UNIVERSIDADE FEDERAL DE JUIZ DE FORA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

LARISSA GABRIELA DE ABREU TOLEDO

METAHEURÍSTICA SIMULATED ANNEALING APLICADA EM PROBLEMA DE JOB SHOP SCHEDULING COM RESTRIÇÕES

JUIZ DE FORA
2014

LARISSA GABRIELA DE ABREU TOLEDO

METAHEURÍSTICA SIMULATED ANNEALING APLICADA EM PROBLEMA DE JOB SHOP SCHEDULING COM RESTRIÇÕES

Trabalho de Conclusão de Curso apresentado a Faculdade de Engenharia da Universidade Federal de Juiz de Fora, como requisito parcial para a obtenção do título de Engenheiro de Produção.

Orientador: D. Sc., Fernando Marques de Almeida Nogueira

JUIZ DE FORA

2014

Toledo, Larissa Gabriela de Abreu.
Metaheurística Simulated Annealing aplicada em problema de Job Shop Scheduling com restrições / Larissa Gabriela de Abreu Toledo. -- 2014.
98 f.

Orientador: Fernando Marques de Almeida Nogueira
Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Juiz de Fora, Faculdade de Engenharia, 2014.

1. Otimização de scheduling . 2. Simulated Annealing. 3. Job Shop. I. Nogueira, Fernando Marques de Almeida, orient. II. Título.

LARISSA GABRIELA DE ABREU TOLEDO

METAHEURÍSTICA SIMULATED ANNEALING APLICADA EM PROBLEMA DE JOB SHOP SCHEDULING COM RESTRIÇÕES

Trabalho de Conclusão de Curso apresentado a Faculdade de Engenharia da Universidade Federal de Juiz de Fora, como requisito parcial para a obtenção do título de Engenheiro de Produção.

Aprovada em 17 de julho de 2014.

BANCA EXAMINADORA

D. Sc., Fernando Marques de Almeida Nogueira

Universidade Federal de Juiz de Fora

M. Sc., Eduardo Pestana de Aguiar

Universidade Federal de Juiz de Fora

D. Sc., Roberto Malheiros Moreira Filho

Universidade Federal de Juiz de Fora

AGRADECIMENTOS

Ao meu orientador, Fernando Nogueira pela excelência em ensinar, paciência, incentivo e dedicação que foram essenciais não só para a realização deste trabalho, como também para minha formação profissional durante o período em que fui sua aluna.

Aos membros da banca, Eduardo Aguiar e Roberto Malheiros pelas suas importantes sugestões, comentários e críticas, que contribuíram muito para a consolidação desse trabalho.

Aos meus amigos do curso pelo companheirismo, amizade e por provarem que a passagem pela universidade vai muito além da aquisição de conhecimento.

Aos meus pais, José Mauro e Dacimar pela minha formação, por acreditarem em mim e serem meus exemplos de vida.

À minha irmã, Fernanda pelos conselhos, incentivo nunca me deixando desistir.

Ao Felipe pelo amor, carinho, cumplicidade e por estar sempre ao meu lado.

RESUMO

Este trabalho contempla o tratamento analítico de decisões relacionadas à *scheduling* em um ambiente de máquinas do tipo *Job Shop* por meio da utilização da metaheurística *Simulated Annealing*. Problemas de *Job Shop Scheduling* são de natureza combinatória, para os quais o tempo de processamento aumenta de forma exponencial em função do número de *jobs* e/ou máquinas, para resolvê-los, usualmente, empregam-se métodos heurísticos, porém, sem a garantia de obtenção da solução ótima. O *Simulated Annealing* é uma metaheurística de otimização, que consiste em um método de busca local com probabilidade de aceitação de movimentos de piora, a fim de explorar mais pontos do espaço solução em busca do mínimo global. Tal técnica é inspirada no modo em que um metal em estado líquido se resfria e passa para o estado sólido formando uma estrutura cristalina de energia mínima. O trabalho tem por finalidade adaptar o algoritmo *Simulated Annealing*, proposto por Kirkpatrick et al. (1983), à resolução de problemas genéricos e estáticos de *Job Shop Scheduling* com restrições dos tipos *release date* e *due date*. Foram desenvolvidas duas formas de codificação para as soluções do problema, sendo que, em uma nunca são geradas solução inviável, porém apresenta ambiguidade em sua representação. A outra é uma codificação única, mas pode gerar soluções inviáveis. O desempenho do algoritmo proposto utilizando essas duas formas de codificação de soluções foi avaliado a partir da resolução de instâncias com diferentes níveis de complexidade. Os resultados do trabalho compreendem a superioridade das soluções encontradas em relação às obtidas pelo *software* LiSA.

Palavras-chave: Otimização de *scheduling*, *Simulated Annealing*, *Job Shop*

ABSTRACT

This paper presents an analytical treatment of decisions related to scheduling in an environment of machines of type Job Shop through the use of the metaheuristic Simulated Annealing. Job Shop Scheduling Problems are of combinatorial nature, for which the processing time increases exponentially according to the number of jobs and/or machines, usually for solve them heuristic methods are employed, but without the guarantee of obtaining the optimal solution. Simulated Annealing is a metaheuristic for optimization, which consists in a local search method with probability of accepting worsening moves in order to explore more points in the solution space seeking global minimum. This technique is inspired by the way that a liquid metal cools and passes into the solid state forming a crystalline structure of minimum energy. The research aims to adapt the Simulated Annealing algorithm, as proposed by Kirkpatrick et al. (1983), to the resolution of generic and static Job Shop Scheduling problems with restrictions of release date and due date types. Two forms of encoding the solutions of the problem have been developed, one never generates infeasible solutions, but presents ambiguity in its representation. The performance of the proposed algorithm using these two ways of encoding solutions was evaluated from the resolution of instances with different levels of complexity. The results of this work include the superiority of the solutions compared with those obtained by LiSA software.

Keywords: Scheduling optimization, Simulated Annealing, Job Shop

LISTA DE FIGURAS

Figura 1 – Metodologia de pesquisa em Engenharia de Produção 17
Figura 2 – Problema de *Job Shop* modelado por Grafo Disjuntivo 25
Figura 3 – Seleção completa consistente do grafo da Figura 2 25
Figura 4 – Matriz de tempo de processamento 32
Figura 5 – Exemplo de parâmetros usados na definição do tamanho de lote 33
Figura 6 – Matriz de ordem de processamento 33
Figura 7 – Exemplos das saídas da função *gerar_instancias.m* 35
Figura 8 – Esquema do algoritmo *Simulated Annealing* adaptado a problemas de *Job Shop* 36
Figura 9 – Exemplo de solução para um problema 3x3 representada pela codificação I 38
Figura 10 – Exemplo de falta de unicidade da codificação I 39
Figura 11 – Matriz representativa da codificação II 40
Figura 12 – Exemplo de solução para problema 3x3 representada pela codificação II 40
Figura 13 – Exemplo de solução inviável 41
Figura 14 – Esquematização da função *gera_sol_1.m* 43
Figura 15 – Exemplo saídas *gera_sol_1.m* 43
Figura 16 – Esquematização da função *gera_sol_11.m* 44
Figura 17 – Esquematização da função *gerar_sol_2.m* 45
Figura 18 – Exemplo saídas *gera_sol_2.m* 46
Figura 19 – Esquema da primeira parte da função *calc_energia_2.m* 49
Figura 20 – Esquema da segunda parte da função *calc_energia_2.m* 50
Figura 21 – Esquema da terceira parte da função *calc_energia_2.m* 51
Figura 22 – Matriz de tempo de conclusão das operações 52
Figura 23 – Esquema da função para estimar a temperatura inicial 53
Figura 24 – Gráfico de Gantt para instância 4.1-1 - *Branch and Bound* 56
Figura 25 – Gráfico de Gantt para instância 4.1-1 - *SA_JS_1* 56
Figura 26 – Gráfico de Gantt para instância 4.1-2 - *Branch and Bound* 57
Figura 27 – Gráfico de Gantt para instância 4.1-2 - *SA_JS_1* 58
Figura 28 – Gráfico de Gantt para instância 4.1-2 - *SA_JS_2* 58

Figura 29 – Comparação dos resultados encontrados pela *SA_JS_2* variando a probabilidade de utilização das formas de gerar novas soluções 61
Figura 30 – Janela de *input* dos parâmetros do método *Branch and Bound* 61
Figura 31 – Janela de *input* dos parâmetros do método Busca Tabu 62
Figura 32 – Janela de *input* dos parâmetros do *Simulated Annealing* 64
Figura 33 – Comparação dos resultados obtidos pelas três variações do algoritmo para problemas 66
Figura 34 – Comparação dos resultados obtidos pelas três variações do algoritmo para problemas 67
Figura 35 – Comparação dos resultados obtidos pelas três variações do algoritmo para problemas 68
Figura 36 – Comparação do número de iterações realizadas pela *SA_JS_1* e *SA_JS_11* 70

LISTA DE TABELAS

Tabela 1 – Cronograma de atividades 19

Tabela 2 – Interpretação da terminologia do sistema físico para o domínio computacional..... 29

Tabela 3 – Exemplo de sensibilidade da função objetivo 46

Tabela 4 – Resultados para J | r_i | Lmax 5 máquinas e 5 *job*s (instância 4.1-1).. 55

Tabela 5 – Resultados para J | r_i | Lmax 5 máquinas e 7 *job*s (instância 4.1-2).. 57

Tabela 6 – Resultados dos problemas J | r_i | Lmax – 8 *job*s x 7 máquinas 66

Tabela 7 – Resultados dos problemas J | r_i | Lmax – 10 *job*s x 12 máquinas 67

Tabela 8 – Resultados dos problemas J | r_i | Lmax – 20 *job*s x 17 máquinas 68

SUMÁRIO

1. INTRODUÇÃO 13

1.1 CONSIDERAÇÕES INICIAIS 13
1.2 JUSTIFICATIVA 15
1.3 ESCOPO DO TRABALHO 16
1.4 ELABORAÇÃO DOS OBJETIVOS 16
1.5 DEFINIÇÃO DA METODOLOGIA 17
1.6 ESTRUTURA DO TRABALHO 18
1.7 CRONOGRAMA 19

2. REVISÃO DE LITERATURA 20

2.1 ESTRUTURA E NOTAÇÕES 20
2.2 JOB SHOP SCHEDULING 23
2.2.1 MODELAGEM DO PROBLEMA DE JOB SHOP 24
2.2.2 ENTROPIA EM PROBLEMA DE JOB SHOP 27
2.3 SIMULATED ANNEALING 27
2.3.1 PRINCÍPIOS BÁSICOS DE TERMODINÂMICA ESTATÍSTICA 29
2.3.2 ALGORITMO SIMULATED ANNEALING 29

3. DESENVOLVIMENTO 31

3.1 DESCRIÇÃO DO PROTOCOLO DE PESQUISA 31
3.1.1 GERAÇÃO DE INSTÂNCIAS 31
3.2 ELABORAÇÃO DO ALGORITMO 35
3.2.1 CODIFICAÇÃO DE SOLUÇÕES 37
3.2.2 GERAÇÃO DE NOVAS SOLUÇÕES 41
3.2.3 CÁLCULO DA FUNÇÃO OBJETIVO 46
3.2.4 ESTIMAR A TEMPERATURA INICIAL 52
3.2.5 PARÂMETROS DO ALGORITMO 54

4. RESULTADOS 55

4.1 VALIDAÇÃO DO ALGORITMO 55
4.2 ANÁLISE DE RESULTADOS 59

5. CONSIDERAÇÕES FINAIS 71

5.1 CONCLUSÕES 71
5.2 RECOMENDAÇÕES DE TRABALHOS FUTUROS 72

REFERÊNCIAS 73

ANEXO 1 – INSTÂNCIAS 76

ANEXO 3 – ALGORITMO SIMULATED ANNEALING ADAPTADO 89

ANEXO 4 – FUNÇÃO GERAR_SOL_1.M 91

ANEXO 6 – FUNÇÃO GERAR_SOL_2.M 93

ANEXO 7 – FUNÇÃO CALC_ENERGIA_1.M 94

ANEXO 8 – FUNÇÃO CALC-ENERGIA_2.M 95

ANEXO 9 – FUNÇÃO ESTIMAT0.M 97

ANEXO 10 – TERMO DE AUTENTICIDADE 98

1. INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

Uma das principais consequências da globalização foi o acirramento da competição entre as organizações. Diante de tal fato, há uma incessante busca por diferenciais competitivos, que na maioria dos casos podem ser encontrados através de melhorias no processo produtivo (CORRÊA; CORRÊA, 2012).

As circunstâncias industriais atuais exigem que empresas de diferenciados ramos da economia desenvolvam e adaptem seus processos produtivos em tempos progressivamente menores, além da demanda por níveis de produtividade cada vez maiores. As organizações são levadas a articular melhor seus processos de Planejamento e Controle da Produção (PCP), principalmente entre os diferentes níveis hierárquicos de gestão (NORONHA; MENDES, 2002).

Uma vez que as decisões referentes ao Planejamento e Controle da produção ocorrem em diferentes horizontes de tempo, possuem diferentes períodos de replanejamento, e consideram diferentes níveis de agregação da informação, essas decisões são usualmente classificadas em três níveis: estratégico, tático e operacional. Dentre esses três horizontes de PCP, o planejamento de curto prazo (*shop scheduling*) vem ganhando destaque. Quando observados os crescentes esforços de flexibilização e automação exigidos pela dinâmica produtiva contemporânea, os quais tornam as atividades de programação detalhada da produção gradativamente mais complexas, devido principalmente ao volume de diferentes variáveis envolvidas e à sua capacidade de influenciar os diferentes e, às vezes, conflitantes objetivos de desempenho do sistema produtivo (CORRÊA; CORRÊA, 2012).

Dentre as dez áreas da Engenharia de Produção, segundo a ABEPRO (2008), cabem à área número um, Engenharia de Operações e Processos de Produção, o desenvolvimento de projetos, operações e melhorias dos sistemas que criam ou entregam os produtos primários da empresa, e à área de Pesquisa Operacional a resolução de problemas envolvendo situações de tomada de decisão, através de modelos matemáticos normalmente processados computacionalmente. Buscando assim, introduzir elementos de objetividade e racionalidade nos processos de tomada de decisão, sem se esquecer dos elementos subjetivos e de enquadramento organizacional que caracterizam o problema.

As origens da Pesquisa Operacional remontam os primórdios da Segunda Guerra Mundial. Com o advento da guerra, havia a necessidade de se alocar de forma eficiente os escassos recursos para as diversas operações militares e atividades internas a cada operação. Por consequência, os comandos militares britânico e norte-americano convocaram vários cientistas para aplicar uma abordagem cientifica nesses e outros problemas táticos e estratégicos (HILLIER; LIEBERMAN, 2006).

Com o fim da guerra, o sucesso da PO no empreendimento bélico despertou interesse por sua aplicação em outros ambientes. A partir do desdobramento do *boom* industrial pós-guerra, os problemas advindos do aumento da complexidade e especialização das organizações foram ganhando relevância e criaram condições necessárias para a aplicação da PO (HILLIER; LIEBERMAN, 2006).

A principal motivação para a realização desse trabalho consiste na percepção de um problema presente no âmbito de PCP, referente ao planejamento de curto prazo da alocação de recursos para atendimento da demanda, também conhecido como problema de *Shop Scheduling.*

Em um problema de *scheduling* real, há grande chance do tempo de processamento de cada uma das operações dos *jobs* variar de um lote de produção para o outro e de nem todos os pedidos estarem presentes para serem sequenciados no instante inicial. Logo, cada produto possui um tempo de partida que é definida pela chegada do pedido na fabrica, e tal data é imprevisível. O sequenciamento nesse tipo de problema pode ser referenciado como ressequenciamento, uma vez que, a cada entrada de um pedido, o agendamento das operações deve ser refeito.

Tais casos ilustram um *scheduling* não determinístico, que deve ser realizado em um período de tempo totalmente aberto, sendo definido como problema dinâmico.

Por outro lado, um problema de *scheduling* estático caracteriza-se pelo fato de que, uma vez inicializado o processo de produção, o sequenciamento não pode ser alterado até que todos os produtos estejam concluídos. Nesse sistema ideal, todos os recursos estão disponíveis durante todo o período de produção, os tempos de processamento são determinísticos, os tempos de transporte dos produtos de uma máquina para outra não são considerados, tampouco o tempo gasto com preparação (VÁZQUEZ; WHITLEY, 2000).

No âmbito de programação da produção, destaca-se o *Job Shop Scheduling* (JSS), um problema notoriamente difícil em otimização combinatória. Há diversos métodos de resolução desse tipo de problema. Como por exemplo, por *Branch and Bound* (BRUCKER;

JURISCH; SIEVERS, 1994), Algoritmo Genético (PEZZELLAA; MORGANTIA; CIASCHETTIB, 2008), GRASP (BINATO et al., 2002), Busca Tabú (AMICO; TRUBIAN, 1993), *Simulated Annealing* (LAARHOVEN; AARTS; LENSTRA, 1992), entre outros.

1.2 JUSTIFICATIVA

Um dos pontos mais marcantes no tipo de processo produtivo *Job Shop* é a grande variedade de produtos, com vários roteiros de fabricação. Nesse caso, múltiplas tarefas devem ser roteirizadas ao longo de uma sequência de postos de trabalho para que possam ser realizadas. Em grande parte dos casos quando uma ordem chega a um centro de trabalho, precisa aguardar em fila, enquanto outros recursos podem estar ociosos. A difícil tarefa de definir a sequência de *jobs* em cada centro de trabalho admite a aplicabilidade de um algoritmo que otimize tal sequenciamento (JAIN; MEERAN, 1998).

Pelo fato do problema de *Job Shop Scheduling* ser um NP-*hard* [1], talvez não seja possível encontrar a solução ótima, por exemplo, para um problema mais complexo com 10 *jobs* e 10 máquinas, ou até mesmo instâncias mais modestas podem ser intratáveis computacionalmente. No entanto, ainda é importante encontrar uma boa solução viável que seja pelo menos razoavelmente próxima da solução ótima (JAIN; MEERAN, 1998).

Um método heurístico é um procedimento que provavelmente vai encontrar uma boa solução viável, mas não se tem a garantia de que essa será a solução ótima. Tal método normalmente é um algoritmo iterativo em que cada iteração conduz a busca por uma nova solução que, eventualmente, poderá ser melhor que a melhor solução encontrada previamente. Em geral, se baseiam em ideias relativamente simples de senso comum de como encontrar uma boa solução. Porém, essas ideias precisam ser cuidadosamente adaptadas à realidade de cada problema em especifico. Portanto, os métodos heurísticos tendem a ser específicos por natureza (HILLIER; LIEBERMAN, 2006).

Uma metaheurística é um conjunto de conceitos que podem ser utilizados para definir métodos heurísticos aplicáveis a um extenso conjunto de problemas. Em geral, para adaptá-la a um problema específico, são exigidas pequenas modificações. Alguns exemplos de metaheurísticas são: *simulated annealing*, busca tabú, *iterated local search*, algoritmos evolutivos e colônia de formigas (METAHEURISTICS NETWORK, 2014).

---

[1] Problemas que aumentam de complexidade de forma não polinomial.

No presente trabalho fez se a escolha por analisar o desempenho, em específico da metaheurística *Simulated Annealing* na resolução de problemas genéricos e estáticos de *Job Shop Scheduling*. O emprego dessa técnica se justifica pela sua capacidade de explorar mais pontos do espaço solução, ao se buscar evitar o direcionamento a mínimos locais, o que se espera ser um diferencial na obtenção de boas soluções para os problemas.

Dado que as condições ideais de um problema estático são praticamente impossíveis em um ambiente real, seria um esforço desperdiçado resolver esse tipo de problema. No entanto, para o desenvolvimento de métodos eficazes na resolução de problemas dinâmicos, esse estudo inicial é de fundamental importância.

Além da relevância desse assunto em âmbito organizacional, a realização deste trabalho se justifica por sua importância acadêmica, representando grande oportunidade de ampliação de conhecimentos na área de pesquisa operacional.

1.3 ESCOPO DO TRABALHO

O escopo desse trabalho engloba problemas genéricos e estáticos de s*cheduling,* mais especificamente o ambiente de máquinas *Job Shop* com restrições do tipo *release date e due date* e função objetivo minimizar o atraso máximo.

Os dados para composição desses problemas genéricos são gerados de forma pseudoaleatória por um procedimento desenvolvido, permitindo que se resolvam problemas de *job shop* com diferentes tamanhos e complexidade.

1.4 ELABORAÇÃO DOS OBJETIVOS

Inicialmente o objetivo deste trabalho consiste em desenvolver um algoritmo utilizando a técnica *Simulated Annealing* que seja capaz de gerar boas soluções para problemas estáticos e genéricos de *Job Shop Scheduling* com restrições do tipo *release date* e *due date* e função objetivo minimizar atraso máximo.

Posteriormente, objetiva-se analisar a interferência da entropia dos tempos e da ordem de processamento dos *jobs* no desempenho do algoritmo utilizando duas diferentes maneiras de codificar as soluções, uma delas nunca gera soluções inviáveis, porém apresenta ambiguidade em sua representação. A outra é uma codificação única, mas pode gerar soluções inviáveis.

1.5 DEFINIÇÃO DA METODOLOGIA

Segundo Miguel (2010) a metodologia de pesquisa em Engenharia de Produção pode ser classificado a partir de circunstâncias sobre a natureza, os objetivos, a abordagem e os métodos utilizados, conforme Figura 1.

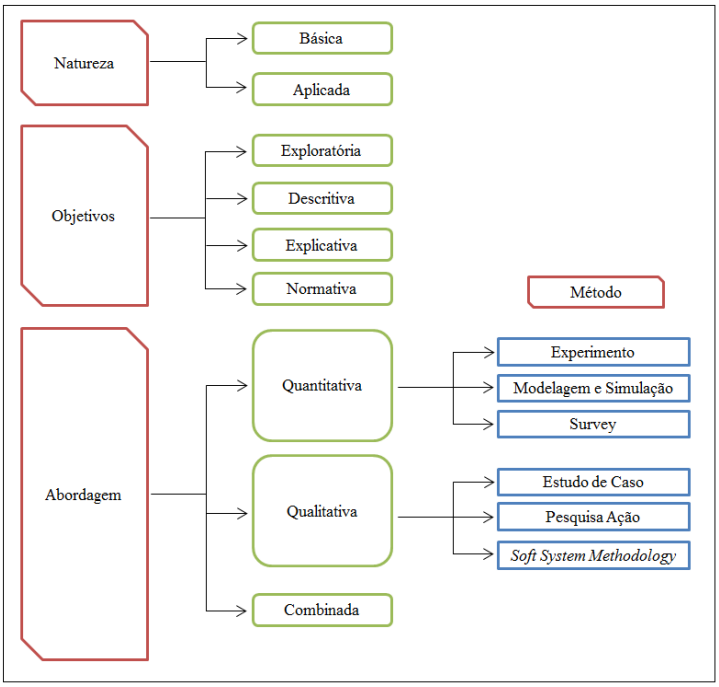

Figura 1 – Metodologia de pesquisa em Engenharia de Produção
Fonte: Miguel, 2010 (Adaptado)

No trabalho em questão, a pesquisa desenvolvida é de natureza aplicada, direcionada a parir de objetivos exploratórios e descritivos. E foi empregada uma abordagem quantitativa baseada em métodos de modelagem e simulação.

As fases de pesquisa abrangem: revisão bibliográfica referente ao ambiente de *scheduling* com ênfase nos problemas de *Job Shop*, metaheurística *Simulated Annealing*, linguagem em Mathworks Matlab, implementação e posterior validação do algoritmo através da resolução de algumas instâncias pelo *software* LiSA.

LiSA – *Library of Scheduling Algorithms* é um *software* para solução determinística de *Shop Scheduling* . Permite-se que se escolha o ambiente de máquinas, os tipos de restrições e a função objetivo. Estão disponíveis heurísticas construtivas, como por exemplo FCFS - *first come, first served*, EDD - *earliest due date first*, SPT - *shortest processing time first*,

algoritmo exato, *branch and bound* e algoritmos iterativos, a exemplo de *Iterative Improvement, Simulated Annealing, Threshold Accepting e Tabu Search.*

Os dados que compõem os problemas são gerados pseudo-aleatoriamente por um procedimento desenvolvido de forma parametrizada permitindo que se defina o número de *jobs*, o número de máquinas, a faixa de tempo de processamento por unidade de *job* em cada máquinas, tamanho médio do lote de produção, amplitude de variação e variação mínima no tamanho do lote, entropia dos tempos e da ordem de processamento dos *jobs* em cada máquina. Quanto à metodologia de resolução do problema, adaptou-se a metaheurística *Simulated Annealing*, proposta por Kirkpatrick et al. (1983), à aplicação em questão.

1.6 ESTRUTURA DO TRABALHO

O capítulo 1 trata das considerações introdutórias sobre o tema de pesquisa, a justificativa de escolha, a determinação do escopo e os objetivos, assim como considerações sobre a metodologia adotada e o cronograma previsto para as fases do trabalho.

O capítulo 2 contempla a revisão bibliográfica, sendo essa desdobrada nos principais conceitos e técnicas necessárias à compreensão e direcionamento do trabalho. Tal revisão foi baseada em artigos científicos e livros das áreas de Planejamento e Controle de Produção e de Pesquisa Operacional, com ênfase em planejamento de curto prazo, ambiente de máquinas do tipo *job shop* e metaheurística *Simulated Annealing*.

No capítulo 3 encontra-se o desenvolvimento do trabalho, inicialmente a descrição da metodologia aplicada, acompanhada pelo detalhamento do problema em questão. Em seguida, é apresentada a função geradora de instâncias. Posteriormente, as duas formas de representação e geração de soluções são expostas e exemplificadas. Por fim, o algoritmo é detalhado, enfatizando o alinhamento do método e dos parâmetros para adaptação ao problema em questão.

O capítulo 4 apresenta um estudo descritivo dos resultados computacionais obtidos através de cada uma das formas de se representar as soluções, diante de alguns cenários de entropia tanto dos tempos de processamento como da ordem de processamento.

O capítulo 5 finaliza o trabalho, com as considerações finais e a consolidação das principais conclusões e sugestões para desenvolvimento de trabalhos futuros.

1.7 CRONOGRAMA

O cronograma exposto abaixo representação a evolução de desenvolvimento das etapas presentes no trabalho:

Tabela 1 – Cronograma de atividades

Meses / Quinzenas / Itens	Novembro 13		Dezembro 13		Janeiro 14		Fevereiro 14		Março 14		Abril 14		Maio 14		Junho 14	
	1ª	2ª	1ª	2ª	1ª	2ª	1ª	2ª	1ª	2ª	1ª	2ª	1ª	2ª	1ª	2ª
Definição do tema		X														
Especificações do trabalho			X													
Revisão Bibliográfica				X	X	X	X	X								
Desenvolvimento do algoritmo em MatLab®										X	X	X	X	X	X	
Validação do algoritmo														X	X	
Simulação de Cenários															X	
Consolidação e análise dos resultados																X

2. REVISÃO DE LITERATURA

Esse capítulo aborda os conceitos e técnicas mais importantes para o desenvolvimento desse trabalho. No item 2.1, são definidos alguns conceitos e notações sobre problemas de *Shop Scheduling.* Na seção 2.2 conceitua-se o problema de *Job Shop Scheduling* A seção 2.3 abrange a técnica de otimização *Simulated Annealing*.

2.1 ESTRUTURA E NOTAÇÕES

Segundo Pinedo (2002), a notação usual representativa do número de *jobs* é *n* e para o número de máquinas é *m*. O subscrito *j* refere-se a um *job* e o índice *i* a uma máquina. Sendo que um *job* necessita ser processado por várias máquinas, o par (j,i) representa um passo desse processamento, isto é, a operação realizada no *job j* pela máquina *i*.

O tempo de processamento (p_{ji}): representa o tempo de processamento do *job j* na máquina *i*.

Peso (w_j): determina a importância do *job j* em relação aos demais no sistema.

Em problemas de *scheduling*, um grupo de tarefas devem ser processadas por um grupo de máquinas, sob diferentes restrições adicionais, de forma a minimizar um ou vários objetivos.

Convencionalmente, os problemas são expressos pela forma $\alpha \mid \beta \mid \gamma$ em que:

- α: descreve a configuração das máquinas e possui apenas uma entrada;
- β: fornece detalhes das características de processamento e restrições;
- γ : representa a função objetivo.

Alguns dos possíveis ambientes de máquinas especificados no campo α são:

Máquina única (1): consiste de uma única máquina, na qual todos os *jobs* são processados.

Máquinas idênticas em paralelo (P_m) : ambiente com *m* maquinas idênticas em paralelo, sendo que um *job j* requer uma única operação e pode ser processado em qualquer uma dessas *m* maquinas.

Máquinas em paralelo com velocidades diferentes (Q_m): ambiente semelhante ao anterior, porém as *m* máquinas possuem produtividades diferentes.

Flow Shop (F_m): ambiente com *m* máquinas em série e todos os *jobs* possuem uma mesma rota.

Job Shop (J_m): cada *job* possui sua rota pré-estabelecida, sendo que os *jobs* podem passar pela mesma máquina mais de uma vez.

Flexible Job Shop (FJ_C): é uma junção do ambiente de máquinas em paralelo com a configuração *job shop*. Em vez de *m* máquinas em série, existem centros de trabalho *c*, com *m* máquinas idênticas. E cada *job* precisa ser processado por apenas uma máquina desses centros de trabalho.

Open Shop (O_m): é um ambiente com *m* máquinas, no qual as operações de cada *job* são totalmente independentes, não existem restrições quanto à ordem de execução das operações.

No campo β são explicitadas as restrições específicas da linha de produção:

Release Dates (r_j): indica que um *job j* não pode começar a ser processado antes da sua data de lançamento r_j.

Due date (d_j): é o prazo de conclusão de cada trabalho.

Preemptions $(prmp)$: significa que não é necessário manter o *job* na máquina até a conclusão da operação, ou seja, o processo pode ser interrompido e recomeçado no mesmo ponto.

Sequence dependent setup time (S_{ijk}): essa limitação faz referência ao fato da sequência de *jobs* em cada máquina influenciar o tempo *setup*.

Breakdowns $(brkdwn)$: considera o fato das máquinas não estarem disponíveis durante todo o período de produção.

Blocking $(block)$: significa que um *job* concluído em uma determinada máquina só poderá desocupá-la quando o *job* que está sendo processado na máquina sucessora estiver concluído, portanto o *job* bloqueado impede que os próximos *jobs* sejam processados nessa máquina.

No-wait (nwt): restrição de processamento de um *job j* que não pode esperar entre duas máquinas sucessivas para ser processado, isso implica no atraso da data de início dos demais *jobs* para garantir que o *job j* percorra toda a linha de produção sem interrupção.

O campo γ refere se ao(s) objetivo(s) do problema, e está diretamente relacionado com o tempo de finalização de cada *job* nos postos de trabalho.

O tempo de finalização de um *job j* na máquina *i* é denotado por C_{ij} e o tempo total que o *job* gasta para sair do sistema é denotado por $C_{j.}$

Algumas possíveis funções de desempenho, que devem ser minimizadas são:

Makespan (Cmax): é definido por $\max(C_1, \cdots, C_n)$, isto é, o tempo de conclusão do último trabalho no sistema.

Tempo total de conclusão $\left(\sum C_j\right)$: corresponde a soma dos tempos de conclusão de todos os *jobs*, também pode ser chamado de tempo de escoamento ou de fluxo.

Tempo total de conclusão ponderada $\left(\sum w_j C_j\right)$: similar ao tempo de conclusão total, porém ponderado pelos pesos dos *jobs*.

Atraso máximo $(L_{\max})$: é definido por $\max(L_1, \cdots, L_n)$, refere-se ao *job* mais atrasado.

$$L_j = \max(C_j - d_j, 0) \tag{1}$$

Atraso total $\left(\sum L_j\right)$: refere-se à soma dos tempos de atraso de todos os *jobs*.

Número de *job*s atrasados $\left(\sum U_j\right)$: representa quantos *jobs* não foram concluídos dentro do prazo.

$$U_j = \begin{cases} 1, & se \quad C_j > d_j \\ 0, & se \quad C_j \leq d_j \end{cases} \tag{2}$$

2.2 JOB SHOP SCHEDULING

Segundo Morton e Pentico (1993) a programação da produção, ou *scheduling*, envolve a consideração de uma série de elementos que disputam vários recursos por um período de tempo, sendo que tais recursos possuem capacidade limitada.

Os componentes de um sistema de sequenciamento são as tarefas, os recursos, a função objetivo e as restrições de potencialidade. As tarefas são o elemento principal do sequenciamento, e devem ser ordenadas de modo a cumprir as atividades otimizando a função objetivo sem deixar extrapolar a utilização dos recursos. Para realização das tarefas demanda-se alocação de recursos, que podem ser mão-de-obra, tempo de máquinas, trecho de uma ferrovia, ou qualquer outro bem que esteja sendo consumidos durante a sua execução (JAIN; MEERAN, 1998).

Segundo Jain e Meeran (1998) as restrições de potencialidade são divididas em restrições de precedência e restrições de localização no tempo. As restrições de precedência não permitem que uma tarefa seja inicializada antes da conclusão de sua antecessora. E as restrições de localização no tempo estabelecem que uma tarefa não pode ser inicializada antes de um determinado instante (*release date*).

O objetivo principal do *scheduling* é otimizar a função objetivo, que pode assumir diversas formas, dependendo da área em que está sendo aplicado. Como por exemplo, duração total do sequenciamento (*makespan*), o ciclo de produção, o ciclo de produção ponderado, o atraso máximo, o atraso total, atraso total ponderado, e o número de *jobs* em atraso (BAKER, 1974 apud Massa, 2004).

Os casos, em que o sequenciamento é aplicado a um conjunto de máquinas utilizadas para a confecção de produtos é conhecido como problemas de fábrica. Os três tipos clássicos de problemas de fábrica são: *Flow Shop*, *Job Shop*, e *Open Shop* (JAIN; MEERAN, 1998).

Ilustrando tais tipos de problemas, pode se citar como *Flow Shop*, uma indústria automotiva, que produz diversos tipos de carros, tendo que executar, para cada modelo, as mesmas operações na mesma ordem, porém com pequenas alterações, o que caracterizada cada modelo. Uma fábrica de confecções de camisetas pode ser classificada como *Job Shop*, onde todos os modelos passam por operações semelhantes em máquinas em comum, mas possuem operações peculiares para cada modelo, como máquinas específicas para fazer detalhes bordados e máquinas próprias de estampar. Como exemplo de *Open Shop* pode-se citar uma fábrica de pré-moldados, onde um móvel é composto de peças que podem ser

produzidas de forma totalmente independentes e em qualquer ordem, sendo unidas somente no final do processo.

Nesses casos os produtos representam as atividades (*jobs*), as operações realizadas na confecção de cada produto são as tarefas e as máquinas são os recursos à serem compartilhados.

Tal problema pode ser entendido como um conjunto de *n jobs* a serem processados em *m* máquinas. O processamento do *job j* pela máquina *i* é representado pela operação O_{ji} e requer uso exclusivo da máquina *i* por um tempo P_{ji}.

2.2.1 Modelagem do Problema de Job Shop

Já foram propostas diferentes forma de se modelar um problema de *Job Shop Scheduling*, dentre elas, pode-se ressaltar a Modelagem por Grafos Disjuntivos e à Modelagem Matemática de Manne (MUTH;THOMPSON, 1963).

2.2.1.1 Modelagem por Grafos Disjuntivos

Uma abordagem gráfica de uma situação tende a facilitar sua compreensão, permitindo a visualização dos relacionamentos entre os componentes em um único contexto.Os problemas de *Job Shop* podem ser representados por um grafo disjuntivo $G = (V, C \cup D)$, em que V, o conjunto dos vértices, representa as operações O_{ji} do *job* j na máquina i, unidas às duas operações especiais fonte (O_0) e sorvedouro (O_*). C, o conjunto dos arcos conjuntivos, retratando a sequência tecnológica das operações de um mesmo produto. D, o conjunto dos arcos disjuntivos, representa os pares de operações (O_{ji}, O_{ki}) que devem ser realizadas na mesma máquina i. O tempo de processamento de cada operação P_{ji} é o peso de cada nó correspondente (MASSA, 2004).

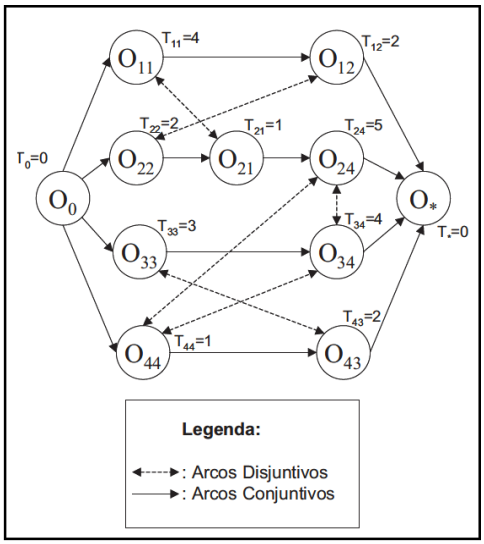

Figura 2 – Problema de *Job Shop* modelado por Grafo Disjuntivo
Fonte: Massa, 2004

Os problemas de *job shop* também podem ser representados como a definição da ordem de todas as operações que devem ser realizadas em uma dada máquina. No modelo de grafo disjuntivo, isto é possível definindo-se a direção de todos os grafos disjuntivos (MASSA, 2004).

Uma seleção é um conjunto de arcos disjuntivos direcionados. E, é dita completa se todas as disjunções forem selecionadas, e será consistente se o grafo direcionado resultante for acíclico.

A Figura 3 ilustra uma seleção completa consistente do grafo exibido na Figura 2.

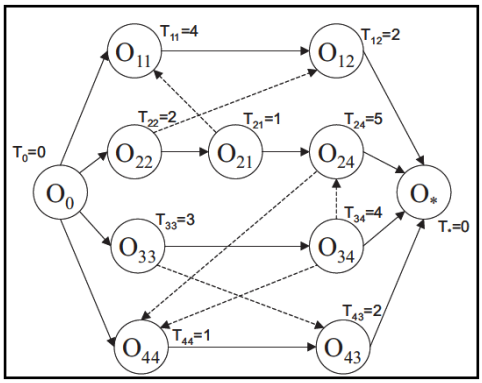

Figura 3 – Seleção completa consistente do grafo da Figura 2
Fonte: Massa, 2004

2.2.1.2 Modelagem Matemática do Job Shop

Outra maneira de se modelar um problema é descrevê-lo através de notações matemáticas e relações de equivalência. Já foram propostas diversas formas de se descrever problemas da classe *job shop*. Dentre elas, tem-se a proposta por Manne, na qual é definida uma variável binária (y_{jki}) para cada par de operações (O_{ji}, O_{ki}) que são processadas na máquina *i* (MUTH; THOMPSON, 1963).

$$y_{jki} = \begin{cases} 0, & se \quad O_{ji} \quad precede \quad O_{ki} \\ 1, & se \quad O_{ki} \quad precede \quad O_{ji} \end{cases} \tag{3}$$

Este modelo supõe o conhecimento da borda superior (*BS*) da duração total do sequenciamento. Tal borda pode ser inicialmente considerada como a soma das durações de todas as operações do problema:

$$BS = \sum_{j=1}^{n} \sum_{i=1}^{m} P_{ji} \tag{4}$$

As restrições podem ser expressas por:

$$I_{u+1,j} \geq I_{uj} + T_{uj}; \forall((u, j),(u+1, j)) \in C_j \tag{5}$$

$$I_{ji} \geq I_{ki} + T_{ki} - BS(y_{jki}); \forall((j,i),(k,i)) \in D_j \tag{6}$$

$$I_{ki} \geq I_{ji} + T_{ji} - BS(1 - y_{jki}); \forall((j,i),(k,i)) \in D_j \tag{7}$$

sendo I_{ji} o tempo de início da operação O_{ji}.

As restrições (5) representam a ordem de processamento de cada *job* (restrições conjuntivas). As restrições (6) e (7) representam as restrições disjuntivas. Nesse modelo para cada par de arcos disjuntivos, somente uma das restrições (6) ou (7) é ativadas. Assim, quando for feito O_{ki} antes de O_{ji} , y_{jki} valerá zero, e a restrição (7) será inativada, já que [$I_{ki} \geq I_{ji} + T_{ji} - BS$], sendo *BS* definido pela equação (4), faria com que a operação O_{ki}

tivesse tempo de início I_{ki}, muito alto, mais especificamente seria no mínimo igual à soma das durações de todas as operações, menos a durações da operação O_{kj}.

Matematicamente ter-se-ia:

$$I_{ki} \geq I_{ji} + T_{ji} - \sum_{j=1}^{n} \sum_{i=1}^{m} P_{ji} \tag{8}$$

Assim a equação (8) torna-se inviável, pois o tempo de inicio de I_{ji} teria que aumentar bastante, influenciando negativamente na minimização do tempo total de produção. Enquanto isso a equação (6) é ativada, já que se torna:

$$I_{ji} \geq I_{ki} + T_{ki} \tag{9}$$

2.2.2 Entropia em Problema de Job Shop

Em termodinâmica, entropia é uma grandeza que mede o grau de irreversibilidade de um sistema físico. Usualmente, é utilizada para caracterizar o grau de desordem de um dado sistema, podendo ser um bom indicador para a distribuição das relações de precedência entre as operações de um *job* (BOLTZMANN, 1995 apud PINEDA, 2006).

Para uma instância com total liberdade entre as precedências das operações de cada *job* tem-se uma entropia associada igual a um $(E = 1)$, o que caracterizaria um ambiente de máquinas do tipo *Open Shop*. Já problemas com entropia igual a zero (E=0), representam casos de *Flow Shop*. Problemas de *Job Shop* pode ser retratados por instâncias com valor de entropia de processamento entre zero ou um $(0 < E < 1)$ (MATTFELD; BIERWIRTH, 1999).

Segundo Mattfeld e Bierwirth (1996), a entropia das instâncias difíceis de *Job Shop* são aproximadamente $1/3$ maiores do que a entropia das instâncias fáceis. As restrições tecnológicas (restrições de ordem de processamento dos *jobs*) dos problemas fáceis permitem um grande número de combinações de ordenamento das operações dos *jobs* em cada máquina, já nas instâncias difíceis, quanto maior o número de restrições, menor será o número de soluções viáveis, portanto mais difícil de encontrá-las.

2.3 SIMULATED ANNEALING

Simulated Annealing (SA), conhecido também por Têmpera ou Recozimento Simulado surgiu no contexto da termodinâmica estatística, desenvolvido por Kirkpatrick et al.

(1983), é um método de busca local que aceita movimentos de piora como forma de escapar de ótimos locais.

Foi baseando-se no procedimento de Metropolis et al. (1953) originalmente proposto como uma estratégia de determinação de estados (configurações) de equilíbrio de uma coleção de átomos a uma dada temperatura que Kirkpatrick et al. (1983) desenvolveram tal procedimento de otimização combinatória.

No método original de Metropolis et al. (1953) escolhe-se o estado inicial de um sistema termodinâmico, energia E e temperatura T, mantendo T constante, perturba-se a configuração inicial e computa-se a diferença de energia dE. Se a variação de energia for negativa $(dE < 0)$, a nova configuração é aceita. Caso, a variação de energia for positiva $(dE > 0)$ a nova configuração será aceita com probabilidade dada pelo fator de Boltzmann $e^{-(dE/KT)}$. Sendo K a constante de Boltzmann, uma constante física que relaciona temperatura e energia de moléculas em um sistema físico, no SI tem-se $K = 1{,}3806503x10^{-23} J/K$. Esse procedimento é repetido diversas vezes para T constante, em seguida a temperatura é reduzida e todo o processo é repetido até que o sistema chegue ao estado mínimo T=0.

Kirkpatrick et al. (1983) perceberam uma semelhança entre o procedimento de recozimento proposto no algoritmo de Metropolis et al. (1953) e os processos de otimização combinatória. A partir da analogia com o recozimento de sólidos seria possível desenvolver um algoritmo genérico de otimização que fosse capaz de escapar de mínimos locais. Portanto, tomaram por base o processo de levar um material a seu estado de equilíbrio máximo, ou seja, de energia mínima, descrito a seguir:

- Um determinado material é inicialmente aquecido a uma alta temperatura, de forma que derreta e seus átomos possam se mover com liberdade.
- A temperatura desta substância derretida é lentamente reduzida de forma que, os átomos possam se mover o suficiente para adotarem uma orientação mais estável.
- Se a substância derretida for resfriada apropriadamente, seus átomos serão capazes de atingir um estado de equilíbrio máximo (energia mínima), produzindo um cristal. Caso contrário, seria produzida uma substância amórfica e imperfeita.

Esse processo de aquecimento seguido de um resfriamento lento denominou-se de recozimento (*annealing*).

2.3.1 Princípios Básicos de Termodinâmica Estatística

Em termodinâmica estatística, grandes sistemas a uma dada temperatura tendem a se aproximar espontaneamente de um estado de equilíbrio, caracterizado por um valor médio de energia que é função da temperatura. A simulação da transição para um estado de equilíbrio e a redução da temperatura do sistema permitem a determinação de valores cada vez menores para a energia média do sistema (ERNÝ, 1985 apud ZUBEN e CASTRO, 2002).

Sendo *x*, a configuração recorrente do sistema, *E(x)* a energia de *x*, e *T* a temperatura. Em equilíbrio, o sistema muda aleatoriamente seu estado de uma configuração a outra de forma que a probabilidade de se encontrar o sistema em uma configuração especifica é dada pela distribuição de Boltzmann-Gibbs:

$$P(x) = K.\exp(-E(x)/KT) \tag{10}$$

Assumindo um sistema com quantidade discreta de possíveis configurações, o cálculo da energia média do sistema torna-se complicado. Entretanto, utilizando simulação de Monte Carlo, como proposto por Metropolis et al. (1953) para as variações aleatórias de estado de uma configuração a outra do sistema, de maneira que, em equilíbrio a equação (10) seja satisfeita (ZUBEN e CASTRO, 2002).

Tabela 2 – Interpretação da terminologia do sistema físico para o domínio computacional

Termodinâmica	Algoritmo Simulated Annealing
Estado (configuração)	Solução (factível) do problema. Ponto no espaço de busca
Energia	Valor retornado pela função de avaliação
Estado de Equilíbrio	Ótimo (local)
Estado de Equilíbrio máximo	Ótimo global
Temperatura	Parâmetro de controle
Recozimento	Busca através da redução de T
Distribuição de Boltzmann-Gibbs	Probabilidade de seleção de um novo ponto

Fonte: ZUBEN e CASTRO (2002)

2.3.2 Algoritmo Simulated Annealing

Supondo o problema genérico de minimizar uma função:

- Inicializar (aleatoriamente) uma solução *S* na região factível do problema;

- Medir a energia de *S*, *E(S)*;
- A cada iteração, um novo ponto *S'* é encontrado aplicando-se uma pequena perturbação no ponto recorrente, selecionando dessa forma um ponto *S'* que esteja na vizinhança de *S*;
- Medir a energia de *S'*, *E(S')*;
- Verificar a variação de energia do sistema: $\Delta E = E(S') - E(S)$;
- Se $\Delta E \leq 0$, então *S'* torna-se a solução recorrente $(S \leftarrow S')$;
- Senão, a probabilidade de *S'* ser aceita como solução recorrente é dado por um caso particular da distribuição de Boltzmann-Gibbs:

  $$P(\Delta E) = \exp(-\Delta E / T) \quad (11)$$

- O parâmetro *T* é inicializado com um valor elevado e é lentamente reduzido durante o processo de busca:
  - geralmente, implementa-se um decaimento geométrico para *T*; $T \leftarrow \alpha.T$; sendo $0 < \alpha < 1$
- O método é finalizado quando pelo menos um dos critérios for satisfeito:
  - nenhuma melhoria significativa for alcançada;
  - atingir o número de iterações pré-definido;
  - atingir o tempo de processamento estipulado;
  - temperatura T receber seu valor mínimo.

A sequência de temperaturas e o número de rearranjos dos parâmetros até a chegada ao estado de equilíbrio a cada temperatura é denominado sequenciamento de recozimento (*annealing schedule*) (KIRKPATRICK et al., 1983 apud ZUBEN e CASTRO, 2002).

3. DESENVOLVIMENTO

3.1 DESCRIÇÃO DO PROTOCOLO DE PESQUISA

O escopo desse trabalho compreende problemas estáticos e genéricos de *Job Shop Scheduling*, como definido na seção 1.3. As instâncias (Anexo 1) utilizadas tanto na validação como na análise de desempenho do algoritmo ao se empregar cada uma das duas formas de codificação da solução foram geradas pela função *gerar_instancias.m*, que será explanada na sequência.

3.1.1 Geração de Instâncias

Uma vez que o trabalho em questão trata de problemas de *Job Shop Scheduling* genéricos, fez-se necessário o desenvolvimento de um mecanismo que fosse capaz de gerar os dados exigidos a composição do problema.

Como um dos objetivos era analisar o desempenho do algoritmo empregando as duas formas de se codificar as soluções, diante de problemas com entropia, tamanho e complexidade diferentes, exigia-se que tal mecanismo fosse flexível suficiente para possibilitar ao usuário construir problemas com as características de seu interesse.

Para tanto, foi elaborada a função em *matlab gerar_instancias.m* disponível no Anexo 2, os argumentos de entrada da função podem ser definidos pelo usuário no arquivo principal SA_JS (Anexo 3), são eles:

- número de *job*s presentes na instância;
- número de máquinas do problema;
- entropia dos tempos de processamento;
- entropia da ordem de processamento;
- tempo mínimo de processamento por unidade de *job* em cada máquina;
- tempo máximo de processamento por unidade de *job* em cada máquina;
- tamanho médio do lote de produção;
- amplitude de variação do tamanho do lote de produção;
- variação mínima permitida do tamanho do lote;
- *release date* mínimo;
- *release date* máximo;

- acréscimo mínimo no tempo de processamento para definição do *due date*;
- acréscimo máximo no tempo de processamento para definição do *due date*.

Como saídas da função têm-se:

- tempo de processamento dos *jobs* em cada máquina;
- ordens de processamento;
- *release date* de cada *job*;
- *due date* de cada *job*;
- dados de entrada *software* LiSA.

A função pode ser subdividida em quatro etapas: na primeira, são gerados os tempo de processamento; na segunda, a ordem das máquinas em que cada *job* deve ser processado, na terceira, *release date* e *due date* e por fim os dados são dispostos no formato de entrada do *software* LiSA.

O tempo de processamento dos *jobs* em cada máquina é disposto em uma matriz, na qual as linhas referem-se aos *jobs* e as colunas, às máquinas, conforme Figura 4.

$$
\begin{array}{cc}
 & \text{Máquinas} \\
\text{Jobs} & \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ j \end{array}
\begin{array}{c}
\begin{array}{cccc} 1 & 2 & \dots & m \end{array} \\
\begin{bmatrix}
p_{11} & p_{12} & \cdots & p_{1m} \\
p_{21} & p_{22} & \cdots & p_{2m} \\
\vdots & \vdots & \vdots & \vdots \\
p_{j1} & p_{j2} & \cdots & p_{jm}
\end{bmatrix}
\end{array}
\end{array}
$$

Figura 4 – Matriz de tempo de processamento

Esses tempos são gerados aleatoriamente dentro dos limites estipulados pelo usuário a partir da definição dos tempos mínimo e máximo de processamento por unidade de *job* em cada máquina. Na sequência, tais tempos são multiplicados pelo tamanho do lote de cada *job*, que são gerados seguindo os seguintes passos:

1. Calcular o número máximo de variações no tamanho do lote, que é definido pela razão entre amplitude de variação e variação mínima permitida;
2. Sortear um valor:
    - se, for menor que a entropia de processamento:

- sortear, dentro do intervalo de 1 à número máximo de variação no tamanho do lote, o número de variações no lote do job_i;
- calcular a variação total no lote do job_j definida pela multiplicação entre o número de variações e variação mínima permitida;
- determinar se a variação total será acrescida ou subtraída no lote médio;
- calcular tamanho do lote do job_i;

▪ caso contrário:

- lote do job_i será igual ao lote médio.

3. O procedimento descrito no item 2 deve ser repetido para todos os *jobs*.

A Figura 5 ilustra um exemplo dos parâmetros utilizados na definição do tamanho dos lotes.

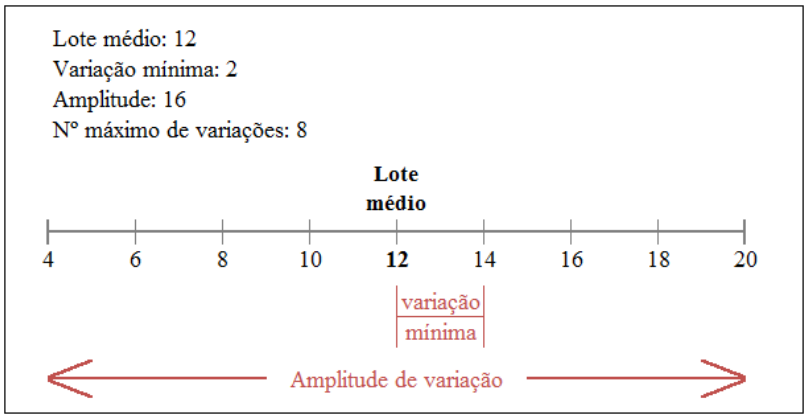

Figura 5 – Exemplo de parâmetros usados na definição do tamanho de lote

A ordem de processamento de cada *job* é armazenada em uma matriz semelhante à ilustrada pela Figura 6, cada linha é reservada para um *job*, as colunas representam as operações, e no interior da matriz têm-se as máquinas.

$$\begin{array}{cc} & \text{Operações} \\ & \begin{array}{cccc} 1 & 2 & \dots & m \end{array} \\ \text{Jobs} \begin{array}{c} 1 \\ 2 \\ \vdots \\ j \end{array} & \begin{bmatrix} M_{11} & M_{12} & \dots & M_{1m} \\ M_{21} & M_{22} & \dots & M_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ M_{j1} & M_{j2} & \dots & M_{jm} \end{bmatrix} \end{array}$$

Figura 6 – Matriz de ordem de processamento

Para definir o caminho das máquinas que cada *job* deverá passar, primeiramente é gerada uma matriz, em que todos os *jobs* seguem a mesma ordem, começando pela máquina 1 e terminando na máquina *m*, sendo *m* o número de máquinas presentes no problema.

Posteriormente, a posição que cada máquina ocupa na ordem de processamento dos *job* poderá ser alterada, conforme procedimento abaixo:

1. Sortear um número:
    - se, for menor que a entropia da ordem de processamento:
        - sortear uma máquina para assumir a primeira posição na ordem de processamento.
    - caso contrário:
        - na primeira posição continua a máquina 1.
2. O procedimento acima é repetido para todas as posições de ordem de processamento de todos os *jobs*.

As *release dates* são armazenados em um vetor de comprimento igual ao número de *jobs* da instância. Tais valores são gerados aleatoriamente, dentro dos limites estabelecidos pelo usuário, a partir da definição dos valores mínimo e máximo de *release date*.

Na geração dos *due dates*, inicialmente para cada um dos *jobs* soma-se o tempo de processamento por máquina com seu *release date* e posteriormente, sorteia-se um acréscimo, dentro dos limites informados pelo usuário, para esse tempo.

A Figura 7 exemplifica duas instâncias geradas a partir dos parâmetros previamente estabelecidos, presentes na mesma figura. As entropias da instância I iguais à zero passaram a valer um na instância II e os demais parâmetros permaneceram inalterados, ilustrando assim, a influência das entropias na geração das instâncias.

Pode-se observar que na instância I todos os *jobs* possuem a mesma ordem de processamento, começando pela máquina 1 e terminando na 4. O tempo de duração das tarefas em cada máquina também são iguais para todos.

Quando as duas entropias forem iguais a um, caso da instância II, nenhum dos *jobs* terá ordem de processamento igual a sequência 1,2,3,4, nem os tempos de processamento de diferentes *jobs* em uma mesma máquina serão iguais.

Inputs:

Instância:	I	II
Número de jobs:	3	3
Número de máquinas:	4	4
Entropia de tempos de processamento:	**0**	**1**
Entropia da ordem de processamento:	**0**	**1**
Tempo de processamento mínimo:	0	0
Tempo de processamento máximo:	7	7
Lote médio:	10	10
Amplitude de variação:	10	10
Variação mínima:	2	2
Release date mínimo:	0	0
Release date máximo:	15	15
Acréscimo mínimo:	0	0
Acréscimo máximo:	50	50

Outputs:

Tempos de Processamento I

$$\begin{bmatrix} 10 & 40 & 40 & 70 \\ 10 & 40 & 40 & 70 \\ 10 & 40 & 40 & 70 \end{bmatrix}$$

Tempos de Processamento II

$$\begin{bmatrix} 8 & 32 & 32 & 56 \\ 26 & 104 & 104 & 182 \\ 2 & 8 & 8 & 14 \end{bmatrix}$$

Ordem de Processamento I

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

Ordem de Processamento II

$$\begin{bmatrix} 4 & 3 & 2 & 1 \\ 2 & 1 & 4 & 3 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

Release date I

$$\begin{bmatrix} 11 & 9 & 5 \end{bmatrix}$$

Release date II

$$\begin{bmatrix} 11 & 13 & 14 \end{bmatrix}$$

Due date I

$$\begin{bmatrix} 235 & 242 & 243 \end{bmatrix}$$

Due date II

$$\begin{bmatrix} 142 & 451 & 54 \end{bmatrix}$$

Figura 7 – Exemplos das saídas da função *gerar_instancias.m*

3.2 ELABORAÇÃO DO ALGORITMO

Essa parte do trabalho busca esclarecer como o algoritmo *Simulated Annealing* proposto por Kirkpatrick et al. (1983), descrito na seção 2.3, foi adaptado ao problema de *Job Shop Scheduling*, na Figura 8 pode ser visto uma esquematização dessa adaptação e no Anexo 3 encontra-se o algoritmo na integra.

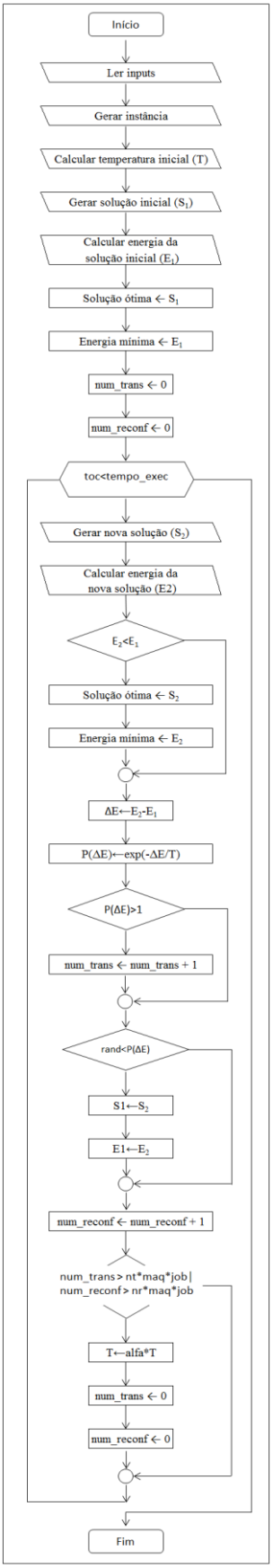

Figura 8 – Esquema do algoritmo *Simulated Annealing* adaptado a problemas de *Job Shop*

Como ilustrado no esquema da Figura 8, para adequar o *Simulated Annealing* à resolução do *Job Shop* foi preciso desenvolver uma representação computacional para as soluções (*scheduling*), e mecanismos aptos a medir a função objetivo, gerar novas soluções, e estimar a temperatura inicial. Além de configurar os parâmetros básicos do algoritmo, como critérios de parada e de decaimento da temperatura.

Nas seções seguintes serão descritas: as duas formas de se codificar as soluções; as funções desenvolvidas para: calcular a função a objetivo (*calc_energia_1.m* e *calc_energia_2.m*), estimar a temperatura inicial (*estima_T0_1.m* e *estima_T0_2.m),* e gerar novas soluções (*gera_sol_1.m*, *gera_sol_11.m* e *gera_sol_2.m*); e os parâmetros básicos do algoritmo.

3.2.1 Codificação de soluções

Segundo Pinedo (2002), *scheduling* pode ser definido como a alocação de recursos a tarefas por um período de tempo determinado. Portanto, soluções para esse tipo de problema devem representar no tempo a sequência de tarefas a serem processadas por cada um dos recursos presentes.

No caso de problemas de *Job Shop*, as restrições tecnológicas de cada *job*, sequência de máquinas a serem percorridas a fim de sua conclusão devem ser respeitas. Logo, as tarefas representam as operações dos *job* em cada máquina.

No presente trabalho foram elaboradas duas formas de se codificar as soluções. A *codificação I* nunca corresponderá a uma solução inviável, porém sua limitação está em não ser capaz de representar unicamente um *scheduling*, ou seja, um mesmo sequenciamento pode ser codificado por diferentes soluções. Já na *codificação II*, isso nunca acontecerá, mas poderá ser gerado um sequenciamento inviável, correspondente a uma solução que não respeite as restrições tecnológicas de pelo menos um dos *jobs*.

3.2.1.1 Codificação I

Cada solução corresponde a um vetor de comprimento igual à soma do número de operações que cada *job* realiza, ou seja, a identificação de cada *job* aparecerá no vetor solução quantas vezes for seu número de operações.

À medida que a identificação de um determinado *job* aparece na solução, esse será sequenciado no recurso referente à operação correspondente ao número de vezes que ele estiver sendo chamado no vetor. Por exemplo, a segunda vez que o job_1 aparece na solução refere-se à segunda operação que esse deve realizar.

A Figura 9 exemplifica uma possível solução para um problema 3x3. O primeiro elemento da solução identifica a primeira operação que o *job* 3 deve realizar. Observando na matriz de Ordem de Processamento o elemento da terceira linha, primeira coluna corresponde à máquina 2. Buscando na matriz Tempo de Processamento, o tempo gasto pelo *job* 3 na máquina 2, encontra-se zero, logo o *job* 3 não precisa passar pela máquina 2. Observando novamente as duas matrizes, pode-se perceber que a primeira operação do *job* 3 é realizada pela máquina 1 e tem duração de 21 unidades de tempo. Porém, antes de alocar a máquina 1 para realizar essa operação deve ser considerado o *release date* do *job* 3, que no caso é igual a zero.

O segundo elemento do vetor solução representa que a máquina 2 deverá ser alocada para realizar a primeira operação o *job* 2 a partir do instante 13. O terceiro elemento, que a máquina 3 deverá ser alocada a partir do instante igual a 14 para processar a primeira operação do *job* 1. O quarto elemento diz que a segunda operação do *job* 2 será na máquina 1 e que essa deverá ser inicializada no primeiro instante em que ocorrer a conclusão da primeira operação do *job* 2 e a máquina 1 estiver desocupada. E assim sucessivamente até que se todos os elementos sejam lidos.

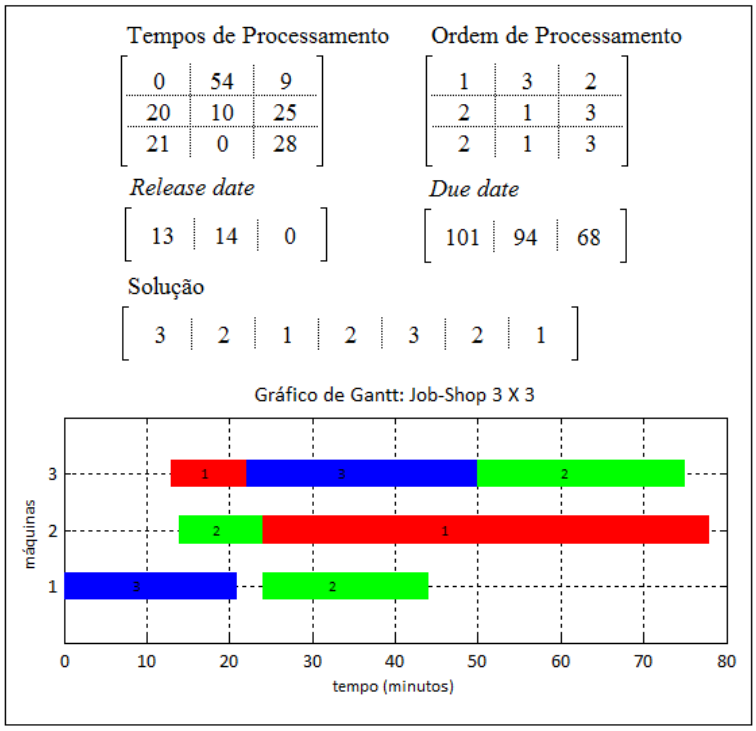

Figura 9 – Exemplo de solução para um problema 3x3 representada pela codificação I

Como citado anteriormente, a limitação dessa forma de codificar soluções se encontra no fato de vários vetores solução corresponderem a um mesmo sequenciamento.

Essa falta de unicidade ocorre, devido a cada elemento representar uma dada operação de um determinado *job*. Com isso, alterações nas posições desses elementos, o que caracteriza uma nova solução, caso tais elementos forem diferentes, não necessariamente corresponderá às operações realizadas em uma mesma máquina, logo não irá gerar um sequenciamento diferente.

Na Figura 10 encontram-se quatro diferentes soluções gerando um mesmo sequenciamento. Invertendo o quinto com o sexto elemento da solução I tem-se a solução II, porém um novo sequenciamento não é gerado, pois o quinto elemento (1) representa a terceira operação do *job* 1 que deve ocorrer na máquina 2 enquanto que o sexto elemento (2) identifica a segunda operação do *job* 2 que será na máquina 3, não impactando na sequencia de *jobs* de nenhuma das máquinas.

Na solução II o quarto elemento, primeira operação do *job* 3 à ser realizada pela máquina 1, foi alterado com o sexto, terceira operação do *job* 1 processada pela máquina 3 formando a solução III, que também não representa um novo sequenciamento.

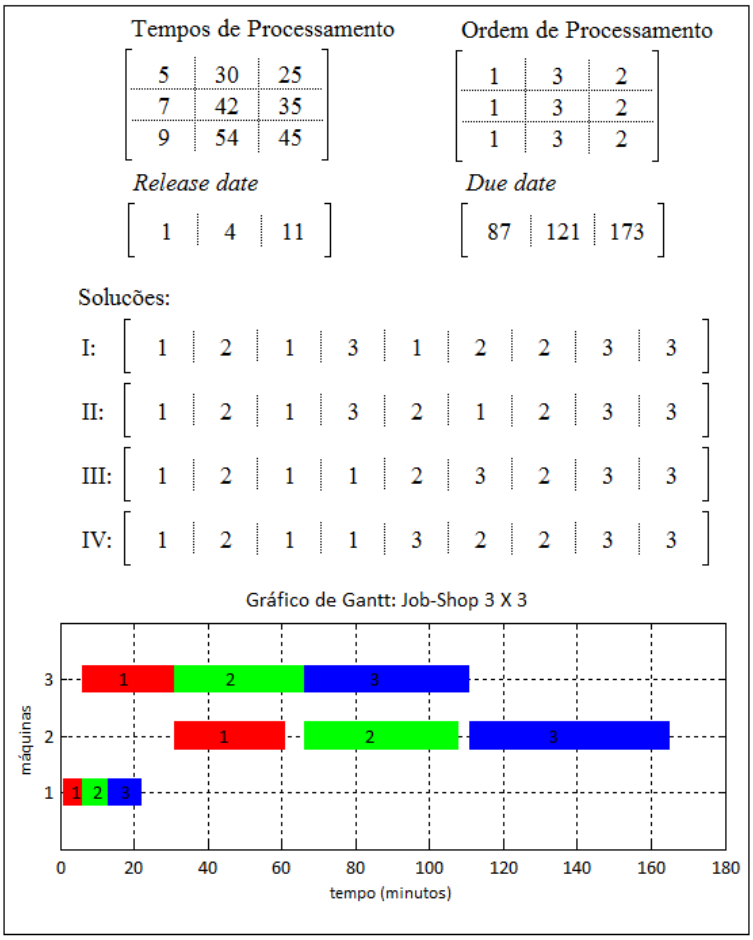

Figura 10 – Exemplo de falta de unicidade da codificação I

3.2.1.2 Codificação II

A outra forma proposta para se codificar uma solução consiste em uma matriz (Figura11), em que cada coluna é reservada para uma máquina, e as linhas representam a ordem que o *job* que estiver contido nela será processado pela máquina correspondente.

$$
\begin{array}{cc}
 & \text{Máquinas} \\
 & \begin{array}{cccc} 1 & 2 & \dots & m \end{array} \\
\text{Ordem} \begin{array}{c} 1 \\ 2 \\ \vdots \\ j \end{array} & \begin{bmatrix} J_{11} & J_{12} & \dots & J_{1m} \\ J_{21} & J_{22} & \dots & J_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ J_{j1} & J_{j2} & \dots & J_{jm} \end{bmatrix}
\end{array}
$$

Figura 11 – Matriz representativa da codificação II

A Figura 12 ilustra um exemplo de solução para um problema 3x3, como pode ser visto no gráfico de Gantt a ordem em que os *jobs* aparecem em cada máquina corresponde a sequência de *jobs* em cada coluna da matriz solução.

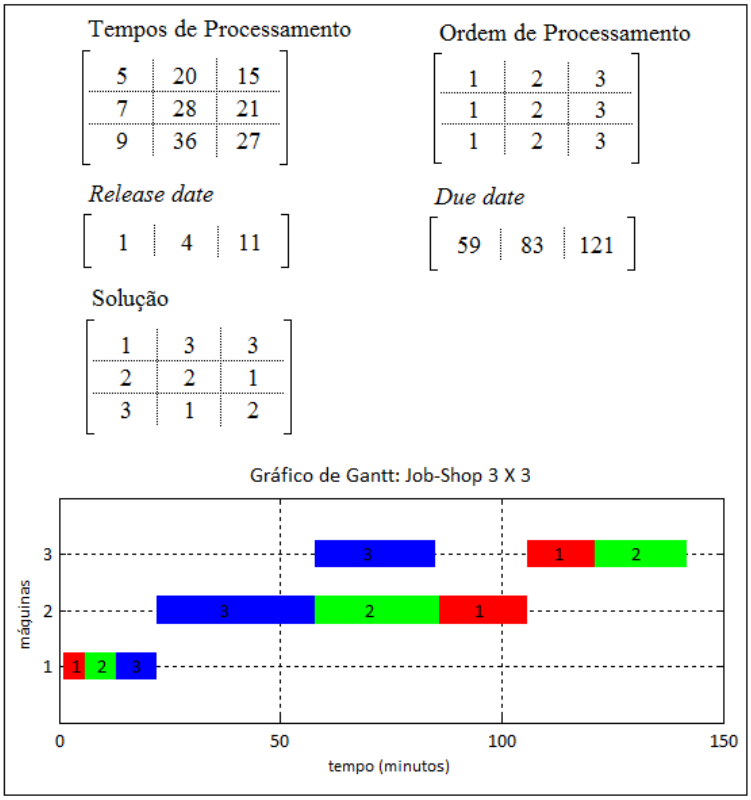

Figura 12 – Exemplo de solução para problema 3x3 representada pela codificação II

Conforme citado anteriormente, essa maneira de codificar as soluções também possui uma limitação. Diferentemente da codificação I, um sequenciamento nunca será representado por diferentes soluções. Porém, pode ser que seja gerada uma solução que não respeite as precedências de processamento de pelo menos um dos *job*, ou seja, uma solução inviável.

Na Figura 13 tem-se um exemplo de solução inviável. Observando o gráfico de Gantt pode-se perceber que a ordem em que os *jobs* aparecem nas máquinas corresponde o que está representado na solução, mas ao considerar a matriz Ordem de Processamento percebe-se que as precedências do *job* 2, como sinalizado na figura, não foram respeitadas.

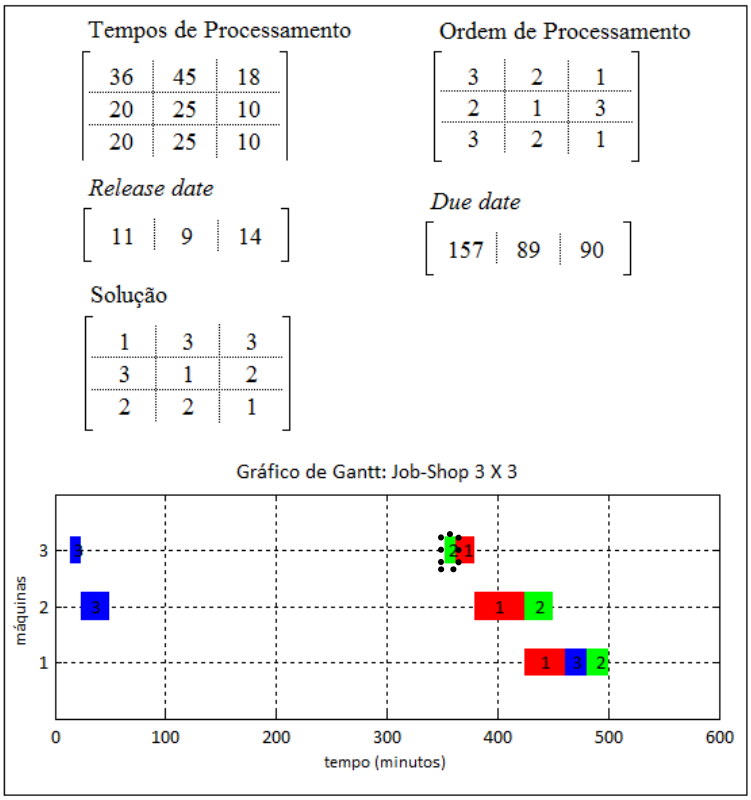

Figura 13 – Exemplo de solução inviável

3.2.2 Geração de novas soluções

Uma das exigências de cada iteração de um algoritmo *Simulated Annealing* consiste na geração de uma nova solução. A energia dessa nova solução (E_2) é comparada à energia da solução recorrente (E_1). Em problemas de minimização, se E_2 for menor que E_1 a solução nova se torna a recorrente, caso contrário, poderá se tornar recorrente com probabilidade dada pela equação11.

Como se tem duas formas de se codificar as soluções, também é necessário se ter dois procedimentos para gerá-las. As funções desenvolvidas com esse intuito serão descritas nas subseções seguintes.

3.2.2.1 Codificação I

Para gerar uma nova solução na forma de *codificação I*, descrita na seção 3.2.1.1, é necessário apenas alterar as posições de diferentes elementos. E isso pode ser feito de várias maneiras. Foram implementadas duas função com esse propósito, uma pseudoaleatória e outra direcionada por probabilidades de priorização dos *job* de acordo com seu atraso.

- Pseudoaleatória

O procedimento para reescrever o vetor solução de maneira aleatória a partir de um vetor já existente está implementado na função *gera_sol_1.m* disponível no Anexo 4 e esquematizado na Figura 14. Mesmo quando os elementos do vetor que mudarão de posição são escolhidos de forma não tendenciosa, faz-se necessário definir como serão feitas tais alterações de posição.

O mecanismo implementado foi:

- Escolher duas posições do vetor solução:
  - Sortear uma das formas a seguir para embaralhar o vetor:
    1) Inverter o subvetor limitado por essas duas posições;
    2) Inverter os três subvetores formados;
    3) Recortar o subvetor limitado por essas duas posições, sortear uma terceira posição e encaixá-lo.

Para que tais formas tenham aproximadamente a mesma probabilidade de serem escolhidas é sorteado um valor entre 0 e 1, se esse valor estiver entre 0 e 0.33, o vetor solução será reescrito pela forma 1, caso tal valor esteja entre 0.33 e 0.66 será escolhida a forma 2 e por fim, se estiver contido no intervalo entre 0.66 e 1, utilizará a forma 3.

A Figura 15 mostra possíveis saídas (nova solução) da função *gera_sol_1.m.* Para que as três maneiras fossem exemplificadas, os valores das posições de corte do vetor foram mantidos inalterados e variou-se o valor de x.

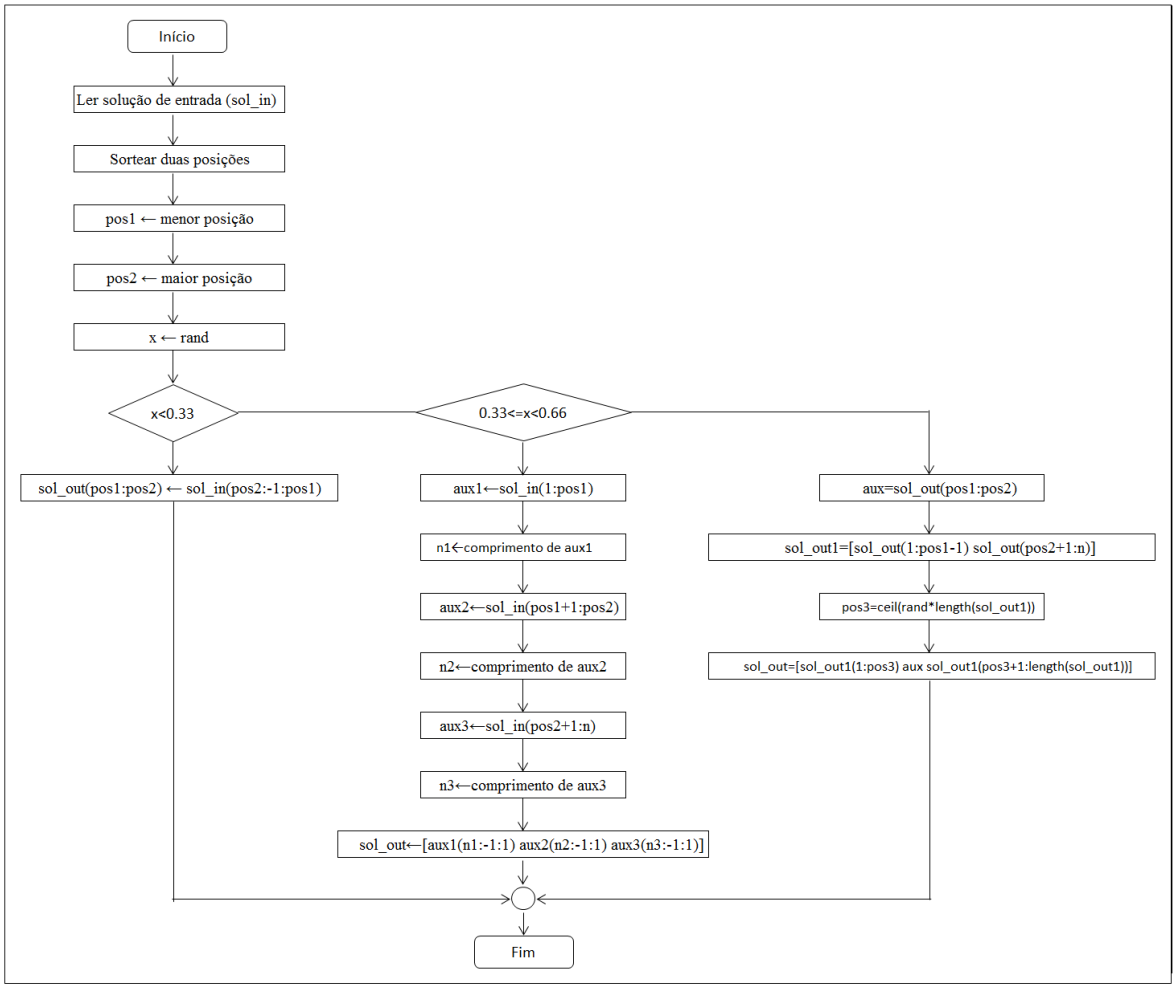

Figura 14 – Esquematização da função *gera_sol_1.m*

pos1:	3
pos2:	9
x I:	0.1254
x II:	0.4562
x III:	0.9573

sol_in: [1 1 1 2 2 2 2 3 3 3 3]

sol_out I: [1 1 3 3 2 2 2 2 1 3 3]

sol_out II: [1 1 3 3 1 2 2 2 2 3 3]

sol_out III: [1 1 3 3 1 2 2 2 2 3 3]

Figura 15 – Exemplo saídas *gera_sol_1.m*

- <u>Direcionada por probabilidades</u>

Esse procedimento para geração de soluções consiste em construir um novo vetor antecipando o agendamento das operações dos *jobs* que tiverem os maiores atrasos. Para isso, *jobs* atrasados $(C_j > d_j)$ recebem probabilidade de ter sua operação antecipada diretamente proporcional a seu atraso enquanto os *jobs* adiantados $(C_j < d_j)$, inversamente proporcional ao seu adiantamento.

Tais atrasos e probabilidades de antecipação são calculados de maneira dinâmica, ou seja, à medida que a posição de um elemento no vetor é alterada, um sequenciamento é gerado e a partir dele, medem-se os atrasos de cada *job* e as probabilidades. A Figura 16 ilustra o esquema desse procedimento, que está implementado na função *gerar_sol_11.m* (Anexo 5).

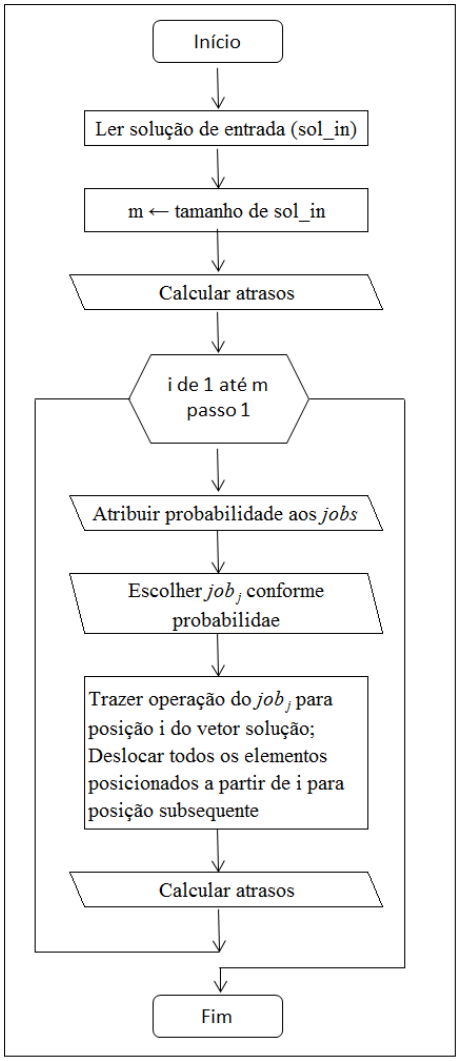

Figura 16 – Esquematização da função *gera_sol_11.m*

3.2.2.2 Codificação II

A geração de uma nova solução na forma de *codificação II*, apresentada na seção 3.2.1.2, é definida pela alteração da posição de pelo menos dois *jobs* em uma das máquinas.

Tal procedimento está implementada na função *gerar_sol_2.m* disponível no Anexo 6 e esquematizada na Figura 17.

Como pode ser visto, no esquema abaixo, os dois procedimentos para modificar a solução de entrada gerando uma nova, são:

1. Permutar algumas linhas da matriz de solução inicial, o que significa alterar a ordem de *jobs* em todas as máquinas
2. Permutar as linhas de algumas das colunas da matriz de entrada, o que corresponde alterar, de forma independente, a ordem de *jobs* em apenas algumas máquinas.

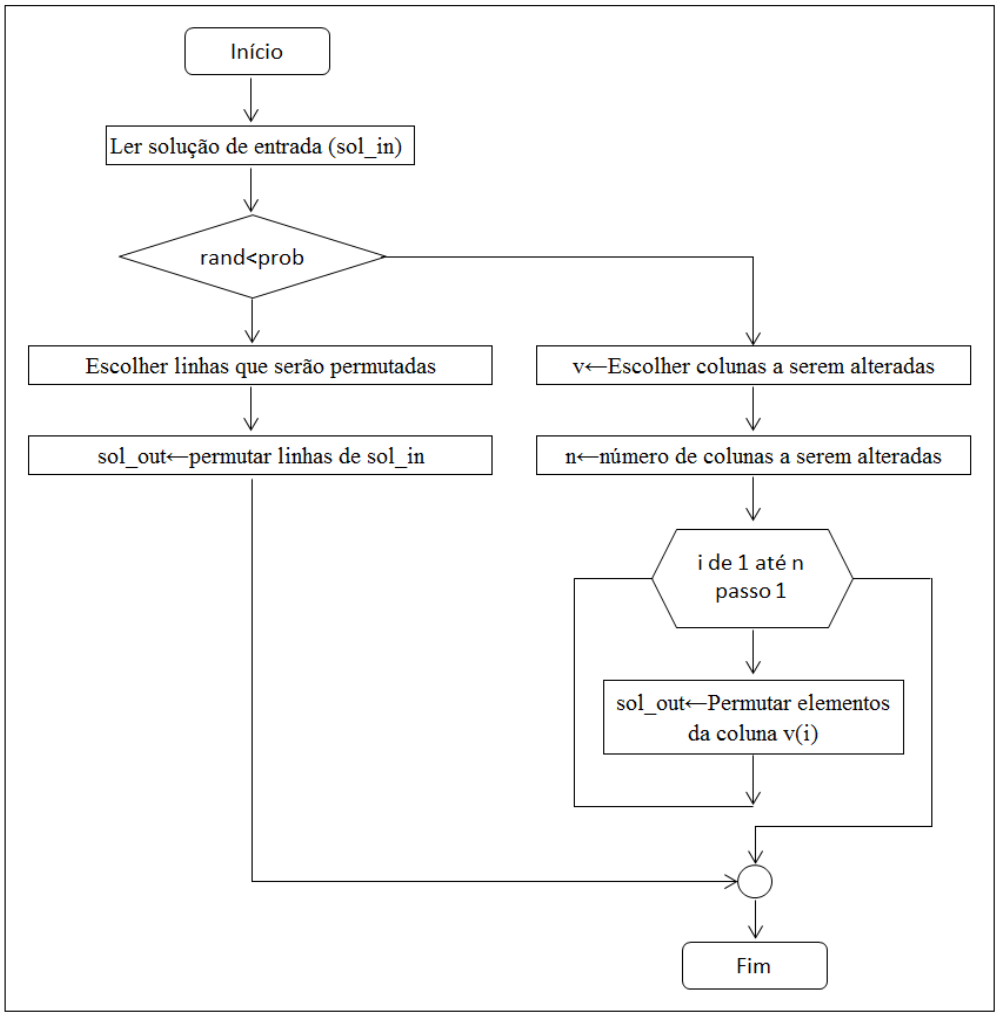

Figura 17 – Esquematização da função *gerar_sol_2.m*

A chance de usar um ou outro procedimento para gerar a nova solução pode impactar no valor da função objetivo encontrado dependendo da instância a ser resolvida, portanto deve ser calibrado pelo usuário todas as vezes que for resolver um problema.

A partir da resolução da dada instância 3.2.2.2-1, disponível no Anexo 1, fica claro tal interferência. Variando a probabilidade de se usar o procedimento representado mais a esquerda no esquema da Figura 17, os diferentes valores para a função objetivo foram compilados na tabela 3, nesse caso seria assumido prob=0.8.

Tabela 3 – Exemplo de sensibilidade da função objetivo

prob	Função objetivo
0.1	1162
0.3	1128
0.5	1112
0.6	1138
0.7	1136
0.8	1084
0.9	1108

A Figura 18 ilustra uma possível saída para cada um dos mecanismos.

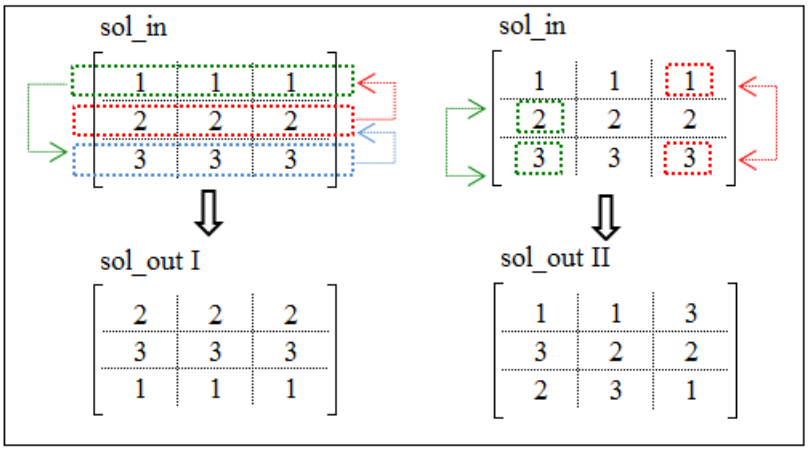

Figura 18 – Exemplo saídas *gera_sol_2.m*

3.2.3 Cálculo da função objetivo

Dado uma solução, seja ela representada pela codificação I ou II, descritas nas seções 3.2.1.1 e 3.2.1.2 respectivamente, é necessário que se tenha um procedimento que transforme tal solução no sequenciamento das operações dos *jobs* nas máquinas, ou seja, que consiga ler a codificação da solução e respaldar a construção do gráfico de Gantt.

A partir da representação matricial do gráfico de Gantt criam-se as condições necessárias ao cálculo da função objetivo (minimizar o atraso máximo) e de outras funções de desempenho (*makespan*, atraso total, número de *jobs* atrasados) que também poderiam ter sido usadas como função objetivo, porém nesse trabalho são meramente informativas.

Uma vez que se têm duas formas de codificar as soluções, foi necessário desenvolver duas funções para transformar tais codificações em sequenciamento, apresentadas nas subseções seguintes.

3.2.3.1 Codificação I

Na *codificação I* cada elemento do vetor representa uma operação de um determinado *job* em uma máquina. Para se chegar ao valor da função objetivo é preciso determinar o instante inicial e final de cada uma dessas operações.

O instante inicial é encontrado quando duas condições são satisfeitas concomitantemente: a operação predecessora já foi concluída e a máquina, na qual tal operação será executada está desocupada.

O instante final é facilmente encontrado pela soma do instante inicial e do tempo de processamento da operação. Com o tempo de conclusão de todas as operações chega-se na conclusão do *job*.

Para medir a função objetivo, primeiro é calculado o atraso de cada *job*, definido pela equação 1, sendo C_j o tempo de conclusão do *job*$_j$ e d_j o *due date*. O atraso máximo será o maior dentre tais atrasos.

As demais funções de desempenho são definidas como:

- *Makespan* corresponde ao maior tempo de complementação (C_j);
- Atraso total é a soma dos atrasos $\left(\sum L_j\right)$;
- Número de *jobs* atrasados $\left(\sum U_j\right)$.

Tais procedimentos estão implementadas na função *calc_energia_1.m* disponível na integra no Anexo 7.

3.2.3.2 Codificação II

Na *codificação II*, descrita na seção 3.2.1.2, o sequenciamento é representado na forma de uma matriz, em que cada coluna é reservada a uma máquina, as linhas representam a ordem de processamento e no interior da matriz ficam armazenados os *jobs*.

Uma vez que já se tem exatamente a ordem de *jobs* a serem processados pelas máquinas, para se chegar ao *scheduling* é necessário fazer cumprir as restrições tecnológicas dos *jobs*, ou seja, respeitar a ordem de processamento de cada um. A função *calc_energia_2.m* (Anexo 8), foi implementada com tal intuito.

Essa função pode ser dividida em três partes, nas quais são realizados os seguintes procedimentos:

1. A primeira operação de cada *job* é inicializada no instante permitido de entrada de cada um (*release date*) e as demais operações são inicializadas no instante de conclusão de sua predecessora;
2. As operações são ressequenciadas, sempre prosseguindo no tempo, máquina a máquina conforme ordem de *jobs* definidas na solução;
3. Começando pelo primeiro *job* (j) agendado na máquina 1 e terminando no último *job* da máquina *m*, sendo *m* o número de máquinas do problema
    - reagendar operação a operação conforme ordem de processamento;
    - reagendar em cada máquina as operações dos demais *jobs*, que estiverem ordenadas depois das operações do job_j.

A partir da execução da primeira parte da função, esquematizada na Figura 19, tem-se o sequenciamento do problema relaxado, as operações foram agendadas sem considerar restrição de capacidade dos recursos, ou seja, o fato de cada máquina só processar um *job* de cada vez.

O procedimento 2 (Figura 20) foi definido com o propósito de atender à restrição de capacidade das máquinas e garantir a sequência de *jobs* por máquina definidas na solução. Porém quando se ajusta cada máquina de forma independente, perde-se a sequência de processamento dos *jobs*.

A terceira parte da função, representada na Figura 21, foi elaborada com o objetivo de reestabelecer a ordem de processamento dos *jobs* e ao mesmo tempo não deixar que se perca nem as restrições de capacidade das máquinas nem a ordem de *jobs* definida na solução.

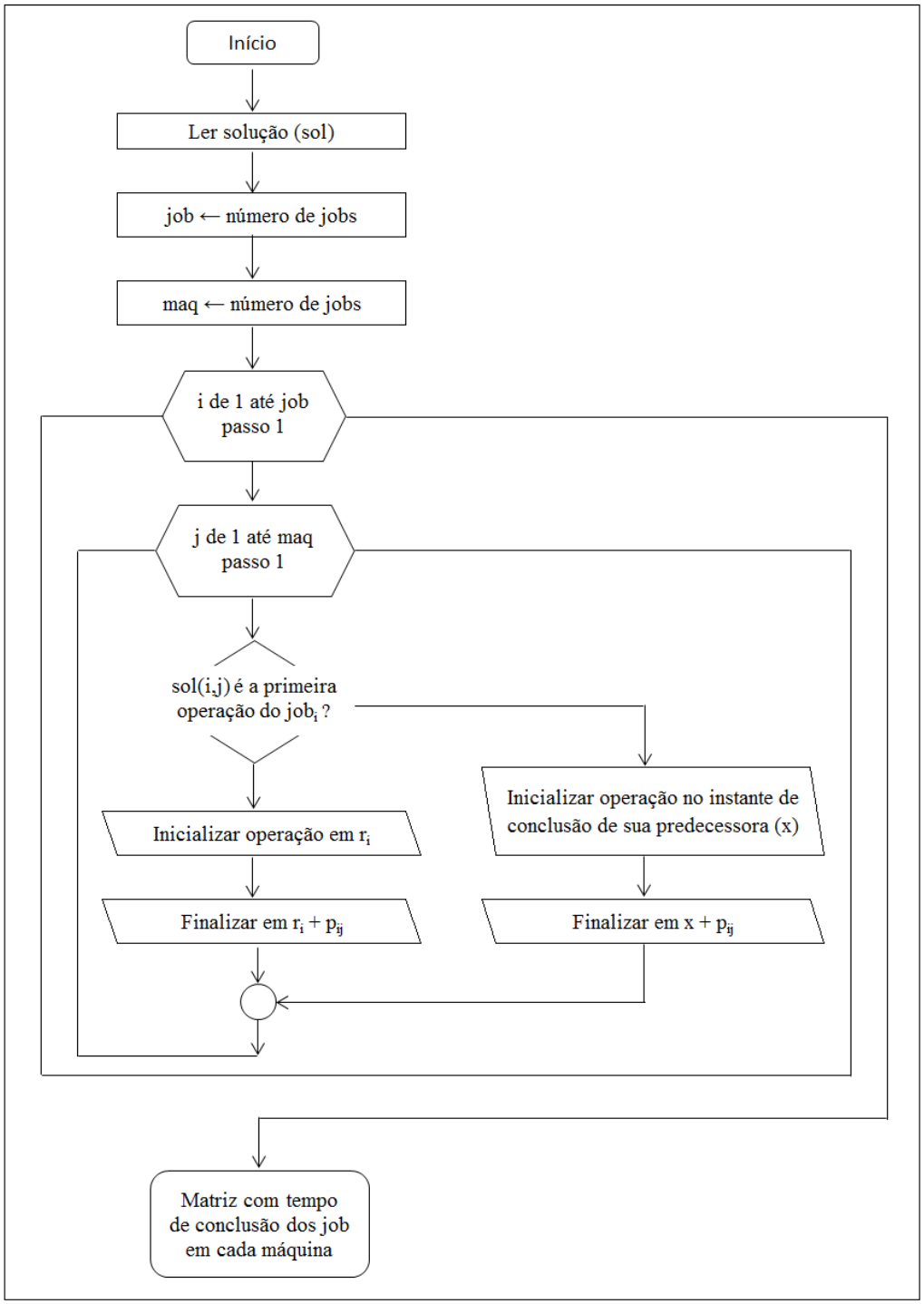

Figura 19 – Esquema da primeira parte da função *calc_energia_2.m*

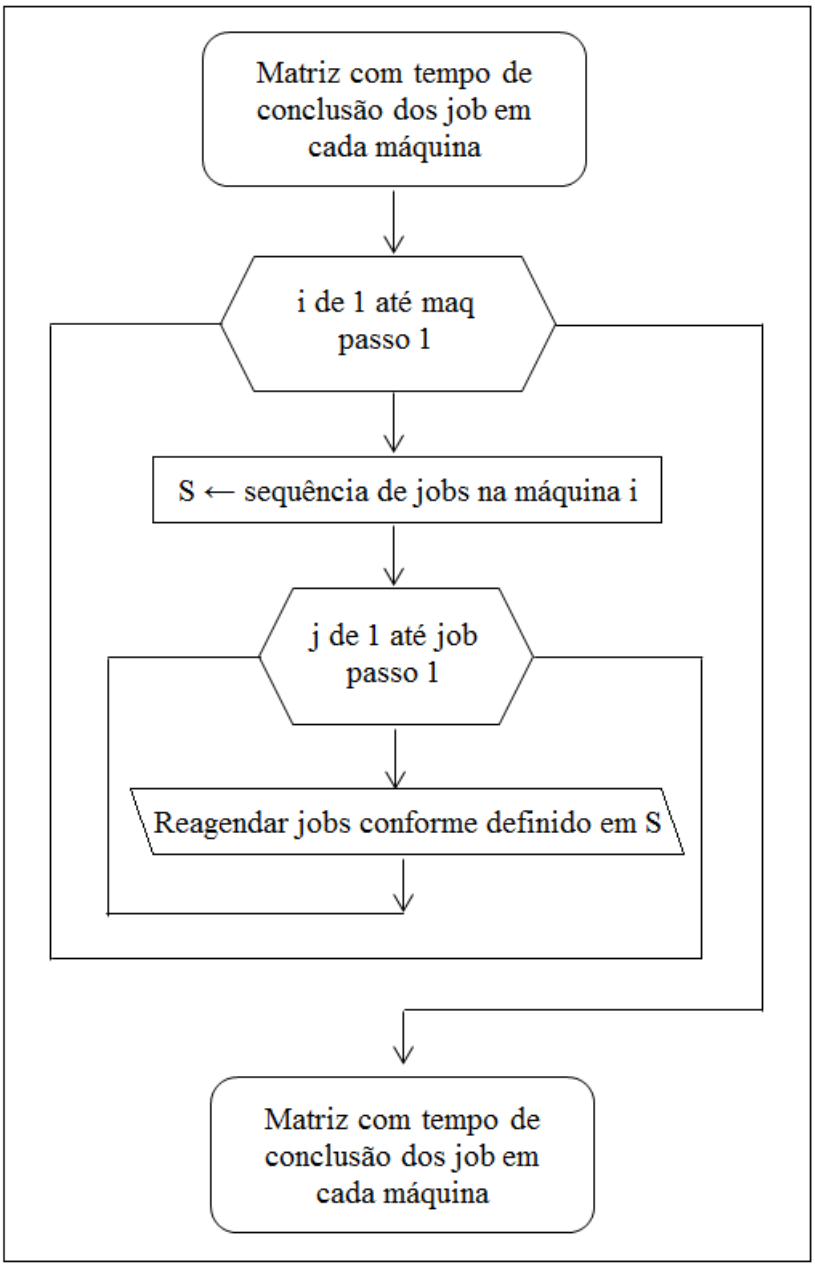

Figura 20 – Esquema da segunda parte da função *calc_energia_2.m*

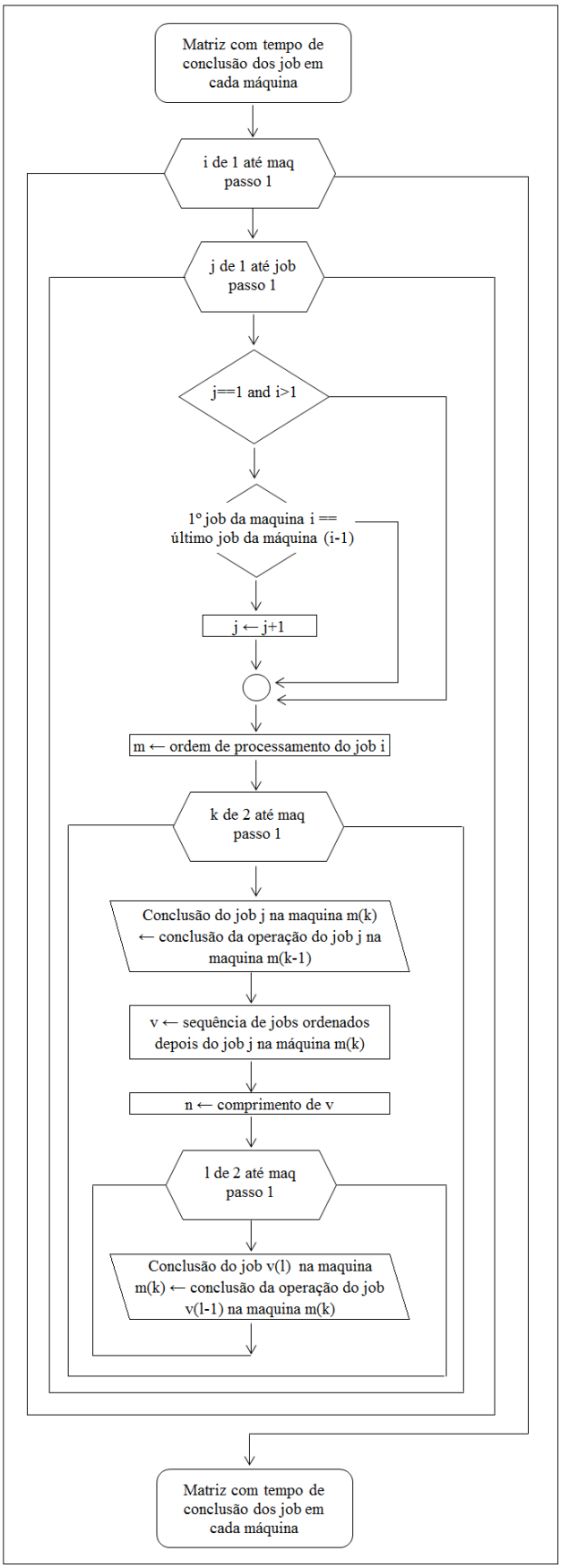

Figura 21 – Esquema da terceira parte da função *calc_energia_2.m*

Como saída desses três procedimentos tem-se uma matriz, semelhante a da Figura 22, com os tempos de conclusão das operações em cada máquina.

$$
\begin{array}{cc}
 & \text{Máquinas} \\
 & \begin{array}{cccc} 1 & 2 & \dots & m \end{array} \\
\text{Jobs} \begin{array}{c} 1 \\ 2 \\ \vdots \\ j \end{array} & \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ p_{j1} & p_{j2} & \cdots & p_{jm} \end{bmatrix}
\end{array}
$$

Figura 22 – Matriz de tempo de conclusão das operações

Como mencionado em seções anteriores, a forma de *codificação II* pode gerar uma solução inviável, então é preciso validar se o sequenciamento representado pela matriz de tempos de conclusão das operações em cada máquina respeitou as restrições tecnológicas dos *jobs*.

Se a solução for inviável é atribuído um valor muito alto à função objetivo. Caso contrário, essa e as demais funções de desempenho serão calculadas.

O valor máximo de cada linha (j) da matriz de tempo de conclusão das operações corresponde ao *makespan* (C_j) do job_j. A partir daí, mede-se a função objetivo e as demais de forma semelhante à apresentada na seção anterior.

3.2.4 Estimar a temperatura inicial

A temperatura do sistema (termodinâmica) representa no algoritmo *Simulated Annealing* um critério regulador de aceitação de soluções piores.

Para estimar a temperatura inicial tem-se as funções *estima_T0_1.m* e *estima_T0_2.m* (Anexo 9), a diferença entre elas está relacionada às duas forma de codificação da solução.

A Figura 23 ilustra a esquematização dessas funções.

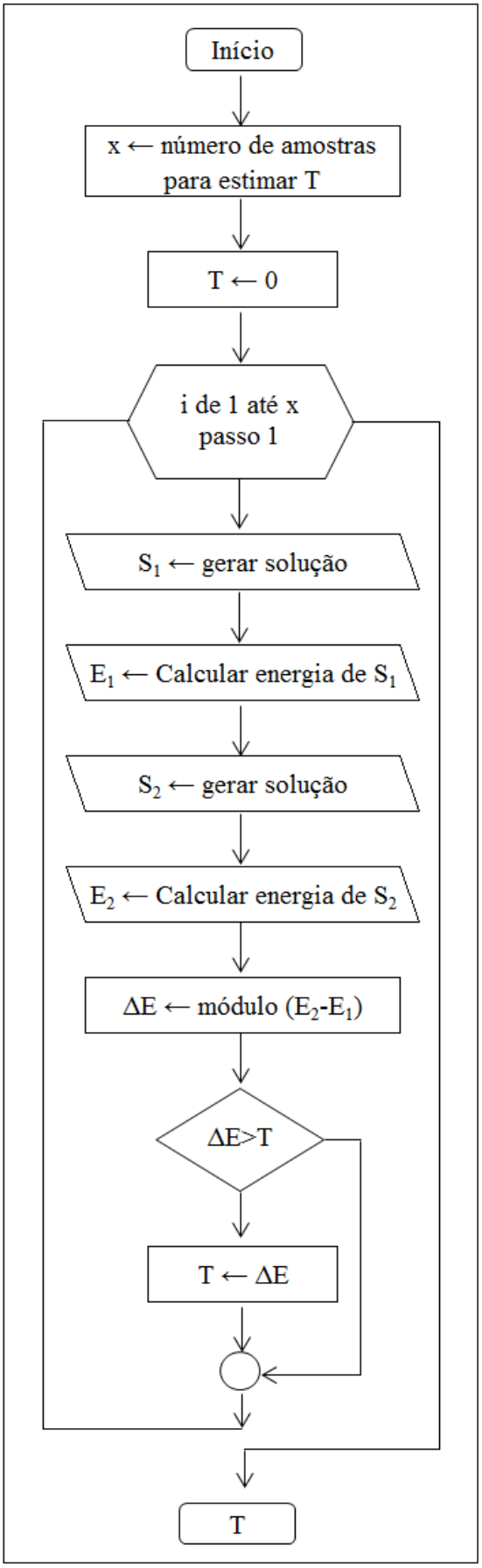

Figura 23 – Esquema da função para estimar a temperatura inicial

3.2.5 Parâmetros do algoritmo

Nesse trabalho, uma das exigências do algoritmo é a definição dos parâmetros e da taxa de decaimento da temperatura.

A cada iteração com a temperatura constante, são contabilizados o número de soluções testadas e quantas dessas soluções levaram a função objetivo a um valor menor. Em termodinâmica, representaria respectivamente o número de reconfigurações e o número de transições de estado do sistema.

É necessário determinar quantas transições ou reconfigurações serão feitas antes de se reduzir a temperatura, porém tais números precisam condizer com o tamanho do problema. Instâncias maiores tendem a ter mais possibilidades de soluções do que as menores.

Optou-se por definir o número máximo de transições e reconfigurações permitidas antes de reduzir a temperatura do sistema como: a multiplicação entre parâmetro regulador, número de *jobs* e número de máquinas do problema. O parâmetro que regula o número de reconfigurações deve ser maior que o que regula o número de estados.

Tais parâmetros e a taxa de decaimento da temperatura podem ser ajustados pelo usuário, dependendo da chance que se pretende aceitar soluções piores no intuito de escapar de mínimos locais.

Como critério de parada deve ser satisfeita uma das duas condições a seguir:

- tempo máximo de processamento informado;
- número máximo permitido de quedas na temperatura sem que se reduza o valor da função objetivo, que também deve ser informado pelo usuário.

4. RESULTADOS

4.1 VALIDAÇÃO DO ALGORITMO

O algoritmo desenvolvido (Anexo 3) será validado a partir da resolução de duas instâncias (4.1-1 e 4.1-2), disponibilizadas no Anexo 1.

Tais instâncias serão resolvidas pelas três variações do algoritmo proposto (*SA_JS_1*, *SA_JS_11*, *SA_JS_2*):

1. A combinação da *codificação I*, descrita na seção 3.2.1.1 e da forma aleatória de se gerar novas soluções (seção 3.2.2.1) implementada na função *gera_sol_1.m* disponível no Anexo 4, denominou-se de SA_JS_1
2. Chamou-se de SA_JS_11 a combinação da *Codificação I* e geração de novas soluções direcionada por probabilidades (*gera_sol_11.m*), presente no Anexo 5.
3. *Codificação II*, apresentada na seção 3.2.1.2 e a forma de se gerar novas soluções descrita na seção 3.2.2.2, implementada na função gera_sol_2 (Anexo 6), foi designada por SA_JS_2

Posteriormente, comparou-se os resultados encontrados com a solução ótima obtida através do *software* LiSA via algoritmo *Branch and Bound* .

Na instância 4.1-1 tem-se 5 *jobs* e 5 máquinas, os dados foram gerados com entropia dos tempos de processamento igual a zero e entropia da ordem de processamento igual a um. Os resultados encontrados para função objetivo estão copilados na tabela 4, e o sequenciamento dos *jobs* nas máquinas obtido pelo LiSA via algoritmo *Branch and Bound* está ilustrado na Figura 24 e o encontrado pela variação *SA_JS_1* do algoritmo proposto, está na Figura 25.

Tabela 4 – Resultados para J | r_i | Lmax 5 máquinas e 5 *jobs* (instância 4.1-1)

	LiSA	SA_JS_1[3]	SA_JS_1.1[4]	SA_JS_2[5]
Objetivo[1]	104	104	104	104
Time[2]	-	0,018	0,041	0,362

[1] Atraso máximo | [2] Tempo gasto em minutos para encontar solução | [3] *Codificação I* e geração de soluções aleatórias
[4] *Codificação I* e geração de soluções direcionada por probabilidade | [5] *Codificação II*

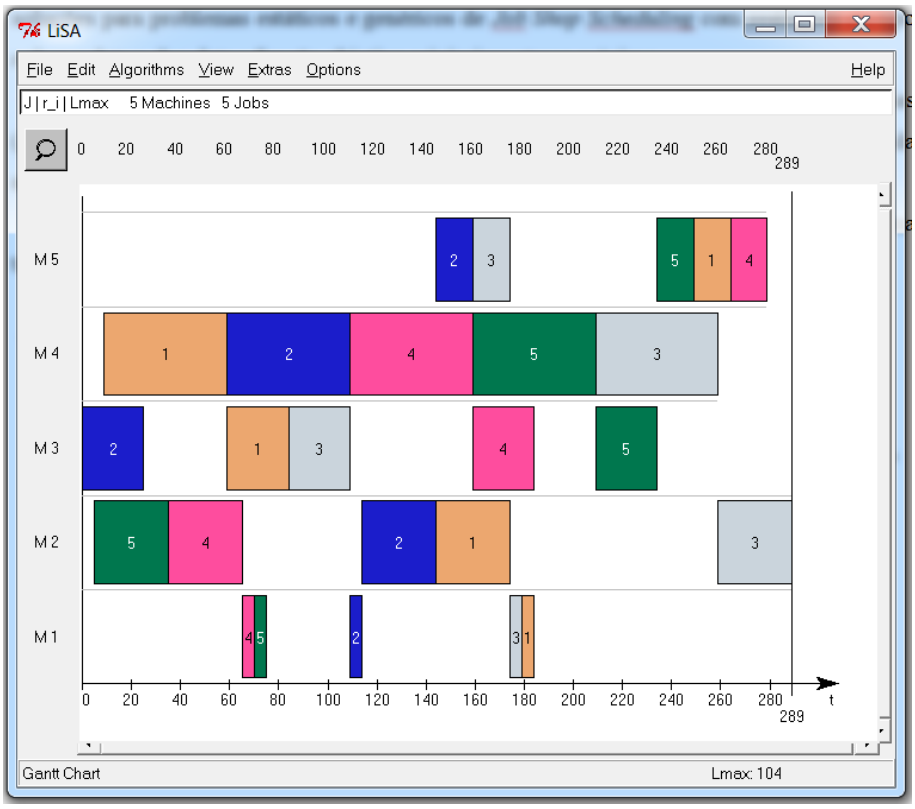

Figura 24 – Gráfico de Gantt para instância 4.1-1 - *Branch and Bound*
Fonte: *Software* LiSA

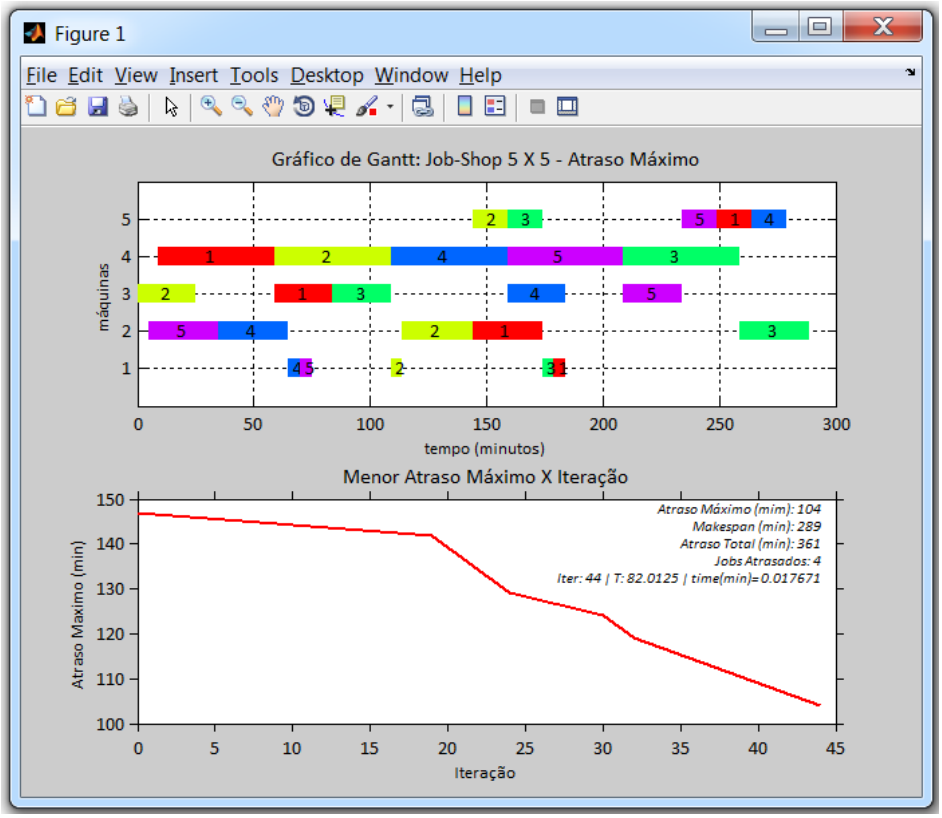

Figura 25 – Gráfico de Gantt para instância 4.1-1 - *SA_JS_1*

A segunda instância resolvida foi uma 7x5 (7 *jobs* e 5 máquinas), com entropia dos tempos de processamento e das ordens de processamento iguais a zero. Os valores encontrados de atraso máximo estão reunidos na tabela 5. A solução via LiSA está ilustrada na Figura 26, o sequenciamento encontrado pelas variações *SA_JS_1*, *SA_JS_2* estão ilustrados nas Figuras 27 e 28.

Tabela 5 – Resultados para J | r_i | Lmax 5 máquinas e 7 *jobs* (instância 4.1-2)

	LiSA	SA_JS_1[3]	SA_JS_1.1[4]	SA_JS_2[5]
Objetivo[1]	465	465	465	465
Time[2]	-	0,066	0,330	0,097

[1] Atraso máximo | [2] Tempo gasto em minutos para encontar solução | [3] *Codificação I* e geração de soluções aleatórias
[4] *Codificação I* e geração de soluções direcionada por probabilidade | [5] *Codificação II*

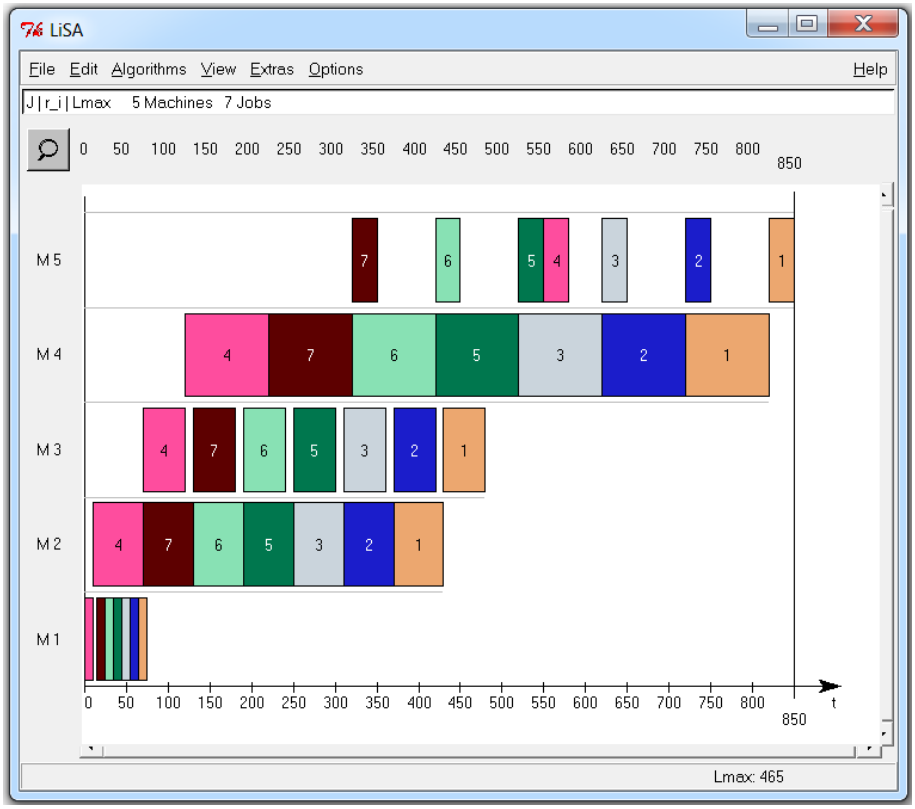

Figura 26 – Gráfico de Gantt para instância 4.1-2 - *Branch and Bound*
Fonte: *Software* LiSA

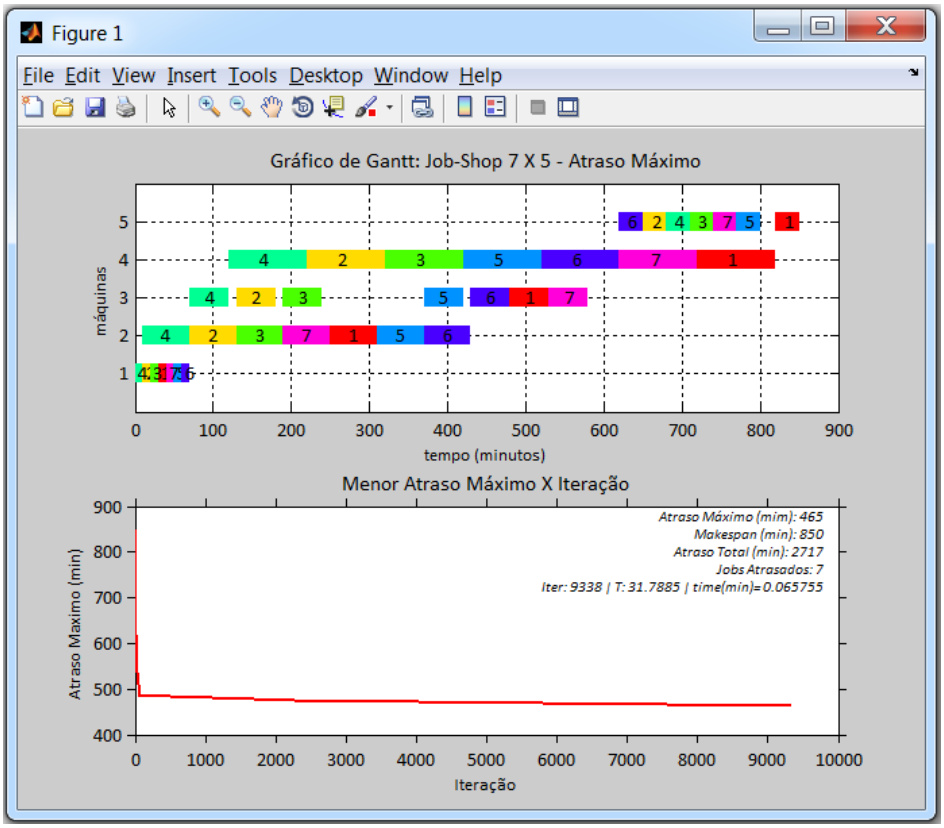

Figura 27 – Gráfico de Gantt para instância 4.1-2 - *SA_JS_1*

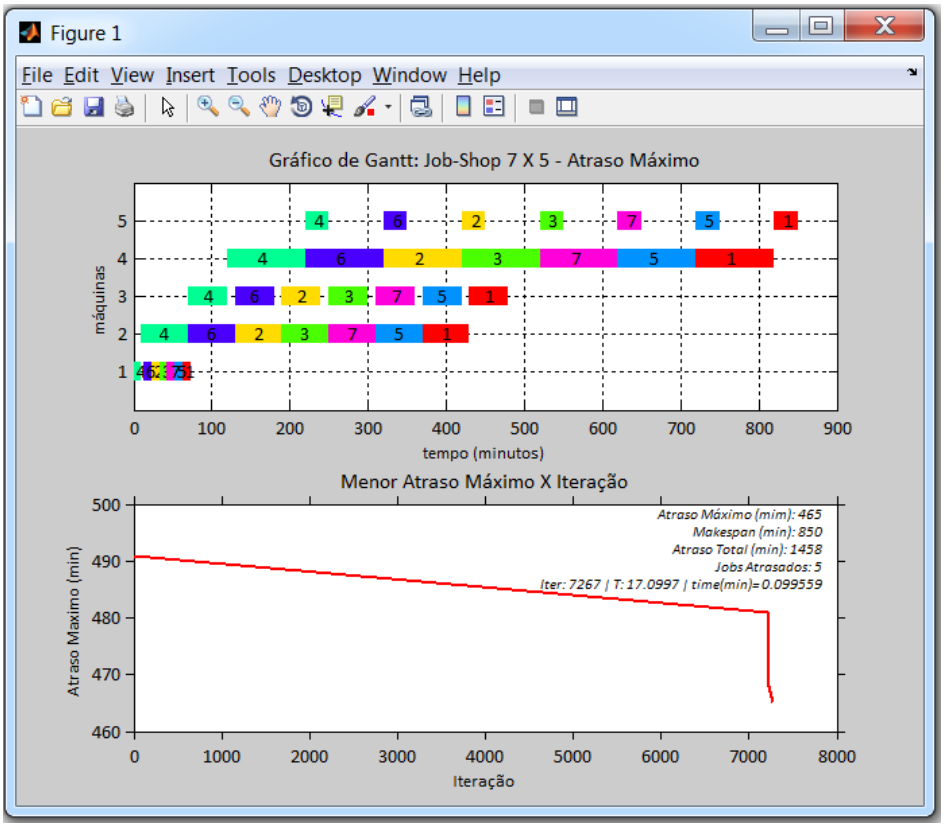

Figura 28 – Gráfico de Gantt para instância 4.1-2 - *SA_JS_2*

Observando as tabelas 4 e 5, pode-se perceber que as três variações do algoritmo elaborado conseguiram encontrar rapidamente a solução ótima para as duas instâncias consideradas.

No caso da instância 4.1-1, as três variações do algoritmo proposto e o LiSA via *Branch and Bound* chegaram no mesmo *scheduling*, como pode ser observado comparando as Figuras 24 e 25.

Já para a instância 4.1-2, os sequenciamentos encontrados tanto pelo LiSA como pelas variações do algoritmo desenvolvido são diferentes, apesar de terem a mesma função objetivo, representando um caso de problema com múltiplas soluções.

Comparando as demais funções de desempenho, tem-se que o *makespan* é igual nos três sequenciamentos (850), mas o atraso total e o número de *jobs* atrasados é menor na solução encontrada pela variação *SA_JS_2* (Figura 28).

4.2 ANÁLISE DE RESULTADOS

Nessa seção será analisada a interferência das entropias de tempo e de ordem de processamento no desempenho das três variações do algoritmo proposto.

Foram resolvidas 18 instâncias, disponíveis no Anexo 1, com três tamanhos diferentes, 8 *jobs* x 7 máquinas, 10 *jobs* x 12 máquinas e 20 *jobs* x 17 máquinas. Para cada um desses tamanhos de problema têm-se seis diferentes instâncias, criadas a partir das combinações de 0, 0.5 e 1 para os valores das duas entropias.

Os resultados encontrados pelas três variações do algoritmo desenvolvido foram comparados ao valor ótimo (LiSA via *Branch and Bound*), nos casos em que foi possível encontrá-lo, e com os valores obtidos através dos algoritmos *Simulated Annealing* e Busca Tabu implementados no LiSA.

Nas três variações do algoritmo admitem-se os seguintes valores para os parâmetros reguladores:

- Número de amostras consideradas para estimar a temperatura inicial: 100;
- Regulador do número de transições de estados para a temperatura constante:
  - *SA_JS_1* e *SA_JS_11:* 5;
  - *SA_JS_2*: 1;
- Regulador do número de reconfigurações para a temperatura constante: 10;
- Taxa de decaimento da temperatura: 0.9;

- Tempo máximo de execução:
  - Instâncias de 8 *jobs* x 7 máquinas : 3 minutos;
  - Instâncias de 10 *jobs* x 12 máquinas: 4 minutos;
  - Instâncias de 20 *jobs* x 17 máquinas: 5 minutos;
- Número máximo de quedas na temperatura sem alteração de estado: 6.

Para os parâmetros que definem a instância a ser gerada adotaram-se os seguintes valores:

- Número de *jobs*: 8, 10 ou 20;
- Número de máquinas: 7, 10 ou 17;
- Entropia de tempos de processamento: 0, 0.5 ou 1;
- Entropia de ordem de processamento: 0, 0.5 ou 1;
- Tempo mínimo de processamento por unidade de *job* em cada máquina: 1;
- Tempo máximo de processamento por unidade de *job* em cada máquina: 5;
- Tamanho médio do lote de produção: 10
- Amplitude de variação do tamanho do lote de produção: 20;
- Variação mínima permitida do tamanho do lote: 2;
- *Release date* mínimo: 0;
- *Release date* máximo: 25;
- Acréscimo mínimo no tempo de processamento para definição do *due date*: 30;
- Acréscimo máximo no tempo de processamento para definição do *due date*:60.

Na variação do algoritmo *SA_JS_2* é necessário ainda definir qual será a probabilidade de utilizar, a cada iteração, uma ou outra forma de gerar a nova solução, como exposto na seção 3.2.2.2. Variando de 0.5 a 0.9 o valor dessa probabilidade, foram resolvidas seis instâncias e a partir da comparação dos resultados encontrados, adotou-se 0.7 para tal probabilidade.

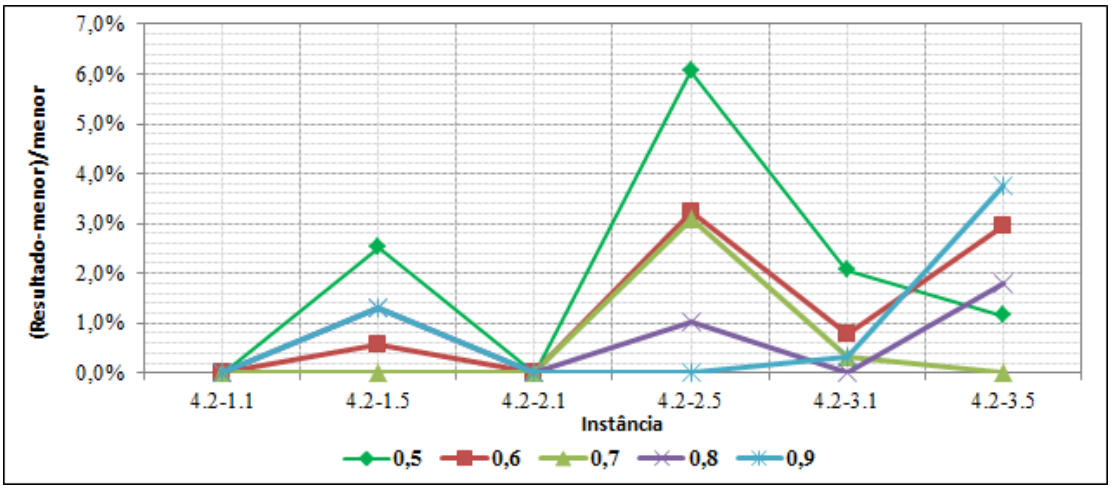

Figura 29 – Comparação dos resultados encontrados pela *SA_JS_2* variando a probabilidade de utilização das formas de gerar novas soluções

No *software* LiSA também é necessário inserir valores para os parâmetros dos métodos a serem utilizados. As definições a seguir sobre tais parâmetros foram retiradas do Manual do LiSA versão 3.0 elaborado por ANDRESEN et al. (s.d).

No caso do método exato, *Branch and Bound*, como pode ser visto na Figura 30, é necessário definir:

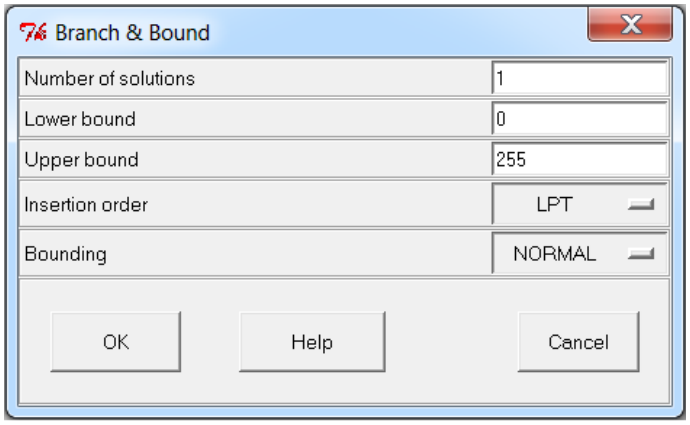

Figura 30 – Janela de *input* dos parâmetros do método *Branch and Bound*
Fonte: *Software* LiSA

- *Number of solution*: representa o limite superior do número de sequenciamentos que podem gerar o valor ótimo da função objetivo. O padrão para esse campo é 1, portanto, o que foi adotado.
- *Lower bound:* pode-se definir um limite inferior já conhecido para a função objetivo, assim o LiSA ira considerar todo *scheduling* que tiver função objetivo maior ou igual a tal número. Caso esse valor seja maior que o valor ótimo, irá conduzir a construção de sequenciamentos não ótimos. Como o

objetivo escolhido nesse trabalho é minimizar o atraso máximo, o valor atribuído a esse campo foi 0.

- *Upper bound:* nesse campo deve-se definir um limite superior previamente conhecido, e então, serão desconsiderados todos os sequenciamentos parciais que já estiverem gerando um valor para função objetivo maior que tal limite. Optou-se por inicializar o *Branch and Bound* a partir da solução encontrada pela Busca Tabu.
- *Insertion order:* representa a maneira como as tarefas serão ordenadas para formar o sequenciamento, têm-se duas opções:
  - *LPT* (*longest processing time*): as operações são anexadas de acordo com seu tempo de processamento, começando pela que tiver o maior tempo;
  - *RANDOM*: as operações são selecionadas de maneira aleátoria.

A esse campo foi atribuída a forma *RANDOM*.

- *Bouning*: corresponde ao procedimento utilizado no cálculo dos limites inferiores para a função objetivo:
  - NORMAL: utiliza-se o valor da função objetivo da sequencia parcial;
  - EXTENDED: este método não está implementado na versão 3.0.

Como critério de parada definiu-se um tempo de processamento de 5 minutos, portanto, ao fim desse tempo o valor encontrado pelo método *Branch and Bound* poderá ou não ser o valor ótimo. Para cada uma das instâncias resolvidas, como pode ser visto nas tabelas 6, 7 e 8 foi especificado se o valor encontrado é ou não o valor ótimo.

Os parâmetros exigidos pelo algoritmo iterativo Busca Tabu podem ser vistos na Figura 31.

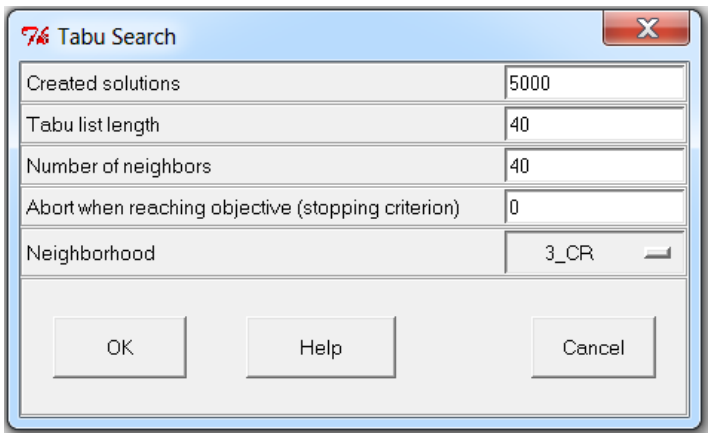

Figura 31 – Janela de *input* dos parâmetros do método Busca Tabu
Fonte: *Software* LiSA

- *Created solutions*: fixa-se o número máximo de soluções geradas, como se tem três diferentes tamanhos de instâncias padronizou-se os seguintes valores:
  - Instâncias de 8 *jobs* x 7 máquinas: 5.000;
  - Instâncias de 10 *jobs* x 12 máquinas: 10.000;
  - Instâncias de 20 *jobs* x 17 máquinas: 25.000.
- *Tabu list length*: corresponde ao número de soluções mantidas na memória do processo, que já foram testadas durante a pesquisa, e com as quais se pretende guiar a procura para uma região do espaço ainda não explorada. (JONES e RABELO, 1998). Foram adotados os seguintes valores:
  - Instâncias de 8 *jobs* x 7 máquinas: 30;
  - Instâncias de 10 *jobs* x 12 máquinas:50;
  - Instâncias de 20 *jobs* x 17 máquinas: 70.
- *Number of neighbors:* define-se o número de soluções vizinhas a serem geradas em cada iteração. Para esse campo adotou-se 50 vizinhos.
- *Abort when reaching objective (stopping criterion):* nessa lacuna pode-se definir um valor para a função objetivo, que quando for atingido a busca será concluída. No trabalho em questão, sendo o objetivo minimizar o atraso máximo, adotou-se 0.
- *Neighborhood:* define-se a forma como as soluções vizinhas serão geradas a partir da solução recorrente, são elas:
  - TRANS: inverte-se a subsequência de operações em uma máquina;
  - K_API: k+1 operações adjacentes são ressequenciadas;
  - SHIFT: desloca-se uma operação na sequência;
  - PI: duas operações aleatórias são trocadas;
  - CR_ API: operação correspondente ao $C_{\max}$ troca de posição com a adjacente no sequenciamento;
  - BL_AP: várias operações críticas são trocam de posição com suas adjacentes;
  - CR_SHIFT: a operação crítica é deslocada no sequenciamento;
  - BL_SHIFT: várias operações críticas são deslocadas no sequenciamento;

- CR_TRANS: inverte as posições de duas operações críticas em uma máquina;
- 3_CR: a operação crítica é trocada com uma operação diretamente adjacente, e o antecessor e o sucessor de tal operação são intercambiados com outras operações.
- K_REINSERTION: k operações são removidos da sequencia e reinseridos em posições aleatórias.
- CR_TRANS_MIX: em 75% dos casos a solução vizinha é gerada pela CR_TRANS, no restante pelo CR_API;
- CR_SHIFT_MIX: tem-se 75% de chance de uma solução vizinha ser gerada por CR_SHIFT e 25% por CR_API.

Nesse trabalho adotou-se a forma 3_CR.

A Figura 32 ilustra os parâmetros utilizados pelo algoritmo *Simulated Annealing* implementado no LiSA versão 3.0.

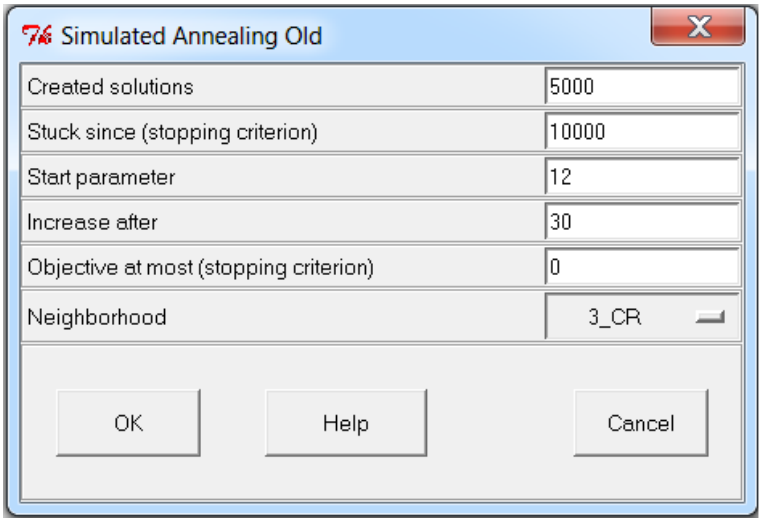

Figura 32 – Janela de *input* dos parâmetros do *Simulated Annealing*
Fonte: *Software* LiSA

- *Created solutions*: fixa-se o número máximo de soluções geradas, como se tem três diferentes tamanhos de instâncias padronizou-se os seguintes valores:
  - Instâncias de 8 *jobs* x 7 máquinas: 5.000;
  - Instâncias de 10 *jobs* x 12 máquinas: 10.000;
  - Instâncias de 20 *jobs* x 17 máquinas: 25.000.

- *Stuck since (stopping criterion):* nessa lacuna define se o número máximo de iterações a serem realizadas, para cada tamanho de instância admitiu-se um valor:
    - Instâncias de 8 *jobs* x 7 máquinas: 10.000
    - Instâncias de 10 *jobs* x 12 máquinas: 20.000 ;
    - Instâncias de 20 *jobs* x 17 máquinas: 50.000;
- *Objective at most (stopping criterion):* nessa lacuna pode-se definir um valor para a função objetivo, que quando for atingido a busca será concluída. No trabalho em questão, sendo o objetivo minimizar o atraso máximo, adotou-se 0.
- *Neighborhood:* Define o procedimento de geração de novas soluções a partir da solução recorrente, pode-se optar por qualquer uma das formas apresentadas na definição de parâmetros da Busca Tabu. Adotou-se a mesma maneira escolhida para a Busca Tabu, o 3_CR.

Na resolução através do LiSA de todas as instâncias abordadas nesse trabalho, tanto via Busca Tabu como pelo *Simulated Annealing* a solução inicial foi gerada pela regra de despacho RAND, na qual as operações são sequenciadas em cada máquina de forma aleatória.

Após executar as três variações do algoritmo desenvolvido por três vezes para cada uma das dezoito instâncias, os melhores resultados, o tempo gasto para encontrá-los e o número de iterações realizadas foram copilados nas tabelas 6, 7 e 8, de acordo com o tamanho do problema.

Em tais tabelas também estão reunidos os resultados encontrados pelo LiSA via *Branch and Bound*, Busca Tabu e *Simulated Anealing.* No entanto, para algumas instâncias não foi possível encontrar a solução ótima, a partir dos valores previamente definidos para os parâmetros do método *Branch and Bound*, porém esses resultados foram apresentados nas tabelas como forma de referência.

Para facilitar a comparação e análise dos valores obtidos optou-se por relativizá-los segundo a diferença em relação ao menor valor encontrado. Tais comparações estão representadas nas Figuras 33, 34 e 35.

Tabela 6 – Resultados dos problemas J | r_i | Lmax – 8 *jobs* x 7 máquinas

	Instância	Entropia 1[3]	Entropia 2[4]	*Branch and Bound*[5]	*Busca Tabu*[5]	*Simulated Annealing*[5]	SA_JS_1[6]	SA_JS_1.1[7]	SA_JS_2[8]
Objetivo[1]	4.2-1.1	0	0	**255***	259	295	259	259	**255**
Time[2]				-	-	-	0,22	2,78	0,01
Iterações				-	-	-	37.026	7.948	26.714
Objetivo	4.2-1.2	1	0	365	755	935	**355**	426	**355**
Time				-	-	-	0,27	2,37	0,50
Iterações				-	-	-	38.148	7.964	26.505
Objetivo	4.2-1.3	0	1	**160***	160	170	**160**	175	526
Time				-	-	-	0,07	0,03	0,82
Iterações				-	-	-	35.343	8.172	22.426
Objetivo	4.2-1.4	0.5	0.5	**24***	160	424	130	164	462
Time				-	-	-	0,28	0,10	0,41
Iterações				-	-	-	39.831	8.063	21.698
Objetivo	4.2-1.5	1	1	**190***	257	309	226	322	1073
Time				-	-	-	0,29	1,06	0,97
Iterações				-	-	-	39.831	8.044	22.326
Objetivo	4.2-1.6	1	0.5	**180***	218	451	**180**	272	1172
Time				-	-	-	0,26	2,10	1,11
Iterações				-	-	-	35.904	8.010	22.257

[1] Atraso máximo | [2] Tempo gasto em minutos para encontar solução | 3 Tempo de processamento | [4] Ordem de processamento | [5] algoritmos implementados no LiSA
[6] *Codificação I* e geração de soluções aleatórias| [7] *Codificação I* e geração de soluções direcionada por probabilidade | [8] *Codificação II* | * Solução Ótima

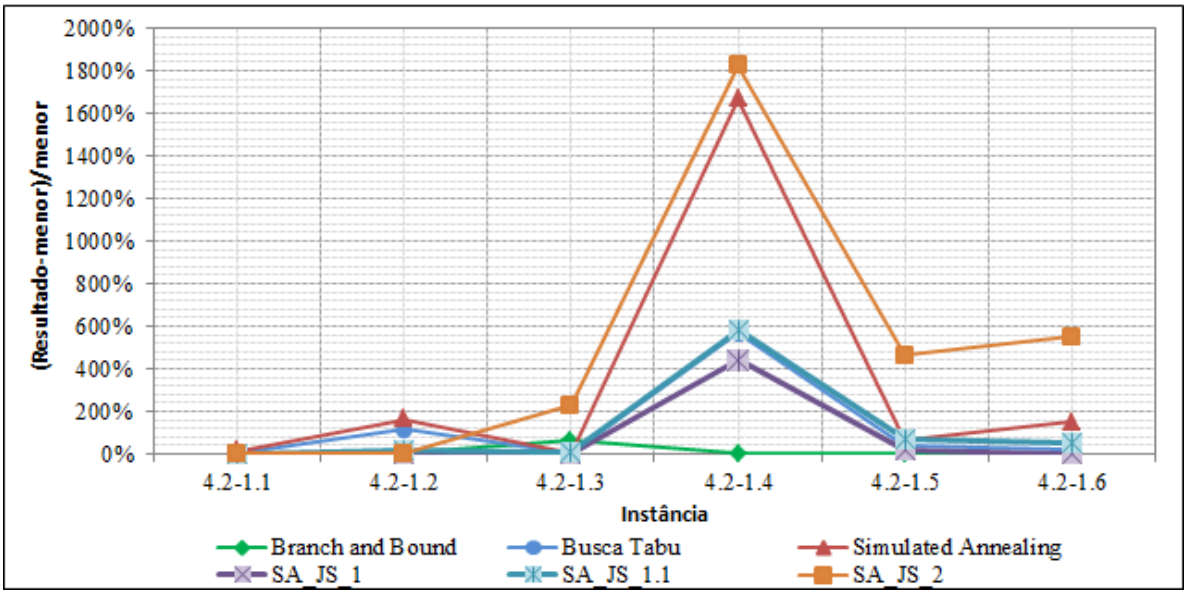

Figura 33 – Comparação dos resultados obtidos pelas três variações do algoritmo para problemas J | r_i | Lmax – 8 *jobs* x 7 máquinas

Tabela 7 – Resultados dos problemas J | r_i | Lmax – 10 *jobs* x 12 máquinas

	Instância	Entropia 1[3]	Entropia 2[4]	*Branch and Bound*[5]	*Busca Tabu*[5]	*Simulated Annealing*[5]	SA_JS_1[6]	SA_JS_1.1[7]	SA_JS_2[8]
Objetivo[1]	4.2-2.1	0	0	246	272	279	269	405	**204**
Time[2]				-	-	-	0,89	2,39	0,28
Iterações				-	-	-	358.937	2.585	97.518
Objetivo	4.2-2.2	1	0	**1***	587	765	**1**	**1**	**1**
Time				-	-	-	1,25	1,74	1,47
Iterações				-	-	-	340.411	2.532	96.947
Objetivo	4.2-2.3	0	1	349	173	349	**102**	312	1666
Time				-	-	-	0,94	2,96	0,85
Iterações				-	-	-	341.250	2.102	96.853
Objetivo	4.2-2.4	0.5	0.5	269	90	418	**45**	261	1393
Time				-	-	-	15946,00	1,50	0,21
Iterações				-	-	-	349.863	2.576	96.998
Objetivo	4.2-2.5	1	1	325	172	575	**14**	140	799
Time				-	-	-	39852,00	0,77	2,38
Iterações				-	-	-	346.330	2.619	95.769
Objetivo	4.2-2.6	1	0.5	472	91	472	**1**	172	564
Time				-	-	-	1,86	1,44	2,71
Iterações				-	-	-	336.013	2.583	96.124

[1] Atraso máximo | [2] Tempo gasto em minutos para encontar solução | 3 Tempo de processamento | [4] Ordem de processamento | [5] algoritmos implementados no LiSA
[6] *Codificação I* e geração de soluções aleatórias| [7] *Codificação I* e geração de soluções direcionada por probabilidade | [8] *Codificação II* | * Solução Ótima

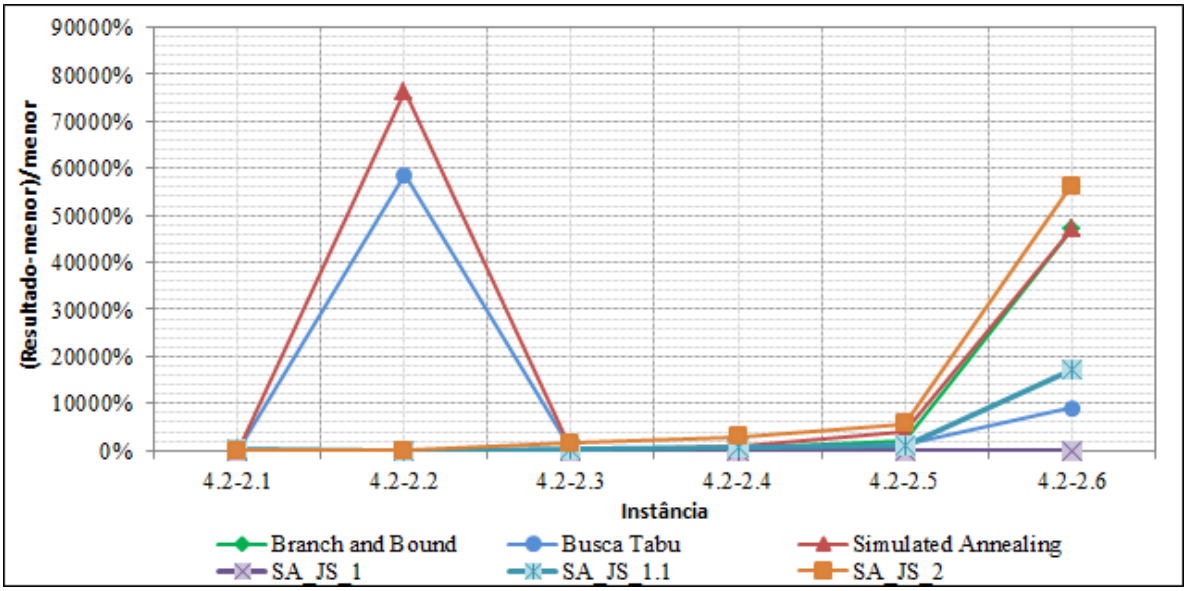

Figura 34 – Comparação dos resultados obtidos pelas três variações do algoritmo para problemas J | r_i | Lmax – 10 *jobs* x 12 máquinas

Tabela 8 – Resultados dos problemas J | r_i | Lmax – 20 *jobs* x 17 máquinas

	Instância	Entropia 1[3]	Entropia 2[4]	*Branch and Bound*[5]	*Busca Tabu*[5]	*Simulated Annealing*[5]	SA_JS_1[6]	SA_JS_1.1[7]	SA_JS_2[8]
Objetivo[1]	4.2-3.1	0	0	698	842	706	1092	1515	**632**
Time[2]				-	-	-	4,70	3,22	0,09
Iterações				-	-	-	155.506	424	14.042
Objetivo	4.2-3.2	1	0	**1000**	2346	2423	1636	2097	**1000**
Time				-	-	-	4,89	0,64	0,09
Iterações				-	-	-	147.540	413	14.617
Objetivo	4.2-3.3	0	1	540	540	467	**440**	917	5512
Time				-	-	-	4,96	1,48	1,65
Iterações				-	-	-	157.050	428	5.984
Objetivo	4.2-3.4	0.5	0.5	1170	1170	1054	1162	**1051**	5433
Time				-	-	-	4,98	4,73	4,27
Iterações				-	-	-	156.939	409	10.774
Objetivo	4.2-3.5	1	1	1804	1804	1872	1326	**1690**	8446
Time				-	-	-	4,99	0,72	0,71
Iterações				-	-	-	156.900	427	10.383
Objetivo	4.2-3.6	1	0.5	1546	1546	1648	**1499**	1845	8581
Time				-	-	-	4,89	0,67	2,03
Iterações				-	-	-	152.011	425	6.166

[1] Atraso máximo | [2] Tempo gasto em minutos para encontar solução | 3 Tempo de processamento | [4] Ordem de processamento | [5] algoritmos implementados no LiSA
[6] *Codificação I* e geração de soluções aleatórias| [7] *Codificação I* e geração de soluções direcionada por probabilidade | [8] *Codificação II* | * Solução Ótima

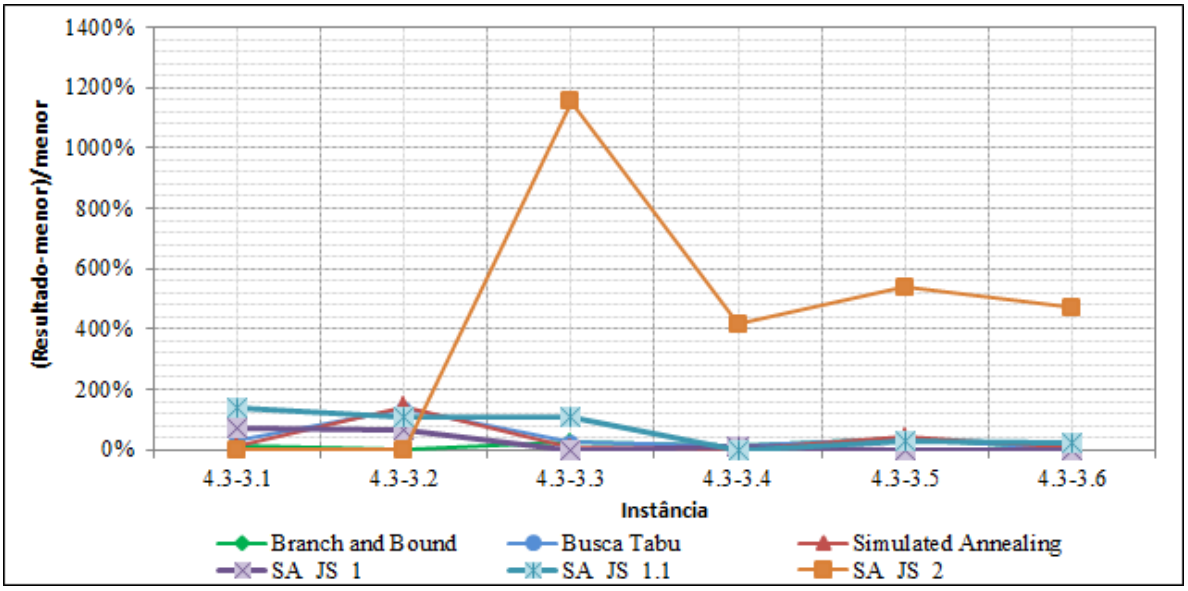

Figura 35 – Comparação dos resultados obtidos pelas três variações do algoritmo para problemas J | r_i | Lmax – 20 *jobs* x 17 máquinas

A partir da análise das tabelas e gráficos expostos acima, pode-se chegar a algumas conclusões iniciais baseando se nas instâncias estudadas, como:

- Em problemas com entropia de processamento próximo de 1 não se deve utilizar a variação *SA_JS_2*;
- A variação *SA_JS_2* é a mais indicada para resolução de problemas com entropia de processamento baixa;
- As variações *SA_JS_ 1* e *SA_JS_11* apresentam desempenho melhor em problemas com entropia de processamento mais alta;
- A implementação da variação *SA_JS_11* é ineficiente;
- A Entropia dos tempos de processamento não favorece uma ou outra variação do algoritmo;
- Problemas com entropia de tempos de processamento alta são mais fáceis.

Instâncias em que os caminhos de máquinas percorridos pelos *jobs* são iguais não há restrição de combinação de sequência de *jobs* em cada máquina para formar uma solução viável. Portanto, o espaço de busca é consideravelmente maior do que em problemas com entropia de ordem de processamento próximo a 1, nas quais as restrições tecnológicas dos *jobs* limitam o número de soluções factíveis.

O fato da variação *SA_JS_2* não obter bons resultados comparados aos encontrados pelas demais variações, para problemas com entropia de ordem de processamento alta está diretamente relacionado à limitação da forma de *Codificação II*, possibilidade de geração de soluções inviáveis. Em instâncias em que as ordens de máquinas percorridas pelos *jobs* são muito diferentes uma das outras, aumenta-se a chance de se gerar soluções inviáveis.

Como apresentado na seção 3.2.3.2, para calcular a função objetivo a partir de uma dada solução deve-se primeiro descodificá-la, ou seja, construir o sequenciamento, e depois verificar se é viável, portanto soluções inviáveis representam perda de processamento na execução de tal procedimento.

As variações *SA_JS_1* e *SA_JS_11* nunca geram soluções inviáveis, porém possuem a limitação de diferentes soluções representarem o mesmo sequenciamento. Para problemas com *entropia 2* igual a 0, tende-se a gerar mais soluções não únicas, pois alterações de posições entre os elementos do vetor solução só fará diferença se representarem operações em

uma mesma máquina (seção 3.2.1.2). Em contra partida na *Codificação II*, soluções diferentes nunca irão representar o mesmo sequenciamento.

Da mesma forma que a *SA_JS_2*, para medir a função objetivo nas variações *SA_JS_1* e *SA_JS_11* também é preciso descodificar a solução, logo soluções diferentes para um mesmo sequenciamento representam tempo de processamento perdido. Isso explica as observações iniciais referentes à variação *SA_JS_2* ser mais indicada para instâncias com *entropia 2* mais baixa, enquanto as variações *SA_JS_1* e *SA_JS_11* para entropia 2 mais alta.

O algoritmo *SA_JS_11* mostrou-se bastante lento comparado às outras duas variações, isso se deve ao método de geração de soluções direcionado por probabilidades, uma vez que para cada posição do vetor solução é realizada uma série de procedimentos apresentados na seção 3.2.2.1, quanto maior o problema mais tempo se gasta em cada iteração. Na Figura 33, pode-se observar a relação entre número de iterações realizadas pela variação *SA_JS_1* e pela *SA_JS_11*.

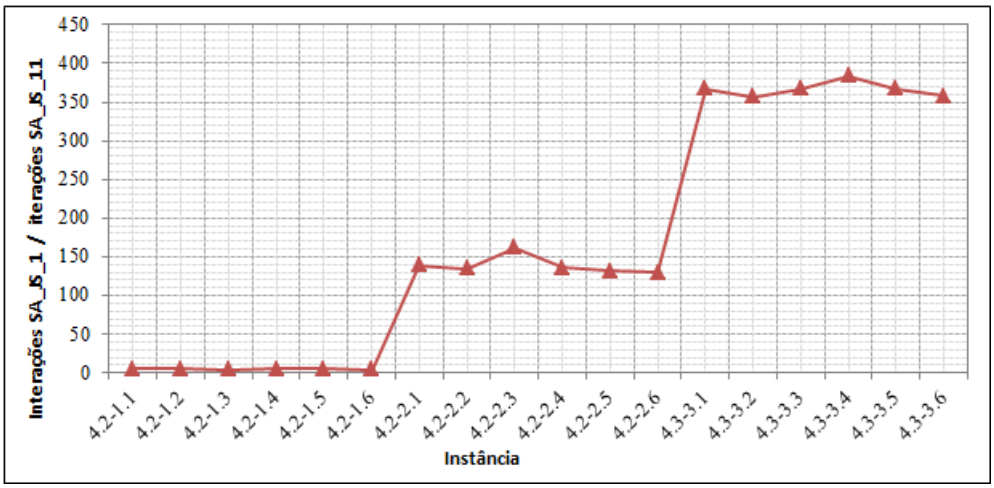

Figura 36 – Comparação do número de iterações realizadas pela *SA_JS_1* e *SA_JS_11*

Alterações na entropia de tempos de processamento não favorecem nenhuma das variações do algoritmo, porém nas instâncias com *entropia 1* igual a 1, os resultados são encontrados em um tempo menor do que nas instâncias com *entropia 1* igual a 0 ou 0.5.

Observando os resultados encontrados pelo LiSA, das 18 instâncias abordadas, apenas para 6 delas foi possível encontrar o valor ótimo, sendo que para essas 6 alguma das três variações do algoritmo desenvolvido também encontrou tal resultado.

Para as demais instâncias em que não se possível descobrir o valor ótimo, o menor resultado veio sempre de alguma das três variações do algoritmo desenvolvido.

5. CONSIDERAÇÕES FINAIS

5.1 CONCLUSÕES

Este trabalho propôs inicialmente a resolução de problemas genéricos e estáticos de *Job Shop Scheduling* e posterior análise de como algumas características das instâncias interferem na eficácia do algoritmo desenvolvido. O modelo deve ser suficientemente flexível para criar problemas com características variadas, como: tamanho, valores dos tempos de processamento e grau de desordem.

A escolha por uma metaheurística fez se necessária, uma vez que o espaço de busca dos problemas tratados crescem de forma não polinomial à medida que se aumenta o número de *jobs* ou o número de máquinas, tornando-os intratáveis em um tempo computacional viável pelo método exato.

As três variações do algoritmo desenvolvido são compostas por combinações das duas formas de codificar soluções com os três procedimentos de gerar novas soluções. De acordo com a avaliação, apresentada na seção 4.2, da interferência das características das instâncias nos resultados e no tempo gasto para encontrá-los, pode-se perceber a baixa chance de se desenvolver um método de otimização que seja capaz de encontrar bons resultados para problemas com características muito diferentes.

Ao se comparar os resultados obtidos pelo LiSA via *Branch and Bound*, Busca tabu e *Simulated Annealing* com os encontrados pela técnica proposta, percebe-se, principalmente nas instâncias maiores, a superioridade de desempenho do algoritmo desenvolvido.

As limitações desse trabalho estão diretamente relacionadas à modelagem dos problemas de *Job Shop*. Assumiram-se as premissas de *scheduling* estático, ou seja, inicializado o processo de produção, o sequenciamento não pode ser alterado até que todos os produtos estejam concluídos, o que impossibilita um pedido feito após o início de uma batelada ser incluído nela. Pressupõe-se, também, que todos os recursos estejam disponíveis durante todo o período de produção, o que impede que paradas programadas e chances de quebras sejam consideradas. Os tempos de processamento são determinísticos, transporte dos produtos de uma máquina para outra não são considerados tampouco o tempo gasto com *setup*.

Apesar da fragilidade de tais premissas o algoritmo proposto pode ser utilizado como suporte na tomada de decisão de problemas reais, uma vez que se adotem tais considerações,

ou ser tomado por base para a evolução de um procedimento que resolva problemas dinâmicos.

5.2 RECOMENDAÇÕES DE TRABALHOS FUTUROS

Dentro do contexto em que esse trabalho foi desenvolvido é possível destacar duas linhas de pesquisa: os vários tipos de problemas de *scheduling* e a adaptação de técnicas computacionais, em especial metaheurísticas, na resolução de tais problemas.

No que se refere aos problemas de *scheduling*, propõe-se:

- Considerar as restrições de tempo de *setup* e tempo gasto no transporte dos *jobs* de uma máquina para outra;
- Modelar o problema com tempo de processamento probabilístico;
- Considerar a confiabilidade das máquinas;
- Aplicar S*imulated Annealing* em outros tipos de problemas como *flow shop*, *open shop*, problemas de máquina única.

No que tange a metaheurística utilizada, recomenda-se:

- Desenvolver um método de definição dos parâmetros de entrada do algoritmo, para melhor adaptá-lo a cada problema a ser resolvido;
- Elaborar um procedimento que verifique se a solução é viável antes de decodificá-la;
- Criar uma forma de codificação para as soluções que não gere soluções inviáveis e que represente unicamente um sequenciamento.

Apesar do *Simulated Annealing* ter sido facilmente adaptado aos problemas de *Job Shop Scheduling* é valido considerar a aplicação de outras técnicas de otimização, como Busca Tabu, algoritmo genético, lógica fuzzy, colônia de formigas, etc.

Outro ponto a ser ponderado é a escolha da função objetivo, como *makespan*, atraso total, utilização das máquinas, que dependendo das restrições modeladas podem ser mais indicadas que minimizar o atraso máximo.

REFERÊNCIAS

ANDRESEN, Michael. **LiSA – A Library of Scheduling Algorithms Handbook for Version 3.0.** Otto-von-Guericke Universität Magdeburg. Disponível em: <http://www.math.uni-magdeburg.de/~werner/handbuch-en.pdf > Acesso em: julho/2014.

AMICO, M.; TRUBIAN, M. Applying tabu search to the job-shop scheduling problem. **Annals of Operations Research.** v. 41, n. 3, p. 231-252, 1993.

ASSOCIAÇÃO BRASILEIRA DE ENGENHARIA DE PRODUÇÃO. Referências de conteúdos da Engenharia de Produção. Disponível em: <http://www.abepro.org.br/interna.asp?p=399&m=424&ss=1&c=362 >. Acesso em: jun/2014

BAKER, K.R. **Introduction to sequencing and scheduling.** 1. ed. New York: John Wiley e Sons, 1974.

BINATO, S.; HERY, W. J.; LOEWENSTERN, D. M.; RESENDE, M. G. C. A Grasp for Job Shop Scheduling. **Operations Research/Computer Science Interfaces Series.** v. 15, p. 59-79, 2002.

BOLTZMANN, L. **Lectures on gas theory**. New York: Dover Publications, 1995.

BRUCKER, Peter; JURISCH, Bernd; SIEVERS, Bernd. A branch and bound algorithm for the job-shop scheduling problem. **Discrete Applied Mathematics.** v. 49, p. 107-127, março, 1994.

CORRÊA, H. L.; CORRÊA, C. A. **Administração de produção e operações:** manufatura e serviços: uma abordagem estratégica. 3. ed. São Paulo: Atlas, 2012.

CASTRO, L.; ZUBEN, F. V. Hill Climbing e Simulated Annealing. Relatório Técnico - DCA/FEEC/Unicamp. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia707_02/topico1_02.pdf>. Acesso em: jun/2014.

HILLIER, Frederick S.; LIEBERMAN, Gerald J. **Introdução à pesquisa operacional**. 8. ed. São Paulo: McGraw-Hill, 2006.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by Simulated Annealing. **Science**. v. 220, n. 4598, p. 671 – 680, 1983.

JAIN, A. S.; MEERAN, S. Deterministic *Job*-Shop Scheduling: Past, Present and Future. **Operations Research.** v. 113, p. 390-434, 1998.

JONES, Albert; RABELO, Luis. Survey of Job shop Scheduling Techniques. **MSID Publications**. Institute of Standards and Technology, Gaithersburg, MD, 1998.

LAARHOVEN, Peter; AARTS, Emile; LENSTRA, Jan. Job Shop Scheduling by Simulated Annealing. **Operations Research**. v. 40, n. 1, p. 113-125, 1992.

MASSA NETO, E. S. **Hibridação de Algoritmos Genéticos com Relaxação Lagrangeana Aplicada ao Job Shop**. Dissertação (Mestrado em Matemática) – Universidade Federal da Bahia, Bahia, 2004

MATTFELD, D.C.; BIERWIRTH, C. A Search Space Analysis of the Job Shop Scheduling Problem. **Annals of Operations Research,** v.86, p. 441-453, 1999.

METAHEURISTICS NETWORK. Referências de conteúdos de metaheurística. Disponível em: <http://www.metaheuristics.net/index.php?main=1>. Acesso em: jun/2014.

METROPOLIS, N.; ROSENBLUTH, A. W.; ROSENBLUTH, M. N.; TELLER, A. H. ; TELLER, E. Equation of State Calculations by Fast Computing Machines. **Journal of Chemical Physics**. v.21, n. 6, p. 1087 - 1092, 1953.

MIGUEL, P. A. C. **Metodologia de pesquisa em engenharia de produção e gestão de operações.** Rio de Janeiro: Elsevier, 2010.

MORTON, Thomas E.; PENTICO, David W. **Heuristic scheduling systems: with applications to production systems and project management.** New York: John Wiley e Sons, 1993.

MUTH, J. F.; THOMPSON, G. L. **Industrial Scheduling.** Englewood Cliffs, N.J.: Prentice-Hall Inc, 1963.

NORONHA, A. B.; MENDES, R. S. Controle de transitórios em escalonamento de job shop cíclico. **Controle & Automação**, v. 13, n. 3, p. 231-247, 2002.

PEZZELLA, F.; MORGANTI, G.; GIASCHETTI, G. A genetic algorithm for the Flexible Job-shop Scheduling Problem. **Computers & Operations Research.** v. 35, p. 3202-3212, 2008.

PINEDA, J. O. C. **Entropia segundo Claude Shannon**: O desenvolvimento do conceito fundamental da teoria da informação. Dissertação (Mestrado em História da Ciência) - Pontifícia Universidade Católica de São Paulo, São Paulo, 2006.

PINEDO, Michael. **Scheduling**: Theory, Algorithms ans Systems. Prentice-Hall Inc., 2002.

VÁZQUEZ, M.; WHITLEY, L. A comparison of Genetic Algorithms in solving the Static Job Shop Scheduling Problem. **Parallel Problem Solving from Nature.** p. 303-312, 2000.

ANEXO 1 – INSTÂNCIAS

Instância 3.2.2.2-1

Tempos de Processamento

26	104	78	130	52	26	26	104	104	52
6	24	18	30	12	6	6	24	24	12
2	8	6	10	4	2	2	8	8	4
8	32	24	40	16	8	8	32	32	16
26	104	78	130	52	26	26	104	104	52
2	8	6	10	4	2	2	8	8	4
28	112	84	140	56	28	28	112	112	56
8	32	24	40	16	8	8	32	32	16
2	8	6	10	4	2	2	8	8	4
4	16	12	20	8	4	4	16	16	8

Ordem das máquinas

2	1	6	8	7	3	5	4	10	9
3	5	1	8	2	9	10	4	6	7
4	10	5	1	3	7	6	9	8	2
3	6	1	8	10	2	9	4	7	5
5	6	7	8	1	2	3	4	10	9
6	3	2	8	7	1	5	4	10	9
2	1	4	3	8	9	10	5	6	7
4	9	8	1	6	5	10	3	2	7
2	1	9	5	4	10	8	7	3	6
2	1	5	7	3	10	4	9	8	6

Release date

3	2	9	0	8	10	4	3	10	7

Due date

1002	238	94	307	1044	88	988	329	91	171

Instância 4.1-1

Tempos de Processamento

5	30	25	50	15
5	30	25	50	15
5	30	25	50	15
5	30	25	50	15
5	30	25	50	15

Ordem das máquinas

4	3	2	1	5
3	4	1	2	5
3	5	1	4	2
2	1	4	3	5
2	1	4	3	5

Release date

9	0	2	4	5

Due date

185	167	185	175	175

Instância 4.1-2

Tempos de Processamento

10	60	50	100	30
10	60	50	100	30
10	60	50	100	30
10	60	50	100	30
10	60	50	100	30
10	60	50	100	30
10	60	50	100	30

Ordem das máquinas

1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5

Release date

10	3	9	0	4	13

Due date

385	337	337	325	364	366

Instância 4.2-1.1

Tempos de Processamento

10	40	30	50	20	10	10
10	40	30	50	20	10	10
10	40	30	50	20	10	10
10	40	30	50	20	10	10
10	40	30	50	20	10	10
10	40	30	50	20	10	10
10	40	30	50	20	10	10
10	40	30	50	20	10	10

Ordem das máquinas

1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7

Release date

10	8	10	6	5	16	1	25

Due date

247	250	260	231	231	266	252	260

Instância 4.2-1.2

Tempos de Processamento

24	96	72	120	48	24	24
30	120	90	150	60	30	30
20	80	60	100	40	20	20
24	96	72	120	48	24	24
20	80	60	100	40	20	20
4	16	12	20	8	4	4
24	96	72	120	48	24	24
26	104	78	130	52	26	26

Ordem das máquinas

1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7
1	2	3	4	5	6	7

Release date

6	8	11	1	1	11	14	2

Due date

539	788	531	606	451	111	553	604

Instância 4.2-1.3

Tempos de Processamento

10	40	30	50	20	10	10
10	40	30	50	20	10	10
10	40	30	50	20	10	10
10	40	30	50	20	10	10
10	40	30	50	20	10	10
10	40	30	50	20	10	10
10	40	30	50	20	10	10
10	40	30	50	20	10	10

Ordem das máquinas

4	5	7	1	2	6	3
7	3	2	5	4	6	1
7	3	2	6	5	4	1
3	5	1	6	2	4	7
3	4	1	2	6	5	7
3	7	1	5	4	6	2
5	4	6	2	1	3	7
5	6	4	3	1	2	7

Release date

10	8	10	6	5	16	1	25

Due date

247	250	260	231	231	266	252	260

Instância 4.2-1.4

Tempos de Processamento

10	40	30	50	20	10	10
10	40	30	50	20	10	10
30	120	90	150	60	30	30
10	40	30	50	20	10	10
6	24	18	30	12	6	6
10	40	30	50	20	10	10
20	80	60	100	40	20	20
4	16	12	20	8	4	4

Ordem das máquinas

1	2	6	4	5	3	7
3	5	1	4	2	7	6
5	2	4	3	1	7	6
2	1	3	5	4	7	6
1	5	4	3	2	6	7
1	3	2	6	7	4	5
2	1	7	4	6	5	3
1	5	4	3	2	7	6

Release date

5	16	1	25	6	8	11	1

Due date

231	266	752	260	141	271	531	103

Instância 4.2-1.5

Tempos de Processamento

24	96	72	120	48	24	24
30	120	90	150	60	30	30
20	80	60	100	40	20	20
24	96	72	120	48	24	24
20	80	60	100	40	20	20
4	16	12	20	8	4	4
24	96	72	120	48	24	24
26	104	78	130	52	26	26

Ordem das máquinas

7	3	2	5	4	6	1
2	1	5	7	3	6	4
3	5	1	7	2	6	4
4	6	7	1	5	2	3
7	6	4	3	5	2	1
4	3	2	1	6	5	7
4	7	5	1	3	6	2
4	3	2	1	6	5	7

Release date

6	8	11	1	1	11	14	2

Due date

539	788	531	606	451	111	553	604

Instância 4.2-1.6

Tempos de Processamento

24	96	72	120	48	24	24
30	120	90	150	60	30	30
20	80	60	100	40	20	20
24	96	72	120	48	24	24
20	80	60	100	40	20	20
4	16	12	20	8	4	4
24	96	72	120	48	24	24
26	104	78	130	52	26	26

Ordem das máquinas

1	2	4	3	6	5	7
1	3	2	6	5	4	7
3	5	1	7	2	6	4
4	6	7	1	5	2	3
7	2	3	5	4	6	1
1	5	4	3	2	7	6
1	6	3	5	4	2	7
4	3	2	1	5	7	6

Release date

6	8	11	1	1	11	14	2

Due date

539	788	531	606	451	111	553	604

Instância 4.2-2.1

Tempos de Processamento

10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40

Ordem das máquinas

1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12

Release date

24	15	21	8	0	25	14	23	1	8

Due date

565	589	549	571	476	578	494	606	531	516

Instância 4.2-2.2

Tempos de Processamento

6	24	18	30	12	6	6	24	24	12	30	24
2	8	6	10	4	2	2	8	8	4	10	8
8	32	24	40	16	8	8	32	32	16	40	32
26	104	78	130	52	26	26	104	104	52	130	104
2	8	6	10	4	2	2	8	8	4	10	8
28	112	84	140	56	28	28	112	112	56	140	112
8	32	24	40	16	8	8	32	32	16	40	32
2	8	6	10	4	2	2	8	8	4	10	8
4	16	12	20	8	4	4	16	16	8	20	16
6	24	18	30	12	6	6	24	24	12	30	24

Ordem das máquinas

1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12

Release date

14	22	12	21	2	16	1	23	14	9

Due date

364	126	477	1465	104	1618	460	146	241	329

Instância 4.2-2.3

Tempos de Processamento

10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40

Ordem das máquinas

11	3	2	7	8	9	4	5	6	12	1	10
8	4	7	2	6	5	3	1	10	9	12	11
9	8	6	12	7	3	5	2	1	11	10	4
6	4	8	2	10	1	12	3	11	5	9	7
6	7	5	8	3	1	2	4	10	9	12	11
2	1	9	5	4	11	8	7	3	12	6	10
7	6	4	3	9	2	1	12	5	11	10	8
3	12	1	5	4	10	8	7	11	6	9	2
10	3	2	6	8	4	12	5	11	1	9	7
3	4	1	2	10	8	11	6	12	5	7	9

Release date

24	15	21	8	0	25	14	23	1	8

Due date

565	589	549	571	476	578	494	606	531	516

Instância 4.2-2.4

Tempos de Processamento

6	24	18	30	12	6	6	24	24	12	30	24
10	40	30	50	20	10	10	40	40	20	50	40
20	80	60	100	40	20	20	80	80	40	100	80
4	16	12	20	8	4	4	16	16	8	20	16
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
26	104	78	130	52	26	26	104	104	52	130	104
10	40	30	50	20	10	10	40	40	20	50	40
10	40	30	50	20	10	10	40	40	20	50	40
6	24	18	30	12	6	6	24	24	12	30	24

Ordem das máquinas

5	11	6	10	1	3	7	9	8	4	2	12
6	10	5	7	3	1	4	8	12	2	11	9
11	2	3	5	4	10	9	12	7	6	1	8
10	2	9	4	8	7	6	5	3	1	12	11
8	12	6	5	4	3	10	1	9	7	11	2
9	2	3	7	6	5	4	10	1	8	12	11
7	8	3	4	5	6	1	2	10	9	11	12
2	1	3	5	4	11	12	9	8	10	6	7
9	6	3	4	5	2	8	7	1	10	12	11
1	5	3	10	2	6	7	8	9	4	12	11

Release date

25	14	23	1	8	14	22	12	21	2

Due date

362	494	1174	214	516	591	1284	592	583	306

Instância 4.2-2.5

Tempos de Processamento

6	24	18	30	12	6	6	24	24	12	30	24
2	8	6	10	4	2	2	8	8	4	10	8
8	32	24	40	16	8	8	32	32	16	40	32
26	104	78	130	52	26	26	104	104	52	130	104
2	8	6	10	4	2	2	8	8	4	10	8
28	112	84	140	56	28	28	112	112	56	140	112
8	32	24	40	16	8	8	32	32	16	40	32
2	8	6	10	4	2	2	8	8	4	10	8
4	16	12	20	8	4	4	16	16	8	20	16
6	24	18	30	12	6	6	24	24	12	30	24

Ordem das máquinas

5	9	6	8	1	3	10	4	2	7	12	11
6	9	10	7	12	1	4	11	2	3	8	5
2	1	7	5	4	10	3	11	12	6	8	9
4	7	8	1	9	10	2	3	5	6	12	11
6	3	2	10	7	1	5	12	11	4	9	8
2	1	8	7	6	5	4	3	10	9	12	11
9	3	2	12	6	5	11	10	1	8	7	4
11	10	4	3	7	9	5	12	6	2	1	8
9	4	5	2	3	11	8	7	1	12	6	10
10	8	6	12	11	3	9	2	7	1	5	4

Release date

14	22	12	21	2	16	1	23	14	9

Due date

362	494	1174	214	516	591	1284	592	583	306

Instância 4.2-2.6

Tempos de Processamento

6	24	18	30	12	6	6	24	24	12	30	24
2	8	6	10	4	2	2	8	8	4	10	8
8	32	24	40	16	8	8	32	32	16	40	32
26	104	78	130	52	26	26	104	104	52	130	104
2	8	6	10	4	2	2	8	8	4	10	8
28	112	84	140	56	28	28	112	112	56	140	112
8	32	24	40	16	8	8	32	32	16	40	32
2	8	6	10	4	2	2	8	8	4	10	8
4	16	12	20	8	4	4	16	16	8	20	16
6	24	18	30	12	6	6	24	24	12	30	24

Ordem das máquinas

5	2	3	7	1	9	4	10	6	8	11	12
6	9	10	7	5	1	4	8	2	3	11	12
2	1	3	8	6	5	12	4	9	11	10	7
1	7	6	8	11	3	2	4	12	10	5	9
6	3	2	10	7	1	5	8	9	4	11	12
2	1	8	7	5	11	4	3	10	9	6	12
1	2	3	5	4	6	8	7	12	11	10	9
11	10	4	3	5	9	8	7	6	2	1	12
1	2	5	4	3	10	7	9	8	6	11	12
1	2	3	4	5	7	6	8	9	10	12	11

Release date

14	22	12	21	2	16	1	23	14	9

Due date

364	126	477	1465	104	1618	460	146	241	329

Instância 4.2-3.1

Tempos de Processamento

10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30

Ordem das máquinas

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Release date

9	13	18	2	7	21	12	21	22	2	8	20	12	16	16	24	2	3	0	17

Due date

759	710	762	768	776	690	784	670	687	684	757	750	698	810	749	730	778	671	711	700

Instância 4.2-3.2

Tempos de Processamento

4	16	12	20	8	4	4	16	16	8	20	16	12	4	16	8	12
24	96	72	120	48	24	24	96	96	48	120	96	72	24	96	48	72
26	104	78	130	52	26	26	104	104	52	130	104	78	26	104	52	78
30	120	90	150	60	30	30	120	120	60	150	120	90	30	120	60	90
6	24	18	30	12	6	6	24	24	12	30	24	18	6	24	12	18
2	8	6	10	4	2	2	8	8	4	10	8	6	2	8	4	6
28	112	84	140	56	28	28	112	112	56	140	112	84	28	112	56	84
6	24	18	30	12	6	6	24	24	12	30	24	18	6	24	12	18
24	96	72	120	48	24	24	96	96	48	120	96	72	24	96	48	72
8	32	24	40	16	8	8	32	32	16	40	32	24	8	32	16	24
4	16	12	20	8	4	4	16	16	8	20	16	12	4	16	8	12
20	80	60	100	40	20	20	80	80	40	100	80	60	20	80	40	60
26	104	78	130	52	26	26	104	104	52	130	104	78	26	104	52	78
4	16	12	20	8	4	4	16	16	8	20	16	12	4	16	8	12
6	24	18	30	12	6	6	24	24	12	30	24	18	6	24	12	18
4	16	12	20	8	4	4	16	16	8	20	16	12	4	16	8	12
8	32	24	40	16	8	8	32	32	16	40	32	24	8	32	16	24
28	112	84	140	56	28	28	112	112	56	140	112	84	28	112	56	84
28	112	84	140	56	28	28	112	112	56	140	112	84	28	112	56	84
8	32	24	40	16	8	8	32	32	16	40	32	24	8	32	16	24

Ordem das máquinas

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Release date

18	9	17	22	22	4	22	1	3	7	19	14	8	25	15	10	24	5	13	6

Due date

285	1825	1834	2358	490	152	2091	419	1592	543	340	1571	1770	352	479	330	620	2121	1884	566

Instância 4.2-3.3

Tempos de Processamento

10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30

Ordem das máquinas

6	9	14	8	10	1	12	4	2	5	13	7	11	3	17	16	15
4	3	2	1	11	8	13	6	15	16	5	14	7	12	9	10	17
7	6	13	17	10	2	1	9	8	5	14	15	3	11	12	16	4
12	4	5	2	3	11	13	9	8	15	6	1	7	17	10	16	14
11	3	2	7	6	5	4	10	13	8	1	15	9	16	12	14	17
5	9	17	14	1	8	10	6	2	7	15	13	12	4	11	16	3
11	17	14	5	4	12	8	7	15	13	1	6	10	3	9	16	2
12	16	8	6	14	4	17	3	15	13	11	1	10	5	9	2	7
5	13	8	7	1	17	4	3	15	14	12	11	2	10	9	16	6
5	4	16	2	1	13	14	9	8	12	15	10	6	7	11	3	17
9	3	2	14	13	7	6	17	1	12	15	10	5	4	11	16	8
14	12	6	13	16	3	11	17	10	9	7	2	4	1	15	5	8
12	17	5	14	3	16	15	9	8	11	10	1	13	4	7	6	2
4	3	2	1	12	8	9	6	7	13	14	5	10	11	17	16	15
14	6	11	13	10	2	8	7	17	5	3	16	4	1	15	12	9
2	1	5	11	3	10	8	7	15	6	4	14	16	12	9	13	17
7	8	5	9	3	15	1	2	4	16	17	13	12	14	6	10	11
13	8	10	5	4	7	6	2	15	3	16	17	1	14	9	11	12
8	12	13	9	7	14	5	1	4	16	15	2	3	6	11	10	17
2	1	12	14	6	5	15	11	17	16	8	3	13	4	7	10	9

Release date

9	13	18	2	7	21	12	21	22	2	8	20	12	16	16	24	2	3	0	17

Due date

759	710	762	768	776	690	784	670	687	684	757	750	698	810	749	730	778	671	711	700

Instância 4.2-3.4

Tempos de Processamento

4	16	12	20	8	4	4	16	16	8	20	16	12	4	16	8	12
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
26	104	78	130	52	26	26	104	104	52	130	104	78	26	104	52	78
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
6	24	18	30	12	6	6	24	24	12	30	24	18	6	24	12	18
2	8	6	10	4	2	2	8	8	4	10	8	6	2	8	4	6
28	112	84	140	56	28	28	112	112	56	140	112	84	28	112	56	84
6	24	18	30	12	6	6	24	24	12	30	24	18	6	24	12	18
24	96	72	120	48	24	24	96	96	48	120	96	72	24	96	48	72
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
6	24	18	30	12	6	6	24	24	12	30	24	18	6	24	12	18
2	8	6	10	4	2	2	8	8	4	10	8	6	2	8	4	6
6	24	18	30	12	6	6	24	24	12	30	24	18	6	24	12	18
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
4	16	12	20	8	4	4	16	16	8	20	16	12	4	16	8	12
6	24	18	30	12	6	6	24	24	12	30	24	18	6	24	12	18
10	40	30	50	20	10	10	40	40	20	50	40	30	10	40	20	30
22	88	66	110	44	22	22	88	88	44	110	88	66	22	88	44	66

Ordem das máquinas

6	2	3	4	7	1	5	14	13	11	10	17	9	8	15	16	12
1	2	9	8	10	14	17	4	3	5	13	15	11	6	12	16	7
11	3	2	4	5	6	12	10	14	8	1	7	17	9	16	15	13
9	2	3	4	5	7	6	8	1	11	10	16	14	13	17	12	15
14	4	3	2	13	10	7	8	9	6	12	11	5	1	15	17	16
1	6	3	12	5	2	7	8	9	11	10	4	13	14	15	17	16
1	2	15	12	17	6	16	14	9	13	11	4	10	8	3	7	5
1	2	3	4	8	6	7	5	12	11	10	9	17	16	15	14	13
1	7	11	5	4	14	2	8	15	10	3	12	13	6	9	17	16
3	8	1	4	14	7	6	2	9	16	12	11	17	5	15	10	13
13	9	15	14	7	6	5	8	2	10	11	12	1	4	3	16	17
1	6	3	4	5	2	17	9	8	15	16	14	13	12	10	11	7
8	4	6	2	7	3	5	1	9	10	13	14	11	12	16	15	17
9	2	3	15	5	14	7	16	1	12	11	10	17	6	4	8	13
1	11	3	5	4	6	17	12	15	13	2	8	10	14	9	16	7
14	9	8	10	7	11	5	3	2	4	6	12	13	1	15	16	17
1	2	9	4	5	12	13	10	3	8	14	6	7	11	17	16	15
1	12	15	7	5	13	4	11	9	14	8	2	6	10	3	17	16
1	5	16	6	2	4	7	14	9	11	10	12	15	8	13	3	17
1	13	8	4	5	10	7	3	16	6	11	14	2	12	15	9	17

Release date

16	16	24	2	3	0	17	18	9	17	22	22	4	22	1	3	7	19	14	8

Due date

340	749	730	2017	671	711	430	155	2127	442	1893	794	445	180	419	666	277	495	797	1499

Instância 4.2-3.5

Tempos de Processamento

4	16	12	20	8	4	4	16	16	8	20	16	12	4	16	8	12
24	96	72	120	48	24	24	96	96	48	120	96	72	24	96	48	72
26	104	78	130	52	26	26	104	104	52	130	104	78	26	104	52	78
30	120	90	150	60	30	30	120	120	60	150	120	90	30	120	60	90
6	24	18	30	12	6	6	24	24	12	30	24	18	6	24	12	18
2	8	6	10	4	2	2	8	8	4	10	8	6	2	8	4	6
28	112	84	140	56	28	28	112	112	56	140	112	84	28	112	56	84
6	24	18	30	12	6	6	24	24	12	30	24	18	6	24	12	18
24	96	72	120	48	24	24	96	96	48	120	96	72	24	96	48	72
8	32	24	40	16	8	8	32	32	16	40	32	24	8	32	16	24
4	16	12	20	8	4	4	16	16	8	20	16	12	4	16	8	12
20	80	60	100	40	20	20	80	80	40	100	80	60	20	80	40	60
26	104	78	130	52	26	26	104	104	52	130	104	78	26	104	52	78
4	16	12	20	8	4	4	16	16	8	20	16	12	4	16	8	12
6	24	18	30	12	6	6	24	24	12	30	24	18	6	24	12	18
4	16	12	20	8	4	4	16	16	8	20	16	12	4	16	8	12
8	32	24	40	16	8	8	32	32	16	40	32	24	8	32	16	24
28	112	84	140	56	28	28	112	112	56	140	112	84	28	112	56	84
28	112	84	140	56	28	28	112	112	56	140	112	84	28	112	56	84
8	32	24	40	16	8	8	32	32	16	40	32	24	8	32	16	24

Ordem das máquinas

11	9	5	15	3	12	16	13	2	14	1	6	8	10	4	7	17
5	3	2	9	1	12	8	7	4	15	13	6	11	17	10	16	14
7	5	9	15	2	13	1	12	3	17	16	8	6	14	4	11	10
16	4	13	2	6	5	15	14	10	9	12	11	3	8	7	1	17
7	17	6	12	13	3	1	10	15	8	16	4	5	14	9	11	2
11	7	17	15	9	16	2	14	5	13	1	12	10	8	4	6	3
11	17	16	8	15	13	14	4	10	9	1	12	6	7	5	3	2
6	5	11	8	2	1	10	4	12	7	3	9	14	13	16	15	17
11	5	12	13	2	15	8	7	14	16	1	3	4	9	6	10	17
3	10	1	9	6	5	13	16	4	2	14	17	7	11	15	8	12
5	16	17	12	1	7	6	11	14	15	8	4	13	9	10	2	3
3	7	1	10	6	5	2	12	14	4	13	8	11	9	16	15	17
10	6	5	7	3	2	4	12	14	1	17	8	16	9	15	13	11
12	5	9	14	2	11	10	16	3	7	6	1	15	4	13	8	17
17	7	13	5	4	16	2	14	11	12	9	10	3	8	15	6	1
8	4	13	2	12	17	14	1	11	16	9	5	3	7	15	10	6
5	3	2	10	1	12	8	7	14	4	15	6	16	9	11	13	17
6	4	17	2	7	1	5	10	12	8	14	9	15	11	13	16	3
3	13	1	5	4	10	14	12	11	6	9	8	2	7	17	16	15
6	10	16	9	12	1	14	11	4	2	8	5	15	7	13	3	17

Release date

18	9	17	22	22	4	22	1	3	7	19	14	8	25	15	10	24	5	13	6

Due date

285	1825	1834	2358	490	152	2091	419	1592	543	340	1571	1770	352	479	330	620	2121	1884	566

Instância 4.2-3.6

Tempos de Processamento

4	16	12	20	8	4	4	16	16	8	20	16	12	4	16	8	12
24	96	72	120	48	24	24	96	96	48	120	96	72	24	96	48	72
26	104	78	130	52	26	26	104	104	52	130	104	78	26	104	52	78
30	120	90	150	60	30	30	120	120	60	150	120	90	30	120	60	90
6	24	18	30	12	6	6	24	24	12	30	24	18	6	24	12	18
2	8	6	10	4	2	2	8	8	4	10	8	6	2	8	4	6
28	112	84	140	56	28	28	112	112	56	140	112	84	28	112	56	84
6	24	18	30	12	6	6	24	24	12	30	24	18	6	24	12	18
24	96	72	120	48	24	24	96	96	48	120	96	72	24	96	48	72
8	32	24	40	16	8	8	32	32	16	40	32	24	8	32	16	24
4	16	12	20	8	4	4	16	16	8	20	16	12	4	16	8	12
20	80	60	100	40	20	20	80	80	40	100	80	60	20	80	40	60
26	104	78	130	52	26	26	104	104	52	130	104	78	26	104	52	78
4	16	12	20	8	4	4	16	16	8	20	16	12	4	16	8	12
6	24	18	30	12	6	6	24	24	12	30	24	18	6	24	12	18
4	16	12	20	8	4	4	16	16	8	20	16	12	4	16	8	12
8	32	24	40	16	8	8	32	32	16	40	32	24	8	32	16	24
28	112	84	140	56	28	28	112	112	56	140	112	84	28	112	56	84
28	112	84	140	56	28	28	112	112	56	140	112	84	28	112	56	84
8	32	24	40	16	8	8	32	32	16	40	32	24	8	32	16	24

Ordem das máquinas

11	9	5	15	3	6	14	8	2	13	1	16	10	7	4	12	17
5	2	17	9	1	7	6	13	4	10	12	11	8	14	15	16	3
7	5	9	15	2	13	1	12	3	10	11	8	6	14	4	16	17
16	4	13	2	6	5	15	14	10	9	11	17	3	8	7	1	12
7	2	3	4	8	6	1	5	14	10	13	12	11	9	17	16	15
1	2	3	5	4	6	7	8	12	10	11	9	17	16	15	14	13
1	2	17	16	5	15	7	8	9	10	11	13	12	14	6	4	3
1	6	5	17	3	2	14	13	11	15	9	16	8	7	10	12	4
11	2	13	4	5	6	15	9	8	14	1	12	3	10	7	17	16
3	2	1	13	6	5	17	11	9	14	8	16	4	10	15	12	7
1	5	3	4	2	6	7	8	9	17	13	12	11	14	15	16	10
1	4	15	2	17	16	9	12	7	13	14	8	10	11	3	6	5
10	2	6	5	4	3	9	11	7	1	8	15	13	17	12	16	14
12	2	13	9	5	11	7	15	4	10	6	1	3	16	8	14	17
17	7	13	5	4	6	2	8	15	12	14	10	3	11	9	16	1
8	4	13	2	5	6	7	1	9	10	11	15	3	14	12	16	17
5	3	2	10	1	12	7	14	16	4	17	6	15	8	13	9	11
1	12	8	4	6	5	10	3	9	7	13	2	11	17	16	15	14
1	13	3	5	4	6	11	8	9	16	7	15	2	14	12	10	17
6	2	15	4	10	1	7	14	9	5	11	13	12	8	3	17	16

Release date

18	9	17	22	22	4	22	1	3	7	19	14	8	25	15	10	24	5	13	6

Due date

285	1825	1834	2358	490	152	2091	419	1592	543	340	1571	1770	352	479	330	620	2121	1884	566

ANEXO 2 – FUNÇÃO GERAR_INSTANCIAS.M

```
function[JM,JO,RD,DD]=gerar_instancias(job,maq,Ent1,Ent2,tmin,tmax,tl,amp,
vm,rdmin,rdmax,amin,amax)

%Gerar matriz tempo de processamento(JM)
A=randi([tmin,tmax],1,maq); %Tempo de processamento em cada máquina
B=tl*ones(job,1); %Tamanho do lote de cada job

n=ceil(amp/vm);
for i=1:job
    if rand<Ent1
        x=1/n;
        for j=1:n
            prob(j)=j*x;
        end
        s=rand;
        for k=1:n
            if prob(k)>s;
                break
            end
        end
        if k<n/2
            B(i)=B(i)-k*vm;
        else
            B(i)=B(i)+k*vm;
        end
    else
        B(i)=B(i);
    end
end

for i=1:job
    for j=1:maq
        JM(i,j)=B(i)*A(j);
    end
end

%Gerar sequencia de processamentos(JO)
J=1:job;
M=1:maq;
for i=1:job
    for j=1:maq
        JO(i,j)=M(j);
    end
end

for i=1:job
    M1=M;
    for j=1:maq
        if JO(i,j)==j
            if rand<Ent2
                A=M1;
                A(find(M1(:)==j))=[];
                n=length(A);
                if n~=0
                    x=1/n;
                    for k=1:n
                        prob(k)=k*x;
```

```
                    end
                    s=rand;
                    for l=1:n
                        if prob(l)>s
                            break
                        end
                    end
                    JO(i,j)=A(l);
                    JO(i,A(l))=M(j);
                    M1(find(M1(:)==A(l)))=[];
                    M1(find(M1(:)==M(j)))=[];
                    prob=[];
                end
            else
                JO(i,j)=j;
                M1(find(M1(:)==j))=[];
            end
        end
    end
end

% Reordenar a sequência de processamento, colocando as máquinas com TP=0 em
%primeiro

for i=1:job
    for j=maq:-1:1
        if JM(i,j)==0
            ord=find(JO(i,:)==j);
            if ord>1
                aux=JO(i,1:ord-1);
                JO(i,1:ord)=[JO(i,ord) aux];
            end
        end
    end
end

%Gerar Release Date(RD)
RD=randi([rdmin,rdmax],1,job);

%Gerar Due Date
for i=1:job
    DD(i)=ceil((1+(randi([amin,amax],1))/100)*(RD(i)+sum(JM(i,:))));
end

%.....................................................................

%Organizar os dados no formato LISA
MO=zeros(job,maq);
for i=1:job
    for j=1:maq
        if JO(i,j)~=0
            MO(i,JO(i,j))=j;
        end
    end
end
Machine_Order=MO;
Processing_Times=JM;

end
```

ANEXO 3 – ALGORITMO SIMULATED ANNEALING ADAPTADO

```
clc
clear
close all

%---------------Paramentros caracteristicos do processo--------------------

job=20; %Numero de jobs
maq=17; %Numero de máquinas
Ent1=1; %Entropia dos tempos de processamento(JM)
Ent2=0; %Entropia da sequencia de processamento(JO)
tmin=1; %Tempo minimo de processo/unidade em cada maquina
tmax=5; %Tempo maximo de processo/unidade em cada maquina
tl=10; %Tamanho medio de lote de produção
amp=20; %Amplitude de variação do tamanho do lote de produção
vm=2; %Variação minima do lote de produção
rdmin=0; %Release date minimo
rdmax=25; %Release date maximo
amin=30;%Acresc. min no tempo de processamento para definição do due date
amax=60;%Acresc. max no tempo de processamento para definição do due date

%----------------Paramentros do Simulated Annealing------------------------

%Numero de amostras consideradas para estimar a temperatura inicial
num_amostra=100;
nt=5; %Regula o numero de transições de estado para T constante
nr=10; %Regula o numero de reconfigurações para T constante
alfa=0.9; %Taxa de queda da temperatura
tempo_exec=180; %Tempo de execução do programa
n_stop=5;

%---------------------------------------------------------------------------

%Gerar instancia
[JM,JO,RD,DD]=gerar_instancias(job,maq,Ent1,Ent2,tmin,tmax,tl,amp,vm,rdmin,
rdmax,amin,amax)

tic
[job,maq]=size(JM);
[job,ope]=size(JO);
cont1=0;
cont2=0;
for i=1:job
    for j=1:ope
        if JM(i,JO(i,j))~=0
            cont1=cont1+1;
            cont2=cont2+1;
            list(cont1,1)=i;%job
            list(cont1,2)=cont2;%operação
            list(cont1,3)=JO(i,j);%maquina
            list(cont1,4)=JM(i,JO(i,j));%tempo de processamento
        end
    end
      cont2=0;
end
[l,c]=size(list);
```

```
%Solução inicial
sol1=(list(randperm(l),1))';

%Calcular a temperatura inicial
T=estima_T0_1(num_amostra,list,job,maq,RD,DD);

%Calcular energia da solução inicial
E1=calc_energia_1(sol1,list,job,maq,RD,DD);

energia_min=E1;
sol_otima=sol1;

stop=0;
iter=0;
num_trans=0;%numero de transicoes de um estado para outro para T constante
num_reconf=0;%numero de reconfiguracoes para T constante

vetx=0;
vety=energia_min;

while toc<tempo_exec && stop<n_stop
    iter=iter+1;
    sol2=gera_sol_1(sol1);
    E2=calc_energia_1(sol2,list,job,maq,RD,DD);
    if E2<=energia_min
        energia_min=E2;
        sol_otima=sol2;
        vetx=[vetx iter];
        vety=[vety energia_min];
        gantt_1(sol_otima,list,job,maq,vetx,vety,iter,T,RD,DD)
    end
    dE=E2-E1;
    prob=exp(-dE/T); %Probabilidade de aceitação de soluções piores
    if prob>1 %E2<E1
        num_trans=num_trans+1;
    end
    if rand<prob %E2>E1
        sol1=sol2;
        E1=E2;
    end
    num_reconf=num_reconf+1;
    if num_trans>nt*job*maq | num_reconf>nr*job*maq
        T=alfa*T;
        if num_trans==0
            stop=stop+1;
        end
        num_trans=0;
        num_reconf=0;
    end
end
```

ANEXO 4 – FUNÇÃO GERAR_SOL_1.M

```
function sol_out=gera_sol_1(sol_in)

n=length(sol_in);
p1=ceil(rand*n);
p2=ceil(rand*n);
pos1=min(p1,p2);
pos2=max(p1,p2);
sol_out=sol_in;
x=rand;

if x<0.33
    sol_out(pos1:pos2)=sol_in(pos2:-1:pos1);
elseif 0.33<=x<0.66
    aux1=sol_in(1:pos1);
    n1=length(aux1);
    aux2=sol_in(pos1+1:pos2);
    n2=length(aux2);
    aux3=sol_in(pos2+1:n);
    n3=length(aux3);
    sol_out=[aux1(n1:-1:1) aux2(n2:-1:1) aux3(n3:-1:1)];
else
    aux=sol_out(pos1:pos2);
    sol_out1=[sol_out(1:pos1-1) sol_out(pos2+1:n)];
    pos3=ceil(rand*length(sol_out1));
    sol_out=[sol_out1(1:pos3) aux sol_out1(pos3+1:length(sol_out1))];
end
```

ANEXO 5 – FUNÇÃO GERAR_SOL_11

```
function sol_out=gera_sol_11(sol_in,list,RD,DD,job,maq)

%Calculo dos tempos de complementacao e das folgas
%JM: vetor que armazena o tempo de complementação de cada job
%I: matriz com os tempos de saída dos jobs de cada máquina

[JM,I]=calc_TempoParc(sol_in,list,job,maq,RD,DD);
sol_out=sol_in;
for i=1:job
    v=find(list(:,1)==i);
    n=length(v);
    Q(i)=n;%Total de operações que cada job realiza
    TC(i)=max(I(i,:)); %Tempos de complementacao
    F(i)=DD(i)-TC(i); %Folgas
end
J=1:job;
M=sortrows([J' Q' DD' TC' F']);
m=sum(M(:,2));
JF=([M(:,1) M(:,5)]);

for i=1:m
    [prob]=calc_prob(JF);
    l=length(prob);
    s=rand;
    for j=1:l
        if prob(j,2)>s
            break
        end
    end
    if sol_out(i)~=prob(j,1)
        v=find(sol_out(i+1:end)==prob(j,1)); %Partindo da posição seguinte
%da posição de modificação procurar o job sorteado
        sol_out(v(1)+i)=[]; %Apagar a primeira vez que aparecer(ele foi
%realocado na posição de modificação)
        aux1=sol_out(1:i-1);
        aux2=sol_out(i:end);
        sol_out=[aux1 prob(j,1) aux2];
    end
    M(sol_out(i),2)= M(sol_out(i),2)-1; %Reduzir em um o número de
%operações do job que foi sorteado
    MJ=calc_TempoParc(sol_out,list,job,maq,RD,DD);%Calcular os tempos de
%complementação gerado pela solução modificada
    M(:,4)=MJ;%Atualizar tempos de complementação
    M(:,5)=M(:,3)-M(:,4);%Atualizar as folgas
    JF=([M(:,1) M(:,5)]);
    v=find(M(:,2)==0);
    JF(v,:)=[]; %Armazenar só as folgas dos jobs que faltam ser realocados
end
end
```

ANEXO 6 – FUNÇÃO GERAR_SOL_2.M

```
function sol_out=gera_sol_2(prob,sol_in,job,maq)
if rand<prob
    ordL=randperm(job);
    for i=1:job
        sol_out(i,:)=sol_in(ordL(i),:);
    end
else
    for i=1:2*maq
        if i<=maq
            ordV(i)=0;
        else
            ordV(i)=1;
        end
    end
    p1=ceil(rand*2*maq);
    p2=ceil(rand*2*maq);
    pos1=min(p1,p2);
    pos2=max(p1,p2);
    aux1=ordV(1:pos1);
    aux2=ordV(pos1+1:pos2);
    aux3=ordV(pos2+1:end);
    ordV=[aux2 aux1 aux3];
    ordV=ordV(1:maq);
    sol_out=sol_in;
    for i=1:maq
        if ordV(i)==1
            sol_out(:,i)=randperm(job);
        end
    end
end
```

ANEXO 7 – FUNÇÃO CALC_ENERGIA_1.M

```
function [Lmax,Cmax,AT,JA,sch]=calc_energia_1(sol,list,job,maq,RD,DD)

n=length(sol);
MM=zeros(maq,1);%Armazena tempo de conclusão do ultimo job processado por
%cada máquina
MJ=RD; % Armazena tempo de conclusão de ca job
for i=1:n
    v=find(list(:,1)==sol(i)); %posições na matriz list do job que será
%agendado
    if MM(list(v(1),3))<=MJ(list(v(1),1)); %Verificar se a maquina está
%desocupada
        MM(list(v(1),3))=MJ(list(v(1),1))+list(v(1),4); %Somar tempo de
%processamento da operação no tempo de conclusão da predecessora
    else
        MM(list(v(1),3))=MM(list(v(1),3))+list(v(1),4); %Somar tempo de
%processamento da operação no tempo de conclusão do ultimo job na maquina
    end
    MJ(list(v(1),1))=MM(list(v(1),3));
    sch(list(v(1),1),list(v(1),3))=MM(list(v(1),3));%Armazena tempo de
%conclusão de cada job por máquina
    list(v(1),:)=0;
end
L=MJ-DD;
Lmax=max(L); % Atraso máximo
Cmax=max(MM);% Makespan
%Atraso total e número de jobs atrasados
AT=0;
JA=0;
for i=1:job
    if L(i)>0
        AT=AT+L(i);
        JA=JA+1;
    end
end
```

ANEXO 8 – FUNÇÃO CALC-ENERGIA_2.M

```
% Agendar todas as operações sem considerar capacidade dos recursos
for i=1:job
    for j=1:maq
        v=find(list(:,1)==i);
        n=length(v);
        for k=1:n
            if k==1
                sch(i,list(v(k),3))=RD(i)+list(v(k),4);
            elseif list(v(k),4)==0
                sch(i,list(v(k),3))=0;
            else
                sch(i,list(v(k),3))=sch(i,list(v(k-1),3))+list(v(k),4);
            end
        end
    end
end

% Reagendar as operações considerando a sequencia de entradas de cada
%máquina
for i=1:maq
    s=sol(:,i); %Sequencia de jobs na maq i
    for j=2:job
        if sch(s(j),i)~=0 %Verificar se o job s(j) realmente é processado
%pela maquina i
            for k=j-1:-1:1 %Encontrar o job antecessor ao job s(j)
%processado pela maq i
                if sch(s(k),i)~=0
                    if sch(s(j),i)-JM(s(j),i)<sch(s(k),i) %Verificar se a
%data de início do job s(j) é menor que a data fim do job antecessor s(k)
%na máquina i
                        sch(s(j),i)=sch(s(k),i)+JM(s(j),i); %Reagendar a
%data de início do job s(j) para a data final do job s(k) na máquina i
                    end
                    break
                end
            end
        end
    end
end
for i=1:maq
    for j=1:job
        if j==1 && i>1
            if sol(j,i)==sol(job,i-1) %Verifica se o primeiro job da maq i
%é igual ao último da máquina anterior
                j=j+1; %Seguir para o próximo
            end
        end
        m=JO(sol(j,i),:); %Sequência de máquinas do job i
        for k=2:maq
            if sch(sol(j,i),m(k))~=0 % Verificar se o job é realmente
%processado pela maq (m(k))
                for q=k-1:-1:1 % Encontrar a máquina antecessora a m(k)no
%processamento do job (sol_out(j,i))
                    if sch(sol(j,i),m(q))~=0
```

```
                        if sch(sol(j,i),m(k))-
JM(sol(j,i),m(k))<sch(sol(j,i),m(q)) %Se a data de início da maq seguinte
%(m(k)) for menor que a data fim da maquina antecessora (m(q))

sch(sol(j,i),m(k))=sch(sol(j,i),m(q))+JM(sol(j,i),m(k)); %Agendar o início
%do processamento na maq (m(k)) para a data final do proc. na maq (m(q))
                            v=sol(find(sol(:,m(k))==sol(j,i)):end,m(k));
%Job (sol_out(j,i)) mais os jobs que estão agendados para a maq m(k) depois
%dele
                            n=length(v);
                            if n>1
                                for l=2:n
                                    if sch(v(l),m(k))~=0 % Verificar se o
%job (v(l)) é realmente processado pela maq (m(k))
                                        for r=l-1:-1:1 % Encontrar o job
%antecessor ao job (v(l)) processado na maq m(k)
                                            if sch(v(r),m(k))~=0
                                                if sch(v(l),m(k))-
JM(v(l),m(k))<sch(v(r),m(k)) %Se a data de início do job seguinte (v(l))
%for menor que a data fim do job antecessor (v(r)) na maq m(k)

sch(v(l),m(k))=sch(v(r),m(k))+JM(v(l),m(k)); %Agendar o início do job
%(v(l)) para a data final do proc. do job (v(r))
                                                end
                                                break
                                            end
                                        end
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end
[sch1,JO1]=sort(sch,2);
%Verificar se a solução é viável (se sequência de jobs nas máquinas
%respeitam a ordem de processamento dos jobs)
if eq(JO,JO1)==ones(job,maq)
    for i=1:job
        FJ(i)=max(sch(i,:)); %Data final dos job
        L(i)=FJ(i)-DD(i); %Atraso/adiantamento dos job
    end
    Lmax=max(L); %Atraso máximo
    Cmax=max(FJ); %Makespan
    L(find(L(:)<0))=0 ;% Desconsiderar os adiantamentos
    AT=sum(L); % Atraso total
    JA=job-length(find(L==0)); %Número de jobs atrasados

else
    Lmax=1000000;
    Cmax=1000000000;
    AT=100000000000;
    JA=1000000000;
end
```

ANEXO 9 – FUNÇÃO ESTIMAT0.M

```
function T=estima_T0_1(num_amostra,list,job,maq,RD,DD)
[l,c]=size(list);
T=0;
for i=1:num_amostra
    sol=list(randperm(l),1);
    E1=calc_energia_1(sol,list,job,maq,RD,DD);
    sol=list(randperm(l),1);
    E2=calc_energia_1(sol,list,job,maq,RD,DD);
    dE=abs(E2-E1);
    if dE>T
        T=dE;
    end
end
```

ANEXO 10 – TERMO DE AUTENTICIDADE

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA

Termo de Declaração de Autenticidade de Autoria

Declaro, sob as penas da lei e para os devidos fins, junto à Universidade Federal de Juiz de Fora, que meu Trabalho de Conclusão de Curso do Curso de Graduação em Engenharia de Produção é original, de minha única e exclusiva autoria. E não se trata de cópia integral ou parcial de textos e trabalhos de autoria de outrem, seja em formato de papel, eletrônico, digital, áudio-visual ou qualquer outro meio.
Declaro ainda ter total conhecimento e compreensão do que é considerado plágio, não apenas a cópia integral do trabalho, mas também de parte dele, inclusive de artigos e/ou parágrafos, sem citação do autor ou de sua fonte.
Declaro, por fim, ter total conhecimento e compreensão das punições decorrentes da prática de plágio, através das sanções civis previstas na lei do direito autoral[1] e criminais previstas no Código Penal[2], além das cominações administrativas e acadêmicas que poderão resultar em reprovação no Trabalho de Conclusão de Curso.

Juiz de Fora, _____ de _____ de 20____.

_____	_____
NOME LEGÍVEL DO ALUNO (A)	Matrícula
_____	_____
ASSINATURA	CPF

[1] LEI N° 9.610, DE 19 DE FEVEREIRO DE 1998. Altera, atualiza e consolida a legislação sobre direitos autorais e dá outras providências.
[2] Art. 184. Violar direitos de autor e os que lhe são conexos: Pena – detenção, de 3 (três) meses a 1 (um) ano, ou multa.