s_v + p^{'}_v - (1 - y_{v,w})L \leq s_w, \quad \forall (v,w) \in B, \tag{5.7}$$

$$s_v \geq 0, \quad \forall v \in V. \tag{5.8}$$

A função objetivo é indicada na expressão (5.1), que considera z o *makespan*, que deve ser o menor valor possível (restrição (5.2)). Como x é binário, a restrição (5.3) garante que x represente uma operação atribuída a uma máquina. Os tempos de processamentos das operações são representados na matriz $p^{'}$ (restrição (5.4)). A restrição (5.5) define restrições de precedência entre operações. Considerando que y é uma variável binária que

representa uma seleção e p' configura os tempos de preocessamento, as restrições (5.6), (5.7) e (5.8) fazem com que s represente um escalonamento (solução).

5.3 Meta-heurísticas desenvolvidas para o FJSP

Para solucionar o FJSP, inicialmente desenvolveu-se um algoritmo GRASP, que é descrito na Seção 5.3.1. Com o objetivo de buscar melhoria nos resultados, optou-se por implementar um segundo método, que explore o espaço de soluções de forma diferente. Assim, em seguida, implementou-se um algoritmo baseado na meta-heurística *Simulated Annealing* (SA), que é explicitado na Seção 5.3.2. Considerando que o SA apresentou melhoria nos resultados em relação ao GRASP, houve o estímulo de experimentar novos métodos que utilizem a combinação de características de intensificação/exploração da busca. Assim, foi proposto outros dois métodos com características híbridas para solucionar o FJSP: *Iterated Local Search* (ILS) e *Clustering Search* (CS), abordados respectivamente nas Seções 5.3.3 e 5.3.4.

5.3.1 *GRASP*

A descrição genérica da meta-heurística GRASP é realizada na Seção 2.3.1. Para o FJSP implementou-se um algoritmo baseado no GRASP clássico, no qual subdivide-se nas etapas construção, intensificação e avaliação da solução, conforme apresentado no Algoritmo 1.

5.3.1.1 Fase de construção

A fase de construção visa gerar uma solução s (veja Algoritmo 8). Essa etapa baseia-se em adicionar em s, uma operação de um *job* por vez, até que todas sejam selecionadas. Para isso, ordena-se as operações candidatas (aquelas que não possuem restrição de precedência) dos *jobs* em uma lista CL, de modo que as operações com menores custos se apresentem primeiro (linha 3). Quando uma operação possui flexibilidade, opta-se pela máquina que a processe e proporcione o menor custo em s. De acordo com o parâmetro de entrada α, $(0 \leq \alpha \leq 1)$, constrói-se a Lista Restrita de Candidatos (LRC), que contém as primeiras $\lceil(\alpha \times |CL|)\rceil$ operações de CL (linha 4). Dessa forma, caso possua mais de uma operação disponível para ser adicionada à solução em construção, a eleatoriedade é utilizada para realizar a seleção. Quanto mais perto de zero os valores de α, a LRC se torna mais restrita, tornando menor a diversidade de soluções construídas. Por outro lado, para valores de α mais próximos de 1, ocorre um comportamento mais aleatório, o que tende a diversificar as soluções construídas, porém podendo gerar soluções com qualidade inferior e tornar mais lento o processo de busca local. Assim, verifica-se em s o tempo inicial que a operação selecionada na LRC pode ser adicionada (respeitando as restrições

de precedência) (linha 6). Após uma operação ser adicionada a solução em construção s (linha 7), reinicia-se o processo (linhas 2 a 8). A etapa de construção é finalizada quando não possuir mais operações candidatas a serem adicionadas à solução.

Algoritmo 8: Pseudocódigo da construção do GRASP para o FJSP

entrada : máquinas M, sequência de operações $(O_{i,j})_{i\in m, j\in n_i}$, α
saída : solução construída s

1 Conjunto de operações $\bar{O} \leftarrow (O_{i,j})_{i\in m, j\in n_i}$
2 **enquanto** $\bar{O} \neq \emptyset$ **faça**
3 $CL \leftarrow$ lista ordenada das $O_{i,j}$ operacoes candidatas
4 $LRC \leftarrow$ lista das $\lceil(\alpha \times |CL|)\rceil$ melhores operacoes de CL
5 Selecione aleatoriamente uma operacao $O_{i,j}$ da LRC
6 Verifica o tempo disponível em s_i para adicionar $O_{i,j}$
7 $s_i \leftarrow s_i + O_{i,j}$
8 Exclua a operacao $O_{i,j}$ de $\bar{O}$
9 **retorna** s

5.3.1.2 Busca local

A Busca Local (BL1) baseia-se no *best improvement method* (HANSEN; MLADENOVIĆ, 2006). Seu funcionamento consiste em explorar a vizinhança de uma solução s com o objetivo de obter uma solução ótima local s^*, que representa o melhor vizinho de s. O pseudocódigo da BL1 é apresentado no Algoritmo 9. A busca local se divide em duas vizinhanças (a) e (b):

(a) A primeira etapa (linhas 2 a 10) consiste em a cada iteração obter uma solução s' ao realizar troca na ordem de processamento de um *job* de uma máquina da solução s. Assim, ao selecionar uma determinada máquina, a partir do segundo *job* do escalonamento, efetua-se a troca com seu vizinho à esquerda. Após a troca, é verificado se a solução não possui violação de precedência, e caso positivo, efetua-se a reconstrução da solução à direita da troca efetuada. Em seguida é avaliado se houve melhora na solução ($s' < s^*$). Caso afirmativo, atualiza-se a melhor solução ($s^* \leftarrow s'$). Após cada troca, retorna-se ($s' \leftarrow s$) e seleciona-se o próximo *job* à direita. O processo se repete até todos os *jobs* de todas as máquinas da solução s sejam avaliados.

(b) Ao finalizar a etapa (a), inicia-se a etapa de exploração da flexibilidade da vizinhança da solução s (linhas 11 a 27). A ideia baseia-se em, para os *jobs* que possuem flexibilidade, realizar todas as trocas entre as máquinas alternativas e para cada substituição, explorar toda vizinhança da máquina inserida do mesmo modo que em (a). Para isso, seleciona-se um *job* na lista de flexibilidade (linha 13), e de acordo

com sua máquina alternativa, efetua-se a troca em s'. Em seguida é verificado se houve violação na restrição de precedência conforme (a). Posteriormente, apura-se $s' < s$, e em caso afirmativo, atualiza-se a melhor solução ($s^* \leftarrow s'$). Após, inicia-se a exploração da vizinhança na máquina que foi alterada, conforme (a), a partir da operação inserida, até toda sua vizinhança seja explorada. Ao final de cada troca, retorna-se a solução $s' \leftarrow s$ e seleciona-se o próximo *job* $flexJobs[k+1]$ que possui flexibilidade. Tal processo repete-se até que toda a flexibilidade seja avaliada.

Algoritmo 9: Pseudocódigo da Busca Local 1 (BL1)

entrada : $s, m, n_m, flexJobs[n_{flex}]$

```
1  s* ← s' ← s;
2  i ← 1;
3  enquanto i <= m faça
4      j ← 2;
5      enquanto j <= n_{m_i} faça
6          Troca os jobs s'(m_i[j]) and s'(m_i[j − 1]) ;
7          Reconstrói a solução s' ;
8          se f(s') < f(s*) então
9              s* ← s' ;
10         s' ← s; j + +;
11 k ← 1;
12 enquanto k <= n_{flex} faça
13     j_{flex} ← flexJobs[k];
14     i ← máquina alternativa que será alocado o j_{flex} ;
15     Destrua a solução s' da direita para esquerda até o j_{flex};
16     Insira o j_{flex} na máquina i;
17     j ← índice do j_{flex} na máquina i ;
18     Reconstrói a solução s';
19     se f(s') < f(s*) então
20         s* ← s'; s ← s';
21     enquanto j <= n_{m_i} faça
22         troca os jobs s'(m_i[j]) and s'(m_i[j − 1]) ;
23         Reconstrói a solução s' ;
24         se f(s') < f(s*) então
25             s* ← s';
26         s' ← s; j + +;
27     k + +;
28 retorna s*
```

5.3.2 *Simulated Annealing*

A descrição genérica do SA é realizada na Seção 2.3.2. Na presente implementação (veja Algoritmo 10), aplica-se SA_{max} iterações em s para cada temperatura (linhas 6-15), que é decrementada de acordo com o valor β (linha 16). A cada iteração, realiza-se uma perturbação na solução atual, gerando $s^{'}$ (linha 9). A perturbação baseia-se em realizar um movimento de troca, que é definido aleatoriamente dentre as estruturas de vizinhanças N^1, N^2 ou N^3 (linha 8). A variação da estrutura de vizinhança possibilita explorar diferentes regiões no espaço de soluções, diversificando a busca. Em seguida, a solução atual é avaliada (linha 10). Caso $s^{'}$ seja melhor que a solução global, atualiza-se s^* (linhas 11-13). Caso contrário, avalia-se, considerando uma dada probabilidade, se a busca continua a partir de $s^{'}$ ou em s (linhas 14-15). O SA_{max} foi configurado com base em Cruz-Chávez, Martínez-Rangel e Cruz-Rosales (2015), que definiu o número I (Equação 5.9) de iterações do SA de acordo com característica de cada instância:

$$I = 2 \times (m \times (n - 1)) \tag{5.9}$$

sendo m o número de máquinas e n o número de *jobs*.

Diferentemente do SA tradicional descrito na Seção 2.3.2, Cruz-Chávez, Martínez-Rangel e Cruz-Rosales (2015) propuseram restringir a vizinhança, permutando pares de operações adjacentes que não têm tempo de folga entre elas, daí surgindo o termo *SA-Partial* e também utilizaram o ajuste do parâmetro T_c a partir do valor do desvio padrão dos valores de sua calibração, com intenção de acelerar a busca. O presente estudo, de forma a extender a versão clássica do SA, foca em explorar várias estruturas de vizinhança como estratégia de acelerar a convergência da busca.

5.3.2.1 Solução Inicial

A construção da solução inicial s é realizada de forma parcialmente aleatória. A ideia baseia-se em selecionar os *jobs* aleatoriamente para serem adicionados em s e, quando há flexibilidade, optar pela máquina que processa a respectiva operação que proporcione o menor custo em s. Para cada operação de um *job*, é inserido seu respectivo número num vetor v com objetivo de representar a quantidade de operações que cada *job* possui. Como exemplo, ilustra-se na Figura 17 a construção de uma solução de uma instância com dois *jobs*, o primeiro com duas operações a serem processadas na máquina M1 com custo 6 e na máquina M2 com custo 10, respectivamente ($O_{1,1}(M1[6])$ e $O_{1,2}(M2[10])$) e o segundo *job* com uma operação a ser processada na máquina M1 com custo 20 ($O_{2,1}(M1[20])$) ou alternativamente na máquina M2 com custo 7 ($O_{2,1}(M2[7])$), gerando o vetor $v = [1, 1, 2]$. Inicialmente sorteou-se o índice $i = 2$, que representa o *job* 1. Então a primeira operação disponível do *job* 1 ($O_{1,1}(M2[6])$) foi selecionada para ser adicionada à solução parcial s,

Algoritmo 10: Pseudocódigo do *Simulated Annealing* para o FJSP

entrada : T_0, T_f, β, SA_{max}, Solução s, $Time_{max}$
saída : Solução s^*

```
1  s* ← s;
2  T ← T0;
3  enquanto Time < Time_max faça
4      enquanto T > Tf faça
5          iter ← 0 ;
6          enquanto iter < SA_max faça
7              iter ← iter + 1 ;
8              k ← random[1, 3] ;
9              s' ← N^k(s);
10             se f(s') < f(s) então
11                 s ← s';
12                 se f(s') < f(s*) então
13                     s* ← s';
14             senão
15                 s ← s', com probabilidade e^(−(f(s')−f(s))/T);
16         T ← T × β;
17 retorna s* ;
```

preenchendo o tempo da máquina M1 de 1 a 6 e decrementando o tamanho de v (etapa (a)). Em seguida, sorteou-se $i = 1$, e então a segunda operação do *job* 1 ($O_{1,2}(M2[10])$) é adicionada à solução, iniciando ao final da $O_{1,1}$, isto é, preenchendo o tempo da máquina M2 de 6 a 16 (etapa (b)). Por fim, o índice $i = 1$ é sorteado, e considerando que há flexibilidade na operação do *job* 2, tem-se a seguinte alternativa: M1 (preenchendo o tempo de 6 a 26) ou M2 (preenchendo o tempo de 16 a 23). Selecionou-se a máquina M2, pois proporciona o menor *makespan* (22) e inseriu-a em s (etapa (c)).

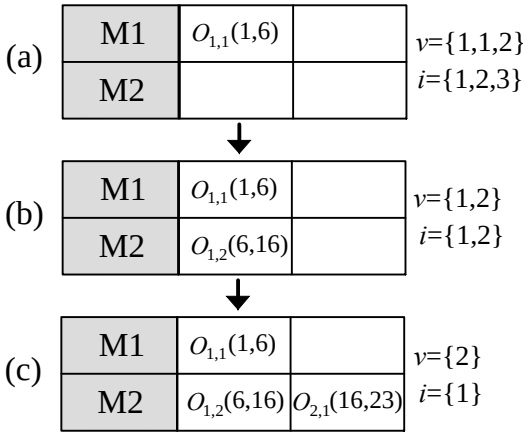

Figura 17 – Exemplo de geração de uma solução inicial para dois *jobs* com três operações

5.3.2.2 Estruturas de Vizinhança

De modo a buscar melhores soluções, o SA possui a característica de explorar a vizinhança de forma iterativa, realizando trocas entre os vizinhos. No contexto do FJSP, os movimentos são realizados alterando a atribuição (por exemplo, movendo uma operação de uma máquina para outra) ou alterando uma sequência (por exemplo, deslocando uma operação ou trocando duas operações) (BRUCKER, 2007; BRUCKER; KNUST, 2012). Dessa forma, três formas de explorar a vizinhança foram utilizadas: N^1, N^2 e N^3, que são descritas a seguir.

5.3.2.2.1 Estrutura de Vizinhança N^1

A estratégia usada na estrutura de vizinhança N^1 é baseada na troca de posições relativas entre um par de operações adjacentes em uma máquina. Para isso, selecionamos aleatoriamente uma máquina e, em seguida, selecionamos aleatoriamente uma operação entre as alocadas na máquina. Somente a primeira operação na máquina selecionada não pode ser escolhida. Isto é devido a como a troca é realizada: as duas operações consideradas para trocar posições relativas são a operação selecionada, O_{ij}, e a operação à sua esquerda imediata, diz O_{pq}. Para não violar a solução, essa estrutura não realiza uma troca quando a operação selecionada e seu vizinho à esquerda pertencem ao mesmo *job*. Quando isso ocorre, nenhum movimento é executado e o processo é finalizado. Mesmo assim, após realizar uma troca, ainda é possível obter uma solução inviável, que é então reparada, como exemplificado na Figura 18.

A Figura 18 ilustra um movimento de troca de N^1. A solução atual é mostrada como a sequência de operações (à esquerda) e o gráfico de Gantt que resultaram da alocação das operações na sequência determinada (à direita). A máquina M1 é selecionada aleatoriamente e, em seguida, uma das operações alocadas para a máquina M1 ($O_{3,2}$) é escolhida aleatoriamente (a). No vetor sequência, uma troca é realizada entre $O_{3,2}$ e a operação $O_{1,2}$, localizada imediatamente à sua esquerda (já que não são operações do mesmo job) (b). Percebe-se que tal troca produziu uma solução inviável em (b), uma vez que a operação $O_{3,2}$ deve ser processada após a conclusão da operação $O_{3,1}$, esta solução deve ser reparada. Portanto, no vetor de sequência, a ordem de processamento entre as operações $O_{3,1}$ e $O_{3,2}$ é alterada, gerando uma solução viável (c). Após um movimento de troca N^1, a solução resultante pode ser inviável. Neste caso, a solução é reparada

5.3.2.2.2 Estrutura de Vizinhança N^2

A estratégia usada na estrutura de vizinhança N^2 é baseada explorar a flexibilidade. Uma operação é considerada flexível se pudermos processá-la em mais de uma máquina. O método procura reprogramar em outra máquina uma operação flexível. Primeiro, uma

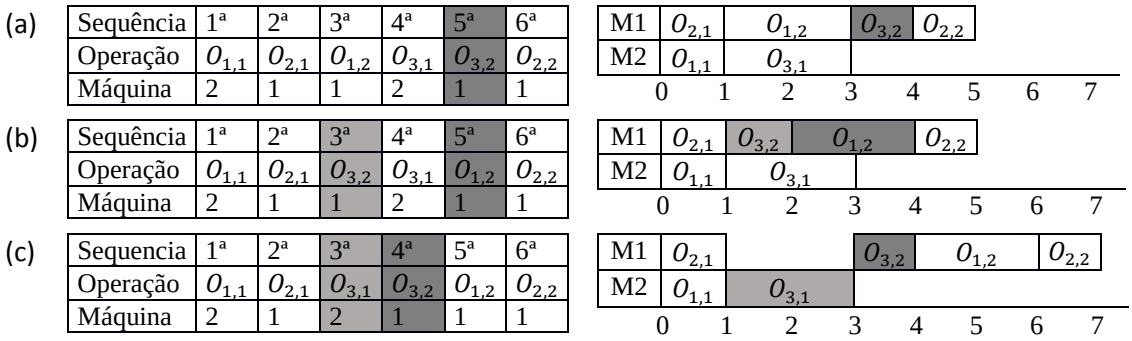

Figura 18 – Movimento de troca: N^1.

operação flexível é selecionada aleatoriamente e a máquina na qual essa operação está alocada é identificada de acordo com a solução atual. Outra máquina que pode processar essa operação é selecionada aleatoriamente. Em seguida, a operação é movida da máquina que está processando para a máquina selecionada. A solução pode ter que ser reparada, a fim de manter sua viabilidade.

A Figura 19 mostra uma troca N^2. Inicialmente, a operação $O_{3,2}$ foi selecionada aleatoriamente entre as operações flexíveis. A operação $O_{3,2}$ pode, alternativamente, ser processada nas máquinas M1 ou M2. Verificou-se que na solução atual, a operação $O_{3,2}$ é alocada na máquina M1 (a) e então foi definida que a operação $O_{3,2}$ será movida para a máquina M2. O resultado da troca é mostrado em (b).

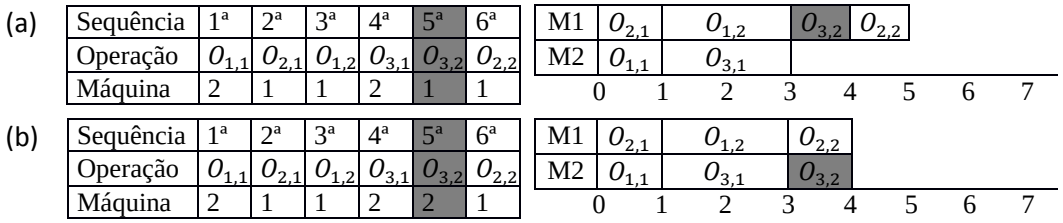

Figura 19 – Movimento de troca: N^2.

5.3.2.2.3 Estrutura de Vizinhança N^3

A estrutura de vizinhança chamada de N^3 é uma extensão da N^1, sendo que a estratégia utilizada baseia-se em realizar a troca de operações considerando a ordem na qual essas foram inseridas na solução. Para isso, utiliza-se um vetor auxiliar que armazena o histórico da sequência de inserções das operações nas máquinas na solução. A troca N^3 apresenta a seguinte característica:

1. Na estrutura de dados que armazenou a ordem de inserção das operações nas máquinas seleciona-se aleatoriamente uma operação e denomina-se esta por X;

2. Se a operação Y mais à esquerda de X não pertence ao mesmo *job* de X: realiza-se a troca entre as posições de X e Y no vetor auxiliar;

3. Caso contrário, denomina-se a operação Y como operação X e volta-se para o passo 2.

A Figura 20 ilustra um movimento N^3 quando a precedência é violada. A operação de $O_{3,3}$ foi selecionada aleatoriamente e a operação $O_{3,2}$ está à esquerda no vetor de sequência. Ambas as operações pertencem ao mesmo *job* e haveria uma violação da restrição de precedência se uma troca fosse feita. Desta forma, a operação $O_{3,2}$ é agora considerada para efetuar a troca. A operação à esquerda de $O_{3,2}$ é $O_{1,2}$. Como neste caso não há violação de precedência, uma troca foi feita entre $O_{3,2}$ e $O_{1,2}$;

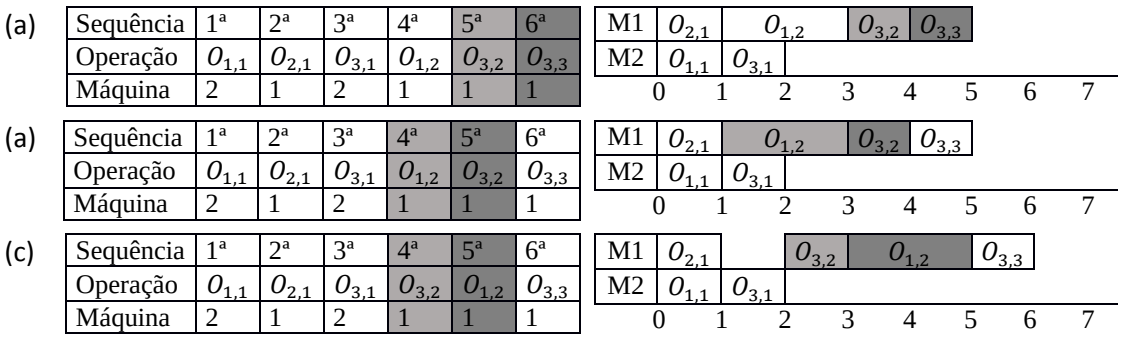

Figura 20 – Movimento de troca N^3 - Quando a ordem de precedência é violada.

5.3.3 *Iterated Local Search*

Para solucionar o FJSP, desenvolveu-se um algoritmo baseado na meta-heurística *Iterated Local Search* (ILS), que foi apresentado na Seção 2.3.3. O Algoritmo 11 apresenta o pseudocódigo genérico do ILS desenvolvido para o FJSP. Após gerar uma solução inicial s (linha 1), o ILS consiste em aplicar uma busca local em s (linha 2), com o objetivo de obter um ótimo local e posteriormente iniciar o processo iterativo (linhas 3-7). Até atingir o limite ILS_{max} de iterações, o ILS realiza pertubação na melhor solução s, para em seguida submetê-la a uma busca local e posteriormente avaliá-la por um critério de aceitação. Na presente implementação, o parâmetro ILS_{max} é determinado por um tempo

limite. Tais etapas são descritas a seguir.

Algoritmo 11: Pseudocódigo do ILS para o FJSP.

entrada : ILS_{max}, k_{max}
saída : Solução s

1 $s \leftarrow$ `SolucaoInicial`();
2 $s \leftarrow$ `BL1`(s);
3 **para** $i \leftarrow 1$ ***até*** ILS_{max} **faça**
4 $\quad s_i \leftarrow$ `Pertubacao`(s, k_{max});
5 $\quad s_i \leftarrow$ `BL2`($s_i, T_0, T_f, \beta, SA_{max}$);
6 $\quad s \leftarrow$ `Aceitacao` (s, s_i);
7 **fim**

5.3.3.1 Solução Inicial

A solução inicial é gerada como descrito na Seção 5.3.2.1.

5.3.3.2 Busca Local 1

A busca local utilizada é a mesma descrita na Seção 5.3.1.2.

5.3.3.3 Busca Local 2: *Simulated Annealing*

Ao iniciar o processo iterativo do ILS, utiliza-se como busca local uma variação do SA descrito na Seção 5.3.2. Essa variação consiste na incorporação de uma busca local (BL1) no SA, conforme especificado a seguir.

Apresenta-se no Algoritmo 12 o pseudocódigo do SA proposto como busca local (BL2). O SA inicia em uma temperatura T_0, que é decrementada conforme o valor de β (linha 18), sendo executado até atingir temperatura de congelamento T_f. A cada temperatura realiza-se SA_{max} iterações (linhas 5 a 17). A cada iteração do SA realiza-se uma pertubação na solução atual s de acordo com uma das três vizinhanças (N^1, N^2 ou N^3), que são selecionadas aleatoriamente, obtendo s' (linhas 7 a 8). Caso s' seja melhor que a solução global (linhas 9 a 15), atualiza-se s^* e aplica-se nesta a busca local BL1 (descrita na Seção 5.3.3.2). Caso contrário, avalia-se, considerando uma dada prioridade, se a busca continua a partir de s' ou s (linhas 16 a 17). O SA_{max} foi configurado da mesma forma que no SA, utilizando a Equação 5.9.

5.3.3.4 Pertubação

Como estratégia de perturbação, realiza-se k_{max} pertubações em s. Com o objetivo de não tornar a busca determinística, a cada iteração é selecionado aleatoriamente um dos

Algoritmo 12: Pseudocódigo do *Simulated Annealing* com busca local (BL2)

entrada : T_0, T_f, β, SA_{max}, Solução s
saída : Solução s^*

```
1  s* ← s;
2  T ← T0;
3  enquanto T > Tf faça
4      iter ← 0 ;
5      enquanto iter < SAmax faça
6          iter ← iter + 1 ;
7          k ← random[1, 3] ;
8          s' ← N^k(s);
9          se f(s') < f(s) então
10             s ← s';
11             se f(s') < f(s*) então
12                 s* ← s';
13                 s' ← BL1(s');
14                 se f(s') < f(s*) então
15                     s* ← s';
16         senão
17             s ← s', com probabilidade e^(−(f(s')−f(s))/T);
18     T ← T × β;
19 retorna s* ;
```

movimentos N^1, N^2 ou N^3. O Algoritmo 13 apresenta o pseudocódigo desta etapa.

Algoritmo 13: Perturbação do ILS para o FJSP.

entrada : s, k_{max}
saída : Solução s

```
1  para i ← 1 até kmax faça
2      k ← random[1, 3] ;
3      s ← N^k(s);
4  fim
```

5.3.3.5 Critério de Aceitação

Para o ILS aceitar uma solução intermediária s_i, esta tem de ser melhor que a solução atual s. Assim, é avaliado se $f(s_i) < f(s)$, e em caso afirmativo, a solução s é descartada e a busca continua em s_i. Caso contrário, a busca continua em s.

5.3.4 *Clustering Search*

A meta-heurística *Clustering Search* (CS) foi desenvolvida como um método genérico que combina meta-heurísticas e clusterização. O CS é considerado um método híbrido por unir meta-heurísticas para geração de soluções e buscas locais. Tem o foco em particionar o espaço de busca e identificar "regiões" (*clusters*) supostamente promissoras, para assim intensificar a busca nessa região aplicando busca local (OLIVEIRA; LORENA, 2007).

O CS foi introduzido na Seção 2.3.4. Apresenta-se no Algoritmo 14 o pseudocódigo do CS proposto para solucionar o FJSP, utilizando o SA como gerador de soluções. O SA é executado até atingir o tempo limite $Time_{max}$ e a cada decremento da temperatura, realiza-se SA_{max} iterações e em seguida a solução atual do SA é enviada para o CS (linhas 4-17). O SA_{max} foi configurado com base em Cruz-Chávez, Martínez-Rangel e Cruz-Rosales (2015), que definiu o número I de iterações do SA de acordo com característica de cada instância (Equação 5.9).

De modo a proporcionar maior diversificação na busca, ao final do SA definiu-se que a cada 30 segundos initerruptos sem melhora na solução, o número SA_{max} de iterações do SA é dobrado e o volume máximo λ é incrementado em 3 unidades, desde que o SA_{max} não ultrapasse seu valor inicial $I \times 100$.

Inicializa-se o método criando o número γ de *clusters*, com suas respectivas características de volume máximo λ e índice de ineficácia r_{max}. Em seguida, inicia-se o SA, que recebe uma solução inicial aleatória, que é definida como a melhor solução (linha 2). Por conseguinte, onde a cada iteração escolhe-se randomicamente uma vizinhança (N^1, N^2 ou N^3) para gerar uma solução vizinha da solução atual (linhas 9-10). Uma solução vizinha é aceita caso for melhor que a solução atual (linhas 11-14) ou, caso contrário, com uma dada probabilidade (linhas 15-16).

A temperatura do SA é decrementada (linha 17) e caso o número máximo de *clusters* não seja atingido, atribui-se ao centro de um novo *cluster* a solução atual do SA (linhas 18-20). Após inicializados todos os γ *clusters*, as novas soluções geradas pelo SA passam a serem incluídas nos *clusters* existentes (linha 23).

Em seguida, inicia-se o processo de assimilação, que é definido com base na semelhança entre a solução atual do SA e o centro de todos os *clusters*. Naquele *cluster* que possuir seu centro mais adequado, insere-se a solução atual do SA e verifica se a região tornou-se promissora e se será aplicada a busca local ou uma perturbação ou se uma nova melhor solução foi encontrada (linhas 25 a 35).

Os parâmetros utilizados no CS proposto são:

- γ: Número máximo de *clusters*;
- λ: Volume máximo dos *clusters*;

- r_{max}: Limite para o índice de ineficácia dos *clusters*;
- T_0: Temperatura inicial do SA;
- β: Taxa de resfriamento do SA;
- T_f: Temperatura final do SA;
- SA_{max}: Número máximo de iterações por temperatura do SA; e
- $Time_{max}$: Tempo limite de execução do CS.

5.3.4.1 Estruturas de Vizinhança

Utilizou-se as estruturas de vizinhança N^1 (seção 5.3.2.2.1), N^2 (seção 5.3.2.2.2) e N^3 (seção 5.3.2.2.3).

5.3.4.2 Similaridade

O objetivo dessa etapa é identificar qual *cluster* possui seu centro mais similar à solução atual (linhas 22-23 do Algoritmo 14). Essa similaridade é obtida através do cálculo da distância de Hamming (D_i) (HAMMING, 1950), no qual é contabilizado o número de posições que as soluções se diferenciam entre si. Ao final da comparação, a solução é atribuída ao *cluster* que esteja menos distante, com isso, o volume do *cluster* v_i é incrementado. Logo após, compara-se a solução do centro do *cluster* com a solução gerada pelo SA e se a solução gerada pelo SA for melhor, torna-se o centro do *cluster*.

A Figura 21 apresenta duas soluções distintas s_1 e s_2. Essa estrutura se refere a ordem de processamento dos jobs nas máquinas. As soluções se diferenciam na máquina que atende o último job ($O_{2,2}$). A distância de Hamming entre as duas soluções é calculada e seu valor é igual um.

s_1

Operação	$O_{1,1}$	$O_{1,2}$	$O_{2,1}$	$O_{2,2}$
Máquina	1	2	1	2

s_2

Operação	$O_{1,1}$	$O_{1,2}$	$O_{2,1}$	$O_{2,2}$
Máquina	1	2	1	1

Figura 21 – Representação do cálculo da distância de Hamming entre duas soluções s_1 e s_2: $D_i(s_1, s_2) = 1$.

Algoritmo 14: Pseudocódigo do CS para o FJSP

entrada: γ, λ, r_{max}, T_0, T_f, β, SA_{max},$Time_{max}$, s

```
1  cluster ← 0; r_i ← 0; v_i ← 0; ∀i = 1...γ;
2  s* ← s; SA_maxIni ← SA_max;
3  enquanto Time < Time_max faça
4      T ← T_0;
5      enquanto T > T_f faça
6          iter ← 0 ;
7          enquanto iter < SA_max faça
8              iter ← iter + 1 ;
9              k ← random[1, 3] ;
10             s' ← N^k(s);
11             se f(s') < f(s) então
12                 s ← s';
13                 se f(s') < f(s*) então
14                     s* ← s';
15             senão
16                 s ← s', com probabilidade e^(−(f(s')−f(s))/T);
17         T ← T × β;
18         se cluster < γ então
19             cluster + +; v_cluster + +;
20             c_cluster ← s;
21         senão
22             i ← armin_{i∈{1...γ}}{D_i};
23             v_i + +; c_i ← best(c_i, s);
24             se v_i = λ então
25                 v_i ← 1;
26                 se r_i = r_max então
27                     r_i ← 0;
28                     c_i ← N^(random[1,3])(c_i);
29                 senão
30                     BL3(c_i);
31                     se c_i melhorou então
32                         r_i ← 0;
33                     senão
34                         r_i + +;
35                 s* ← best(s*, c_i);
36 retorna s*
```

5.3.4.3 Busca Local 3

Como busca local implementou-se um algoritmo baseado nos métodos propostos por Hansen e Mladenović (2006). Seu funcionamento consiste em explorar a vizinhança

da solução s que representa o centro c_i de um *cluster* que atingiu o limite de volume λ. A diferença em relação a busca local BL1 apresentada na Seção 5.3.1.2 se resume em ao encontrar um vizinho melhor, encerra-se a busca naquela máquina, iniciando na próxima, caso exista. A estratégia se divide em duas vizinhanças (a) e (b):

(a) A primeira etapa consiste em obter uma solução s' ao realizar troca da ordem de processamento dos *jobs* de cada máquina de s. Assim, ao selecionar uma determinada máquina, a partir do segundo *job* do escalonamento, efetua-se a troca com seu vizinho à esquerda. Após a troca, é verificado se a solução não possui violação de precedência, e caso positivo, efetua-se a reconstrução da solução à direita da troca efetuada. Em seguida é avaliado se houve melhora na solução ($s' < s$). Caso afirmativo, atualiza-se a solução ($s \leftarrow s'$) e a busca continua a partir da próxima máquina. Caso contrário, retorna-se a solução $s' \leftarrow s$ e seleciona-se o próximo *job* à direita. O processo se repete até que todos os *jobs* de todas as máquinas sejam avaliados.

(b) Ao finalizar a etapa (a), inicia-se a etapa de exploração da flexibilidade da vizinhança s. A ideia baseia-se em, para os *jobs* que possuem flexibilidade, realizar todas as trocas entre as máquinas alternativas e para cada substituição, explorar toda sua vizinhança do mesmo modo que em (a). Para isso, seleciona-se um *job* em $flexJobs[k]$ que possui flexibilidade, e de acordo com sua máquina alternativa, efetua-se a troca em s'. Em seguida é verificado se houve violação na restrição de precedência conforme (a). Posteriormente, apura-se $s' < s$, e em caso afirmativo, atualiza-se a melhor solução atual ($s \leftarrow s'$). Após, inicia-se a exploração da vizinhança na máquina que foi alterada, conforme (a), a partir da operação inserida, até que encontre um vizinho melhor ou toda a vizinhança seja explorada. Caso não encontre um vizinho melhor, retorna-se a solução $s' \leftarrow s$ e seleciona-se o próximo job $flexJobs[k+1]$ que possui flexibilidade. Tal processo repete-se até que toda a flexibilidade seja avaliada.

O pseudocódigo da Busca Local 3 é apresentado no Algoritmo 15.

5.4 Experimentos Computacionais

Os experimentos computacionais são apresentados da seguinte forma: Inicialmente apresenta-se a configuração dos experimentos para cada método, em seguida, os resultados individuais dos algoritmos são exibidos que, por fim, são comparados com a literatura.

5.4.1 Configuração dos Experimentos

Para verificar a eficácia dos algoritmos descritos neste estudo, considera-se conjuntos tradicionais de instâncias do FJSP encontradas na literatura: *BRdata* (BRANDIMARTE,

Algoritmo 15: Pseudocódigo da Busca Local 3 (BL3)

entrada : s

$s^* \leftarrow s' \leftarrow s$;
$i \leftarrow 1$;
enquanto $i <= m$ **faça**
    $j \leftarrow 2; repita \leftarrow VERDADEIRO$;
    enquanto $j <= n_{m_i}$ ***e*** $repita$ **faça**
        troca os jobs $s'(m_i[j])$ e $s'(m_i[j-1])$;
        reconstrói s' ;
        se $f(s^{'}) < f(s^*)$ **então**
            $s^* \leftarrow s^{'}$;
            $repita \leftarrow FALSO$;
        $s' \leftarrow s$;
        $j++$;
$k \leftarrow 1$;
enquanto $k <= n_{flex}$ **faça**
    Seleciona-se o job j_{flex} na lista de jobs que possuem flexibilidade $flexJobs[k]$;
    $i \leftarrow$ máquina alternativa que será alocado o j_{flex} ;
    Destrua a solução s' da direita para esquerda até o j_{flex};
    Insira o j_{flex} na máquina i;
    $j \leftarrow$ índice do j_{flex} na máquina i ;
    Reconstrua a solução $s'; repita \leftarrow VERDADEIRO$;
    se $f(s^{'}) < f(s^*)$ **então**
        $s^* \leftarrow s^{'}; s \leftarrow s^{'}$;
    enquanto $j <= n_{m_i}$ ***e*** $repita$ **faça**
        troca os jobs $s'(m_i[j])$ e $s'(m_i[j-1])$;
        reconstrói s' ;
        se $f(s^{'}) < f(s^*)$ **então**
            $s^* \leftarrow s^{'}$;
            $repita \leftarrow FALSO$;
        $s' \leftarrow s$;
        $j++$;
    $k++$;
retorna s^* ;

1993), que possui de 10 a 20 *jobs*, 4 a 15 máquinas e fator de flexibilidade de 1,43 a 4,10; *HUdata* (HURINK; JURISCH; THOLE, 1994), que possui como características instâncias com número de *jobs* que vão de 6 a 30, número de máquinas que vão de 5 a 15 e flexibilidade de 1,15 a 7,50; *BCdata* (BARNES; CHAMBERS, 1996), possuindo instâncias de 10 a 15 *jobs* e 11 a 18 máquinas, com flexibilidade variando de 1,10 a 1,30; e *DPdata* (DAUZÈRE-PÉRÈS; PAULLI, 1997), com instâncias possuindo de 10 a 20 *jobs*, 5 a 8 máquinas e flexibilidade de 1,13 a 5,02. Esse conjunto proporciona um total de 178 instâncias com diferentes tamanhos e complexidade.

A descrição detalhada das instâncias de cada conjunto de instâncias está relacionada na Seção Apêndice B.1.

Os algoritmos GRASP, SA, ILS e CS foram implementados em linguagem C/C++ e compilados com o compilador Visual C++. Os testes foram executados em um computador com processador Intel Core i5-750 2,67GHz com 8GB de memória RAM, utilizando sistema operacional Windows 7 Ultimate 64 Bits. Os algoritmos foram aplicados 10 vezes para cada instância, e registrou-se o melhor e a média do valor *makespan* encontrado.

5.4.1.1 Configuração dos Experimentos - GRASP

Para o GRASP implementado, considera-se apenas o tempo computacional como parâmetro de entrada, uma vez que o parâmetro α é definido aleatoriamente dentre o intervalo $\{0,1; 0,2; ...; 0,8; 0,9\}$ a cada início da etapa de construção. Dessa forma, configurou-se o tempo máximo em 1800 segundos para cada execução do GRASP. O algoritmo foi aplicado dez vezes para cada instância relacionada e armazenou-se o melhor valor da função objetivo, assim como o tempo que o mesmo foi encontrado.

5.4.1.2 Configuração dos Experimentos - SA

Para calibrar o SA, foram realizados testes com três instâncias de pequeno, médio e grande porte (mk01, 10a e mk10, respectivamente). A Tabela 6 apresenta o resultado final do processo de calibração.

Tabela 6 – Definição dos parâmetros do SA.

Param.	**Valor definido**	**Valores testados**
T_0	1000	100, 1000 e 10000
β	0,998	-
T_f	0,001	-
SA_{max}	I	I, $I \times 10$ e $I \times 100$
$Time_{max}$	1800	600, 1200 e 1800

5.4.1.3 Configuração dos Experimentos - ILS

Para calibrar o ILS foram realizados testes com três instâncias de pequeno, médio e grande porte (mk01, 10a e mk10, respectivamente). A Tabela 7 apresenta o resultado final do processo de calibração.

Tabela 7 – Definição dos parâmetros do ILS.

Param.	Valor definido	Valores testados
T_0	100	100, 1000 e 10000
β	0,998	-
T_f	0,001	-
SA_{max}	I	I, $I \times 10$ e $I \times 100$
k_{max}	$I \times 0,1$	10, $I \times 0,1$ e 100
$Time_{max}$	1800	600, 1200 e 1800

Considerando que o ILS utiliza a meta-heurística SA como método de busca local, o T_0 foi o único parâmetro específico do SA que sofreu alteração em relação a parametrização apresentada na Seção 5.4.1.2. Um valor menor do T_0 possibilitou explorar melhor a característica de diversificação do ILS, sendo que com um valor menor para o respectivo parâmetro, o ILS executava poucas iterações, consequentemente se comportando de maneira proxima ao SA.

De modo a buscar uma maior diversificação e intensificação da busca, ao final de cada iteração do ILS, o SA_{max} e o k_{max} têm seus valores dobrados (com a limitação do tamanho de uma variável inteira). Assim, a cada nova iteração do ILS, a solução s sofrerá um número maior de perturbações, o que tende a diversificar a busca, e o SA realizará um maior número de iterações a cada temperatura, intensificando a busca.

5.4.1.4 Configuração dos Experimentos - CS

Para calibrar o CS foram realizados testes com três instâncias de pequeno, médio e grande porte (mk01, la10 e mk10, respectivamente). A Tabela 8 apresenta o resultado final do processo de calibração. Considerando a diferença das instâncias, verificou-se que algumas necessitavam de maior grau de diversificação da busca. Dessa forma, definiu-se a estratégia de a cada trinta segundos sem o algoritmo apresentar melhora na solução, que seja dobrado o SA_{max} e incrementado em três unidades o volume máximo λ dos *clusters*.

Tabela 8 – Definição dos parâmetros do CS

Param.	Valor definido	Valores testados
γ	20	5, 10 e 20
λ	5	5, 10 e 20
r_{max}	2	1, 2, 5
T_0	1000	100, 1000 e 10000
β	0,998	-
T_f	0,001	-
SA_{max}	I	I, $I \times 10$ e $I \times 100$
$Time_{max}$	1800	600, 1200 e 1800

5.4.2 Resultados computacionais dos algoritmos propostos

5.4.2.1 Resultados computacionais do algoritmo GRASP

O primeiro algoritmo desenvolvido para o FJSP foi o baseado na meta-heurística GRASP. O GRASP foi aplicado a algumas instâncias de diferentes complexidades e os resultados são comparados com os apresentados pelo TS (MASTROLILLI; GAMBARDELLA, 2000) na Tabela 9, que é descrita como segue. As três primeiras colunas indicam características da instância utilizada, compreendendo o conjunto na qual ela se encontra (Conjunto), o nome da instância (I) e seu respectivo número de *jobs* e número de máquinas ($n \times m$). Para cada algoritmo (TS e GRASP), relaciona-se o melhor valor da função objetivo encontrado para cada instância (FO), assim como a média dos valores da função objetivo obtidos em dez execuções (Av) e a média dos tempos em que foram necessários para atingir os melhores resultados (T(s)). Por fim, é apontado o desvio entre os valores das melhores soluções encontradas pelo GRASP em relação ao TS na coluna Desvio.

Ao analisar os resultados da Tabela 9, constatou-se que, dentre as dezesseis instâncias consideradas, o GRASP consegue chegar nos mesmos melhores resultados do TS (MASTROLILLI; GAMBARDELLA, 2000) em apenas cinco destas (HUdata/edata/mt06, HUdata/rdata/mt06, HUdata/vdata/mt06, HUdata/vdata/mt10 e HUdata/vdata/mt20), isto é, em aproximadamente 31%, porém com um custo computacional bastante superior nas instâncias duas últimas instâncias, que possuem a característica de possuir um maior número de *jobs* e máquinas. Em relação aos desvios, a média geral ficou em 8,28%, porém na instância Mk10, o GRASP apresentou um desvio de 43,94% em relação ao TS. Nas instâncias maiores, o GRASP apresentou o resultado médio mais distante do melhor resultado da função objetivo obtido, e com um alto custo computacional, quando comparados com os resultados do TS. Com isso, percebeu-se que, para instâncias mais complexas, o GRASP não conseguiu obter bons resultados. Assim sendo, optou-se por implementar outras meta-heurísticas, que utilizam diferentes estratégias de exploração do espaço de soluções, como o SA, ILS e CS. Tais métodos são descritos nas seções seguintes.

5.4.2.2 Resultados computacionais dos algoritmos SA, ILS e CS

Apresenta-se na Tabela 10 o resumo dos resultados dos algoritmos propostos - SA, ILS e CS - conforme descrito a seguir. Na primeira coluna (Conjunto) lista-se o nome do conjunto de instâncias utilizado. As demais colunas subdividem-se em três informações obtidas pela execução dos respectivos algoritmos: O média do melhor valor da função objetivo encontrado para cada instância a partir das dez execuções de cada algoritmo (FO); A média geral dos melhores valores da função objetivo obtidos nas dez execuções para cada instância dos respectivos algoritmos (Av); e, a média geral dos tempos necessários para as meta-heurísticas chegarem aos melhores valores da função objetivo encontrados nas respectivas execuções (T(s)).

Tabela 9 – Resultados do GRASP proposto e comparação com os result'ados de algumas instâncias clássicas.

			TS			GRASP			
Conjunto	I	$n \times m$	FO	Av	T(s)	FO	Av	T(s)	Desvio
BRdata	Mk01	10×6	**40**	40,0	0,0	41	41,6	60,0	2,50%
BRdata	Mk10	20×15	**198**	199,2	7,7	285	288,1	668,0	43,94%
BCdata	mt10c1	10×11	**928**	928,0	2,3	976	982,3	897,4	5,17%
DPdata	10a	15×8	**2291**	2305,6	56,1	2681	2722,5	840,6	17,02%
HUdata/edata	mt06	6×6	**55**	55,0	0,0	**55**	55,0	0,0	0,00%
HUdata/edata	mt10	10×10	**871**	873,0	1,6	916	930,1	324,2	5,17%
HUdata/edata	mt20	20×5	**1088**	1088,8	3,5	1218	1245,5	313,4	11,95%
HUdata/edata	la40	15×15	**1150**	1151,6	7,8	1296	1313,9	943,1	12,70%
HUdata/rdata	mt06	6×6	**47**	47,0	0,0	**47**	47,00	0,4	0,00%
HUdata/rdata	mt10	10×10	**686**	686,0	2,7	714	727,5	401,9	4,08%
HUdata/rdata	mt20	20×5	**1022**	1022,6	3,5	1026	1028,1	510,9	0,39%
HUdata/rdata	la40	15×15	**970**	974,0	6,1	1138	1163,9	715,7	17,32%
HUdata/vdata	mt06	6×6	**47**	47,0	0,0	**47**	47,0	0,0	0,00%
HUdata/vdata	mt10	10×10	**655**	655,0	0,0	**655**	660,0	216,3	0,00%
HUdata/vdata	mt20	20×5	**1022**	1022,0	3,8	**1022**	1023,0	438,4	0,00%
HUdata/vdata	la40	15×15	**955**	955,0	0,3	1072	1084,5	814,2	12,25%
Média			751,56	753,11	5,97	824,31	835,00	446,5	8,28%

Tabela 10 – Resumo dos resultados

	SA			ILS			CS		
Conjunto	FO	Av	T(s)	FO	Av	T(s)	FO	Av	T(s)
BRdata	173,60	175,43	129,52	**172,70**	173,22	322,76	173,00	173,47	165,92
DPdata	2309,89	2390,34	434,72	**2211,28**	2216,46	884,45	2211,39	2217,08	629,38
BCdata	997,95	1000,60	690,86	997,57	1001,09	630,38	**997,05**	999,53	316,44
HUdata/edata	1011,53	1006,41	268,63	**1003,19**	1004,88	289,27	1003,47	1004,98	226,29
HUdata/rdata	910,60	911,86	395,85	**909,05**	910,69	362,27	909,28	910,95	350,47
HUdata/vdata	895,86	896,36	234,19	**895,40**	895,74	243,24	895,51	895,92	254,80
Média	1049,91	1063,50	358,96	1031,53	1033,68	455,40	1031,62	1033,65	323,88

Analisando os resultados da Tabela 10, percebe-se que o ILS e o CS apresentam-se sempre mais competitivos que o SA nos conjuntos de instâncias considerados. Na média, o ILS obteve resultados ligeiramente melhores que o CS nos conjuntos de instâncias *BRdata*, *DPdata* e *HUdata*, e um pouco inferiores no conjunto de instâncias *BCdata*. No geral, o ILS alcançou os melhores resultados, porém, quando considera-se o resultado médio de cada algoritmo, assim como o custo computacional, o CS se mostrou discretamente mais robusto que o ILS.

Os resultados detalhados dos algoritmos propostos - SA, ILS e CS - são apresentados no Apêndice B e estão organizados da seguinte forma: Tabelas 33, 34, 35 apontam, respectivamente, os resultados dos conjuntos de instâncias *BRdata*, *DPdata* e *BCdata*. Cada tabela apresenta o nome da instância (I) e, para cada algoritmo proposto, lista-se o melhor valor da função objetivo (*makespan*) encontrado em 10 execuões (FO), seguido da média desse valor (Av), da média dos tempo computacionais (T(s)) e do desvio padrão (DP) dos melhores valores de FO obtidos nas dez execuções dos algoritmos. De modo a facilitar a visualização, os melhores valores de FO da comparação são grifados em negrito.

Para o conjunto de instâncias *HUdata*, os resultados são subdivididos em *edata*, *vdata* e *rdata*, apresentando-se respectivamente nas Tabelas 36, 37, 38, 39, 40 e 41, que são organizadas da seguinte forma: a primeira coluna indica o nome da instância e, para cada algoritmo, lista-se o melhor valor da função objetivo (*makespan*) encontrado em 10 execuões (FO), seguido da média desses valores (Av), da média dos tempos computacionais (T(s)) e do desvio padrão (DP) dos melhores valores da função objetivo obtidos nas dez execuções dos algoritmos.

Considerando que somente em Mastrolilli e Gambardella (2000) são relatados os resultados detalhados para o conjunto de instâncias *HUdata*, nas Tabelas 36, 37, 38, 39, 40 e 41, além de apresentar os resultados dos algoritmos propostos, compara-se os mesmos com os gerados pelo TS (MASTROLILLI; GAMBARDELLA, 2000). De modo a facilitar a visualização, os melhores valores de FO da comparação são grifados em negrito e na última linha é indicado o número de BKS que cada algoritmo atingiu.

Apresenta-se na Figura 22 o comportamento do *makespan* da instância Mk10 do conjunto *BRdata* a partir da aplicação do GRASP, SA, ILS e CS. O tempo total de processamento foi de 1800 segundos, no qual registrou-se o valor do makespan e o tempo de cada nova melhor solução encontrada por cada método. Quando comparado com os demais algoritmos, constata-se que o GRASP tende a ficar preso em ótimos locais, sendo que não conseguiu melhorar o valor do *makespan* ao atingir a faixa de 290. Considerando que o SA, ILS e CS tem como base a utilização do SA, seus resultados apresentam características semelhantes, atingindo resultados superiores ao GRASP. Porém, o SA apresenta uma convergência prematura quando comparado ao CS, isto é, antes de 200 segundos de execução, o SA não consegue apresentar novas melhoras. Por outro lado, o CS consegue

obter resultados superiores ao SA, ficando um pouco atrás do ILS, que alcança melhores resultados depois de 1400 segundos de execução do algoritmo. Isso se deve à incorporação de características de diversificação/intensificação incorporadas ao ILS e CS.

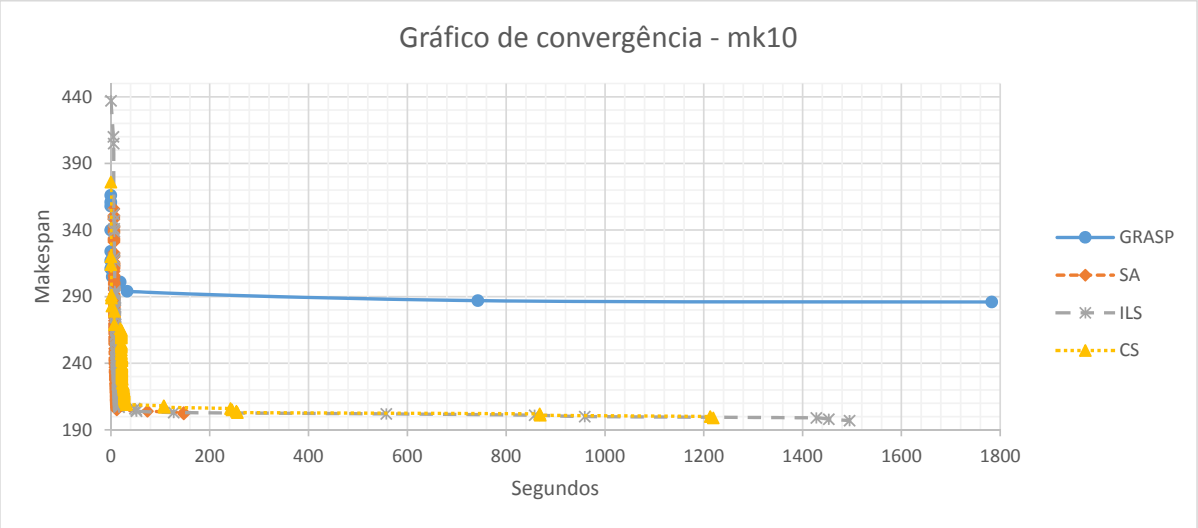

Figura 22 – Gráfico de convergência - Instância Mk10.

5.4.3 Comparação dos Resultados computacionais com os encontrados na literatura

Considerando o conjunto de instâncias *BRdata*, a Tabela 11 apresenta uma comparação dos resultados dos algoritmos propostos (SA, ILS e CS), com os apresentados em EOPDHS (GAHAM; BOUZOUIA; ACHOUR, 2017), HABC (LI et al., 2017), QPSO (SINGH; MAHAPATRA, 2016), DHS (GAO et al., 2016), melhores resultados de Previero (2016), SA-PC (CRUZ-CHÁVEZ; MARTÍNEZ-RANGEL; CRUZ-ROSALES, 2015), HmcDGA (ISHIKAWA; KUBOTA; HORIO, 2015), SSPR (GONZÁLEZ; VELA; VARELA, 2015), eGA (ZHANG; GAO; SHI, 2011) e TS (MASTROLILLI; GAMBARDELLA, 2000). Os resultados da Tabela 11, assim como da Tabela 12 e da Tabela 13, são apresentados da seguinte forma: a coluna I lista o nome das instâncias e as demais colunas indicam os melhores valores de *makespan* encontrado por cada método; ao final da tabela, apresenta-se a média geral dos melhores valores do *makespan* de cada método, assim como o número de vezes que cada método atingiu o melhor valor conhecido (#BKS). Os valores BKS estão grifados em negrito.

Ao analisar a Tabela 11, percebe-se que não há um algoritmo que obtém todos os melhores valores conhecidos. O SSPR (GONZÁLEZ; VELA; VARELA, 2015) possui um discreto destaque, obtendo o melhor valor médio de *makespan* e chega ao melhor valor conhecido (BKS) em 8 de 10 instâncias. O pior resultado da comparação é o apresentado em Previero (2016), sendo que chegou em apenas 3 melhores valores conhecidos para o *makespan* e na média, seu resultado foi 6,5% pior que o SSPR. O ILS proposto alcançou 6

Tabela 11 – Comparação dos resultados com a literatura para o conjunto de instâncias *BRdata*

I	SA	ILS/SA	CS	EOPDHS	HABC	QPSO	DHS	Previero	SA-PC	HmcDGA	SSPR	eGA	TS
Mk01	40	40	40	40	41	**37**	40	40	40	40	40	40	40
Mk02	27	**26**	**26**	26	**26**	**26**	28	**26**	28	**26**	**26**	**26**	**26**
Mk03	**204**	**204**	**204**	**204**	206	**204**	**204**	**204**	216	**204**	**204**	**204**	**204**
Mk04	**60**	**60**	**60**	**60**	62	**60**	**60**	**60**	**60**	**60**	**60**	**60**	**60**
Mk05	173	173	173	172	176	173	172	176	**168**	175	172	173	173
Mk06	60	58	59	60	60	64	67	65	59	60	**57**	58	58
Mk07	140	**139**	**139**	**139**	140	**139**	143	150	147	144	**139**	144	144
Mk08	**523**	**523**	**523**	**523**	526	**523**	**523**	524	524	**523**	**523**	**523**	**523**
Mk09	**307**	**307**	**307**	**307**	**307**	**307**	309	341	**307**	311	**307**	**307**	**307**
Mk10	202	197	199	207	208	205	212	250	197	217	**196**	198	198
Média	173,60	172,70	173,00	173,80	175,20	173,80	175,80	183,60	174,60	176,00	172,40	173,30	173,30
#BKS	4	6	6	5	2	7	3	3	3	4	8	5	5

de 10 BKS, ficando distante aproximadamente 0,17% do resultado médio do SSPR. No entanto, ao analisar a média geral de cada algoritmo, entende-se que estão muito próximos.

A Tabela 12 apresenta os resultados dos algoritmos propostos para o conjunto de instâncias *DPdata* (SA, ILS e CS). Os resultados são comparados com o QPSO (SINGH; MAHAPATRA, 2016), os melhores resultados de Previero (2016), SSPR (GONZÁLEZ; VELA; VARELA, 2015), eGA (ZHANG; GAO; SHI, 2011) e TS (MASTROLILLI; GAMBARDELLA, 2000).

Tabela 12 – Comparação dos resultados com a literatura para o conjunto de instâncias *DPdata*

I	SA	ILS/SA	CS	QPSO	Previero	SSPR	eGA	TS
1a	2531	**2505**	**2505**	**2505**	2762	**2505**	2516	2518
2a	2231	2231	2231	2230	2424	**2229**	2231	2231
3a	2230	2229	2229	2229	2355	**2228**	2232	2229
4a	2506	2503	2503	**2498**	2670	2503	2515	2503
5a	2218	2215	2213	**2207**	2376	2211	2208	2216
6a	2216	2202	2202	**2170**	2254	2183	2174	2203
7a	2320	2286	2291	2264	2595	2274	**2217**	2283
8a	2072	2067	2066	2073	2265	**2064**	2073	2069
9a	2067	2064	2064	2066	2378	**2062**	2066	2066
10a	2313	2277	2277	2205	2585	2269	**2189**	2291
11a	2068	2064	2064	**2050**	2229	2051	2063	2063
12a	2042	2029	2033	2019	2492	**2018**	2019	2034
13a	2274	2259	2261	2253	2708	2248	**2194**	2260
14a	2940	2169	2166	2167	2448	**2163**	2167	2167
15a	2168	2163	2165	2165	3287	**2162**	2165	2167
16a	2269	2255	2253	2252	2595	2244	**2211**	2255
17a	2962	2151	2145	2134	2696	2130	**2109**	2141
18a	2151	2134	2137	2123	3164	2119	**2089**	2137
Média	2309,9	2211,3	2211,4	2200,6	2571,3	2203,5	2191,0	2212,9
#BKS	0	1	1	5	0	8	**6**	0

Em relação aos resultados da Tabela 12, ressalta-se que algumas referências da Tabela 11 não a utilizam, isto é, não realizaram testes no conjunto *DPdata* e por esse motivo não são listados. Ao avaliar a comparação, constata-se que não há também neste conjunto de instâncias, um algoritmo que domine totalmente os resultados. O eGA obteve a melhor média dos melhores valores alcançados no conjunto de instâncias. No entando, o SSPR alcançou o maior número de BKS, 8 de 18 melhores valores conhecidos, 2 a mais que o eGA. Considerando os algoritmos propostos, o ILS e o CS atingiram apenas um melhor valor conhecido, mas na média geral, ficam aproximadamente apenas 0,93% distante da melhor média obtida, pelo eGA. Para esse conjunto de instâncias, o SA proposto, assim como Previero, não alcançou nenhum valor BKS, porém, na média, o SA apresentou

resultados aproximadamente 11% melhores que os do Previero.

Na Tabela 13 apresenta-se os resultados dos algoritmos propostos para o conjunto de instâncias *BCdata*. Os resultados são comparados com o os melhores resultados de Previero (2016), SSPR (GONZÁLEZ; VELA; VARELA, 2015), eGA (ZHANG; GAO; SHI, 2011) e TS (MASTROLILLI; GAMBARDELLA, 2000).

Tabela 13 – Comparação dos resultados com a literatura para o conjunto de instâncias *BCdata*

I	SA	ILS/SA	CS	Previero	SSPR	eGA	TS
mt10c1	**927**	**927**	**927**	**927**	**927**	928	928
mt10cc	910	910	910	**908**	**908**	910	910
mt10x	922	922	922	**918**	**918**	**918**	**918**
mt10xx	**918**	**918**	**918**	**918**	**918**	**918**	**918**
mt10xxx	**918**	**918**	**918**	**918**	**918**	**918**	**918**
mt10xy	906	906	**905**	**905**	**905**	906	906
mt10xyz	858	858	858	**847**	**847**	**847**	**847**
setb4c9	**914**	**914**	**914**	**914**	**914**	919	919
setb4cc	909	909	909	**907**	**907**	909	909
setb4x	**925**	**925**	**925**	**925**	**925**	**925**	**925**
setb4xx	**925**	**925**	**925**	**925**	**925**	**925**	**925**
setb4xxx	**925**	**925**	**925**	**925**	**925**	**925**	**925**
setb4xy	913	**910**	**910**	**910**	**910**	916	916
setb4xyz	905	905	905	**902**	905	905	905
seti5c12	1174	1174	1174	1174	**1170**	1174	1174
seti5cc	1136	1136	1136	1136	**1135**	1136	1136
seti5x	1205	1202	**1198**	1205	**1198**	1201	1201
seti5xx	1198	1198	1198	1200	**1197**	1199	1199
seti5xxx	1203	1203	1198	1200	**1194**	1197	1197
seti5xy	1138	1136	1136	1136	**1135**	1136	1136
seti5xyz	1128	1128	1127	1129	**1125**	**1125**	**1125**
Média	998,0	997,6	997,0	996,6	995,5	997,0	997,0
#BKS	7	8	10	14	20	8	8

Assim como na Tabela 12, na Tabela 13 utiliza-se um menor número de tabalhos na comparação em relação a Tabela 11, pois não são todas as referências que utilizaram o conjunto de instâncias *BCdata* nos testes computacionais. Na média, os resultados dos algoritmos dessa comparação apresentam-se muito próximos, sendo que da pior média (SA) de resultados para a melhor (SSPR), tem-se uma pequena diferença de aproximadamente 0,21%. Assim como nos demais conjuntos de instâncias analisados até então, não há um algoritmo que obtenha todos os melhores valores conhecidos. O SSPR obteve os melhores resultados da comparação, alcançando 20 de 21 BKS, e o melhor valor médio de 995,5. Dentre os algoritmos propostos, o CS possui o melhor resultado e a mesma média relatada pelo eGA e TS, mas alcançou 2 BKS a mais que os mesmos.

As Tabelas 14, 15 e 16 indicam os valores médios dos tempos de execução necessários para encontrar a melhor solução em cada execução dos respectivos algoritmos avaliados nos conjuntos de instâncias *BRdata*, *DPdata* e *BCdata*. Considerou-se nessas comparações alguns dos trabalhos que apresentaram detalhadamente os respectivos valores. Para o conjunto de instâncias *BRdata*, o SA-PC apresentou um custo computacional bastante elevado, quando comparado com os demais algoritmos. Quando considerado os algoritmos propostos (SA, ILS e CS) numa comparação com o SSPR, eGA e TS, percebe-se que nos conjuntos de instâncias *BRdata*, *DPdata* e *BCdata*, o TS apresenta-se como o algoritmo que os resultados convergem mais rapidamente. É importante ressaltar que por serem executados em computadores com configurações distintas, os resultados não podem ser comparados com exatidão, apenas como referência.

Tabela 14 – Comparação dos tempos médios de execução dos algoritmos para o conjunto de instâncias *BRdata*

I	SA [1]	ILS/SA [1]	CS [1]	EOPDHS [2]	SA-PC [3]	SSPR [4]	eGA [5]	TS [6]
Mk01	3	2	3	0	3200	11	2	0
Mk02	7	30	36	3	4600	15	3	1
Mk03	2	3	1	0	6786	24	1	0
Mk04	7	14	11	6	5467	19	6	0
Mk05	128	115	65	14	6781	57	7	1
Mk06	411	582	396	23	3783	40	16	3
Mk07	686	1120	495	42	7526	84	17	9
Mk08	4	5	1	1	8795	83	2	0
Mk09	7	10	20	36	4563	52	30	0
Mk10	40	1346	632	228	7865	94	37	8
Média	129,5	322,8	165,9	35,2	5936,6	47,9	12,1	2,2

[1]Intel Core i5-750 2,67 GHz
[2]Intel Core i7-3770 3,40 GHz
[3]PC 2,00 GHz
[4]Intel Core 2 Duo 2,66 GHz
[5]Intel Pentium IV 1,80 GHz
[6]Intel Pentium 266 MHz

Ao avaliar isoladamente as variações implementados do método SA na comparação das Tabelas 11 e 14, isto é, os resultados do SA proposto e o SA-PC (CRUZ-CHÁVEZ; MARTÍNEZ-RANGEL; CRUZ-ROSALES, 2015), percebe-se comportamentos bem diferentes. O SA proposto obteve quatro BKS, um a mais que o SA-PC, e na média, os valores da FO foram superiores em aproximadamente 0,5% apenas. Porém, ao avaliar o custo computacional, mesmo considerando que foram executados em computadores distintos, constata-se uma difereça de aproximaxamente 45 vezes que o SA proposto, na média, gerou os resultados em relação ao SA-PC. Isso reforça que, o SA proposto, mesmo alcançando resultados inferiores ao ILS e o CS, ainda se apresenta como uma alternativa robusta para

Tabela 15 – Comparação dos tempos médios de execução dos algoritmos para o conjunto de instâncias *DPdata*

I	SA [1]	ILS/SA [1]	CS [1]	SSPR [2]	eGA [3]	TS [4]
1a	721	906	270	68	98	28
2a	725	570	520	100	165	42
3a	25	746	638	110	112	64
4a	742	745	361	57	109	28
5a	628	731	655	112	155	46
6a	5	896	621	181	133	54
7a	926	1136	644	139	396	68
8a	429	1250	639	181	389	53
9a	174	602	734	213	390	81
10a	881	1480	601	120	373	56
11a	758	1099	744	193	406	61
12a	125	1262	758	280	332	82
13a	812	1112	677	119	523	49
14a	0	170	635	269	555	111
15a	6	874	616	376	765	138
16a	863	1444	665	131	494	75
17a	0	67	759	299	645	121
18a	4	828	793	409	735	149
Média	434,7	884,5	629,4	186,5	376,5	72,6

[1]Intel Core i5-750 2,67 GHz
[2]Intel Core 2 Duo 2,66 GHz
[3]Intel Pentium IV 1,80 GHz
[4]Intel Pentium 266 MHz

solucionar o FJSP.

Em relação ao conjunto de instâncias *HUdata*, o resumo dos resultados são apresentados na Tabela 17. Constata-se que o ILS chegou a 120 melhores valores conhecidos de um total de 129 instâncias, perante 109 pelo CS, 90 pelo SA e 98 pelo TS. Ressalta-se que o ILS obteve 27 novos BKS e o CS gerou 24 novos BKS. O ILS apresentou resultados, na média, aproximadamente 0,02% superior ao CS nesse conjunto de instâncias, porém todos se mostrando competitivos. Em relação ao custo computacional, o TS (MASTROLILLI; GAMBARDELLA, 2000), mesmo executado os testes em um processador mais lento (Intel Pentium 266 MHz), na média, sempre atingiu seus melhores resultados mais rapidamente que todos os métodos propostos nesse trabalho. os resultados detalhados são relacionados no Apêndice B.2 nas Tabelas 36, 37, 38, 39, 40 e 41,

Tabela 16 – Comparação dos tempos médios de execução dos algoritmos para o conjunto de instâncias *BCdata*

I	SA [1]	ILS/SA [1]	CS [1]	SSPR [2]	eGA [3]	TS [4]
mt10c1	28	25	70	26	23	2
mt10cc	189	138	238	20	19	10
mt10x	938	702	150	23	21	4
mt10xx	833	364	85	19	20	2
mt10xxx	660	691	115	20	25	1
mt10xy	307	926	155	21	24	4
mt10xyz	355	281	172	20	30	6
setb4c9	727	395	335	28	13	14
setb4cc	946	886	272	21	20	10
setb4x	574	404	404	19	9	7
setb4xx	1027	556	358	21	54	15
setb4xxx	701	446	505	22	63	8
setb4xy	937	713	347	32	28	3
setb4xyz	763	476	277	21	40	7
seti5c12	909	752	509	25	71	19
seti5cc	627	934	401	29	70	12
seti5x	823	871	461	41	68	16
seti5xx	737	1009	431	37	78	24
seti5xxx	758	1140	307	38	105	24
seti5xy	791	848	577	29	70	12
seti5xyz	879	681	477	35	71	17
Média	690,9	630,4	316,4	26,0	44,0	10,3

[1]Intel Core i5-750 2,67 GHz
[2]Intel Core 2 Duo 2,66 GHz
[3]Intel Pentium IV 1,80 GHz
[4]Intel Pentium 266 MHz

5.5 Considerações finais

Este estudo apresenta as meta-heurísticas GRASP, *Simulated Annealing* (SA) e dois métodos híbridos, *Iterated Local Search* que utiliza o *Simulated Annealing* como busca local e o *Clustering Search* para o FJSP. O FJSP é um problema que vem sendo bastante estudado recentemente, e considera que algumas operações podem ser processadas em parte das m máquinas disponíveis. Este problema é considerado um dos problemas de escalonamento mais difíceis de serem resolvidos.

Para avaliar a performance dos algoritmos propostos, utilizou-se um extenso conjunto de instâncias padrão e comparou-se os resultados com os de algoritmos bem conhecidos na literatura. Os experimentos computacionais mostraram que os métodos híbridos propostos são competitivos quando comparado com os melhores resultados existentes na literatura.

Tabela 17 – Resumo dos resultados para o conjunto *HUdata*.

	TS				ILS				CS			
I	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP
edata	1004	1005	2,8	0,9	**1003**	1005	289,3	1,1	1004	1005	226,3	1,1
rdata	909,9	911,4	4,2	1,3	**909**	910,7	362,3	1,1	909,3	910,9	350,5	0,9
vdata	895,6	895,9	4,3	0,3	**895,4**	895,7	243,2	0,2	895,5	895,9	254,8	0,3
Média	936,4	937,4	3,7	0,9	**935,9**	937,1	298,3	0,8	936,1	937,3	277,2	0,8
#BKS	98				**120**				109			

6 Conclusões e Trabalhos Futuros

O presente estudo constituiu na exploração de algumas técnicas baseadas em heurísticas para solucionar alguns problemas encontrados no contexto real de planejamento e controle da produção.

Em relação ao SALBP-2, propôs-se uma meta-heurística GRASP. A implementação proposta do GRASP foi testada em um grande conjunto de instâncias da literatura. Suas soluções foram comparadas com as melhores soluções conhecidas na literatura. Os resultados computacionais mostram que o algoritmo GRASP obtém bons resultados quando comparado à literatura, isto é, alcançando, na média, melhores valores da função objetivo. Além disso, a característica de obter automaticamente valores para o parâmetro alfa, incorporada no GRASP clássico, tende a facilitar sua aplicação em casos reais do problema.

Para o JSP, foi apresentada uma implementação do GRASP, que foi testado em um conjunto de 43 instâncias padrões e comparado com os melhores valores conhecidos (KURDI, 2016). Os resultados computacionais mostram que o algoritmo GRASP atinge os mesmos valores das melhores soluções conhecidas (BKS) ou próximos nas instâncias testadas. De maneira geral, o algoritmo GRASP produz resultados com desvio relativo médio de $4,1\%$ em relação ao BKS.

Considerando o FJSP, extensão do JSP, desenvolveu-se as meta-heurísticas GRASP, *Simulated Annealing* (SA) e dois métodos híbridos, *Iterated Local Search* que utiliza o *Simulated Annealing* como busca local e o *Clustering Search* para o FJSP. O FJSP estende o JSP, considerando que algumas operações podem ser processadas em parte das m máquinas disponíveis, isto é, trata-se de dois problemas simultâneos: roteamento e escalonamento. Para avaliar a performance dos algoritmos propostos, utilizou-se um total de 178 instâncias padrões com diferentes tamanhos e complexidade. Os resultados foram comparados com os apresentados por algoritmos bem conhecidos e recentes na literatura. Os experimentos computacionais mostraram que os métodos híbridos propostos são competitivos quando comparado com os melhores resultados existentes na literatura.

6.1 Trabalhos futuros

Os métodos aplicados nesse estudo apresentaram resultados interessantes para os problemas abordados. Entretanto, com o objetivo de aproximar os problemas do real contexto industrial, que tem a necessidade de considerar mais de um critério simultaneamente, sugere-se a pesquisa de extensões dos problemas, em especial, a inserção de

novos objetivos nos problemas clássicos, para serem considerados simultaneamente, isto é, realizando abordagens multi-objetivo, especialmente quando relacionados à datas de entrega.

Outra abordagem interessante pode ser realizada visando integrar os problemas abordados nessa tese, tratando-os de forma dinâmica. Fatores de incerteza, como cancelamento de pedidos, falta de funcionários ou quebra de máquinas, se forem considerados, tendem a agregar economia no processo de produção.

Referências

AHONEN, H.; ALVARENGA, A. de; AMARAL, A. Simulated annealing and tabu search approaches for the corridor allocation problem. *European Journal of Operational Research*, v. 232, n. 1, p. 221 – 233, 2014. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221713005808>. Citado na página 32.

AIEX, R.; BINATO, S.; RESENDE, M. Parallel grasp with path-relinking for job shop scheduling. *Parallel Computing*, v. 29, n. 4, p. 393 – 430, 2003. ISSN 0167-8191. Parallel computing in numerical optimization. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167819103000140>. Citado 3 vezes nas páginas 49, 50 e 56.

AL-HINAI, N.; ELMEKKAWY, T. Y. An efficient hybridized genetic algorithm architecture for the flexible job shop scheduling problem. *Flexible Services and Manufacturing Journal*, v. 23, n. 1, p. 64–85, Mar 2011. ISSN 1936-6590. Disponível em: <https://doi.org/10.1007/s10696-010-9067-y>. Citado 2 vezes nas páginas 63 e 64.

APPLEGATE, D.; COOK, W. A computational study of the job-shop scheduling problem. *ORSA Journal on Computing*, v. 3, n. 2, p. 149–156, 1991. Disponível em: <https://doi.org/10.1287/ijoc.3.2.149>. Citado na página 63.

ARENALES, M. et al. *Pesquisa operacional*: Para cursos de engenharia. [S.l.]: Elsevier, 2007. Citado na página 23.

BAGHERI, A. et al. An artificial immune algorithm for the flexible job-shop scheduling problem. *Future Generation Computer Systems*, v. 26, n. 4, p. 533 – 541, 2010. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X0900154X>. Citado na página 64.

BARNES, J.; CHAMBERS, J. *Flexible job shop scheduling by tabu search.* 1996. Report Series: ORP96-09. Graduate program in operations research and industrial engineering. The University of Texas at Austin. Citado na página 83.

BAUTISTA, J.; BATALLA-GARCÍA, C.; ALFARO-POZO, R. Models for assembly line balancing by temporal, spatial and ergonomic risk attributes. *European Journal of Operational Research*, v. 251, n. 3, p. 814 – 829, 2016. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221715011819>. Citado na página 37.

BAUTISTA, J.; PEREIRA, J. A dynamic programming based heuristic for the assembly line balancing problem. *European Journal of Operational Research*, v. 194, n. 3, p. 787 – 794, 2009. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221708001446>. Citado 2 vezes nas páginas 37 e 38.

BAYBARS, İ. A survey of exact algorithms for the simple assembly line balancing problem. *Management Science*, v. 32, n. 8, p. 909–932, 1986. Disponível em: <https://doi.org/10.1287/mnsc.32.8.909>. Citado 3 vezes nas páginas 36, 38 e 40.

BAYKASOGLU, A. Multi-rule multi-objective simulated annealing algorithm for straight and u type assembly line balancing problems. *Journal of Intelligent Manufacturing*, v. 17, n. 2, p. 217–232, Apr 2006. ISSN 1572-8145. Disponível em: <https://doi.org/10.1007/s10845-005-6638-y>. Citado na página 37.

BAZARAA, M. S.; JARVIS, J. J.; SHERALI, H. D. *Linear Programming and Network Flows*. 4th. ed. UHoboken, New Jersey, USA: John Wiley & Sons, 2009. 768 p. ISBN 0-13-152462-3. Citado na página 27.

BECKER, C.; SCHOLL, A. A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, v. 168, n. 3, p. 694 – 715, 2006. ISSN 0377-2217. Balancing Assembly and Transfer lines. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221704004801>. Citado 3 vezes nas páginas 35, 36 e 38.

BESTEN, M. den; STÜTZLE, T.; DORIGO, M. Design of iterated local search algorithms. In: _____. *Applications of Evolutionary Computing: EvoWorkshops 2001: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM Como, Italy, April 18–20, 2001 Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. p. 441–451. ISBN 978-3-540-45365-9. Disponível em: <https://doi.org/10.1007/3-540-45365-2_46>. Citado na página 32.

BINATO, S. et al. A grasp for job shop scheduling. In: _____. *Essays and Surveys in Metaheuristics*. Boston, MA: Springer US, 2002. p. 59–79. ISBN 978-1-4615-1507-4. Disponível em: <https://doi.org/10.1007/978-1-4615-1507-4_3>. Citado na página 53.

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, Apr 2014. ISSN 1862-4480. Disponível em: <https://doi.org/10.1007/s11590-013-0669-7>. Citado 5 vezes nas páginas 62, 63, 65, 67 e 68.

BISSOLI, D. C.; ALTOÉ, W. A. S.; AMARAL, A. R. S. Resolução do problema job shop scheduling flexível utilizando a meta-heurística simulated annealing. In: *XX ENMC - Encontro Nacional de Modelagem Computacional*. Nova Friburgo-RJ: [s.n.], 2017. ISBN 978-85-5676-019-7. ISSN 2527-2357. Citado na página 32.

BISSOLI, D. C.; AMARAL, A. R. S. Um grasp simples e robusto para o job shop scheduling problem. In: *XLVII SBPO - Simpósio Brasileiro de Pesquisa Operacional*. Porto de Galinhas, Pernambuco-PE: [s.n.], 2015. p. 2057–2067. Disponível em: <http://www.din.uem.br/sbpo/sbpo2015/pdf/142810.pdf>. Citado na página 31.

BISSOLI, D. C.; AMARAL, A. R. S. Um algoritmo grasp para o salbp-2. In: *XLVIII SBPO - Simpósio Brasileiro de Pesquisa Operacional*. Vitória-ES: [s.n.], 2016. p. 2081–2091. Disponível em: <http://www.din.uem.br/sbpo/sbpo2016/pdf/155608.pdf>. Citado na página 31.

BLUM, C. Iterative beam search for simple assembly line balancing with a fixed number of work stations. *SORT - Statistics and Operations Research Transactions*, v. 35(2), p. 145–164, 2011. Disponível em: <https://www.raco.cat/index.php/SORT/article/view/248347/332463>. Citado na página 44.

BOWMAN, E. H. Assembly-line balancing by linear programming. *Operations Research*, v. 8, n. 3, p. 385–389, 1960. Disponível em: <https://doi.org/10.1287/opre.8.3.385>. Citado 2 vezes nas páginas 37 e 38.

BOYSEN, N.; FLIEDNER, M. A versatile algorithm for assembly line balancing. *European Journal of Operational Research*, v. 184, n. 1, p. 39 – 56, 2008. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221706011362>. Citado na página 39.

BOZEJKO, W.; UCHROŃSKI, M.; WODECKI, M. Parallel hybrid metaheuristics for the flexible job shop problem. *Computers & Industrial Engineering*, v. 59, n. 2, p. 323 – 333, 2010. ISSN 0360-8352. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360835210001191>. Citado na página 63.

BRANDIMARTE, P. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, v. 41, n. 3, p. 157–183, Sep 1993. ISSN 1572-9338. Disponível em: <https://doi.org/10.1007/BF02023073>. Citado 3 vezes nas páginas 63, 64 e 83.

BRUCKER, P. The job-shop problem: Old and new challenges. In: *in Proceedings of the MISTA Conference 2007*. [S.l.: s.n.], 2007. p. 15–22. Citado na página 74.

BRUCKER, P.; KNUST, S. *Complex Scheduling*. 2. ed. [S.l.]: Springer-Verlag Berlin Heidelberg, 2012. (GOR-Publications). ISBN 3642239285,9783642239281. Citado na página 74.

BRUCKER, P.; SCHLIE, R. Job-shop scheduling with multi-purpose machines. *Computing*, v. 45, n. 4, p. 369–375, Dec 1990. ISSN 1436-5057. Disponível em: <https://doi.org/10.1007/BF02238804>. Citado 2 vezes nas páginas 25 e 62.

ÇALIŞ, B.; BULKAN, S. A research survey: review of ai solution strategies of job shop scheduling problem. *Journal of Intelligent Manufacturing*, v. 26, n. 5, p. 961–973, Oct 2015. ISSN 1572-8145. Disponível em: <https://doi.org/10.1007/s10845-013-0837-8>. Citado na página 52.

CASTILLO, E. et al. Mixed-integer linear programming. In: _____. *Building and Solving Mathematical Programming Models in Engineering and Science*. John Wiley & Sons, Inc., 2001. p. 25–46. ISBN 9780471225294. Disponível em: <http://dx.doi.org/10.1002/9780471225294.ch2>. Citado na página 67.

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. ISSN 1475-3995. Disponível em: <http://dx.doi.org/10.1111/itor.12199>. Citado 4 vezes nas páginas 62, 63, 65 e 66.

CHAVES, A. A. *Uma Meta-heurística Híbrida com Busca por Agrupamentos Aplicada a Problemas de Otimização Combinatória*. Tese (Doutorado) — Instituto Nacional de Pesquisas Espaciais - INPE, São José dos Campos, 2009. Doutorado em Computação Aplicada. Citado 2 vezes nas páginas 33 e 34.

CHEN, H.; IHLOW, J.; LEHMANN, C. A genetic algorithm for flexible job-shop scheduling. In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. [S.l.: s.n.], 1999. v. 2, p. 1120–1125 vol.2. ISSN 1050-4729. Citado 2 vezes nas páginas 63 e 64.

CHENG, R.; GEN, M.; TSUJIMURA, Y. A tutorial survey of job-shop scheduling problems using genetic algorithms—i. representation. *Computers & Industrial Engineering*, v. 30, n. 4, p. 983 – 997, 1996. ISSN 0360-8352. Disponível em:

<http://www.sciencedirect.com/science/article/pii/0360835296000472>. Citado na página 52.

CHENG, R.; GEN, M.; TSUJIMURA, Y. A tutorial survey of job-shop scheduling problems using genetic algorithms, part ii: hybrid genetic search strategies. *Computers & Industrial Engineering*, v. 36, n. 2, p. 343 – 364, 1999. ISSN 0360-8352. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360835299001369>. Citado na página 52.

ÇIL, Z. A. et al. A beam search approach for solving type ii robotic parallel assembly line balancing problem. *Applied Soft Computing*, v. 61, n. Supplement C, p. 129 – 138, 2017. ISSN 1568-4946. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1568494617304829>. Citado na página 37.

CINAR, D.; TOPCU, Y. I.; OLIVEIRA, J. A. A taxonomy for the flexible job shop scheduling problem. In: _____. *Optimization, Control, and Applications in the Information Age: In Honor of Panos M. Pardalos's 60th Birthday*. Cham: Springer International Publishing, 2015. p. 17–37. ISBN 978-3-319-18567-5. Citado 2 vezes nas páginas 62 e 63.

COLIN, E. C. *Pesquisa Operacional - 170 Aplicações em Estratégia*. 1. ed. [S.l.]: LTC, 2007. ISBN 9788521615590. Citado na página 28.

CRUZ-CHÁVEZ, M. A.; MARTÍNEZ-RANGEL, M. G.; CRUZ-ROSALES, M. H. Accelerated simulated annealing algorithm applied to the flexible job shop scheduling problem. *International Transactions in Operational Research*, p. 1–19, 2015. ISSN 1475-3995. Disponível em: <http://dx.doi.org/10.1111/itor.12195>. Citado 5 vezes nas páginas 63, 72, 79, 90 e 94.

DAUZÈRE-PÉRÈS, S.; PAULLI, J. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research*, v. 70, n. 0, p. 281–306, 1997. ISSN 1572-9338. Citado 3 vezes nas páginas 63, 67 e 83.

DEMIR, Y.; İşLEYEN, S. K. Evaluation of mathematical models for flexible job-shop scheduling problems. *Applied Mathematical Modelling*, v. 37, n. 3, p. 977 – 988, 2013. ISSN 0307-904X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0307904X12001837>. Citado na página 62.

EREL, E.; SARIN, S. C. A survey of the assembly line balancing procedures. *Production Planning & Control*, Taylor & Francis, v. 9, n. 5, p. 414–434, 1998. Disponível em: <http://dx.doi.org/10.1080/095372898233902>. Citado 3 vezes nas páginas 36, 37 e 38.

FATTAHI, A.; TURKAY, M. On the milp model for the u-shaped assembly line balancing problems. *European Journal of Operational Research*, v. 242, n. 1, p. 343 – 346, 2015. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221714008583>. Citado na página 37.

FATTAHI, P.; MEHRABAD, M. S.; JOLAI, F. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, v. 18, n. 3, p. 331–342, Jun 2007. ISSN 1572-8145. Disponível em: <https://doi.org/10.1007/s10845-007-0026-8>. Citado na página 62.

FATTAHI, P.; ROSHANI, A.; ROSHANI, A. A mathematical model and ant colony algorithm for multi-manned assembly line balancing problem. *The International Journal of Advanced Manufacturing Technology*, v. 53, n. 1, p. 363–378, Mar 2011. ISSN 1433-3015. Disponível em: <https://doi.org/10.1007/s00170-010-2832-y>. Citado na página 37.

FEO, T. A.; RESENDE, M. G. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, v. 8, n. 2, p. 67 – 71, 1989. ISSN 0167-6377. Disponível em: <http://www.sciencedirect.com/science/article/pii/0167637789900023>. Citado 2 vezes nas páginas 25 e 29.

FESTA, P.; RESENDE, M. G. Grasp: Basic components and enhancements. *Telecommun. Syst.*, Kluwer Academic Publishers, Norwell, MA, USA, v. 46, n. 3, p. 253–271, mar. 2011. ISSN 1018-4864. Disponível em: <http://dx.doi.org/10.1007/s11235-010-9289-z>. Citado na página 29.

FESTA, P.; RESENDE, M. G. C. An annotated bibliography of grasp – part i: Algorithms. *International Transactions in Operational Research*, Blackwell Publishing Ltd, v. 16, n. 1, p. 1–24, 2009. ISSN 1475-3995. Disponível em: <http://dx.doi.org/10.1111/j.1475-3995.2009.00663.x>. Citado na página 29.

FESTA, P.; RESENDE, M. G. C. An annotated bibliography of grasp–part ii: Applications. *International Transactions in Operational Research*, Blackwell Publishing Ltd, v. 16, n. 2, p. 131–172, 2009. ISSN 1475-3995. Disponível em: <http://dx.doi.org/10.1111/j.1475-3995.2009.00664.x>. Citado 2 vezes nas páginas 29 e 31.

FISHER, H.; THOMPSON, G. L. Probabilistic learning combinations of local job-shop scheduling rules. In: MUTH, J. F.; THOMPSON, G. L. (Ed.). *Industrial Scheduling*. Englewood Cliffs, NJ: Prentice-Hall, 1963. p. 225–251. Citado na página 55.

GAHAM, M.; BOUZOUIA, B.; ACHOUR, N. An effective operations permutation-based discrete harmony search approach for the flexible job shop scheduling problem with makespan criterion. *Applied Intelligence*, Aug 2017. ISSN 1573-7497. Disponível em: <https://doi.org/10.1007/s10489-017-0993-1>. Citado 2 vezes nas páginas 64 e 90.

GAO, J.; SUN, L.; GEN, M. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, v. 35, n. 9, p. 2892 – 2907, 2008. ISSN 0305-0548. Part Special Issue: Bio-inspired Methods in Combinatorial Optimization. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054807000020>. Citado 2 vezes nas páginas 63 e 64.

GAO, K. Z. et al. Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives. *Journal of Intelligent Manufacturing*, v. 27, n. 2, p. 363–374, Apr 2016. ISSN 1572-8145. Disponível em: <https://doi.org/10.1007/s10845-014-0869-8>. Citado 2 vezes nas páginas 64 e 90.

GAREY, M. R.; JOHNSON, D. S.; SETHI, R. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, v. 1, n. 2, p. 117–129, 1976. Disponível em: <https://doi.org/10.1287/moor.1.2.117>. Citado na página 52.

GENDREAU, M.; POTVIN, J.-Y. *Handbook of Metaheuristics*. [S.l.]: Springer US, 2010. v. 2. 648 p. (146, v. 2). ISBN 9781441916655. Citado 2 vezes nas páginas 24 e 32.

GHOSH, S.; GAGNON, R. J. A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *International Journal of Production Research*, Taylor & Francis, v. 27, n. 4, p. 637–670, 1989. Disponível em: <http://dx.doi.org/10.1080/00207548908942574>. Citado na página 36.

GLOVER, F. W.; KOCHENBERGER, G. A. *Handbook of Metaheuristics*. 1. ed. [S.l.]: Springer US, 2003. (International Series in Operations Research & Management Science). ISBN 9780306480560. Citado na página 29.

GONÇALVES, J. F.; ALMEIDA, J. R. de. A hybrid genetic algorithm for assembly line balancing. *Journal of Heuristics*, v. 8, n. 6, p. 629–642, 2002. ISSN 1572-9397. Disponível em: <http://dx.doi.org/10.1023/A:1020377910258>. Citado 2 vezes nas páginas 39 e 45.

GONZÁLEZ, M. A.; VELA, C. R.; VARELA, R. An efficient memetic algorithm for the flexible job shop with setup times. In: *Proceedings of the Twenty-Third International Conference on International Conference on Automated Planning and Scheduling*. AAAI Press, 2013. (ICAPS'13), p. 91–99. Disponível em: <http://dl.acm.org/citation.cfm?id=3038718.3038730>. Citado na página 63.

GONZÁLEZ, M. A.; VELA, C. R.; VARELA, R. Scatter search with path relinking for the flexible job shop scheduling problem. *European Journal of Operational Research*, v. 245, n. 1, p. 35 – 45, 2015. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221715001721>. Citado 4 vezes nas páginas 63, 90, 92 e 93.

GUROBI. *Gurobi Optimizer Reference Manual*. [S.l.], 2016. Disponível em: <https://www.gurobi.com>. Citado na página 63.

GUTIÉRREZ, C.; GARCÍA-MAGARIÑO, I. Modular design of a hybrid genetic algorithm for a flexible job–shop scheduling problem. *Knowledge-Based Systems*, v. 24, n. 1, p. 102 – 112, 2011. ISSN 0950-7051. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0950705110001243>. Citado na página 63.

HAMMING, R. W. Error detecting and error correcting code. *Bell Labs Technical Journal*, v. 29, n. 2, p. 147 – 160, 1950. Citado na página 80.

HAMTA, N. et al. A hybrid pso algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. *International Journal of Production Economics*, v. 141, n. 1, p. 99 – 111, 2013. ISSN 0925-5273. Meta-heuristics for manufacturing scheduling and logistics problems. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925527312001090>. Citado na página 37.

HANSEN, P.; MLADENOVIĆ, N. First vs. best improvement: An empirical study. *Discrete Applied Mathematics*, v. 154, n. 5, p. 802 – 817, 2006. ISSN 0166-218X. IV ALIO/EURO Workshop on Applied Combinatorial Optimization. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0166218X05003070>. Citado 2 vezes nas páginas 70 e 81.

HMIDA, A. B. et al. Discrepancy search for the flexible job shop scheduling problem. *Computers & Operations Research*, v. 37, n. 12, p. 2192 – 2201, 2010. ISSN 0305-0548.

Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054810000638>. Citado na página 63.

HO, N. B.; TAY, J. C. Genace: an efficient cultural algorithm for solving the flexible job-shop problem. In: *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*. [S.l.: s.n.], 2004. v. 2, p. 1759–1766 Vol.2. Citado 2 vezes nas páginas 61 e 65.

HURINK, J.; JURISCH, B.; THOLE, M. Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum*, v. 15, n. 4, p. 205–215, Dec 1994. ISSN 1436-6304. Disponível em: <https://doi.org/10.1007/BF01719451>. Citado 2 vezes nas páginas 63 e 83.

HWANG, R.; KATAYAMA, H. Integrated procedure of balancing and sequencing for mixed-model assembly lines: a multi-objective evolutionary approach. *International Journal of Production Research*, Taylor & Francis, v. 48, n. 21, p. 6417–6441, 2010. Disponível em: <http://dx.doi.org/10.1080/00207540903289755>. Citado na página 37.

IDA, K.; OKA, K. Flexible job-shop scheduling problem by genetic algorithm. *Electrical Engineering in Japan*, Wiley Subscription Services, Inc., A Wiley Company, v. 177, n. 3, p. 28–35, 2011. ISSN 1520-6416. Disponível em: <http://dx.doi.org/10.1002/eej.21194>. Citado 2 vezes nas páginas 63 e 64.

ISHIKAWA, S.; KUBOTA, R.; HORIO, K. Effective hierarchical optimization by a hierarchical multi-space competitive genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, v. 42, n. 24, p. 9434 – 9440, 2015. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417415005394>. Citado 2 vezes nas páginas 63 e 90.

JAIN, A.; MEERAN, S. Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, v. 113, n. 2, p. 390 – 434, 1999. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221798001131>. Citado na página 52.

JOHNSON, S. M. Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, Wiley Subscription Services, Inc., A Wiley Company, v. 1, n. 1, p. 61–68, 1954. ISSN 1931-9193. Disponível em: <http://dx.doi.org/10.1002/nav.3800010110>. Citado 2 vezes nas páginas 24 e 52.

JONES, A.; RABELO, L. C.; SHARAWI, A. T. Survey of job shop scheduling techniques. In: _____. *Wiley Encyclopedia of Electrical and Electronics Engineering*. John Wiley & Sons, Inc., 2001. ISBN 9780471346081. Disponível em: <http://dx.doi.org/10.1002/047134608X.W3352>. Citado na página 52.

KACEM, I.; HAMMADI, S.; BORNE, P. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, v. 32, n. 1, p. 1–13, Feb 2002. ISSN 1094-6977. Citado 3 vezes nas páginas 61, 65 e 66.

KILINCCI, O. A petri net-based heuristic for simple assembly line balancing problem of type 2. *The International Journal of Advanced Manufacturing Technology*, v. 46, n. 1, p. 329–338, Jan 2010. ISSN 1433-3015. Disponível em: <https://doi.org/10.1007/s00170-009-2082-z>. Citado na página 39.

KIM, Y. K.; KIM, Y. J.; KIM, Y. Genetic algorithms for assembly line balancing with various objectives. *Computers & Industrial Engineering*, v. 30, n. 3, p. 397 – 409, 1996. ISSN 0360-8352. IE in Korea. Disponível em: <http://www.sciencedirect.com/science/article/pii/0360835296000095>. Citado 2 vezes nas páginas 39 e 45.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science*, American Association for the Advancement of Science, v. 220, n. 4598, p. 671–680, 1983. ISSN 0036-8075. Disponível em: <http://science.sciencemag.org/content/220/4598/671>. Citado 3 vezes nas páginas 25, 31 e 32.

KLEIN, R.; SCHOLL, A. Maximizing the production rate in simple assembly line balancing – a branch and bound procedure. *European Journal of Operational Research*, v. 91, n. 2, p. 367 – 385, 1996. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/037722179500047X>. Citado na página 38.

KNUTH, D. E. Postscript about np-hard problems. *SIGACT News*, ACM, New York, NY, USA, v. 6, n. 2, p. 15–16, abr. 1974. ISSN 0163-5700. Disponível em: <http://doi.acm.org/10.1145/1008304.1008305>. Citado na página 23.

KOULAMAS, C.; ANTONY, S.; JAEN, R. A survey of simulated annealing applications to operations research problems. *Omega*, v. 22, n. 1, p. 41 – 56, 1994. ISSN 0305-0483. Disponível em: <http://www.sciencedirect.com/science/article/pii/030504839490006X>. Citado na página 32.

KUCUKKOC, I.; ZHANG, D. Z. Type-e parallel two-sided assembly line balancing problem: Mathematical model and ant colony optimisation based approach with optimised parameters. *Computers & Industrial Engineering*, v. 84, n. Supplement C, p. 56 – 69, 2015. ISSN 0360-8352. Intelligent Enterprise Systems. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360835214004665>. Citado na página 37.

KURDI, M. An effective new island model genetic algorithm for job shop scheduling problem. *Computers & Operations Research*, v. 67, n. Supplement C, p. 132 – 142, 2016. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054815002361>. Citado 4 vezes nas páginas 52, 56, 58 e 99.

LAARHOVEN, P. J. M.; AARTS, E. H. L. (Ed.). *Simulated Annealing: Theory and Applications*. Norwell, MA, USA: Kluwer Academic Publishers, 1987. ISBN 9-027-72513-6. Citado na página 32.

LAGUNA, M.; MARTI, R. Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, v. 11, n. 1, p. 44–52, 1999. Disponível em: <https://doi.org/10.1287/ijoc.11.1.44>. Citado na página 52.

LAPIERRE, S. D.; RUIZ, A.; SORIANO, P. Balancing assembly lines with tabu search. *European Journal of Operational Research*, v. 168, n. 3, p. 826 – 837, 2006. ISSN 0377-2217. Balancing Assembly and Transfer lines. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221704004886>. Citado 2 vezes nas páginas 37 e 38.

LAWRENCEN, B. J. Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques. In: *Supplement to resource constrained project scheduling*. Carnegie Mellon University, Pittsburgh, PA: Graduate School of Industrial Administration, Carnegie-Mellon University, 1984. p. 225–251. Citado na página 55.

LI, J.-q.; PAN, Q.-k.; LIANG, Y.-C. An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, v. 59, n. 4, p. 647 – 662, 2010. ISSN 0360-8352. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360835210002056>. Citado na página 64.

LI, J.-Q. et al. A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, v. 52, n. 5, p. 683–697, Feb 2011. ISSN 1433-3015. Disponível em: <https://doi.org/10.1007/s00170-010-2743-y>. Citado na página 64.

LI, X. et al. Hybrid artificial bee colony algorithm with a rescheduling strategy for solving flexible job shop scheduling problems. *Computers & Industrial Engineering*, v. 113, n. Supplement C, p. 10 – 26, 2017. ISSN 0360-8352. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360835217304011>. Citado 2 vezes nas páginas 64 e 90.

LIU, S. B.; ONG, H. L.; HUANG, H. C. Two bi-directional heuristics for the assembly line type ii problem. *The International Journal of Advanced Manufacturing Technology*, v. 22, n. 9, p. 656–661, Nov 2003. ISSN 1433-3015. Disponível em: <https://doi.org/10.1007/s00170-002-1504-y>. Citado na página 39.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search. In: *Handbook of Metaheuristics, volume 57 of International Series in Operations Research and Management Science*. [S.l.]: Kluwer Academic Publisher, 2002. p. 321–353. Citado 2 vezes nas páginas 25 e 32.

MANNE, A. S. On the job-shop scheduling problem. *Operations Research*, v. 8, n. 2, p. 219–223, 1960. Disponível em: <https://doi.org/10.1287/opre.8.2.219>. Citado na página 62.

MASTROLILLI, M.; GAMBARDELLA, L. M. Effective neighbourhood functions for the flexible job shop problem. *Journal of Scheduling*, John Wiley & Sons, Ltd., v. 3, n. 1, p. 3–20, 2000. ISSN 1099-1425. Disponível em: <http://dx.doi.org/10.1002/(SICI)1099-1425(200001/02)3:1<3::AID-JOS32>3.0.CO;2-Y>. Citado 10 vezes nas páginas 63, 64, 65, 86, 89, 90, 92, 93, 95 e 129.

MAURI, G. R. M. *Novas abordagens para representação e obtenção de limitantes e soluções para alguns problemas de otimização combinatória*. Tese (Doutorado) — Instituto Nacional de Pesquisas Espaciais - INPE, São José dos Campos, 2008. Doutorado em Computação Aplicada. Citado na página 32.

MELLOR, P. A review of job shop scheduling. *OR*, Operational Research Society, v. 17, n. 2, p. 161–171, 1966. ISSN 14732858. Disponível em: <http://www.jstor.org/stable/3007281>. Citado na página 52.

MORALES, S. W. G. *Formulações matemáticas e estratégias de resolução para o problema job shop clássico*. Dissertação (Mestrado) — Universidade de São Paulo - USP, 2012. Citado 2 vezes nas páginas 51 e 55.

MURPHEY, R. A.; PARDALOS, P. M.; PITSOULIS, L. S. A greedy randomized adaptive search procedure for the multitarget multisensor tracking problem. In: PARDALOS, P. M.; DU, D. Z. (Ed.). *Network design: Connectivity and facilities location*. [S.l.]: American Mathematical Society, 1998. v. 40 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science, p. 277–301. Citado na página 45.

N., M. R. R.; KOBTI, Z. A memetic algorithm for job shop scheduling using a critical-path-based local search heuristic. *Memetic Computing*, v. 4, n. 3, p. 231–245, Sep 2012. ISSN 1865-9292. Disponível em: <https://doi.org/10.1007/s12293-012-0084-0>. Citado na página 64.

NEARCHOU, A. C. Balancing large assembly lines by a new heuristic based on differential evolution method. *The International Journal of Advanced Manufacturing Technology*, v. 34, n. 9, p. 1016–1029, Oct 2007. ISSN 1433-3015. Disponível em: <https://doi.org/10.1007/s00170-006-0655-7>. Citado 2 vezes nas páginas 39 e 45.

NEARCHOU, A. C. Maximizing production rate and workload smoothing in assembly lines using particle swarm optimization. *International Journal of Production Economics*, v. 129, n. 2, p. 242 – 250, 2011. ISSN 0925-5273. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925527310004093>. Citado 2 vezes nas páginas 37 e 38.

OGAN, D.; AZIZOGLU, M. A branch and bound method for the line balancing problem in u-shaped assembly lines with equipment requirements. *Journal of Manufacturing Systems*, v. 36, n. Supplement C, p. 46 – 54, 2015. ISSN 0278-6125. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0278612515000138>. Citado na página 37.

OLIVEIRA, A. C. A. M.; CHAVES, A. A.; LORENA, L. A. N. Clustering search. *Pesquisa Operacional*, scielo, v. 33, p. 105 – 121, 04 2013. ISSN 0101-7438. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0101-74382013000100007&nrm=iso>. Citado na página 33.

OLIVEIRA, A. C. M. *Algoritmos evolutivos híbridos com detecção deregiões promissoras em espaços de busca contínuo e discreto*. Tese (Doutorado) — Instituto Nacional de Pesquisas Espaciais - INPE, São José dos Campos, 2004. Doutorado em Computação Aplicada. Citado 2 vezes nas páginas 25 e 33.

OLIVEIRA, A. C. M.; LORENA, L. A. N. Hybrid evolutionary algorithms and clustering search. In: _____. *Hybrid Evolutionary Algorithms*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 77–99. ISBN 978-3-540-73297-6. Disponível em: <https://doi.org/10.1007/978-3-540-73297-6_4>. Citado na página 79.

OTTO, A.; SCHOLL, A. Incorporating ergonomic risks into assembly line balancing. *European Journal of Operational Research*, v. 212, n. 2, p. 277 – 286, 2011. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S037722171100110X>. Citado na página 37.

OZBAKIR, L. et al. Multiple-colony ant algorithm for parallel assembly line balancing problem. *Applied Soft Computing*, v. 11, n. 3, p. 3186 – 3198, 2011. ISSN 1568-4946. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1568494610003285>. Citado na página 37.

PACHECO, R. F. I.; SANTORO, M. C. Proposta de classificação hierarquizada dos modelos de solução para o problema de job shop scheduling. *Gestão & Produção*, v. 6, n. 1, p. 44–52, 1999. Disponível em: <http://dx.doi.org/10.1590/S0104-530X1999000100001>. Citado na página 52.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. *Combinatorial Optimization: Algorithms and Complexity*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1982. ISBN 0-13-152462-3. Citado na página 28.

PAPE, T. Heuristics and lower bounds for the simple assembly line balancing problem type 1: Overview, computational tests and improvements. *European Journal of Operational Research*, v. 240, n. 1, p. 32 – 42, 2015. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221714005268>. Citado 2 vezes nas páginas 37 e 38.

PEETERS, M.; DEGRAEVE, Z. An linear programming based lower bound for the simple assembly line balancing problem. *European Journal of Operational Research*, v. 168, n. 3, p. 716 – 731, 2006. ISSN 0377-2217. Balancing Assembly and Transfer lines. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221704004813>. Citado 2 vezes nas páginas 37 e 38.

PENNA, P. H. V.; SUBRAMANIAN, A.; OCHI, L. S. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, v. 19, n. 2, p. 201–232, Apr 2013. ISSN 1572-9397. Disponível em: <https://doi.org/10.1007/s10732-011-9186-y>. Citado na página 32.

PETROPOULOS, D. I.; NEARCHOU, A. C. A particle swarm optimization algorithm for balancing assembly lines. *Assembly Automation*, v. 31, n. 2, p. 118–129, 2011. Disponível em: <https://doi.org/10.1108/01445151111117700>. Citado 2 vezes nas páginas 37 e 38.

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. ISSN 0305-0548. Part Special Issue: Search-based Software Engineering. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054807000524>. Citado 2 vezes nas páginas 63 e 64.

PITAKASO, R. Differential evolution algorithm for simple assembly line balancing type 1 (salbp-1). *Journal of Industrial and Production Engineering*, Taylor & Francis, v. 32, n. 2, p. 104–114, 2015. Disponível em: <http://dx.doi.org/10.1080/21681015.2015.1007094>. Citado 2 vezes nas páginas 37 e 38.

PREVIERO, W. D. *Estratégias de resolução para o problema de job-shop flexível*. Tese (Doutorado) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 9 2016. Doutorado em Ciência da Computação. Citado 4 vezes nas páginas 62, 90, 92 e 93.

RAHMATI, S. H. A.; ZANDIEH, M. A new biogeography-based optimization (bbo)

algorithm for the flexible job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, v. 58, n. 9, p. 1115–1129, Feb 2012. ISSN 1433-3015. Disponível em: <https://doi.org/10.1007/s00170-011-3437-9>. Citado na página 64.

REITER, R. On assembly-line balancing problems. *Operations Research*, v. 17, n. 4, p. 685–700, 1969. Disponível em: <https://doi.org/10.1287/opre.17.4.685>. Citado 2 vezes nas páginas 24 e 38.

RESENDE, M. G.; RIBEIRO, C. C. *Optimization by GRASP*. Springer-Verlag New York, 2003. v. 1. 312 p. ISBN 9781493965304. Disponível em: <http://http://www.springer.com/us/book/9781493965281>. Citado na página 29.

RIBEIRO, G. M.; MAURI, G. R.; LORENA, L. A. N. A simple and robust simulated annealing algorithm for scheduling workover rigs on onshore oil fields. *Computers & Industrial Engineering*, v. 60, n. 4, p. 519–526, 2011. ISSN 0360–8352. Citado na página 32.

SADRZADEH, A. Development of both the ais and pso for solving the flexible job shop scheduling problem. *Arabian Journal for Science and Engineering*, v. 38, n. 12, p. 3593–3604, Dec 2013. ISSN 2191-4281. Disponível em: <https://doi.org/10.1007/s13369-013-0625-y>. Citado na página 64.

SAIDI-MEHRABAD, M.; FATTAHI, P. Flexible job shop scheduling with tabu search algorithms. *The International Journal of Advanced Manufacturing Technology*, v. 32, n. 5, p. 563–570, Mar 2007. ISSN 1433-3015. Disponível em: <https://doi.org/10.1007/s00170-005-0375-4>. Citado na página 64.

SAIF, U. et al. A survey on assembly lines and its types. *Frontiers of Mechanical Engineering*, v. 9, n. 2, p. 95–105, Jun 2014. ISSN 2095-0241. Disponível em: <https://doi.org/10.1007/s11465-014-0302-1>. Citado na página 36.

SCHOLL, A. Publications of Darmstadt Technical University, Institute for Business Studies (BWL). *Balancing and sequencing of assembly lines*. [s.n.], 1999. Disponível em: <https://EconPapers.repec.org/RePEc:dar:wpaper:10881>. Citado na página 40.

SCHOLL, A.; BECKER, C. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, v. 168, n. 3, p. 666 – 693, 2006. ISSN 0377-2217. Balancing Assembly and Transfer lines. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221704004795>. Citado 2 vezes nas páginas 36 e 38.

SCHOLL, A.; VOSS, S. Simple assembly line balancing—heuristic approaches. *Journal of Heuristics*, v. 2, n. 3, p. 217–244, Dec 1997. ISSN 1572-9397. Disponível em: <https://doi.org/10.1007/BF00127358>. Citado na página 39.

SINGH, M. R.; MAHAPATRA, S. A quantum behaved particle swarm optimization for flexible job shop scheduling. *Computers & Industrial Engineering*, v. 93, n. Supplement C, p. 36 – 44, 2016. ISSN 0360-8352. Disponível em: <http://www.sciencedirect.com/science/article/pii/S030683521500474X>. Citado 3 vezes nas páginas 64, 90 e 92.

SINGH, M. R. et al. Particle swarm optimization algorithm embedded with maximum

deviation theory for solving multi-objective flexible job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, v. 85, n. 9, p. 2353–2366, Aug 2016. ISSN 1433-3015. Disponível em: <https://doi.org/10.1007/s00170-015-8075-1>. Citado na página 49.

SIVASANKARAN, P.; SHAHABUDEEN, P. Literature review of assembly line balancing problems. *The International Journal of Advanced Manufacturing Technology*, v. 73, n. 9, p. 1665–1694, Aug 2014. ISSN 1433-3015. Disponível em: <https://doi.org/10.1007/s00170-014-5944-y>. Citado na página 36.

SUBRAMANIAN, A. *Heuristic, Exact and Hybrid Approaches for Vehicle Routing Problems*. Tese (Doutorado) — Universidade Federal Fluminense, Niterói-RJ, 2012. Doutorado em Computação. Citado na página 32.

SUBRAMANIAN, A.; BATTARRA, M. An iterated local search algorithm for the travelling salesman problem with pickups and deliveries. *The Journal of the Operational Research Society*, Palgrave Macmillan Journals, v. 64, n. 3, p. 402–409, 2013. ISSN 01605682, 14769360. Disponível em: <http://www.jstor.org/stable/23355747>. Citado na página 32.

TAILLARD Éric D. Parallel taboo search techniques for the job shop scheduling problem. *ORSA Journal on Computing*, v. 6, n. 2, p. 108–117, 1994. Disponível em: <https://doi.org/10.1287/ijoc.6.2.108>. Citado na página 54.

TASAN, S. O.; TUNALI, S. A review of the current applications of genetic algorithms in assembly line balancing. *Journal of Intelligent Manufacturing*, v. 19, n. 1, p. 49–69, Feb 2008. ISSN 1572-8145. Disponível em: <https://doi.org/10.1007/s10845-007-0045-5>. Citado na página 36.

TEEKENG, W.; THAMMANO, A. Modified genetic algorithm for flexible job-shop scheduling problems. *Procedia Computer Science*, v. 12, n. Supplement C, p. 122 – 128, 2012. ISSN 1877-0509. Complex Adaptive Systems 2012. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050912006321>. Citado na página 64.

TRIKI, H.; MELLOULI, A.; MASMOUDI, F. A multi-objective genetic algorithm for assembly line resource assignment and balancing problem of type 2 (alrabp-2). *Journal of Intelligent Manufacturing*, v. 28, n. 2, p. 371–385, Feb 2017. ISSN 1572-8145. Disponível em: <https://doi.org/10.1007/s10845-014-0984-6>. Citado na página 37.

TUBINO, D. F. *Sistemas de Produção: a produtividade no chão-de-fábrica*. Porto Alegre: Bookman, 1999. Citado na página 23.

UĞURDAĞ, H. F.; RACHAMADUGU, R.; PAPACHRISTOU, C. A. Designing paced assembly lines with fixed number of stations. *European Journal of Operational Research*, v. 102, n. 3, p. 488 – 501, 1997. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221796002482>. Citado na página 39.

WANG, L. et al. A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem. *Computers & Industrial Engineering*, v. 62, n. 4, p. 917 – 926, 2012. ISSN 0360-8352. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0360835211003871>. Citado na página 64.

WANG, L. et al. A hybrid artificial bee colony algorithm for the fuzzy flexible job-shop scheduling problem. *International Journal of Production Research*, Taylor & Francis, v. 51, n. 12, p. 3593–3608, 2013. Disponível em: <http://dx.doi.org/10.1080/00207543.2012.754549>. Citado na página 64.

WANG, X.; DUAN, H. A hybrid biogeography-based optimization algorithm for job shop scheduling problem. *Computers & Industrial Engineering*, v. 73, n. Supplement C, p. 96 – 114, 2014. ISSN 0360-8352. Disponível em: <http://www.sciencedirect.com/science/article/pii/S036083521400117X>. Citado na página 64.

WEN-CHYUAN, C.; KOUVELIS, P.; URBAN, T. L. Line balancing in a just-in-time production environment: balancing multiple u-lines. *IIE Transactions*, v. 39, n. 4, p. 347 – 359, 2007. ISSN 0740817X. Disponível em: <http://search-ebscohost-com.ez43.periodicos.capes.gov.br/login.aspx?direct=true&db=aph&AN=24925185&lang=pt-br&site=ehost-live>. Citado na página 37.

WOLSEY, L. A. *Integer Programming*. New York, NY, USA: Wiley-Interscience, 1998. 288 p. ISBN 978-0-471-28366-9. Citado 2 vezes nas páginas 27 e 28.

XIE, C.; ALLEN, T. T. Simulation and experimental design methods for job shop scheduling with material handling: a survey. *The International Journal of Advanced Manufacturing Technology*, v. 80, n. 1, p. 233–243, Sep 2015. ISSN 1433-3015. Disponível em: <https://doi.org/10.1007/s00170-015-6981-x>. Citado na página 52.

XING, L.-N. et al. A knowledge-based ant colony optimization for flexible job shop scheduling problems. *Applied Soft Computing*, v. 10, n. 3, p. 888 – 896, 2010. ISSN 1568-4946. Disponível em: <http://www.sciencedirect.com/science/article/pii/S156849460900194X>. Citado na página 64.

XING, L.-N.; CHEN, Y.-W.; YANG, K.-W. An efficient search method for multi-objective flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, v. 20, n. 3, p. 283–293, Jun 2009. ISSN 1572-8145. Disponível em: <https://doi.org/10.1007/s10845-008-0216-z>. Citado na página 64.

XING, L.-N.; CHEN, Y.-W.; YANG, K.-W. Multi-objective flexible job shop schedule: Design and evaluation by simulation modeling. *Applied Soft Computing*, v. 9, n. 1, p. 362 – 376, 2009. ISSN 1568-4946. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1568494608000756>. Citado na página 64.

YAGMAHAN, B. Mixed-model assembly line balancing using a multi-objective ant colony optimization approach. *Expert Systems with Applications*, v. 38, n. 10, p. 12453 – 12461, 2011. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417411005422>. Citado na página 37.

YAZDANI, M.; AMIRI, M.; ZANDIEH, M. Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Systems with Applications*, v. 37, n. 1, p. 678 – 687, 2010. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417409005685>. Citado na página 64.

YU, J.; YIN, Y. Assembly line balancing based on an adaptive genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, v. 48, n. 1, p. 347–354, Apr 2010. ISSN 1433-3015. Disponível em: <https://doi.org/10.1007/s00170-009-2281-7>. Citado 2 vezes nas páginas 37 e 38.

YUAN, Y.; XU, H. An integrated search heuristic for large-scale flexible job shop scheduling problems. *Computers & Operations Research*, v. 40, n. 12, p. 2864 – 2877, 2013. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054813001676>. Citado na página 63.

ZACHARIA, P. T.; NEARCHOU, A. C. Multi-objective fuzzy assembly line balancing using genetic algorithms. *Journal of Intelligent Manufacturing*, v. 23, n. 3, p. 615–627, Jun 2012. ISSN 1572-8145. Disponível em: <https://doi.org/10.1007/s10845-010-0400-9>. Citado na página 37.

ZHANG, G.; GAO, L.; SHI, Y. An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, v. 38, n. 4, p. 3563 – 3573, 2011. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S095741741000953X>. Citado 4 vezes nas páginas 64, 90, 92 e 93.

ZHANG, H. et al. An integer-coded differential evolution algorithm for simple assembly line balancing problem of type 2. *Assembly Automation*, v. 36, n. 3, p. 246–261, 2016. Disponível em: <https://doi.org/10.1108/AA-11-2015-089>. Citado 2 vezes nas páginas 39 e 45.

ZHANG, J. et al. Review of job shop scheduling research and its new perspectives under industry 4.0. *Journal of Intelligent Manufacturing*, Aug 2017. ISSN 1572-8145. Disponível em: <https://doi.org/10.1007/s10845-017-1350-2>. Citado na página 52.

ZHANG, Q.; MANIER, H.; MANIER, M.-A. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. *Computers & Operations Research*, v. 39, n. 7, p. 1713 – 1723, 2012. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054811002954>. Citado na página 64.

ÖZGÜVEN, C.; ÖZBAKIR, L.; YAVUZ, Y. Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied Mathematical Modelling*, v. 34, n. 6, p. 1539 – 1548, 2010. ISSN 0307-904X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0307904X09002819>. Citado na página 62.

Apêndices

APÊNDICE A – Resultados Detalhados do GRASP para o SALBP-2

O conjunto de instâncias *Dataset*1 é apresentado nas Tabelas 18, 19, 20 e 21 e do *Dataset*2 nas Tabelas 22, 23, 24, 25 e 26. Nessas tabelas, as duas primeiras colunas fornecem o nome do grafo de precedência e o número de estações de trabalho (m). O valor da melhor solução conhecida para cada instância está listado na coluna BKS; O melhor valor da função objetivo encontrada após dez execuções do GRASP é apresentado na coluna *best*. A média dos tempos de ciclo encontrados após dez execuções é apresentada na coluna avg, assim como os tempos computacionais médios em segundos estão listados em $avgt(s)$; A coluna *std* indica o desvio padrão dos tempos de ciclo encontrados e a última coluna lista o PECT médio para cada instância.

Tabela 18 – Resultados detalhados do Dataset1 - Parte I.

Graph	m	BKS	best	avg	avg t(s)	std	PECT
Buxey	7	47	48	48,00	0,01	0,00	2,13
	8	41	41	41,00	0,00	0,00	0,00
	9	37	37	37,00	0,39	0,00	0,00
	10	34	34	34,00	0,00	0,00	0,00
	11	32	32	32,00	4,40	0,00	0,00
	12	28	28	28,00	23,98	0,00	0,00
	13	27	27	27,00	0,01	0,00	0,00
	14	25	25	25,00	0,02	0,00	0,00
Gunther	6	84	84	84,00	0,00	0,00	0,00
	7	72	72	72,00	33,43	0,00	0,00
	8	63	63	63,00	1,25	0,00	0,00
	9	54	54	54,40	63,49	0,49	0,00
	10	50	50	50,00	0,05	0,00	0,00
	11	48	48	48,00	0,00	0,00	0,00
	12	44	44	44,00	0,00	0,00	0,00
	13	42	42	42,00	0,00	0,00	0,00
	14	40	40	40,00	0,06	0,00	0,00
	15	40	40	40,00	0,06	0,00	0,00
Kilbridge	3	184	184	184,00	0,01	0,00	0,00
	4	138	138	138,00	0,01	0,00	0,00
	5	111	111	111,00	0,00	0,00	0,00
	6	92	92	92,00	0,98	0,00	0,00
	7	79	79	79,00	0,50	0,00	0,00
	8	69	69	69,10	32,35	0,32	0,00
	9	62	62	62,00	0,12	0,00	0,00
	10	56	56	56,00	0,03	0,00	0,00
	11	55	55	55,00	0,00	0,00	0,00
Lutz1	8	1860	1860	1860,00	0,00	0,00	0,00
	9	1638	1664	1664,00	0,19	0,00	1,59
	10	1526	1526	1526,00	8,64	0,00	0,00
	11	1400	1400	1400,00	0,96	0,00	0,00
	12	1400	1400	1400,00	0,00	0,00	0,00
Média		296,19	297,03	297,05	5,34	0,03	0,12

Tabela 19 – Resultados detalhados do Dataset1 - Parte II.

Graph	m	BKS	best	avg	avg t(s)	std	PECT
Lutz2	9	54	55	55,00	0,00	0,00	1,85
	10	49	50	50,00	0,01	0,00	2,04
	11	45	45	45,00	0,00	0,00	0,00
	12	41	41	41,00	6,11	0,00	0,00
	13	38	38	38,00	0,37	0,00	0,00
	14	35	36	36,00	0,36	0,00	2,86
	15	33	33	33,50	55,45	0,53	0,00
	16	31	32	32,00	0,03	0,00	3,23
	17	29	30	30,00	0,12	0,00	3,45
	18	28	28	28,90	3,83	0,32	0,00
	19	26	27	27,00	3,49	0,00	3,85
	20	25	25	25,90	15,78	0,32	0,00
	21	24	24	24,80	11,44	0,42	0,00
	22	23	23	23,00	13,93	0,00	0,00
	23	22	23	23,00	0,07	0,00	4,55
	24	21	22	22,00	0,74	0,00	4,76
	25	20	21	21,00	0,10	0,00	5,00
	26	19	20	20,70	34,71	0,48	5,26
	27	19	20	20,00	40,72	0,00	5,26
	28	18	19	19,00	0,48	0,00	5,56
Sawyer	7	47	48	48,00	0,00	0,00	2,13
	8	41	41	41,00	0,02	0,00	0,00
	9	37	37	37,00	0,73	0,00	0,00
	10	34	34	34,00	0,05	0,00	0,00
	11	31	32	32,00	0,00	0,00	3,23
	12	28	28	28,00	3,38	0,00	0,00
	13	26	26	26,00	0,08	0,00	0,00
	14	25	25	25,00	0,01	0,00	0,00
Média		31,04	31,54	31,67	6,86	0,07	1,89

Tabela 20 – Resultados detalhados do Dataset1 - Parte III.

Graph	m	BKS	best	avg	avg t(s)	std	PECT
Tonge	3	1170	1170	1170,00	0,24	0,00	0,00
	4	878	878	878,00	0,06	0,00	0,00
	5	702	702	702,00	19,06	0,00	0,00
	6	585	586	586,00	10,31	0,00	0,17
	7	502	502	502,20	43,63	0,42	0,00
	8	439	439	439,90	7,22	0,32	0,00
	9	391	391	391,20	94,82	0,42	0,00
	10	352	352	352,90	13,01	0,32	0,00
	11	320	321	321,00	17,64	0,00	0,31
	12	294	295	295,00	45,85	0,00	0,34
	13	271	272	272,50	68,91	0,53	0,37
	14	251	253	253,90	18,49	0,32	0,80
	15	235	237	237,70	38,88	0,67	0,85
	16	221	223	223,10	70,08	0,57	0,90
	17	208	210	210,60	48,22	0,52	0,96
	18	196	198	199,60	26,48	0,70	1,02
	19	186	189	189,80	59,11	0,92	1,61
	20	177	181	181,40	78,92	0,52	2,26
	21	170	174	174,20	68,37	0,42	2,35
	22	162	165	165,70	61,05	0,48	1,85
	23	156	158	158,60	63,63	0,52	1,28
	24	156	156	156,00	0,18	0,00	0,00
	25	156	156	156,00	0,02	0,00	0,00
Arcus1	3	25236	25240	25240,00	39,75	0,00	0,02
	4	18927	18932	18940,00	82,81	4,00	0,03
	5	15142	15162	15162,00	2,27	0,00	0,13
	6	12620	12642	12659,80	93,56	6,38	0,17
	7	10826	10874	10878,00	88,35	3,00	0,44
	8	9554	9577	9578,90	135,42	2,43	0,24
	9	8499	8528	8528,00	0,65	0,00	0,34
	10	7580	7614	7620,10	128,95	6,11	0,45
	11	7084	7091	7091,00	0,07	0,00	0,10
	12	6412	6533	6540,70	89,11	6,08	1,89
	13	5864	6005	6010,40	10,59	1,80	2,40
	14	5441	5511	5530,80	58,24	13,01	1,29
	15	5104	5202	5202,00	11,63	0,00	1,92
	16	4850	4904	4905,70	92,29	5,10	1,11
	17	4516	4624	4641,80	66,49	13,35	2,39
	18	4317	4387	4415,80	117,49	12,24	1,62
	19	4068	4133	4149,60	131,08	9,86	1,60
	20	3882	3990	4024,20	69,07	20,44	2,78
	21	3691	3748	3766,30	87,58	6,94	1,54
	22	3691	3691	3691,00	3,53	0,00	0,00
Média		4080,98	4106,88	4111,47	50,30	2,75	0,83

Tabela 21 – Resultados detalhados do Dataset1 - Parte IV.

Graph	m	BKS	best	avg	avg t(s)	std	PECT
Arcus2	3	50133	50133	50133,00	2,56	0,00	0,00
	4	37600	37600	37600,00	7,06	0,00	0,00
	5	30080	30081	30081,00	30,37	0,00	0,00
	6	25067	25068	25068,90	59,97	0,74	0,00
	7	21486	21488	21489,40	83,61	0,84	0,01
	8	18800	18804	18806,00	111,54	1,15	0,02
	9	16711	16719	16722,00	116,78	1,76	0,05
	10	15040	15049	15058,50	141,50	5,46	0,06
	11	13673	13689	13691,60	114,53	1,58	0,12
	12	12534	12568	12574,30	123,15	6,98	0,27
	13	11570	11611	11623,20	109,44	7,38	0,35
	14	10747	10804	10814,70	119,24	8,65	0,53
	15	10035	10084	10096,10	90,34	10,12	0,49
	16	9412	9450	9486,20	129,87	22,41	0,40
	17	8855	8888	8923,60	144,13	21,02	0,37
	18	8377	8437	8444,10	114,64	5,34	0,72
	19	7924	8028	8056,30	133,00	21,63	1,31
	20	7524	7654	7677,90	54,67	15,04	1,73
	21	7188	7307	7342,10	57,15	15,06	1,66
	22	6858	6972	6995,30	132,55	14,52	1,66
	23	6560	6670	6704,20	55,63	14,44	1,68
	24	6284	6468	6481,90	53,11	8,49	2,93
	25	6106	6201	6218,80	37,74	9,13	1,56
	26	5856	6125	6125,00	0,55	0,00	4,59
	27	5689	5770	5855,70	156,11	46,75	1,42
Média		14404,36	14466,72	14482,79	87,17	9,54	0,88

Tabela 22 – Resultados detalhados do Dataset2 - Parte I.

Graph	m	BKS	best	avg	avg t(s)	std	PECT
Hahn	3	4787	4787	4787,00	0,00	0,00	0,00
	4	3677	3681	3681,00	9,26	0,00	0,11
	5	2823	2823	2823,00	0,00	0,00	0,00
	6	2400	2400	2400,00	0,01	0,00	0,00
	7	2336	2336	2336,00	0,00	0,00	0,00
	8	1907	1910	1910,00	0,21	0,00	0,16
	9	1827	1829	1832,30	40,29	5,31	0,11
	10	1775	1829	1832,30	59,83	5,31	3,04
Warnecke	3	516	516	516,00	0,26	0,00	0,00
	4	387	388	388,00	0,19	0,00	0,26
	5	310	310	310,00	0,84	0,00	0,00
	6	258	259	259,40	28,30	0,52	0,39
	7	222	222	222,90	6,39	0,32	0,00
	8	194	195	195,90	16,57	0,32	0,52
	9	172	175	175,00	42,46	0,00	1,74
	10	155	159	159,00	17,93	0,00	2,58
	11	142	143	143,90	46,66	0,32	0,70
	12	130	132	132,00	6,71	0,00	1,54
	13	120	121	121,90	9,55	0,32	0,83
	14	111	113	113,50	32,30	0,53	1,80
	15	104	107	107,90	16,69	0,32	2,88
	16	98	100	100,20	44,62	0,42	2,04
	17	92	97	97,00	7,08	0,00	5,43
	18	87	90	90,00	35,14	0,00	3,45
	19	84	86	86,20	54,07	0,42	2,38
	20	79	80	81,80	80,27	0,79	1,27
	21	76	77	77,70	25,96	0,48	1,32
	22	73	75	75,70	36,63	0,48	2,74
	23	69	72	72,20	74,97	0,42	4,35
	24	66	70	70,10	41,58	0,32	6,06
	25	64	67	67,00	47,24	0,00	4,69
	26	64	66	66,40	40,09	0,52	3,13
	27	60	64	66,00	0,28	0,00	6,67
	28	59	61	62,60	28,91	0,70	3,39
	29	56	59	59,00	17,16	0,00	5,36
Wee-Mag	3	500	500	500,00	0,00	0,00	0,00
	4	375	375	375,00	0,00	0,00	0,00
	5	300	300	300,00	0,01	0,00	0,00
	6	250	250	250,00	0,15	0,00	0,00
	7	215	215	215,00	0,02	0,00	0,00
	8	188	188	188,00	0,14	0,00	0,00
	9	167	167	167,00	2,54	0,00	0,00
Média		651,79	654,62	655,09	20,75	0,42	1,64

Tabela 23 – Resultados detalhados do Dataset2 - Parte II.

Graph	m	BKS	best	avg	avg t(s)	std	PECT
Wee-Mag	10	150	151	151,00	4,94	0,00	0,67
	11	137	137	137,00	20,57	0,00	0,00
	12	125	126	126,00	33,42	0,00	0,80
	13	116	116	116,50	29,70	0,53	0,00
	14	108	109	109,80	28,83	0,42	0,93
	15	100	102	102,00	9,39	0,00	2,00
	16	94	95	95,00	7,14	0,00	1,06
	17	89	91	91,00	27,91	0,00	2,25
	18	87	88	88,00	56,59	0,00	1,15
	19	85	86	86,00	16,42	0,00	1,18
	20	77	78	78,20	42,87	0,42	1,30
	21	72	73	73,00	6,20	0,00	1,39
	22	69	70	70,00	76,60	0,00	1,45
	23	67	69	69,00	20,76	0,00	2,99
	24	66	68	68,00	31,65	0,00	3,03
	25	65	67	67,00	32,08	0,00	3,08
	26	65	66	66,00	7,10	0,00	1,54
	27	65	65	65,00	7,54	0,00	0,00
	28	64	64	64,80	17,23	0,42	0,00
	29	63	63	63,00	58,91	0,00	0,00
	30	56	56	56,00	1,49	0,00	0,00
Lutz3	3	548	548	548,70	26,98	0,48	0,00
	4	411	411	411,60	26,71	0,52	0,00
	5	329	329	329,00	0,40	0,00	0,00
	6	275	275	275,00	3,99	0,00	0,00
	7	236	237	237,00	1,07	0,00	0,42
	8	207	208	208,00	16,09	0,00	0,48
	9	184	185	185,50	46,30	0,53	0,54
	10	165	166	167,80	30,29	0,63	0,61
	11	151	152	152,00	5,58	0,00	0,66
	12	138	139	139,80	12,21	0,42	0,72
	13	128	129	129,10	27,41	0,32	0,78
	14	118	120	120,30	51,09	0,48	1,69
	15	110	113	113,00	23,62	0,00	2,73
	16	105	106	107,10	65,27	0,74	0,95
	17	98	99	99,20	72,90	0,42	1,02
	18	93	95	95,00	20,39	0,00	2,15
	19	89	90	90,00	11,16	0,00	1,12
	20	85	89	89,30	58,85	0,48	4,71
	21	80	83	83,60	27,37	0,52	3,75
	22	76	80	80,00	2,75	0,00	5,26
	23	74	76	76,00	28,47	0,00	2,70
Média		131,43	132,62	132,84	26,10	0,17	1,31

Tabela 24 – Resultados detalhados do Dataset2 - Parte III.

Graph	m	BKS	best	avg	avg t(s)	std	PECT
Mukherje	3	1403	1403	1403,00	0,02	0,00	0,00
	4	1052	1052	1052,00	0,28	0,00	0,00
	5	844	844	844,00	0,00	0,00	0,00
	6	704	704	704,00	0,08	0,00	0,00
	7	621	621	621,00	19,88	0,00	0,00
	8	532	532	532,30	82,10	0,48	0,00
	9	471	471	471,10	39,49	0,32	0,00
	10	424	424	424,00	0,01	0,00	0,00
	11	391	392	392,00	0,16	0,00	0,26
	12	358	359	359,00	18,46	0,00	0,28
	13	325	326	326,90	1,87	0,32	0,31
	14	311	312	312,00	70,50	0,00	0,32
	15	288	290	290,00	22,89	0,00	0,69
	16	268	271	271,00	0,04	0,00	1,12
	17	251	251	251,80	64,91	0,42	0,00
	18	239	240	240,60	25,73	0,52	0,42
	19	226	226	226,90	14,67	0,32	0,00
	20	220	222	222,20	36,41	0,42	0,91
	21	208	209	209,20	42,37	0,42	0,48
	22	200	202	202,90	15,69	0,32	1,00
	23	189	191	191,60	47,16	0,52	1,06
	24	179	182	182,00	30,78	0,00	1,68
	25	172	175	175,00	8,52	0,00	1,74
	26	171	171	171,00	0,19	0,00	0,00
Barthold	3	1878	1878	1878,00	0,03	0,00	0,00
	4	1409	1409	1409,00	0,01	0,00	0,00
	5	1127	1127	1127,00	0,38	0,00	0,00
	6	939	939	939,00	24,91	0,00	0,00
	7	805	805	805,00	44,18	0,00	0,00
	8	705	705	705,00	1,15	0,00	0,00
	9	626	627	627,00	1,41	0,00	0,16
	10	564	564	564,50	43,98	0,53	0,00
	11	513	513	513,00	23,46	0,00	0,00
	12	470	470	470,90	22,84	0,32	0,00
	13	434	435	435,00	51,38	0,00	0,23
	14	403	404	404,40	41,08	0,52	0,25
	15	383	383	383,00	0,01	0,00	0,00
Média		548,73	549,43	549,63	21,54	0,15	0,29

Tabela 25 – Resultados detalhados do Dataset2 - Parte IV.

Graph	m	BKS	best	avg	avg t(s)	std	PECT
Barthol2	27	157	160	160,20	35,92	0,42	1,91
	28	152	154	155,20	31,39	0,63	1,32
	29	146	150	150,00	53,11	0,00	2,74
	30	142	145	145,60	43,66	0,52	2,11
	31	137	141	141,00	11,11	0,00	2,92
	32	133	135	135,90	40,86	0,32	1,50
	33	129	133	133,10	49,03	0,32	3,10
	34	125	129	129,20	42,72	0,42	3,20
	35	121	126	126,20	40,28	0,42	4,13
	36	118	122	122,40	29,38	0,52	3,39
	37	115	119	120,10	45,05	0,88	3,48
	38	112	116	117,00	67,61	0,47	3,57
	39	109	114	114,80	55,52	0,42	4,59
	40	106	111	112,20	45,49	0,63	4,72
	41	104	109	109,20	96,89	0,42	4,81
	42	101	107	107,30	39,69	0,67	5,94
	43	99	105	105,30	52,21	0,48	6,06
	44	97	102	102,50	64,57	0,53	5,15
	45	95	100	100,10	41,80	0,32	5,26
	46	93	97	97,90	33,17	0,32	4,30
	47	91	95	95,60	44,88	0,52	4,40
	48	89	93	93,70	43,73	0,48	4,49
	49	87	92	92,00	71,98	0,00	5,75
	50	85	90	90,20	70,13	0,42	5,88
	51	84	88	88,60	45,50	0,52	4,76
Média		113,08	117,32	117,81	47,83	0,43	3,98

Tabela 26 – Resultados detalhados do Dataset2 - Parte V.

Graph	m	BKS	best	avg	avg t(s)	std	PECT
Scholl	25	2787	2827	2830,30	138,44	3,16	1,44
	26	2680	2712	2716,60	109,11	2,80	1,19
	27	2580	2612	2615,00	123,48	1,67	1,24
	28	2488	2517	2520,40	107,61	2,37	1,17
	29	2402	2429	2433,20	104,00	2,18	1,12
	30	2322	2349	2357,20	164,85	3,99	1,16
	31	2247	2281	2283,00	134,06	1,48	1,51
	32	2177	2208	2214,10	57,75	2,07	1,42
	33	2111	2155	2160,20	55,73	2,79	2,08
	34	2049	2092	2103,30	109,63	6,65	2,10
	35	1991	2042	2042,00	96,62	0,00	2,56
	36	1935	1997	2008,50	110,02	6,05	3,20
	37	1883	1939	1961,00	54,74	10,75	2,97
	38	1834	1878	1888,20	179,70	5,42	2,40
	39	1787	1834	1839,50	134,74	3,83	2,63
	40	1742	1783	1790,50	141,66	3,50	2,35
	41	1700	1742	1743,60	116,14	1,20	2,47
	42	1659	1703	1711,50	99,74	3,61	2,65
	43	1621	1688	1688,00	2,35	0,00	4,13
	44	1584	1688	1688,00	0,05	0,00	6,57
	45	1549	1603	1617,70	196,68	7,32	3,49
	46	1515	1576	1581,00	86,47	3,35	4,03
	47	1483	1536	1539,50	91,80	1,36	3,57
	48	1452	1518	1526,40	92,35	5,92	4,55
	49	1423	1492	1497,60	116,02	3,85	4,85
	50	1394	1450	1461,10	138,88	4,97	4,02
	51	1386	1419	1422,70	157,17	3,66	2,38
	52	1386	1389	1397,40	130,17	4,50	0,22
Média		1898,82	1944,96	1951,34	108,93	3,52	2,62

APÊNDICE B – Resultados das Meta-heurísticas propostas para o FJSP

B.1 Características dos conjuntos de instâncias utilizados

Nas Tabelas 27, 28, 29, 30, 31 e 32 referenciam-se, respectivamente o detalhamento dos conjuntos de instâncias *BRdata*, *DPdata*, *BCdata*, *HUdata/edata*, *HUdata/rdata* e *HUdata/vdata*, da forma como segue. Na coluna *I* lista-se o nome da instância; em seguida é informado a quantidade de *jobs* e de máquinas ($n \times m$); o fator de flexibilidade é indicado na coluna *Flex.*; e, o valor *lower bound* obtido em Mastrolilli e Gambardella (2000) é apresentado na coluna *LB*.

Tabela 27 – Característica do conjunto de instâncias *BRdata*.

I	$n \times m$	Flex.	LB
Mk01	10×6	2,09	36
Mk02	10×6	4,10	24
Mk03	15×8	3,01	204
Mk04	20×5	1,91	48
Mk05	15×4	1,71	168
Mk06	10×15	3,27	33
Mk07	20×5	2,83	133
Mk08	20×10	1,43	523
Mk09	20×10	2,53	299
Mk10	20×15	2,98	165

Tabela 28 – Característica do conjunto de instâncias *DPdata*.

I	$n \times m$	Flex.	LB
1a	10×5	1,13	2505
2a	10×5	1,69	2228
3a	10×5	2,56	2228
4a	10×5	1,13	2503
5a	10×5	1,69	2189
6a	10×5	2,56	2162
7a	15×8	1,24	2187
8a	15×8	2,42	2061
9a	15×8	4,03	2061
10a	15×8	1,24	2178
11a	15×8	2,42	2017
12a	15×8	4,03	1969
13a	20×8	1,34	2161
14a	20×8	2,99	2161
15a	20×8	5,02	2161
16a	20×8	1,34	2148
17a	20×8	2,99	2088
18a	20×8	5,02	2057

Tabela 29 – Característica do conjunto de instâncias *BCdata*.

I	$n \times m$	Flex.	LB
mt10c1	10×11	1,10	655
mt10cc	10×12	1,20	655
mt10x	10×11	1,10	655
mt10xx	10×12	1,20	655
mt10xxx	10×13	1,30	655
mt10xy	10×12	1,20	655
mt10xyz	10×13	1,30	655
setb4c9	15×11	1,10	857
setb4cc	15×12	1,20	857
setb4x	15×11	1,10	846
setb4xx	15×12	1,20	846
setb4xxx	15×13	1,30	846
setb4xy	15×12	1,20	845
setb4xyz	15×13	1,30	838
seti5c12	15×16	1,07	1027
seti5cc	15×17	1,13	955
seti5x	15×16	1,07	955
seti5xx	15×17	1,13	955
seti5xxx	15×18	1,20	955
seti5xy	15×17	1,13	955
seti5xyz	15×18	1,20	955

Tabela 30 – Característica do conjunto de instâncias *HUdata/edata*.

I	$n \times m$	Flex.	LB	I	$n \times m$	Flex.	LB
mt06	6×6	1,15	55	la20	10×10	1,15	857
mt10	10×10	1,15	871	la21	15×10	1,15	895
mt20	20×5	1,15	1088	la22	15×10	1,15	832
la01	10×5	1,15	609	la23	15×10	1,15	950
la02	10×5	1,15	655	la24	15×10	1,15	881
la03	10×5	1,15	550	la25	15×10	1,15	894
la04	10×5	1,15	568	la26	20×10	1,15	1089
la05	10×5	1,15	503	la27	20×10	1,15	1181
la06	15×5	1,15	833	la28	20×10	1,15	1116
la07	15×5	1,15	762	la29	20×10	1,15	1058
la08	15×5	1,15	845	la30	20×10	1,15	1147
la09	15×5	1,15	878	la31	30×10	1,15	1523
la10	15×5	1,15	866	la32	30×10	1,15	1698
la11	20×5	1,15	1087	la33	30×10	1,15	1547
la12	20×5	1,15	960	la34	30×10	1,15	1592
la13	20×5	1,15	1053	la35	30×10	1,15	1736
la14	20×5	1,15	1123	la36	15×15	1,15	1006
la15	20×5	1,15	1111	la37	15×15	1,15	1355
la16	10×10	1,15	892	la38	15×15	1,15	1019
la17	10×10	1,15	707	la39	15×15	1,15	1151
la18	10×10	1,15	842	la40	15×15	1,15	1034
la19	10×10	1,15	796	-	-	-	-

Tabela 31 – Característica do conjunto de instâncias $HUdata/rdata$.

I	$n \times m$	Flex.	LB	I	$n \times m$	Flex.	LB
mt06	6×6	2,00	47	la20	10×10	2,00	756
mt10	10×10	2,00	679	la21	15×10	2,00	808
mt20	20×5	2,00	1022	la22	15×10	2,00	737
la01	10×5	2,00	570	la23	15×10	2,00	816
la02	10×5	2,00	529	la24	15×10	2,00	775
la03	10×5	2,00	477	la25	15×10	2,00	752
la04	10×5	2,00	502	la26	20×10	2,00	1056
la05	10×5	2,00	457	la27	20×10	2,00	1085
la06	15×5	2,00	799	la28	20×10	2,00	1075
la07	15×5	2,00	749	la29	20×10	2,00	993
la08	15×5	2,00	765	la30	20×10	2,00	1068
la09	15×5	2,00	853	la31	30×10	2,00	1520
la10	15×5	2,00	804	la32	30×10	2,00	1657
la11	20×5	2,00	1071	la33	30×10	2,00	1497
la12	20×5	2,00	936	la34	30×10	2,00	1535
la13	20×5	2,00	1038	la35	30×10	2,00	1549
la14	20×5	2,00	1070	la36	15×15	2,00	1016
la15	20×5	2,00	1089	la37	15×15	2,00	989
la16	10×10	2,00	717	la38	15×15	2,00	943
la17	10×10	2,00	646	la39	15×15	2,00	966
la18	10×10	2,00	666	la40	15×15	2,00	955
la19	10×10	2,00	647	-	-	-	-

Tabela 32 – Característica do conjunto de instâncias *HUdata/vdata*.

I	$n \times m$	Flex.	LB	I	$n \times m$	Flex.	LB
mt06	6×6	3,00	47	la20	10×10	5,00	756
mt10	10×10	5,00	655	la21	15×10	5,00	800
mt20	20×5	2,50	1022	la22	15×10	5,00	733
la01	10×5	2,50	570	la23	15×10	5,00	809
la02	10×5	2,50	529	la24	15×10	5,00	773
la03	10×5	2,50	477	la25	15×10	5,00	751
la04	10×5	2,50	502	la26	20×10	5,00	1052
la05	10×5	2,50	457	la27	20×10	5,00	1084
la06	15×5	2,50	799	la28	20×10	5,00	1069
la07	15×5	2,50	749	la29	20×10	5,00	993
la08	15×5	2,50	765	la30	20×10	5,00	1068
la09	15×5	2,50	853	la31	30×10	5,00	1520
la10	15×5	2,50	804	la32	30×10	5,00	1657
la11	20×5	2,50	1071	la33	30×10	5,00	1497
la12	20×5	2,50	936	la34	30×10	5,00	1535
la13	20×5	2,50	1038	la35	30×10	5,00	1549
la14	20×5	2,50	1070	la36	15×15	7,50	948
la15	20×5	2,50	1089	la37	15×15	7,50	986
la16	10×10	5,00	717	la38	15×15	7,50	943
la17	10×10	5,00	646	la39	15×15	7,50	922
la18	10×10	5,00	663	la40	15×15	7,50	955
la19	10×10	5,00	617	-	-	-	-

B.2 Resultados das Meta-heurísticas propostas para o FJSP

Tabela 33 – Resultados detalhados dos algoritmos propostos para o conjunto de instâncias *BRdata*.

	SA				ILS/SA				CS			
I	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP
Mk01	**40**	40,0	3,1	0,0	**40**	40,0	2,2	0,0	**40**	40,0	2,6	0,0
Mk02	27	27,0	6,5	0,0	**26**	26,8	30,2	0,4	**26**	26,8	36,1	0,4
Mk03	**204**	204,0	2,3	0,0	**204**	204,0	2,6	0,0	**204**	204,0	0,9	0,0
Mk04	**60**	60,0	6,6	0,0	**60**	60,1	13,6	0,3	**60**	60,1	11,3	0,3
Mk05	**173**	173,0	128,4	0,0	**173**	173,0	115,4	0,0	**173**	173,0	64,9	0,0
Mk06	60	60,0	411,5	0,0	**58**	59,4	582,4	0,7	59	59,4	395,9	0,5
Mk07	140	140,9	685,8	0,3	**139**	140,1	1119,9	0,6	**139**	139,9	495,1	0,3
Mk08	**523**	523,0	4,4	0,0	**523**	523,0	5,1	0,0	**523**	523,0	0,5	0,0
Mk09	**307**	307,0	6,7	0,0	**307**	307,0	9,9	0,0	**307**	307,0	20,2	0,0
Mk10	202	219,4	40,0	45,9	**197**	198,6	1346,3	1,2	199	201,5	631,7	1,3
Média	173,6	175,4	129,5	4,6	172,7	173,2	322,8	0,3	173,0	173,5	165,9	0,3

Tabela 34 – Resultados detalhados dos algoritmos propostos para o conjunto de instâncias *DPdata*.

	SA				ILS/SA				CS			
I	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP
1a	2531	2541,3	721,3	6,5	**2505**	2517,6	906,5	5,1	**2505**	2516,0	269,5	4,1
2a	**2231**	2234,3	725,5	1,6	**2231**	2234,2	570,2	1,7	**2231**	2232,1	519,8	1,0
3a	2230	2233,8	24,6	2,4	**2229**	2230,0	745,5	0,5	**2229**	2229,8	638,3	0,4
4a	2506	2517,6	741,9	8,5	**2503**	2508,1	745,2	3,8	**2503**	2507,5	360,9	2,4
5a	2218	2220,9	628,3	1,9	2215	2221,4	731,3	2,8	**2213**	2218,1	654,7	2,6
6a	2216	2310,8	5,0	281,5	**2202**	2206,4	896,3	2,4	**2202**	2206,3	621,4	2,2
7a	2320	2327,7	925,7	6,9	**2286**	2295,2	1136,2	7,1	2291	2299,5	644,2	6,2
8a	2072	2075,1	429,2	1,8	2067	2069,5	1250,1	1,8	**2066**	2069,3	639,3	1,3
9a	2067	2069,9	174,3	1,7	**2064**	2066,3	602,3	1,2	**2064**	2066,0	733,5	1,2
10a	2313	2320,0	881,5	3,9	**2277**	2286,3	1480,3	6,9	**2277**	2296,8	601,2	10,2
11a	2068	2071,9	757,7	2,1	**2064**	2065,8	1099,5	1,3	**2064**	2067,1	744,1	2,1
12a	2042	2050,3	124,8	5,1	**2029**	2037,0	1262,0	3,5	2033	2039,3	758,2	2,9
13a	2274	2282,0	812,4	4,9	**2259**	2262,6	1111,9	4,0	2261	2272,0	677,3	6,9
14a	2940	3135,7	0,0	126,9	2169	2171,2	169,7	1,4	**2166**	2167,6	634,5	1,3
15a	2168	2565,8	5,7	415,1	**2163**	2165,6	874,1	1,3	2165	2166,1	615,5	0,7
16a	2269	2277,9	863,4	4,8	2255	2261,5	1443,6	4,0	**2253**	2264,3	664,6	6,9
17a	2962	3107,4	0,0	87,6	2151	2155,8	67,2	3,6	**2145**	2149,3	758,7	2,2
18a	2151	2683,7	3,7	368,4	**2134**	2141,7	828,3	3,6	2137	2140,3	793,1	1,9
Média	2309,9	2390,3	434,7	74,0	2211,3	2216,5	884,5	3,1	2211,4	2217,1	629,4	3,1

Tabela 35 – Resultados detalhados dos algoritmos propostos para o conjunto de instâncias *BCdata*.

	SA				ILS/SA				CS			
I	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP
mt10c1	**927**	927,0	27,8	0,0	**927**	927,0	24,6	0,0	**927**	927,3	70,1	0,9
mt10cc	**910**	910,0	189,1	0,0	**910**	910,0	138,5	0,0	**910**	910,0	238,1	0,0
mt10x	**922**	922,9	937,6	0,9	**922**	923,8	702,5	2,1	**922**	922,0	149,8	0,0
mt10xx	**918**	919,5	833,4	2,5	**918**	920,1	364,0	3,8	**918**	918,0	85,4	0,0
mt10xxx	**918**	922,0	659,5	4,9	**918**	921,5	691,0	4,7	**918**	918,4	114,9	1,3
mt10xy	906	907,6	306,8	0,9	906	907,7	925,7	0,7	**905**	907,7	155,0	0,9
mt10xyz	**858**	858,0	355,4	0,0	**858**	858,4	280,6	0,8	**858**	858,0	171,5	0,0
setb4c9	**914**	914,5	726,7	1,6	**914**	918,0	395,2	3,7	**914**	914,0	334,7	0,0
setb4cc	**909**	914,7	945,6	3,1	**909**	913,6	886,4	3,4	**909**	914,7	272,2	3,2
setb4x	**925**	929,2	573,7	2,4	**925**	930,0	404,2	4,3	**925**	925,6	404,3	1,0
setb4xx	**925**	927,6	1027,4	3,0	**925**	929,0	555,6	3,6	**925**	925,8	357,5	1,0
setb4xxx	**925**	926,7	701,2	1,8	**925**	930,6	446,1	4,9	**925**	925,0	504,6	0,0
setb4xy	913	921,0	937,0	4,5	**910**	918,8	712,6	4,7	**910**	917,9	347,0	4,8
setb4xyz	**905**	907,6	763,1	3,3	**905**	908,1	476,5	3,4	**905**	910,0	277,3	4,4
seti5c12	**1174**	1174,4	908,7	0,5	**1174**	1174,6	751,8	1,0	**1174**	1174,1	509,0	0,3
seti5cc	**1136**	1139,5	626,6	2,5	**1136**	1137,5	934,0	1,3	**1136**	1137,4	400,7	1,0
seti5x	1205	1207,4	823,3	2,3	1202	1207,4	870,8	2,7	**1198**	1203,9	461,2	4,3
seti5xx	**1198**	1205,9	736,9	4,6	**1198**	1208,0	1008,6	4,3	**1198**	1205,8	431,2	4,5
seti5xxx	1203	1206,7	757,9	2,3	1203	1207,3	1140,2	2,6	**1198**	1206,2	307,2	3,7
seti5xy	1138	1138,8	791,4	1,3	**1136**	1141,2	848,5	4,4	**1136**	1137,0	576,7	1,2
seti5xyz	1128	1131,7	878,9	3,3	1128	1130,3	680,7	2,7	**1127**	1131,3	476,8	3,5
Média	998,0	1000,6	690,9	2,2	997,6	1001,1	630,4	2,8	997,0	999,5	316,4	1,7

Tabela 36 – Resultados detalhados dos algoritmos propostos para o conjunto de instâncias *edata* e comparação com a literatura - Parte I.

	TS				SA				ILS/SA				CS			
I	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP
mt06	**55**	55,0	0,0	0,0	**55**	55,0	0,4	0,0	**55**	55,0	0,1	0,0	**55**	55,0	0,0	0,0
mt10	**871**	873,0	1,6	1,4	873	873,8	254,2	1,6	**871**	871,6	280,1	1,0	**871**	872,4	290,7	1,0
mt20	**1088**	1088,8	3,5	0,8	**1088**	1088,2	278,5	0,6	**1088**	1088,0	70,2	0,0	**1088**	1089,0	162,6	1,3
la01	**609**	609,0	0,0	0,0	**609**	609,0	0,9	0,0	**609**	609,0	0,3	0,0	**609**	609,0	0,0	0,0
la02	**655**	655,0	0,0	0,0	**555**	655,0	1,4	0,0	**655**	655,0	0,7	0,0	**655**	655,0	1,3	0,0
la03	**550**	550,0	1,0	0,0	**550**	550,6	88,6	0,5	**550**	550,0	74,7	0,0	**550**	550,0	61,0	0,0
la04	**568**	568,0	0,4	0,0	**568**	568,0	3,6	0,0	**568**	568,0	0,6	0,0	**568**	568,0	2,0	0,0
la05	**503**	503,0	0,0	0,0	**503**	503,0	0,9	0,0	**503**	503,0	0,3	0,0	**503**	503,0	0,0	0,0
la06	**833**	833,0	0,0	0,0	**833**	833,0	1,5	0,0	**833**	833,0	0,6	0,0	**833**	833,0	0,7	0,0
la07	**762**	762,0	0,4	0,0	**762**	762,7	252,6	1,3	**762**	762,7	221,1	1,3	**762**	763,5	144,1	0,0
la08	**845**	845,0	0,0	0,0	**845**	845,0	8,0	0,0	**845**	845,0	3,8	0,0	**845**	845,0	4,3	0,0
la09	**878**	878,0	0,0	0,0	**878**	878,0	2,1	0,0	**878**	878,0	0,7	0,0	**878**	878,0	1,0	0,0
la10	**866**	866,0	0,0	0,0	**866**	866,0	1,5	0,0	**866**	866,0	0,5	0,0	**866**	866,0	0,8	0,0
la11	**1103**	1103,0	1,9	0,0	**1103**	1103,2	211,9	0,4	**1103**	1103,0	310,3	0,0	**1103**	1103,2	273,6	0,0
la12	**960**	960,0	0,0	0,0	**960**	960,0	2,3	0,0	**960**	960,0	0,8	0,0	**960**	960,0	1,8	0,0
la13	**1053**	1053,0	0,0	0,0	**1053**	1053,0	2,2	0,0	**1053**	1053,0	0,8	0,0	**1053**	1053,0	1,9	0,0
la14	**1123**	1123,0	0,0	0,0	**1123**	1123,0	2,4	0,0	**1123**	**1123,0**	0,9	0,0	**1123**	1123,0	2,0	0,0
la15	**1111**	1111,0	0,3	0,0	**1111**	1111,0	12,7	0,0	**1111**	1111,0	3,0	0,0	**1111**	1111,0	7,1	0,0
la16	**892**	892,0	0,3	0,0	**892**	893,5	187,4	4,7	**892**	892,0	97,3	0,0	**892**	892,0	87,0	0,0
la17	**707**	707,0	0,6	0,0	**707**	707,0	4,4	0,0	**707**	707,0	1,4	0,0	**707**	707,0	5,4	0,0
la18	**842**	842,0	0,8	0,0	**842**	843,1	195,6	0,6	**842**	842,8	129,3	0,4	**842**	842,9	87,3	0,3
la19	**796**	796,0	1,5	0,0	**796**	798,5	114,7	1,0	**796**	796,2	100,7	0,6	**796**	797,7	105,2	0,9
Média	803,2	803,3	0,6	0,1	798,7	803,6	74,0	0,5	803,2	803,3	59,0	0,1	803,2	803,5	56,4	0,2
#BKS		22				21				22				22		

Tabela 37 – Resultados detalhados dos algoritmos propostos para o conjunto de instâncias *edata* e comparação com a literatura - Parte II.

	TS				SA				ILS/SA				CS			
I	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP
la20	**857**	857,0	0,9	0,0	**857**	857,0	227,9	0,0	**857**	857,0	97,2	0,0	**857**	857,0	150,7	0,0
la21	**1017**	1023,2	2,8	5,3	**1017**	1025,0	145,0	3,6	**1017**	1020,9	294,1	3,6	**1017**	1017,5	376,4	1,6
la22	**882**	883,0	4,3	1,4	888	891,2	419,1	3,0	883	886,8	323,0	2,0	883	886,6	347,5	1,9
la23	**950**	950,0	3,0	0,0	**950**	950,0	208,6	0,0	**950**	950,2	161,3	0,4	**950**	950,0	269,9	0,0
la24	909	911,6	3,9	2,6	909	914,0	347,8	3,3	**908**	909,6	218,5	1,6	**908**	908,9	369,9	0,3
la25	**941**	945,0	1,8	2,8	945	948,9	304,6	2,3	945	947,5	451,8	1,9	945	949,1	249,9	2,3
la26	1125	1127,0	5,5	3,4	1124	1127,7	780,4	2,8	**1113**	1118,4	1251,6	4,2	**1113**	1121,9	611,6	4,1
la27	1186	1188,8	9,3	3,3	1188	1190,9	982,3	2,9	**1182**	1186,3	633,5	3,2	**1182**	1188,1	571,1	2,6
la28	1149	1149,0	3,4	0,0	1149	1153,0	800,4	4,3	**1147**	1147,7	926,8	0,9	1149	1150,2	337,8	2,6
la29	1118	1120,6	5,5	2,9	1118	1124,1	830,1	4,7	**1116**	1118,6	775,7	2,3	**1116**	1118,1	750,0	1,3
la30	1204	1213,2	9,2	6,5	1209	1212,8	388,5	2,6	**1201**	1204,8	776,0	2,4	1203	1209,9	671,2	3,1
la31	1539	1540,6	9,6	0,9	1541	1541,0	255,0	0,0	**1536**	1540,9	388,5	2,1	1541	1542,3	323,9	2,4
la32	**1698**	1698,0	1,9	0,0	**1698**	1698,0	24,6	0,0	**1698**	1698,0	16,2	0,0	**1698**	1698,0	24,4	0,0
la33	**1547**	1547,0	1,4	0,0	**1547**	1547,0	17,0	0,0	**1547**	1547,0	7,2	0,0	**1547**	1547,0	24,0	0,0
la34	**1599**	1599,2	9,4	0,5	**1599**	1605,0	405,4	5,2	**1599**	1599,0	527,1	0,0	**1599**	1599,3	574,0	0,9
la35	**1736**	1735,0	0,4	0,0	**1736**	1736,0	13,6	0,0	**1736**	1736,0	4,7	0,0	**1736**	1736,0	17,3	0,0
la36	1162	1163,2	8,1	1,8	1164	1167,2	512,2	2,7	**1160**	1165,1	272,4	2,6	**1160**	1163,1	558,3	2,3
la37	**1397**	1397,0	3,5	0,0	**1397**	1404,4	754,5	9,1	**1397**	1408,2	1165,6	5,8	**1397**	1403,3	497,2	7,6
la38	1144	1146,6	6,9	2,6	1548	1157,1	691,4	4,0	**1143**	1156,1	1012,9	4,7	1144	1152,5	722,1	5,5
la39	**1184**	1185,6	8,7	1,3	1190	1192,7	672,9	2,5	1186	1190,2	653,0	1,8	1187	1189,2	540,7	1,5
la40	1150	1151,6	7,8	2,6	1150	1153,1	1142,4	2,1	**1146**	1149,4	1183,1	3,2	1147	1149,3	502,8	2,9
Média	1214,0	1215,8	5,1	1,8	1234,5	1218,9	472,6	2,6	1212,7	1216,1	530,5	2,0	1213,3	1216,1	404,3	2,0
#BKS		11				8				18				13		

Tabela 38 – Resultados detalhados dos algoritmos propostos para o conjunto de instâncias *rdata* e comparação com a literatura - Parte I.

	TS				SA				ILS/SA				CS			
I	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP
mt06	**47**	47,0	0,0	0,0	**47**	47,0	0,8	0,0	**47**	47,0	0,6	0,0	**47**	47,0	0,0	0,0
mt10	**686**	686,0	2,7	0,0	**686**	687,1	134,0	1,9	**686**	686,1	162,3	0,3	**686**	686,0	99,4	0,0
mt20	**1022**	1022,6	3,5	0,6	**1022**	1022,6	441,8	0,5	**1022**	1022,8	191,6	0,4	**1022**	1022,3	125,9	0,5
la01	**571**	571,8	2,0	0,8	**571**	571,6	238,3	0,5	**571**	571,6	209,3	0,5	**571**	571,4	219,7	0,5
la02	**530**	530,6	1,3	0,6	**530**	530,8	82,6	0,4	**530**	530,6	217,9	0,7	**530**	530,7	106,4	0,5
la03	478	478,2	1,4	0,5	**477**	477,9	436,3	0,3	**477**	477,8	374,5	0,4	**477**	477,9	206,7	0,3
la04	**502**	503,0	0,6	0,7	503	503,2	324,3	0,4	**502**	503,1	137,8	0,6	**502**	502,6	212,3	0,7
la05	**457**	457,6	1,8	0,6	**457**	457,7	283,1	0,5	**457**	457,3	244,6	0,5	**457**	457,0	255,4	0,0
la06	**799**	799,4	3,0	0,6	**799**	799,1	375,7	0,3	**799**	799,1	269,9	0,3	**799**	799,0	264,0	0,0
la07	750	750,0	1,1	0,0	**749**	749,9	163,7	0,3	750	750,1	100,6	0,3	750	750,0	183,3	0,0
la08	**765**	765,8	0,4	0,5	**765**	765,5	259,4	0,5	**765**	765,9	66,5	0,3	**765**	765,8	91,1	0,4
la09	**853**	853,4	2,3	0,6	**853**	853,2	449,1	0,4	**853**	853,2	272,8	0,4	**853**	853,1	308,8	0,3
la10	**804**	804,6	1,3	0,6	**804**	804,6	110,7	0,5	**804**	804,7	103,1	0,5	**804**	804,9	70,8	0,3
la11	**1071**	1071,0	2,6	0,0	**1071**	1071,0	54,1	0,0	**1071**	1071,0	64,4	0,0	**1071**	1071,0	69,2	0,0
la12	**936**	936,0	0,1	0,0	**936**	936,0	9,1	0,0	**936**	936,0	20,3	0,0	**936**	936,0	15,2	0,0
la13	**1038**	1038,0	0,9	0,0	**1038**	**1038,0**	29,8	0,0	**1038**	1038,0	53,2	0,0	**1038**	1038,0	37,3	0,0
la14	**1070**	1070,0	0,3	0,0	**1070**	1070,0	19,7	0,0	**1070**	1070,0	24,1	0,0	**1070**	1070,0	13,0	0,0
la15	1090	1090,0	1,8	0,0	1090	1090,0	51,3	0,0	**1089**	1089,9	73,0	0,3	**1089**	1089,9	147,8	0,3
la16	**717**	717,0	0,1	0,0	**717**	717,0	3,5	0,0	**717**	717,0	3,0	0,0	**717**	717,0	3,9	0,0
la17	**646**	646,0	0,0	0,0	**646**	646,0	2,6	0,0	**646**	646,0	1,7	0,0	**646**	646,0	3,1	0,0
la18	**666**	666,0	1,8	0,0	**666**	667,4	161,7	1,3	**666**	666,6	177,9	1,3	**666**	666,0	163,9	0,0
la19	**700**	701,2	1,9	1,6	703	703,0	24,7	0,0	**700**	702,4	248,7	1,3	**700**	707,1	30,4	5,1
Média	736,3	736,6	1,4	0,3	736,4	736,8	166,2	0,4	736,2	736,6	137,2	0,4	736,2	736,8	119,4	0,4
#BKS		19				19				21				21		

Tabela 39 – Resultados detalhados dos algoritmos propostos para o conjunto de instâncias *rdata* e comparação com a literatura - Parte II.

	TS				SA				ILS/SA				CS			
I	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP
la20	**756**	756,0	0,0	0,0	**756**	756,0	5,2	0,0	**756**	756,0	2,9	0,0	**756**	756,0	3,6	0,0
la21	**835**	841,0	7,8	4,2	842	845,7	791,1	2,5	836	840,0	720,3	2,7	839	841,4	666,8	2,4
la22	760	763,6	5,1	2,1	768	770,7	750,3	2,7	**757**	760,5	1051,0	2,2	760	764,5	631,7	3,0
la23	842	845,2	6,5	3,0	841	846,0	591,9	2,7	**840**	843,8	640,1	2,9	842	844,7	564,9	1,8
la24	808	813,8	4,1	3,4	812	814,6	993,0	1,8	**806**	810,6	814,5	2,3	808	810,3	618,6	1,8
la25	791	794,4	3,4	2,2	793	797,3	859,4	3,2	**787**	792,9	614,2	2,7	789	794,9	721,8	2,6
la26	**1061**	1063,8	7,7	1,6	1062	1063,9	720,7	0,9	**1061**	1062,5	756,8	1,0	**1061**	1063,2	569,8	1,1
la27	1091	1092,6	7,5	1,3	1090	1091,8	872,7	1,1	**1088**	1090,3	711,4	1,6	**1088**	1090,5	677,6	1,2
la28	1080	1081,6	7,5	1,1	**1079**	1081,0	682,7	0,8	**1079**	1080,6	517,8	1,0	**1079**	1080,5	689,2	0,7
la29	998	998,6	4,0	1,3	998	999,1	504,7	0,6	997	998,0	488,2	0,9	**996**	997,5	752,2	0,8
la30	1078	1080,8	7,8	1,8	1082	1086,0	482,4	2,0	1078	1080,5	699,6	1,8	**1077**	1081,3	850,2	2,1
la31	**1521**	1522,0	8,6	1,0	**1521**	1521,6	342,1	0,5	**1521**	1521,5	444,6	0,5	**1521**	1521,8	271,8	0,4
la32	**1659**	1659,8	12,7	1,1	**1659**	1659,0	308,5	0,0	**1659**	1659,1	428,7	0,3	**1659**	1659,0	545,3	0,0
la33	1499	1500,0	11,5	1,0	**1498**	1498,8	284,9	0,4	**1498**	1498,9	322,8	0,3	**1498**	1498,9	310,7	0,3
la34	**1536**	1536,2	7,3	0,5	**1536**	1536,0	400,0	0,0	**1536**	1536,3	415,7	0,5	**1536**	1536,0	531,2	0,0
la35	**1550**	1550,6	15,3	0,6	**1550**	1550,1	553,8	0,3	**1550**	1550,5	257,6	0,5	**1550**	1550,2	485,5	0,4
la36	1030	1031,2	4,9	1,1	**1028**	1030,2	690,2	2,4	**1028**	1029,2	713,3	1,8	**1028**	1028,7	720,7	0,9
la37	1077	1080,6	9,5	2,7	1077	1081,2	731,0	2,7	**1068**	1078,5	810,9	5,1	**1068**	1078,5	818,3	3,8
la38	962	968,0	9,3	3,7	969	973,3	708,8	2,6	**958**	969,5	671,5	8,1	959	969,5	655,8	4,3
la39	**1024**	1033,2	2,8	13,3	**1024**	1025,7	705,1	1,3	**1024**	1024,0	670,3	0,0	**1024**	1024,0	579,4	0,0
la40	970	974,0	6,1	3,4	971	973,4	1386,9	2,5	966	970,1	807,7	4,4	**965**	970,7	777,7	3,7
Média	1091,8	1094,6	7,1	2,4	1093,1	1095,3	636,4	1,5	1090,1	1093,0	598,1	1,9	1090,6	1093,4	592,5	1,5
#BKS	8				9				17				15			

Tabela 40 – Resultados detalhados dos algoritmos propostos para o conjunto de instâncias *vdata* e comparação com a literatura - Parte I.

	TS				SA				ILS/SA				CS			
I	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP
mt06	**47**	47,0	0,0	0,0	**47**	47,0	0,7	0,0	**47**	47,0	0,3	0,0	**47**	47,0	0,0	0,0
mt10	**655**	655,0	0,0	0,0	**655**	655,0	2,6	0,0	**655**	655,0	1,2	0,0	**655**	655,0	0,0	0,0
mt20	**1022**	1022,0	3,8	0,0	**1022**	1022,3	92,7	0,5	**1022**	1022,1	106,4	0,3	**1022**	1022,0	104,1	0,0
la01	**570**	570,8	1,0	0,5	**570**	570,9	402,6	0,3	**570**	570,7	316,0	0,5	571	571,0	93,7	0,0
la02	**529**	529,4	2,3	0,6	**529**	529,9	232,6	0,6	**529**	529,9	168,1	0,3	**529**	529,6	97,1	0,5
la03	**477**	477,6	0,6	0,6	**477**	477,9	300,4	0,3	**477**	477,9	171,2	0,3	**477**	477,4	249,5	0,5
la04	**502**	502,0	1,2	0,0	**502**	502,1	121,1	0,3	**502**	502,0	169,8	0,0	**502**	502,0	118,6	1,0
la05	**457**	458,0	1,4	0,7	458	458,4	156,5	0,5	**457**	458,4	169,7	0,6	**457**	457,9	221,7	0,3
la06	**799**	799,0	1,0	0,0	**799**	799,2	193,6	0,4	**799**	799,0	117,6	0,0	**799**	799,0	250,1	0,0
la07	**749**	749,8	3,6	0,8	750	750,3	211,9	0,5	**749**	749,9	318,9	0,3	750	750,0	106,6	0,0
la08	**765**	765,2	1,8	0,5	**765**	765,8	33,3	0,4	**765**	765,9	16,1	0,3	**765**	765,7	42,7	0,5
la09	**853**	853,0	2,6	0,0	**853**	853,5	107,2	0,5	**853**	853,6	56,9	0,5	**853**	853,4	98,9	0,5
la10	**804**	804,0	1,7	0,0	**804**	804,9	21,5	0,3	**804**	804,6	42,0	0,5	**804**	804,7	34,0	0,5
la11	**1071**	1071,0	0,7	0,0	**1071**	1071,0	50,8	0,0	**1071**	1071,0	23,1	0,0	**1071**	1071,0	30,8	0,0
la12	**936**	936,0	0,6	0,0	**936**	936,0	21,4	0,0	**936**	936,0	7,8	0,0	**936**	936,0	10,5	0,0
la13	**1038**	1038,0	0,7	0,0	**1038**	1038,0	66,5	0,0	**1038**	1038,0	42,5	0,0	**1038**	1038,0	72,3	0,0
la14	**1070**	1070,0	0,5	0,0	**1070**	1070,0	22,9	0,0	**1070**	1070,0	19,1	0,0	**1070**	1070,0	19,5	0,0
la15	**1089**	1089,4	1,6	0,6	**1089**	1089,7	161,3	0,5	**1089**	1089,6	85,2	0,5	**1089**	1089,4	223,6	0,5
la16	**717**	717,0	0,0	0,0	**717**	717,0	2,4	0,0	**717**	717,0	1,0	0,0	**717**	717,0	0,0	0,0
la17	**646**	646,0	0,0	0,0	**646**	646,0	2,3	0,0	**646**	646,0	0,9	0,0	**646**	646,0	0,0	0,0
la18	**663**	663,0	0,0	0,0	**663**	663,0	2,5	0,0	**663**	663,0	1,1	0,0	**663**	663,0	0,0	0,0
la19	**617**	617,0	0,1	0,0	**617**	617,0	4,0	0,0	**617**	617,0	2,4	0,0	**617**	617,0	2,3	0,0
Média	730,7	730,9	1,1	0,2	730,8	731,1	100,5	0,2	730,7	731,1	83,5	0,2	730,8	731,0	80,7	0,2
#BKS	22				20				22				20			

Tabela 41 – Resultados detalhados dos algoritmos propostos para o conjunto de instâncias *vdata* e comparação com a literatura - Parte II.

	TS				SA				ILS/SA				CS			
I	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP	FO	Av	T(s)	DP
la20	**756**	756,0	0,0	0,0	**756**	756,0	2,1	0,0	**756**	756,0	0,8	0,0	**756**	756,0	0,0	0,0
la21	806	807,6	4,7	1,5	808	809,8	725,0	1,6	**805**	805,5	846,3	0,7	**805**	806,8	833,6	1,2
la22	739	739,8	6,4	0,8	739	741,1	967,5	1,4	**736**	736,8	1057,6	0,6	**736**	738,2	780,1	1,2
la23	815	816,0	5,7	1,2	817	819,0	837,6	0,9	**813**	813,8	1082,1	0,9	**813**	814,7	819,1	1,3
la24	777	779,0	6,8	1,6	778	781,2	847,1	1,5	**776**	777,0	836,6	0,7	**776**	778,7	708,2	1,6
la25	756	756,4	6,3	0,6	758	759,1	949,7	0,9	**754**	755,0	1099,0	0,8	756	756,9	942,4	1,1
la26	**1054**	1054,6	8,6	0,6	**1054**	1054,7	506,3	0,5	**1054**	1054,0	268,0	0,0	**1054**	1054,3	537,8	0,5
la27	**1085**	1085,8	6,4	0,5	**1085**	1085,9	312,4	0,3	**1085**	1085,1	675,6	0,3	**1085**	1085,9	510,9	0,3
la28	**1070**	1070,4	12,7	0,6	**1070**	1070,7	302,7	0,5	**1070**	1070,0	279,9	0,0	**1070**	1070,1	667,7	0,3
la29	**994**	994,6	13,1	0,6	**994**	994,9	505,8	0,3	**994**	994,5	429,8	0,5	**994**	994,7	687,9	0,5
la30	**1069**	1070,0	6,9	0,7	1070	1070,4	457,3	0,5	**1069**	1069,8	535,4	0,4	**1069**	1069,9	800,1	0,3
la31	**1520**	1520,0	16,3	0,0	**1520**	1520,7	415,9	0,5	**1520**	1520,7	303,8	0,5	**1520**	1520,8	255,2	0,4
la32	**1658**	1658,0	13,0	0,0	**1658**	1658,0	302,6	0,0	**1658**	1658,0	375,3	0,0	**1658**	1658,1	422,9	0,3
la33	**1497**	1497,8	17,1	0,5	1498	1498,0	204,1	0,0	1498	1498,0	278,7	0,0	1498	1498,1	439,0	0,3
la34	**1535**	1535,2	13,5	0,5	**1535**	1535,2	418,2	0,4	**1535**	1535,1	475,7	0,3	**1535**	1535,2	535,9	0,4
la35	**1549**	1549,0	20,7	0,0	1550	1550,0	50,6	0,0	**1549**	1550,0	29,9	0,3	1550	1550,0	174,3	0,0
la36	**948**	948,0	0,3	0,0	**948**	948,0	11,2	0,0	**948**	948,0	10,1	0,0	**948**	948,0	13,3	0,0
la37	**986**	986,0	0,4	0,0	**986**	986,0	11,2	0,0	**986**	986,0	10,0	0,0	**986**	986,0	16,7	0,0
la38	**943**	943,0	0,1	0,0	**943**	943,0	10,0	0,0	**943**	943,0	8,7	0,0	**943**	943,0	3,1	0,0
la39	**922**	922,0	0,2	0,0	**922**	922,0	11,7	0,0	**922**	922,0	10,1	0,0	**922**	922,0	28,3	0,0
la40	**955**	955,0	0,3	0,0	**955**	955,0	10,1	0,0	**955**	955,0	8,7	0,0	**955**	955,0	3,8	0,0
Média	1068,3	1068,8	7,6	0,4	1068,8	1069,5	374,3	0,4	1067,9	1068,3	410,6	0,3	1068,0	1068,7	437,2	0,5
#BKS	16				13				20				18			