UNIVERSIDADE FEDERAL DE GOIÁS
REGIONAL CATALÃO
UNIDADE ACADÊMICA ESPECIAL DE GESTÃO E NEGÓCIOS
PROGRAMA DE PÓS-GRADUAÇÃO *STRICTO SENSU*
MESTRADO EM GESTÃO ORGANIZACIONAL

FERNANDA NEIVA MESQUITA

META-HEURÍSTICA BASEADA EM *SIMULATED ANNEALING* PARA PROGRAMAÇÃO DA PRODUÇÃO EM MÁQUINAS PARALELAS COM DIFERENTES DATAS DE LIBERAÇÃO E TEMPOS DE *SETUP*

CATALÃO-GO
2015

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR AS TESES E DISSERTAÇÕES ELETRÔNICAS NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

1 1. Identificação do material bibliográfico: **[x] Dissertação** **[ ] Tese**

1 **2. Identificação da Tese ou Dissertação**
2

Nome completo do autor: Fernanda Neiva Mesquita

Título do trabalho: META-HEURÍSTICA BASEADA EM SIMULATED ANNEALING PARA PROGRAMAÇÃO DA PRODUÇÃO EM MÁQUINAS PARALELAS COM DIFERENTES DATAS DE LIBERAÇÃO E TEMPOS DE SETUP

3. Informações de acesso ao documento:

Concorda com a liberação total do documento [x] SIM [ ] NÃO[1]

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio do(s) arquivo(s) em formato digital PDF da tese ou dissertação.

_____ Data: 07 / 10 / 2016

Assinatura da autora

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. A extensão deste prazo suscita justificativa junto à coordenação do curso. Os dados do documento não serão disponibilizados durante o período de embargo.

FERNANDA NEIVA MESQUITA

META-HEURÍSTICA BASEADA EM *SIMULATED ANNEALING* PARA PROGRAMAÇÃO DA PRODUÇÃO EM MÁQUINAS PARALELAS COM DIFERENTES DATAS DE LIBERAÇÃO E TEMPOS DE *SETUP*

Dissertação apresentada ao Programa de Pós-Graduação em Gestão Organizacional da Universidade Federal de Goiás – Regional Catalão, área de concentração Gestão Organizacional, linha de pesquisa Inovação, Desenvolvimento e Tecnologia, como requisito parcial à obtenção do título de mestre em Gestão Organizacional.

Orientador: Prof. Dr. Hélio Yochihiro Fuchigami

CATALÃO-GO

2015

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Neiva Mesquita, Fernanda
META-HEURÍSTICA BASEADA EM SIMULATED ANNEALING PARA PROGRAMAÇÃO DA PRODUÇÃO EM MÁQUINAS PARALELAS COM DIFERENTES DATAS DE LIBERAÇÃO E TEMPOS DE SETUP [manuscrito] / Fernanda Neiva Mesquita. - 2015.
LXXX, 80 f.

Orientador: Prof. Dr. Hélio Yochihiro Fuchigami.
Dissertação (Mestrado) - Universidade Federal de Goiás, Unidade Acadêmica Especial de Gestão e Negócios, Catalão, Programa de Pós Graduação em Gestão Organizacional (profissional), Catalão, 2015.
Bibliografia. Apêndice.
Inclui siglas, abreviaturas, símbolos, gráfico, tabelas, algoritmos, lista de tabelas.

1. Máquinas Paralelas. 2. Tempos de Setup Independentes da Sequência. 3. Datas de Liberação. 4. Simulated Annealing. I. Yochihiro Fuchigami, Hélio, orient. II. Título.

CDU 005

SERVIÇO PÚBLICO FEDERAL
UNIVERSIDADE FEDERAL DE GOIÁS
REGIONAL CATALÃO
MESTRADO PROFISSIONAL EM GESTÃO ORGANIZACIONAL

ATA DE SESSÃO PÚBLICA DE EXAME DE DEFESA DA DISSERTAÇÃO DO MESTRADO PROFISSIONAL NO PROGRAMA DE PÓS-GRADUAÇÃO *STRICTO SENSU* EM GESTÃO ORGANIZACIONAL DA UNIVERSIDADE FEDERAL DE GOIÁS.

No dia dezessete (17) de dezembro de 2015, às 08:00 horas, na sala 108, Prédio Administrativo, Regional Catalão da Universidade Federal de Goiás, **FERNANDA NEIVA MESQUITA**, discente do Programa de Pós-Graduação *Stricto Sensu* em Gestão Organizacional (52001016061P6) da Universidade Federal de Goiás, expôs, em Sessão Pública o exame de defesa da dissertação intitulado **META-HEURÍSTICA BASEADA EM SIMULATED ANNELING PARA PROGRAMAÇÃO DA PRODUÇÃO EM MÁQUINAS PARALELAS COM DIFERENTES DATAS DE LIBERAÇÃO E TEMPOS DE SETUP**, para Comissão de Avaliação composta pelos (as) docentes: **Dr. Hélio Yochihiro Fuchigami** (Programa de Pós-Graduação em Gestão Organizacional/Universidade Federal de Goiás, Presidente da Comissão), **Dr. Donaldo Mark Santee** (Programa de Pós-Graduação em Modelagem e Otimização/UFG, Membro Convidado Externo) e **Dr. José Waldo Martínez Espinosa** (Programa de Pós-Graduação em Gestão Organizacional/Universidade Federal de Goiás, Membro Convidado Interno). O trabalho da Comissão de Avaliação foi conduzido pelo docente Presidente que, inicialmente, após apresentar os docentes integrantes da Comissão, concedeu 30 minutos o (a) discente candidato (a) para que este (a) expusesse o trabalho. Após a exposição, o docente Presidente concedeu a palavra a cada membro convidado da Comissão para que estes argüissem o (a) discente candidato (a). Após o encerramento das argüições, a Comissão de Avaliação do trabalho de defesa avaliou a dissertação e o desempenho do (a) discente candidato (a) na exposição, considerando a trajetória deste no curso de mestrado profissional. Como resultado da avaliação, a Comissão de Avaliação deliberou pela:

Aprovação do trabalho de defesa [X]

A Comissão de Avaliação declara o (a) discente candidato (a) APROVADO NO EXAME DE DEFESA PÚBLICA. A Comissão de Avaliação pode sugerir alterações de forma e/ou conteúdo consideradas aceitáveis, as correções, quando identificadas, devem ser realizadas no prazo máximo de 30 dias contados a partir do recebimento da Ata de Defesa. As alterações deverão ser indicadas no Anexo ao presente documento e/ou podem constar na versão lida pelo membro da Comissão de Avaliação para a sessão de defesa do trabalho de dissertação. Neste caso, a versão lida corrigida deverá ser entregue ao (à) discente candidato (a) no final da sessão

Reprovação do trabalho de defesa [ ]

De acordo com a Resolução – CEPEC N° 1109 é previsto a reprovação quando a Comissão de Avaliação determina que o trabalho apresentado não satisfaz as condições mínimas para ser considerado projeto de mestrado válido, em condições de se desenvolver um trabalho de conclusão de mestrado.

A Comissão de Avaliação:

Dr. Hélio Yochihiro Fuchigami
Membro Presidente
Universidade Federal de Goiás

Dr. Donald Mark Santee
Membro Convidado Externo
Universidade Federal de Goiás

Dr. José Waldo Martínez Espinosa
Membro Convidado Interno
Universidade Federal de Goiás

Fernanda Neiva Mesquita
Discente Candidato (a)
Matrícula: 2013-1575

Para uso da Coordenação/Secretaria do PPGGO

Prof. Dr. Vagner Rosalem
Coordenador do Mestrado Profissional no Programa de Pós-Graduação Stricto Sensu em Gestão Organizacional
Universidade Federal de Goiás

Prof. Dr. Geraldo Sadoyama Leal
Vice-Coordenador do Mestrado Profissional no Programa de Pós-Graduação Stricto Sensu em Gestão Organizacional
Universidade Federal de Goiás
Prof. Dr. Geraldo Sadoyama Leal

Observações:

Visto Secretaria:
EX. Defesa. n°.18/2015

Catalão, 15 / 12 /2015

A Deus

que me propiciou o conhecimento

para entender melhor minha missão de vida e de fé.

Agradecimentos

Ao meu querido orientador Prof. Dr. Hélio Yochihiro Fuchigami, como foi dolorosa e ao mesmo tempo prazerosa nossa convivência. Sem suas contribuições e todo o conhecimento/direcionamento esse trabalho não teria nunca passado de um doce sonho. Obrigada por acreditar em mim.

A meus queridos pais Silvio e Lourdes, que sempre estiveram comigo, me apoiando e me dado carinho. Aos meus irmãos: José Neto, meu exemplo de pesquisador, meu irmão Silvio Júnior (*in memoria*) que com grande dor nos deixou tão jovem e ao meu irmãozinho Igor**,** o prodígio da família, que sempre me deu força e amor. E ao meu pequeno sobrinho Matheus que veio para trazer alegria e bênção à nossa família.

Ao meu amado marido Neomar que não tenho palavras que poderia descrever o quanto me ajudou, sempre sendo meu companheiro, amigo e confidente. Sem você essa conquista nunca seria a mesma, te amo.

A empresa TriSolo que compreendeu minhas ausências para cursar as disciplinas do mestrado e toda sua equipe: Sebastião, Marcia, Luciene, Leonardo, Lindomar, Ana Carla, André, Raquel, Lucas, Thiago, Isabela, Bauer, Alexandre, Gnainna e demais amigos.

Ao programa de pós-graduação em Gestão Organizacional e todo seu corpo docente pelo conhecimento repassado. Principalmente a Universidade Federal de Goiás – Regional de Catalão pela oportunidade de cursar o programa.

RESUMO

Mesquita, F. N. (2015). *Meta-Heurística Baseada Em Simulated Annealing para Programação da Produção em Máquinas Paralelas com Diferentes Datas de Liberação e Tempos de Setup*. Dissertação de Mestrado, Programa de Pós-Graduação em Gestão Organizacional, Universidade Federal de Goiás, Regional Catalão, Catalão, GO, Brasil.

Este estudo trata de problemas de máquinas paralelas com tempos de *setup* independentes, diferentes datas de liberação e minimização do *makespan*. Qualquer processo produtivo requer um gerenciamento eficaz por meio do Planejamento e Controle da Produção (PCP). Esta atividade inclui a programação da produção, ou seja, a alocação de recursos para execução de tarefas em uma base de tempo. A atividade de programação é uma das tarefas mais complexas no gerenciamento da produção, pois a necessidade de lidar com diversos tipos diferentes de recursos e atividades simultâneas. Além disso, o número de soluções cresce exponencialmente em várias dimensões, de acordo com a quantidade de tarefas, operações ou máquinas, gerando assim uma natureza combinatória ao problema. No ambiente tratado neste trabalho, cada tarefa tem o mesmo tempo de processamento em qualquer máquina. Considerando a restrição de tempos de *setup* independente apenas da tarefa que espera por processamento e a presença de datas de liberação diferentes de zero, características muito práticas nas indústrias. Como não foram encontrados na literatura trabalho que tratasse desse ambiente de trabalho, ainda menos que utilizasse a meta-heurística *Simulated Annealing*, então foi desenvolvido o método para o problema, juntamente com a solução inicial os respectivos esquemas de perturbação e a definição de limitantes inferiores para o *makespan*. Ao resultados obtidos na experimentação computacional foram analisados e discutidos com base no desvio relativo percentual em relação a solução inicial e aos limitantes inferiores, o tempo médio de computação e a ferramenta de análise estatística ANOVA.

Palavras-chave: máquinas paralelas, tempos de *setup* independentes, datas de liberação, *Simulated Annealing.*

ABSTRACT

Mesquita, F. N. (2015). *Meta-Heuristics Based On Simulated Annealing for Production Scheduling on Parallel Machines with different dates of release and Setup Times*. Doctoral Dissertation, Programa de Pós-Graduação em Gestão Organizacional, Universidade Federal de Goiás, Regional Catalão, Catalão, GO, Brasil

This study deals with problems of parallel machines with independent setup times, different dates of release and minimizing the makespan. Any production process requires effective management by the Production Planning and Control (PCP). This activity includes the planning of production, so the allocation of resources for task execution on a time basis. The programming activity is one of the most complex tasks in the management of production because the need to deal with several different types of resources and concurrent activities. Furthermore, the number of solutions grows exponentially in several dimensions, according to the number of tasks, operations or machines, thereby generating a combinatorial nature of the problem. The environment treated in this work each task has the same processing time on any machine. Considering only the restriction independently of the task setup time waiting for processing and the presence of release dates different from zero very practical characteristics in industries. As were found in the literature work that deals of this work environment, even less that used the meta-heuristic Simulated Annealing, so we developed the method to the problem, along with the initial solution their disturbance schemes and the setting of lower bounds for the makespan. The results obtained from the computational experiments were analyzed and discussed based on the relative percentage deviation from the initial solution and lower bounds, the average computation time and statistical analysis tool ANOVA.

Keywords: parallel machines, independent setup times, release dates, Simulated Annealing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACO	colônia de formigas
LB	limitante inferior (*lower bound*)
LB_{ruim}	limitante inferior ruim
LPT	ordena pelo maior tempo de processamento (p_j)
PCP	planejamento e controle da produção
PO	pesquisa operacional
PS	esquema de perturbação *(perturbation schemes)*
RPD	desvio relativo percentual
SA	*Simulated Annealing*
SI	solução inicial
VNS	busca na vizinhança variável

LISTA DE SÍMBOLOS

b	número de lotes (*batches*)
C_j	data de término das tarefas J_j
C_{max}	data de término máxima das tarefas (*makespan*)
C^*_{max}	*makespan* ótimo
$C_{max}(S)$	*makespan* da solução *S*
$C_{max}(S^*)$	*makespan* da solução S^*
$C_{max}(S')$	*makespan* da solução S'
C_{max}^{heur}	makespan fornecido pelo algoritmo *Simulated annealing*
J	conjunto de *n* tarefas $\{J_1, J_2, \dots, J_n\}$
J_j	tarefa de índice *j*
J_u	última tarefa da máquina de maior carga
L_{max}	maior desvio de pontualidade (*lateness*) das tarefas
m	número de máquinas paralelas idênticas
M_k	conjunto de máquinas no estágio *k*
max C_j	maior data de término das tarefas J_j
min r_j,	menor data de liberação das tarefas J_j
n	número de tarefas
p_j	tempo de processamento da tarefa J_j
p_{jk}	tempo de processamento distinto em cada máquina M_k
p_u	tempo de processamento da última tarefa da máquina de maior carga
Pm	ambiente de produção com *m* máquinas paralelas idênticas
$p_{[j]}$	representa o tempo de processamento da *j*-ésima tarefa da sequência.
$prec$	presença de restrições de precedência geral
$prmp$	operações podem ser interrompidas (*preemption*)
r	fator de resfriamento
r_j	data de liberação da tarefa J_j
s_j	presença de tempos de *setup* independente da sequência
s_{ij}	presença de tempos de *setup* dependente da sequência
s_u	tempo de *setup* da última tarefa da máquina de maior carga
S	solução inicial
S^*	melhor solução encontrada

S'	nova solução da vizinhança de S
T_i	temperatura inicial
T_f	temperatura final
ω_k	fator de proporcionalidade da máquina M_k

SUMÁRIO

1 INTRODUÇÃO

1.1 Considerações iniciais

A matemática, muito mais que uma mera ciência dos números, é uma das formas de expressão da cultura humana, é uma forma de comunicação, significa essencialmente a arte ou a técnica de compreender – do grego, "matema" significa compreensão e "tica", arte ou técnica. Assim, além da aritmética das necessidades cotidianas, a matemática é a chave para a compreensão do mundo físico. Neste contexto, a modelagem matemática se insere na arte de transformar situações da realidade em problemas matemáticos, cujas soluções devem apresentar linguagem simples e de fácil compreensão (BASSANEZI, 2010).

A matemática tem grande influência na humanidade, pois presta sua contribuição a diversos campos do saber. É utilizada para resolver diferentes problemas, que – em sua maioria – abstêm-se de clareza e objetividade o que os tornam de difícil interpretação e resolução. Com isso, constrói-se um modelo esta representação simplificada do problema real, considerando as variáveis que influenciaram a problemática, para então resolvê-lo; e necessariamente a solução deve adequar-se ao problema original.

A Pesquisa Operacional (PO) é uma área de estudo multidisciplinar que se baseia em modelos matemáticos, algoritmos e ferramentas computacionais para a análise de problemas práticos e complexos. Esta ciência evidência a tomada de decisões e resoluções de problemas reais, geralmente sob condições de recursos escassos (PINEDO, 2012). Existem diversas técnicas clássicas de resolução, tais como: método *branch-and-bound*, simplex, relaxação lagrangeana, métodos heurísticos, dentre outros. Pode ser muito bem aplicada em sistemas logísticos de planejamento e controle da produção (CORREA *et al*., 2006)

O Planejamento e Controle da Produção (PCP) é uma importante área de estudo que possui pontos comuns à Pesquisa Operacional; este segmento é responsável pelo planejamento, execução e controle das atividades nos sistemas produtivos, ou seja, desenvolve os planos que guiarão a produção e os métodos de controle (ZACCARELLI, 1987).

Na visão de Martins (2010), o objetivo principal do PCP é comandar o processo produtivo de manufatura e/ou empresas de serviços, administrar o fluxo de materiais e

informações, bem como controlar o desempenho do sistema. Isso exige um gerenciamento dinâmico e integrado às demais áreas funcionais da empresa.

No ponto de encontro destas duas áreas – a PO e o PCP – e muitas vezes utilizando ferramentas como a programação matemática, existem os problemas de Programação da Produção (ou *Scheduling*)*;* área que pode-se definir, genericamente, como: a alocação de recursos disponíveis para a execução de tarefas em um horizonte de tempo; sendo assim, uma importante ferramenta de decisão nos sistemas de controle da produção (BAKER, 1974).

A programação da produção corrobora para a boa ordenação das tarefas que serão executadas; para isso, considera uma fração de tempo e a quantidade de máquinas alocadas, ou seja, trata-se da determinação do local e período de execução de cada operação, que otimizará o processo produtivo (LAWLER, 1989).

De acordo com Baker (1974), o problema de programação da produção requer decisões de sequenciamento e alocação de recursos, como: ferramentas, equipamentos, máquinas, células de produção, operadores, computadores entre outros.

Uma ferramenta na programação da produção é o Gráfico de Gantt, inventado por H. L. Gantt em 1917 e que representa graficamente o tempo como uma barra. Os momentos de inicio e fim das atividades podem ser indicados no gráfico e algumas vezes o processo real do trabalho também é indicado. As vantagens dos gráficos de Gantt é que eles proporcionam uma representação visual simples do que ocorre em cada operação (SLACK *et al.,* 1999).

1.2 Descrição do problema de pesquisa

Neste contexto, serão adotadas algumas definições clássicas da programação da produção. As operações são as atividades em si a serem realizadas; uma tarefa é um conjunto de operações inter-relacionadas por restrições de precedência derivadas de limitações tecnológicas. Já a máquina é o equipamento, dispositivo capaz de executar uma operação e o tempo gasto para execução da operação é o tempo de processamento (HAX e CANDEA, 1984).

As restrições tecnológicas são determinadas principalmente pelo padrão de fluxo das tarefas nas máquinas (MacCARTHY e LI, 1993), isso nos leva aos seguintes problemas clássicos de fluxos:

- Máquina única: apenas uma máquina disponível para o processamento;
- Máquinas paralelas: várias máquinas disponíveis para o processamento – cada tarefa pode ser processada em qualquer máquina, porém em apenas uma;
- *Flow shop*: as tarefas possuem um fluxo linear de processamento passando por várias

máquinas – cada máquina é um estágio de produção;

- *Flow shop* híbrido (*flexible flow shop* ou *flow shop* flexível): os estágios de produção podem conter várias máquinas disponíveis;
- *Job shop*: cada tarefa tem seu fluxo individual ou rota específica nas máquinas;
- *Job shop flexível:* existem estágios com duas ou mais máquinas disponíveis – mas a tarefa é processada por somente uma máquina em cada estágio;
- *Open shop:* semelhante ao *job shop*, porém as tarefas não possuem uma rota específica conhecida previamente, podendo inclusive não passar por uma ou mais máquinas; as tarefas podem ser processadas em qualquer ordem.

Esta pesquisa aborda o problema de máquinas paralelas, caracterizado pela existência de um único estágio de produção formado por vários recursos disponíveis, os quais podem processar igualmente qualquer tarefa requerida. Em geral, este ambiente pode conter máquinas denominadas idênticas, uniformes (ou proporcionais) ou não relacionadas, definidas a seguir.

No problema de máquinas paralelas idênticas, cada tarefa tem o mesmo tempo de processamento em qualquer máquina, ou seja, o tempo de processamento p_j da tarefa J_j independe da máquina M_k em que é executada.

Além deste tipo, existem também as máquinas paralelas uniformes ou proporcionais em que o tempo de processamento das tarefas, nas várias máquinas, se diferem por um fator multiplicativo. Por exemplo, isto ocorre quando em uma máquina o tempo de processamento de uma tarefa leva o dobro do tempo necessário em outra máquina.

O problema de máquinas paralelas não relacionadas ocorre quando os recursos produtivos disponíveis em paralelo possuem configurações bem variadas, por exemplo, com a existência de furadeiras novas e usadas, ou com modelos distintos, operando em conjunto. Pode ainda haver recursos em paralelo com funções diferentes, como a existência de furadeiras, tornos e lixadeiras, porém neste caso, a tarefa deve ser processada por um subconjunto específico de máquinas do estágio (por exemplo, apenas as lixadeiras).

Neste caso, os tempos de processamento p_j da tarefa J_j em quaisquer duas máquinas M_q e M_k seriam representados por:

$$p_{jq} = \omega_k p_{jk} \qquad (1.1)$$

Onde:

ω_k é o fator de proporcionalidade da máquina M_k em relação à máquina M_q.

Assim, todas as tarefas teriam esta mesma proporção nos tempos de processamento destas duas máquinas. No exemplo citado, ficaria $p_{j1} = 2p_{j2}$, indicando que o tempo de processamento da tarefa J_j na máquina M_1 é o dobro do tempo da tarefa J_j na máquina M_2.

O terceiro tipo de problema é o de máquinas paralelas não relacionadas, em cada máquina os tempos de processamento das tarefas são completamente diferentes, sem nenhum fator de proporcionalidade entre eles, ou seja, cada tarefa J_j tem o seu tempo p_{jk} distinto em cada máquina M_k. Ou seja, $p_{j1} \neq p_{j2} \neq ... \neq p_{jm}$.

O ambiente produtivo de máquinas paralelas é frequentemente encontrado nas indústrias, ou mesmo em versões híbridas em meio a processos em fluxo. Um exemplo típico de máquinas paralelas está na indústria automobilística. Podem haver postos de trabalho em meio à linha de produção, em que são adicionadas novas máquinas ou equipamentos para ampliar a capacidade produtiva.

Estes novos recursos possivelmente terão velocidades de processamento superiores aos já existentes, estabelecendo uma razão (ou relação) entre os tempos dessas duas categorias de equipamentos. Este caso configura um exemplo de máquinas paralelas uniformes. Se existe um conjunto independente de tarefas (ou atividades) que podem ser executadas em qualquer uma destas máquinas com o objetivo de se otimizar o tempo total de produção, tem-se um problema de minimização do *makespan* (SENTHILKUMAR e NARAYANAN, 2011).

O presente estudo propõe a minimização do *makespan*, que é a duração total da programação. Em qualquer sistema produtivo, a redução do tempo total necessário para a execução de todas as tarefas acarretará também a maximização da utilização dos recursos e a redução do custo de produção (LAWLER, 1989).

Dentre as restrições da produção práticas que devem ser consideradas estão os tempos de *setup* separado dos tempos de processamento, que constituem os tempos despendidos na preparação das máquinas que receberão as tarefas. Algumas operações típicas de *setup* são a obtenção de ferramentas e materiais, processos de limpeza e esterilização, lubrificações, ajustes, posicionamentos, aquecimentos (ALLAHVERDI *et al.*, 1999)

Nos problemas típicos de programação da produção, os tempos de *setup* podem ser dependentes ou independentes da sequência de execução das tarefas. Quando o *setup* depende apenas da tarefa que espera por processamento, ele é considerado independente e quando o *setup* também depende da tarefa que foi processada anteriormente na máquina é considerado dependente (FUCHIGAMI e MOCCELLIN, 2009).

O tempo de *setup* independente da sequência para a tarefa J_j é representado por s_j, enquanto o tempo de *setup* dependente da sequência para a tarefa J_j tendo processado

anteriormente a tarefa J_i é denotado por s_{ij}. Assim, requer os dados do problema com *setup* independente em forma vetorial desses tempos, enquanto que para o *setup* dependente é necessário uma matriz de *setup* (ALLAHVERDI *et al.*, 1999).

Outra classificação possível para os tempos de *setup* é quanto a sua antecipação em relação à liberação da tarefa. Os *setups* antecipados, também chamados separados (ou *detached*), são aquelas atividades que podem ser iniciadas antes da chegada da tarefa à máquina, ou seja, antes da sua liberação, como por exemplo, operações de limpeza e lubrificação das máquinas. Por outro lado, os *setup*s não antecipados, denominados ainda não separáveis (ou *attached*), requerem que a tarefa já esteja presente na máquina para ser realizada, como nos casos de ajustes, posicionamentos e fixação da tarefa na máquina.

Tanto os *setups* dependentes como os independentes, os antecipados e não antecipados dependem da natureza das operações e do ambiente produtivo. Este trabalho trata do problema com *setup* antecipado e independente da sequência.

Mais uma restrição realista que será considerada neste estudo é a presença de datas de liberação (*release dates*) diferentes de zero. Cada tarefa J_j tem sua data de liberação denotada por r_j. São os instantes que demarcam que cada tarefa pode ser executada ou as chegadas das tarefas nas máquinas (PINEDO, 2012). Esta é uma característica muito prática, pois nas indústrias dificilmente todas as tarefas chegarão simultaneamente. A consideração das datas de liberação diferentes de zero conferem um caráter mais dinâmico ao problema.

De acordo com Souza e Moccellin (2000), os métodos de solução para problemas de programação da produção podem ser de dois tipos: métodos de solução ótima ou exata, que geram uma programação ótima de acordo com o critério de desempenho adotado, tais como técnicas de enumeração do tipo *Branch-and-Bound* e Programação Linear Inteira; e métodos heurísticos, que buscam o melhor caminho dentre os existentes, sem precisar testar todas as opções possíveis ou chegar à solução ótima, demandando um tempo computacional geralmente bem menor.

Os métodos heurísticos podem ainda ser subdivididos em meta-heurísticas, como as clássicas Busca Tabu, *Simulated Annealing* e Algoritmo Genético; heurísticas construtivas, quando se adotam critérios para construir a solução iterativamente, e heurísticas melhorativas, quando se parte de uma solução inicial completa (geralmente gerada aleatoriamente) e se empreendem operações na tentativa de melhorá-la (FUCHIGAMI, 2013).

É bastante comum se optar pela resolução do problema por meio de um método heurístico porque a pequena diferença na qualidade da solução em relação à solução ótima

não justifica o enorme esforço computacional exigido pelos métodos exatos, embora essa restrição tenha se atenuado com os avanços da tecnologia.

Uma pesquisa bibliográfica detalhada, Alcan e Basligil (2010) realizaram apontamentos de inúmeras meta-heurísticas aplicadas ao problema de máquinas paralelas. Os autores demonstraram que em geral a meta-heurística *Simulated Annealing* produz a melhor solução em menor tempo, quando comparada a outros métodos probabilísticos. Ao mesmo tempo, verificaram que relativamente às outras meta-heurísticas, há poucos trabalhos que utilizam o *Simulated Annealing* no problema de máquinas paralelas. Este é um dos motivos pelos quais se optou pela utilização deste método na resolução do problema tratado nesta pesquisa.

Para mensurar o resultado do algoritmo *Simulated Annealing* proposto, foram desenvolvidos limitantes inferiores (*lower bounds*) para o *makespan*. Conforme será detalhado nos próximos capítulos, os limitantes inferiores são valores de referência que são calculados com os dados do problema sem a necessidade de se estabelecer uma solução (neste caso, uma sequência de tarefas). Por definição, a expressão do cálculo do limitante inferior deve garantir que o seu valor seja no máximo igual ao *makespan* ótimo ($LB \leq C^*_{max}$).

Nesta pesquisa, formaliza-se então a seguinte problema: a meta-heurística *Simulated Annealing* é eficaz (em relação à qualidade da solução) e eficiente (em relação ao tempo computacional) na solução do problema de máquinas paralelas idênticas com datas de liberação e *setup* independente?

1.3 Objetivos da pesquisa e estrutura do trabalho

Os objetivos gerais deste estudo são:

- Investigar a estrutura do problema de máquinas paralelas idênticas com datas de liberação diferentes de zero e tempos de *setup* antecipados e independentes da sequência de tarefas;
- Propor o algoritmo *Simulated Annealing;*
- Comparar a eficácia da solução com limitantes inferiores adaptados da literatura.

Os objetivos específicos desta pesquisa englobam:

- Examinar minuciosamente a literatura específica a fim de verificar o estado da arte;
- Identificar as propriedades estruturais do problema;

- Propor um método de solução inicial para a meta-heurística e os respectivos esquemas de perturbação;
- Definir e implementar os limitantes inferiores para o *makespan*;
- Analisar estatisticamente os resultados obtidos.

O texto foi estruturado da seguinte forma: o primeiro capítulo apresenta uma introdução sobre programação da produção enfatizando o problema tratado nessa pesquisa e seus objetivos. O segundo capítulo traz a notação do problema de programação de operações em máquinas, descrevendo de forma genérica o método de solução *Simulated Annealing* e uma análise da revisão bibliográfica do ambiente de máquinas paralelas idênticas com minimização do *makespan.* No terceiro capítulo trata de uma análise da estrutura do problema, propondo o algoritmo de solução inicial, esquemas de perturbações, algoritmo *Simulated Annealing* proposto e os limitantes inferiores. No quarto capítulo apresenta a delimitação da experimentação computacional, análise dos resultados das perturbações em relação ao limitante inferior e também em relação a solução inicial. As conclusões do trabalho são descritas no quinto capítulo.

2 REFERENCIAL TEÓRICO

2.1 Programação da produção

Um problema de produção é denotado na literatura específica por meio de uma notação matemática de três campos. Pinedo (2012) organiza esses parâmetros em um trio $\alpha|\beta|\gamma$ que auxilia a classificação de um problema de sequenciamento. Esses campos determinam o problema, descrevendo "α" como o ambiente de produção com o número de máquinas ou estágios, β fornecendo detalhes das limitações dos recursos e tarefas, e γ que contém o objetivo a ser otimizado, ou seja, a medida de desempenho.

De acordo com essa conhecida notação de três campos, o problema a ser tratado é representado da seguinte forma $P_m \left|s_j, r_j\right|C_{max}$, onde "$P_m$" indica o ambiente de produção com *m* máquinas paralelas idênticas, as restrições do problema denotado pelo "s_j" que é a presença de tempos de *setup* independentes da sequência e "r_j" indicando a presença de diferentes datas de liberação das tarefas. O campo "C_{max}" define a medida de desempenho considerada, ou seja, a duração total da programação (*makespan*).

Diante desse contexto, é necessário ressaltar alguns pressupostos do problema tratado:

a) Cada máquina pode processar apenas uma tarefa de cada vez;

b) As máquinas estão sempre disponíveis para processamento, ou seja, não há quebra de máquinas ou paradas não programadas;

c) O problema possui um único estágio de produção e todas as máquinas pertencem a ele, caracterizando a disposição em paralelo;

d) Não há outros recursos limitantes, como mão de obra ou material;

e) Cada tarefa pode ser executada por somente uma máquina;

f) As tarefas nas diversas máquinas, uma vez iniciadas não devem ser interrompidas nem canceladas;

g) As tarefas são independentes, ou seja, não há restrição de precedência entre elas, e todas deverão ser executadas;

h) Cada tarefa possui necessariamente uma única operação;

i) Os tempos de processamento das tarefas, datas de liberação e tempos de *setup* são fixos e previamente determinados;

j) As tarefas têm diferentes datas de liberação;

k) Os tempos de *setup* das tarefas nas diversas máquinas são antecipados, podendo iniciar antes da liberação da tarefa, e independentes da sequência de execução.

Dentre os diversos parâmetros para os problemas de programação da produção, serão citadas as variáveis de entrada e as variáveis que descrevem a solução do problema tratado. De acordo com Farber e Coves (2005), podemos descrever:

- Tarefa (J_j): uma atividade processada por uma máquina ou estação de trabalho é chamada de tarefa (*job*). O número de tarefas é representado pela letra "*n*", ou seja, o conjunto das tarefas J_j é denotado por: $J = \{J_1, ..., J_n\}$.
- Máquina (M_k): os recursos produtivos que executarão as tarefas serão denominados genericamente de máquinas, podendo ser equipamentos, computadores, células de produção, operadores (funcionários), etc. No problema de máquinas paralelas idênticas existem várias máquinas M_k disponíveis para o processamento, onde cada tarefa J_j pode ser processada em qualquer máquina.
- Tempo de processamento (p_j): é o tempo que a tarefa J_j permanece na máquina enquanto está sendo processada.
- *Release Date* (r_j): é a data de liberação da tarefa J_j, ou seja, o instante a partir do qual a tarefa pode iniciar o seu processamento.
- *Setup* (s_j): também chamado de tempo de preparação ou configuração, o *setup* é o tempo adicional necessário para preparar a máquina, que processará a tarefa J_j. Neste estudo, está sendo abordado um tempo de *setup* independente da sequência, ou seja, o tempo de *setup* depende apenas da tarefa a ser processada.
- *Makespan* ($C_{\max}$): duração total da programação. Mede a eficiência operacional informando o tempo necessário para se executar um conjunto de tarefas. Equivale a maximização da utilização dos recursos.

2.2 Algoritmo *Simulated Annealing*

A Computação Natural ou Computação Bio-inspirada consiste em uma área de pesquisa e desenvolvimento de técnicas computacionais essencialmente fundamentadas em conceitos biológicos e conceitos computacionais evolutivos, aplicando a partir da implementação computacional, o processo de evolução natural como um paradigma de solução de problemas de otimização (SARAMAGO e STEFFEN, 2004).

Os métodos naturais são procedimentos iterativos que tentam simular os processos usados na natureza para resolver problemas difíceis. O *Simulated Annealing*, já referenciado em português como “resfriamento simulado”, “recozimento simulado” ou “tempera simulada” é um método evolutivo baseado em processos naturais, mais especificadamente no processo de resfriamento dos metais utilizados na metalurgia.

O algoritmo *Simulated Annealing* surgiu em 1983, tendo como precursor Osman e Potts (1989), que se baseou nas idéias de Metropolis *et al.* (1953). Consiste em uma útil conexão entre a mecânica (comportamento de um sistema em equilíbrio térmico a uma temperatura finita) e a otimização combinatória (encontrar o mínimo de uma dada função dependendo de vários parâmetros). Alguns resultados publicados utilizando esta técnica, em particular devido aos esquemas de resfriamento rápido têm gerado grandes avanços nas técnicas de otimização probabilísticas (MASTERS, 1993).

A utilização de algoritmos aleatórios está cercada de diversas potencialidades. Sua aplicação é útil em diversos tipos de problemas para o apoio à tomada de decisão. Robustos e flexíveis, os algoritmos randômicos orientados probabilisticamente são técnicas e procedimentos genéricos, adaptáveis, aplicáveis na solução de uma vasta quantidade de problemas complexos, em que a condição primordial é a disponibilidade de uma grande capacidade de recursos computacionais.

O *Simulated Annealing* encontra novos pontos dentro do espaço de busca através da aplicação de operadores estatísticos para os pontos atuais, obtendo novos pontos que orientam a busca de soluções ótimas dentro deste espaço. Esta técnica executa para a otimização uma perturbação randômica que modifica as soluções, mantendo o melhor valor para cada iteração. Após muitas iterações, o conjunto produtor do melhor valor é designado para o centro sobre o qual a perturbação acontecerá na próxima temperatura.

A fim de escapar do ótimo local, o algoritmo reinicia a busca partindo de uma solução diferente e utiliza a melhor solução encontrada como solução do algoritmo. Esta estratégia aumenta a independência à solução inicial, mas acrescenta uma dificuldade quanto a decidir quando interromper o algoritmo, sendo necessário o estabelecimento de critérios adequados de convergência.

Na analogia com o processo de resfriamento (*annealing*) da metalurgia, sabe-se que, se o metal é resfriado em condições apropriadas, uma solução cristalina simples pode ser obtida (KIRKPATRICK *et al.*, 1983). Neste resfriamento, o metal é aquecido a altas temperaturas, causando um choque violento nos átomos. Se o metal for resfriado de forma brusca, a microestrutura tende a um estado randomicamente instável, porém, se o metal é

resfriado de forma suficientemente lenta, o sistema procurará um ponto de equilíbrio caracterizado por uma microestrutura ordenada e estável.

Com uma raciocínio simples baseado em um fenômeno físico, Metropolis *et al*. (1953) introduziram um método numérico que representou o estado de equilíbrio de um conjunto de átomos a uma dada temperatura. Cada configuração é definida por um conjunto de átomos de energia (S) e temperatura (T), e a cada passo do algoritmo, os átomos sofrem um pequeno deslocamento aleatório provocando uma pequena alteração de energia ΔS no sistema.

Na otimização via *Simulated Annealing*, considera-se a perturbação aleatória e a manutenção do melhor valor. A temperatura é então reduzida e novas tentativas executadas. Ao final do processo, pretende-se obter a melhor solução que foi possível de ser encontrada, eventualmente a ótima global.

A configuração inicial das variáveis do problema é adotada pelo algoritmo de geração da solução inicial. O valor inicial é adotado como o melhor valor. Após a temperatura ser reduzida, novas tentativas são executadas. Este procedimento é repetido, ao final do processo pretende-se obter a melhor solução. Na otimização, o processo que encontra um estado de baixa energia e os pontos em que ocorrem defeitos e intrusões são considerados soluções ruins.

De forma genérica nos problemas de otimização, os parâmetros principais da programação do algoritmo são os seguintes fatores: a temperatura inicial e final, o número de iterações do algoritmo com a mesma temperatura e a estratégia de redução de temperatura ao longo da execução do algoritmo.

E as etapas mais sensíveis do método são: a seleção da solução inicial, a definição do esquema de perturbações, a determinação do número de iterações em cada temperatura e o critério de parada.

Então a lógica do método segue a partir de um ponto no espaço de soluções S calcula-se um novo ponto S' vizinho do atual. Se a energia é menor neste novo ponto (ou seja, a medida de desempenho melhorou), ele passa a ser o ponto atual, então um ponto seu vizinho é calculado e o algoritmo continua. Se a energia é maior neste novo ponto (indicando que a nova solução é pior do que a atual), ele não é automaticamente descartado. Há certa probabilidade dele ser aceito como o novo ponto atual, e esta probabilidade é tão maior quanto for o parâmetro temperatura ou quanto menor for a diferença de energia entre os dois pontos.

Essa ideia é baseada no critério de Metrópolis que consiste em avaliar a diferença de energia entre a solução atual e a nova solução, ou seja, (METRÓPOLIS *et al*., 1953),

$$\Delta S = S_{k+1} - S_k \tag{2.1}$$

Onde:

S_k – é a solução encontrada na última iteração;

S_{k+1} – é a nova solução encontrada.

Se ΔS é negativo, isto significa que a nova solução é melhor que a anterior, e esta última é substituída. Entretanto, se é positivo, a probabilidade de esta solução de maior energia substituir a anterior é dada por:

$$P(\Delta S) = exp\left(\frac{-\Delta S}{kT}\right) \tag{2.2}$$

Onde:

$P(\Delta S)$ – probabilidade de uma solução inferior ser aceita;

T– parâmetro de controle;

k– constante de Boltzmann.

A escolha da função de probabilidade $P(\Delta S)$, conforme descrito acima, deve-se ao fato de que o sistema evolui segundo uma distribuição de Boltzmann, mas outras funções de distribuição podem ser testadas.

Assim, a técnica do *Simulated Annealing* consiste numa meta-heurística com diversas aplicações, que parte de uma solução inicial e efetua movimentos de busca local visando melhorar a solução obtida, incluindo a possibilidade de aceitar soluções piores do que a atual no intuito de escapar dos ótimos locais.

Como outros procedimentos heurísticos, não garante que alcançou a solução ótima, embora em geral forneça resultados com razoável qualidade em tempo computacional aceitável. Este método tem sido aplicado em diversos problemas práticos de otimização com resultados bastante satisfatórios. E por este motivo foi escolhido para se resolver o problema de programação da produção tratado nesta pesquisa.

2.3 Análise histórica dos trabalhos

Nesta pesquisa, foi realizada uma extensiva busca na literatura específica de programação da produção, delimitando-se aos problemas de máquinas paralelas idênticas com minimização do *makespan*.

Além disso, é importante ressaltar como uma das contribuições deste trabalho que esta revisão tenta descrever a evolução ao longo dos anos do problema tratado, naturalmente não contemplando exaustivamente todos os artigos existentes na literatura até o momento, visto que seria inviável pela vasta área de pesquisa.

O artigo pioneiro de máquinas paralelas idênticas com objetivo de minimizar o *makespan* foi publicado por McNaughton (1959), que considerou o problema que não permite interrupção de tarefas (*preemption*) e propôs limitantes inferiores usados em inúmeros outros trabalhos posteriores.

Baseando na heurística de McNaughton (1959), Hu (1961) resolveu duas questões, propondo um algoritmo no problema de programação de tarefas: a minimização da duração total no ambiente de máquinas paralelas idênticas e a otimização do número de máquinas paralelas de forma que as tarefas sejam processadas em um dado intervalo de tempo *T*.

Utilizando o algoritmo de Hu (1961) combinado com os limitantes inferiores de McNaughton (1959), os autores Muntz e Coffman (1969) desenvolveram o caso do problema de duas máquinas paralelas idênticas. Em outro momento, Graham (1969) adaptou a conhecida regra de sequenciamento LPT (que prioriza as tarefas com maior tempo de processamento) adaptando uma nova heurística com bons resultados.

Mais tarde, Muntz e Coffman (1970) utilizaram o algoritmo de Hu (1961) com presença de restrições de precedência onde as operações não podem ser interrompidas (*preemption*).

O conhecido algoritmo MULTI-FIT foi adaptado para problema de máquinas paralelas por Coffman *et al.* (1978) com bons resultados comparados aos limitantes inferiores propostos. Friesen (1984) aperfeiçoou os limitantes inferiores para o algoritmo MULTI-FIT proposto por Coffman *et al.* (1978).

Mais adiante, Lee e Massey (1988) combinaram o método LPT e o algoritmo MULTI-FIT, no qual o LPT gerou a solução inicial e o MULTI-FIT como um método de melhoria. Posteriormente, Monma e Potts (1989) determinaram a minimização do *makespan* em duas máquinas paralelas onde as tarefas podem ser interrompidas e o tempo de *setup* é independente de sequência.

O artigo de Tang (1990) considerou uma situação em que os tempos de *setup* dependem do tipo da parte que está sendo produzida e utiliza uma adaptação do método MULTI-FIT para minimizar o *makespan.*

Aperfeiçoaram os limitantes ideais para o *makespan*, Blocher e Chand (1991) propôs um algoritmo eficiente de duas etapas utilizando o método de solução da heurística LPT. Rajgopal e Bidanda (1991) incluíram tempos de *setup* dependente da sequência com o objetivo de minimizar *makespan*, vários procedimentos heurísticos são propostos e testados em relação ao *makespan*, *flow time* médio e tempo gasto dos *setups*.

Modelando o problema de máquinas paralelas, Guignard (1993) utilizou como método de resolução a decomposição lagrangeana para formular o problema. No artigo de Monma e Potts (1993), os trabalhos são divididos em lotes e com tempo de configuração das máquinas sempre que necessário, caso ocorra uma mudança de processamento de um trabalho em um lote para um emprego em outro lote. Os tempos de preparação são assumidos depende apenas do lote do trabalho a ser programado. Dois tipos de heurísticas são propostos e analisados os piores casos.

Os autores Ovacik e Uzsoy (1993) fizeram inclusão de tempo de *setup* dependente abordando para esses casos especiais (maiores e menores *setups* para mudar entre ou dentro das classes, e *setup* delimitados por tempos de processamento).

Os algoritmos genéticos desenvolvido por Hou *et al*. (1994) e Corrêa *et al.* (1999) foram aplicados ao problema de máquinas paralelas idênticas. Comparando o algoritmo genético e o *simulated annealing*, Min e Cheng (1998) desenvolveram o problema de máquinas paralelas, e descobriram que dentre os dois o que gerou melhores soluções foi o algoritmo genético com tempos de execução mais longos que o algoritmo SA.

Baseando-se em vários artigos citados anteriormente, Kurz e Askin (2001) abordaram a programação em máquinas paralelas com tempos de *setup* dependentes, data de liberação diferente de zero com o objetivo de minimizar *makespan*, no qual ao considerar a data de liberação o algoritmo de inserção aleatória obteve melhores resultados comparado ao algoritmo genético. Chang *et al.* (2004) trataram do algoritmo *simulated annealing* para minimizar o *makespan* em máquinas paralelas em processamento em lote. Os resultados demonstraram que o método de programação linear inteira implementado no software CPLEX comparado ao *simulated annealing*, se aproxima dos resultados ótimos na maioria dos casos.

O algoritmo genético híbrido (HGH) foi abordado por Kashan *et al.* (2008), juntamente com o SA comparando com limitantes inferiores, para maiores instâncias o tempo de execução do HGH foi mais demorado comparado ao SA, entretanto com melhor qualidade

de convergência. Damodaran e Chang (2008) criaram quatro heurísticas para minimização do *makespan* em máquinas paralelas idênticas com instâncias de tempos de processamento, comparando a performance ao SA obteve valores aproximados dentre as heurísticas propostas. Lee *et al.* (2006) pesquisaram sobre método *Simulated Annealing* aplicado a máquinas paralelas, demostrando ser um método muito eficaz com 95% dos resultados alcançaram a melhor solução e que supera os algoritmos LISTFIT e PI propostos respectivamente por Gupta e Ruiz-Torres (2001) e Fatemi e Jolai (1998).

Os autores Behnamian *et al.* (2009) desenvolveram três meta-heurísticas híbridas a colônia de formigas (ACO), resfriamento simulado (SA) e busca na vizinhança variável (VNS) para o problema com tempos de *setup* dependente da sequência, destacando as vantagens em termos de generalidade e de qualidade para grandes instâncias.

Montoya-Torres *et al.* (2010) desenvolveram um algoritmo para máquina única e máquinas paralelas idênticas com o objetivo de minimizar o *makespan* com tempos de *setup* dependente e data de liberação, propondo o algoritmo de inserção aleatória e obtendo melhores desempenhos que o artigo Kurz e Askin (2001). Laha (2012) baseou no trabalho de Lee *et al.* (2006) para o mesmo problema e obteve melhores resultados ainda, utilizando o algoritmo *Simulated Annealing*.

A Figura 2.1 apresenta os artigos encontrados nos problemas de máquinas paralelas idênticas e minimizaram o *makespan*. Além disso, consideraram-se outras restrições a medida que os autores os tratassem nos artigos. A organização dos 26 trabalhos está revisada em ordem cronológica e a relação (setas) existente entre eles. As setas pontilhadas, quando um artigo comparou o seu método ou mesmo utilizaram da parte teórica do outro para encontrar melhores resultados. As diferentes cores das setas foram usadas apenas para melhor visualização da representação, não indicando nenhuma diferença.

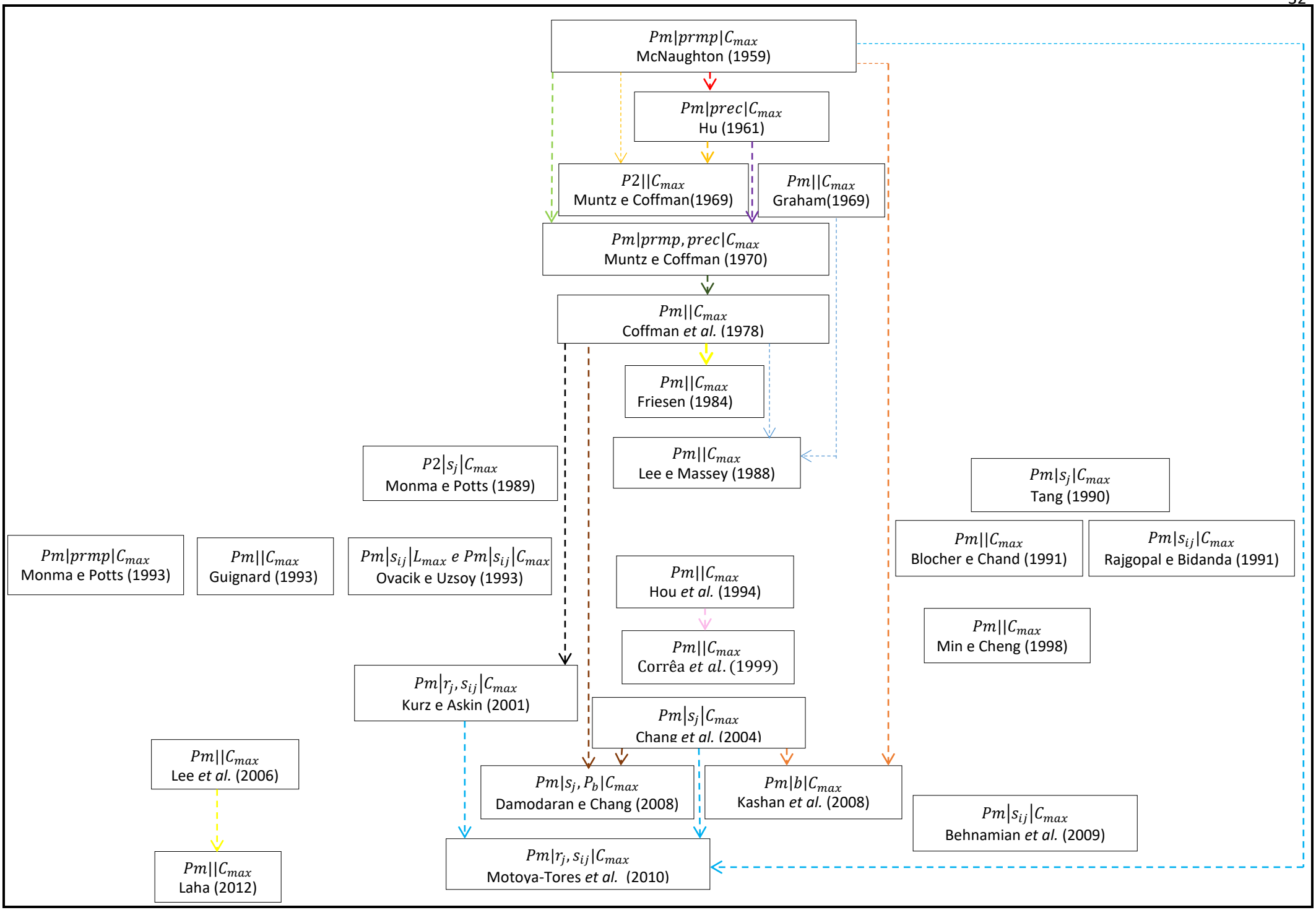


FIGURA 2.1 – Representação em ordem cronológica dos principais artigos de máquinas paralelas idênticas com minimização do *makespan*

Além da representação visual, os trabalhos citados também foram organizados na Tabela 2.1, que inclui o método de solução utilizado no problema.

Dentre os 26 artigos analisados, como mostra a Tabela 2.1 somente seis trabalhos utilizaram ou mesmo compararam o *Simulated Annealing* na resolução do problema de máquinas paralelas idênticas. Sendo que as meta-heurísticas passaram a ser consideradas apenas nos últimos trabalhos apresentados, especificamente, a partir dos anos 90. Os anteriores utilizaram principalmente as heurísticas.

TABELA 2.1 – Descrição dos artigos citados na Figura 2.1 e seus respectivos métodos de solução

Autor(s) /ano	Notação	Método de Solução
McNaughton (1959)	$Pm\|prmp\|C_{max}$	Heurística
Hu (1961)	$Pm\|prec\|C_{max}$	Heurística
Muntz e Coffman (1969)	$P2\|\|C_{max}$	Heurística
Graham (1969)	$Pm\|\|C_{max}$	Heurística
Muntz e Coffman (1970)	$Pm\|prmp, prec\|C_{max}$	Heurística
Coffman *et al.* (1978)	$Pm\|\|C_{max}$	MULTI-FIT
Friesen (1984)	$Pm\|\|C_{max}$	MULTI-FIT
Lee e Massey (1988)	$Pm\|\|C_{max}$	MULTI-FIT
Monma e Potts (1989)	$P2\|s_j\|C_{max}$	Heurística
Tang (1990)	$Pm\|s_j\|C_{max}$	MULTI-FIT
Blocher e Chand (1991)	$Pm\|\|C_{max}$	LPT
Rajgopal e Bidanda (1991)	$Pm\|s_{ij}\|C_{max}$	MULTI-FIT
Guignard (1993)	$Pm\|\|C_{max}$	Decomposição Lagrangeana
Monma e Potts (1993)	$Pm\|prmp\|C_{max}$	Heurística
Ovacik e Uzsoy (1993)	$Pm\|s_{ij}\|L_{max}$ *e* $Pm\|s_{ij}\|C_{max}$	Heurística
Hou *et al*. (1994)	$Pm\|\|C_{max}$	Algoritmo Genético
Min e Cheng (1998)	$Pm\|\|C_{max}$	Algoritmo Genético + *Simulated Annealing*
Corrêa *et al*.(1999)	$Pm\|\|C_{max}$	Algoritmo Genético
Kurz e Askin (2001)	$Pm\|r_j, s_{ij}\|C_{max}$	MULTI-FIT + Algoritmo Genético + Algoritmo de Inserção Múltipla
Chang *et al.* (2004)	$Pm\|s_j\|C_{max}$	*Simulated Annealing* + Programação Linear Inteira
Lee *et al.* (2006)	$Pm\|\|C_{max}$	*Simulated Annealing*

Kashan *et al*. (2008)	$Pm\|b\|C_{max}$	Algoritmo Genético Híbrido + *Simulated Annealing*
Damodaran e Chang (2008)	$Pm\|s_j, P_b\|C_{max}$	Heurística
Behnamian *et al.* (2009)	$Pm\|s_{ij}\|C_{max}$	Colônia de Formigas + *Simulated Annealing* + Busca local
Motoya-Tores *et al*. (2010)	$Pm\|r_j, s_{ij}\|C_{max}$	Algoritmo de Inserção Randômica
Laha (2012)	$Pm\|\|C_{max}$	*Simulated Annealing*

Algumas conclusões foram observadas com essa revisão bibliográfica, representando contribuições teóricas deste trabalho. Como pode ser visto na Figura 2.1, a pesquisa pioneira de McNaughton (1959) serviu de base para muitos outros autores. Nota-se também que os estudos posteriores foram feitos de forma mais independente uns dos outros, não estabelecendo uma hierarquia, tanto em termos de elaboração de métodos como em comparações de resultados.

Uma fragilidade da evolução dos métodos está justamente no fato de não comparar seus resultados com pesquisas anteriores. Isto ocorreu principalmente nos anos iniciais das publicações, provavelmente pela dificuldade em se acessar os trabalhos. Nas décadas mais recentes, é possível que o maior acesso tenha viabilizado as comparações dos resultados dos métodos.

Nas últimas décadas, os problemas avaliados contêm diferentes restrições, mostrando a tendência de se considerar situações mais realistas. Isto acarreta o tratamento de problemas mais difíceis e maior número de restrições práticas.

Não foi encontrado nenhum trabalho que aborde o mesmo problema em estudo, com o mesmo método de solução e que utilize a meta-heurística *Simulated Annealing*. Os trabalhos mais próximos são de Kurz e Askin (2001) e Montoya-Torres *et al.* (2010), porém, ambos abordam tempos de *setup* dependentes da sequência.

Os trabalhos de Kurz e Askin (2001) e de Montoya-Torres *et al.* (2010) foram os únicos que consideraram diferentes datas de liberação das tarefas, mostrando que essa restrição foi introduzida nas pesquisas apenas muito recentemente. Esta característica confere um caráter mais prático ao problema, pois nas indústrias dificilmente todas as tarefas chegam simultaneamente.

3 MÉTODOS DE SOLUÇÃO PROPOSTOS

3.1 Análise da estrutura do problema

Nos problemas clássicos de programação da produção, o *makespan* é simplesmente definido como C_{max} = max C_j, ou seja, a duração total da programação equivale ao maior instante de término dentre as tarefas, a conclusão do processamento da última tarefa programada.

Já nos problemas que existem diferentes datas de liberação das tarefas, normalmente se utiliza a expressão C_{max} = max C_j – min r_j, considerando a duração total da programação como o intervalo entre o primeiro instante em que há pelo menos uma tarefa liberada (min r_j) e o instante de término da última tarefa programada (max C_j). Esta relação no problema de máquinas paralelas pode ser visualizada no exemplo da Figura 3.1.

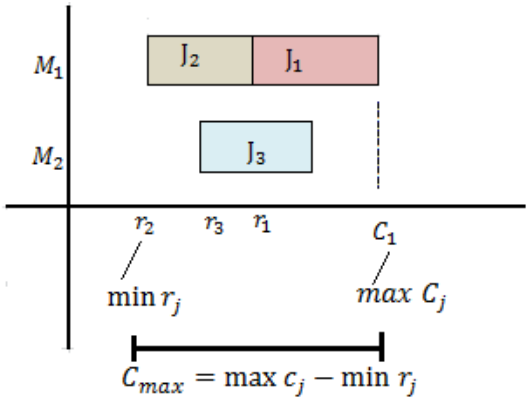


FIGURA 3.1 – Cálculo do *makespan* no problema de máquinas paralelas com datas de liberação e sem *setup*

Entretanto, no problema tratado neste trabalho, além da presença de diferentes datas de liberação das tarefas, existem ainda os tempos de *setup* que podem ser antecipados, ou seja, iniciados antes da liberação das tarefas. Esta relação pode ser observada no exemplo da Figura 3.2.

Como o tempo de *setup* pode ser maior do que a data de liberação (como na tarefa J_2) ou menor do que a data de liberação (como na tarefa J_3), optou-se nesta pesquisa por considerar a expressão do cálculo do *makespan* como C_{max} = max C_j, que garante a execução de toda a carga de trabalho, sendo processamento ou *setup*, desde a data zero da programação até o instante de término da última tarefa.

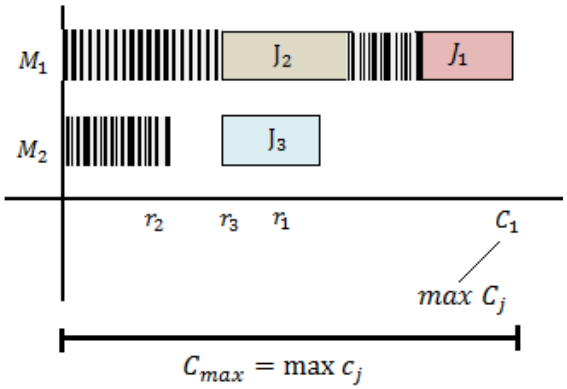


FIGURA 3.2 – Cálculo do *makespan* no problema de máquinas paralelas com datas de liberação e *setup* antecipado

Na aplicação do algoritmo *Simulated Annealing*, a solução do problema será representada por meio vetorial, contendo ordenadamente os índices das tarefas programadas nas máquinas. A posição contendo valor zero representa a mudança de máquina, ou seja, cada sequência de tarefas programada em cada máquina será separada por uma posição contendo valor zero.

Esta é uma dificuldade adicional do problema de máquinas paralelas, uma vez que em outros problemas, como máquina única e *flow shop* permutacional, bastaria a representação da sequência de tarefas para se estabelecer a programação. No caso do problema de máquinas paralelas, além da decisão de sequenciamento, existe também a atividade de alocação das tarefas às máquinas.

A Figura 3.3 apresenta a ilustração vetorial de solução do problema de máquinas paralelas. No exemplo, as tarefas J_1, J_2 e J_3 foram programadas nesta ordem na primeira máquina e as tarefas J_4, J_5 e J_6 foram programas nesta sequência na segunda máquina.

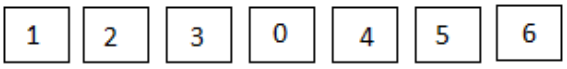


FIGURA 3.3 – Representação da solução em forma vetorial

A programação das tarefas nas máquinas representada anteriormente é ilustrada no gráfico de Gantt da Figura 3.4, considerando os dados das tarefas e máquinas (tempos de processamento e de *setup* e datas de liberação).

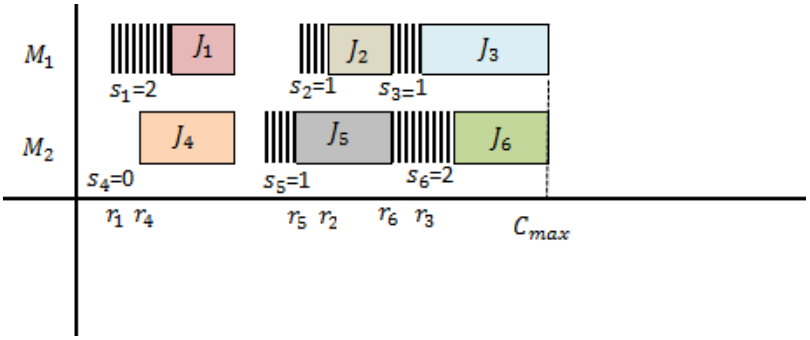


FIGURA 3.4 – Gráfico de Gantt com a programação correspondente à representação vetorial

3.2 Algoritmo para solução inicial

Conforme apresentado na seção 2.2, o algoritmo *Simulated Annealing* parte de uma solução inicial e realiza operações denominadas "perturbações" a fim de investigar a vizinhança em busca de soluções melhores.

A ideia para geração da solução inicial é já partir de uma boa solução para o problema em busca de melhorias maiores. No problema tratado, é desejável que se programe primeiro as tarefas com as menores datas de liberação, ao mesmo tempo em que é vantajoso priorizar as tarefas com as maiores cargas de trabalho, no caso somando os tempos de processamento e de *setup*.

A maior carga de trabalho no início da programação já se mostrou eficaz para minimização do *makespan* no problema clássico de máquinas paralelas (sem *setup* e com datas de liberação iguais a zero) por meio da utilização da regra LPT e da alocação sequencial das tarefas às máquinas de menor carga.

Assim, o algoritmo de solução inicial proposto faz a ordenação crescente pela razão $\frac{r_j}{p_j+s_j}$ e em seguida alocada uma tarefa de cada vez à máquina de menor carga.

Algoritmo para solução inicial

Passo 1

Ordene as tarefas de forma crescente pela razão $\frac{r_j}{p_j+s_j}$.

Passo 2

Aloque uma tarefa de cada vez na máquina de menor carga.

3.3 Esquemas de perturbações propostos

Os possíveis movimentos de busca em vizinhança denominados “esquemas de perturbações” (ou *perturbation schemes*, denotados por PS) foram propostos para o problema de máquinas paralelas idênticas com base em trabalhos publicados na literatura.

Como a solução é representada vetorialmente, ou seja, os conjuntos de tarefas alocadas em cada máquina são separados pelo valor zero, dependendo da posição em que é realizada a perturbação, ocorre não apenas uma troca de tarefas, mas também uma tarefa pode trocar de máquina.

No algoritmo de forma sequencial será realizado cada esquema de perturbação, de forma que o esquema a ser utilizado será o mesmo até chegar ao critério de parada que seria a temperatura final. E posteriormente será salvo seu respectivo *makespan* obtido. Da mesma forma, será testando o procedimento de escolha ao acaso de um esquema de perturbação qualquer para visto de comparação de resultados.

Assim, são descritos a seguir os seis esquemas de perturbação considerados mais apropriados para o problema tratado, separados em intramáquinas, aqueles em que as operações são feitas com as tarefas de uma mesma máquina, e intermáquinas, em que se movimentam tarefas de máquinas diferentes.

Esquemas de perturbação intramáquinas:

PS1: troca de tarefas adjacentes intramáquina

Seleciona aleatoriamente uma tarefa e troca com a da posição imediatamente posterior. Caso a máquina escolhida tenha apenas uma tarefa, não é possível executar a perturbação e uma nova máquina é selecionada. A posição selecionada do vetor não pode conter valor zero nem ser da última tarefa da máquina.

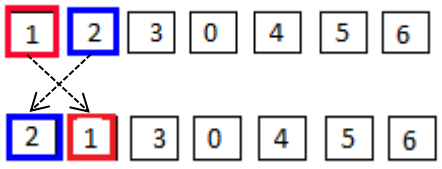


FIGURA 3.5 – Exemplo do esquema de perturbação PS1

PS2: troca de tarefas aleatórias intramáquina

Seleciona aleatoriamente duas tarefas não necessariamente adjacentes de uma mesma máquina, também selecionada aleatoriamente a posição, e faz a troca das duas.

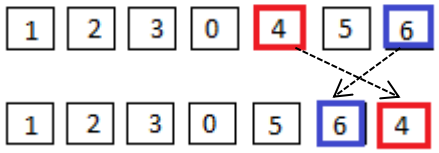


FIGURA 3.6 – Exemplo do esquema de perturbação PS2

PS3: reinserção de uma tarefa intramáquina

Seleciona aleatoriamente uma tarefa de uma máquina, também escolhida aleatoriamente, e em seguida uma nova posição, onde a tarefa selecionada será inserida e as tarefas intermediárias (incluindo a que ocupava a posição selecionada), deslocadas uma posição a direita.

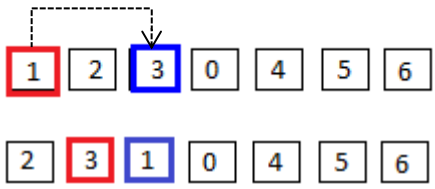


FIGURA 3.7 – Exemplo do esquema de perturbação PS3

Esquemas de perturbação intermáquinas:

PS4: intercâmbio de tarefas intermáquinas

Seleciona aleatoriamente duas tarefas que sejam de duas máquinas, também escolhidas aleatoriamente, e em seguida troca-as.

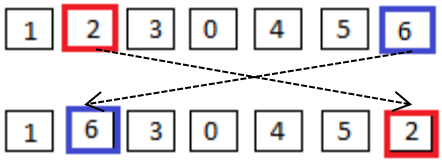


FIGURA 3.8 – Exemplo do esquema de perturbação PS4

PS5: realocação de uma tarefa intermáquinas

Seleciona aleatoriamente uma tarefa de uma máquina, também escolhida aleatoriamente, e em seguida uma posição em uma outra máquina aleatória, onde a tarefa selecionada será inserida e as tarefas a partir desta posição deslocadas à direita na máquina.

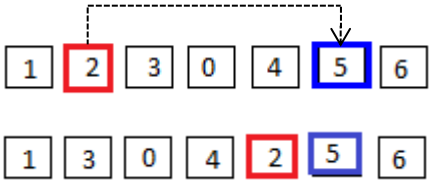


FIGURA 3.9 – Exemplo do esquema de perturbação PS5

PS6: realocação de subsequências intermáquinas

Seleciona aleatoriamente uma máquina e a primeira posição "pos1" da subsequência que será realocada com tamanho no intervalo de 2 a (n_k – pos+1), onde n_k é o número de tarefas programadas na máquina M_k. Em seguida, selecione também aleatoriamente uma segunda máquina e uma posição "pos2", a partir da qual a subsequência será inserida. Retira a subsequência selecionada da primeira máquina (reprogramando adequadamente as tarefas remanescentes) e insere-a na segunda máquina selecionada, de acordo com as posições descritas.

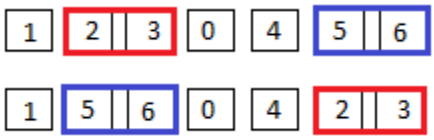


FIGURA 3.10 – Exemplo do esquema de perturbação PS6

3.4 Algoritmo *Simulated Annealing* proposto

A partir do algoritmo de solução inicial e dos esquemas de perturbação descritos, pode-se apresentar a seguinte meta-heurística para programação do problema $P_m|r_j, s_j|C_{max}$. Seja:

carga $_{maior}$: carga da máquina de maior carga;

carga $_{menor}$: carga da máquina de menor carga;

J_u: última tarefa da máquina de maior carga;

p_u: tempo de processamento da última tarefa da máquina de maior carga;

s_u: tempo de *setup* da última tarefa da máquina de maior carga.

Passo 1

Inicialize os seguintes dados:

- Número de tarefas (n)
- Número de máquinas paralelas idênticas (m)
- Tempos de processamento de cada tarefa (p_j)
- Tempos de *setup* de cada tarefa (s_j)
- Datas de liberação de cada tarefa (r_j)
- Temperatura inicial (T_i)
- Temperatura final (T_f)
- Fator de resfriamento (r)
- Esquema de perturbação

Passo 2

Obtenha uma solução inicial (S) por meio do Algoritmo para Solução Inicial.
Calcule o *makespan* desta sequência [$C_{max}(S)$].
Faça $S^* = S$ (onde S^* é a melhor sequência encontrada).
Faça $T = T_i$.

Passo 3

Utilizando de forma sequencial um esquema de perturbação, selecione uma solução S' na vizinhança de S.
Calcule $\Delta = C_{\max}(S') - C_{max}(S)$.

Se $\Delta \leq 0$ (função objetivo melhora), então faça $S = S'$.

Se $C_{max}(S) < C_{max}(S^*)$, faça $S^* = S$.

Senão ($\Delta > 0$), gere um número aleatório u uniformemente distribuído entre 0 e 1. Se $u \leq e^{-\Delta/T}$, então faça $S = S'$.

Passo 4

Faça $T = rT$ (resfriamento).
Se $T > T_f$, então vá para o ***Passo 3***.
Senão, vá para o ***Passo 5***.

Passo 5

Passo 5.1 Se $carga_{maior} - carga_{menor} > p_u + s_u$ (cargas desequilibradas),
então aloque a tarefa J_u na máquina de menor carga,
e faça S' a nova programação.
Calcule $\Delta = C_{\max}(S') - C_{max}(S)$ e vá para o ***Passo 5.2***.
Senão (cargas equilibradas), vá para o ***Passo 5.3***.

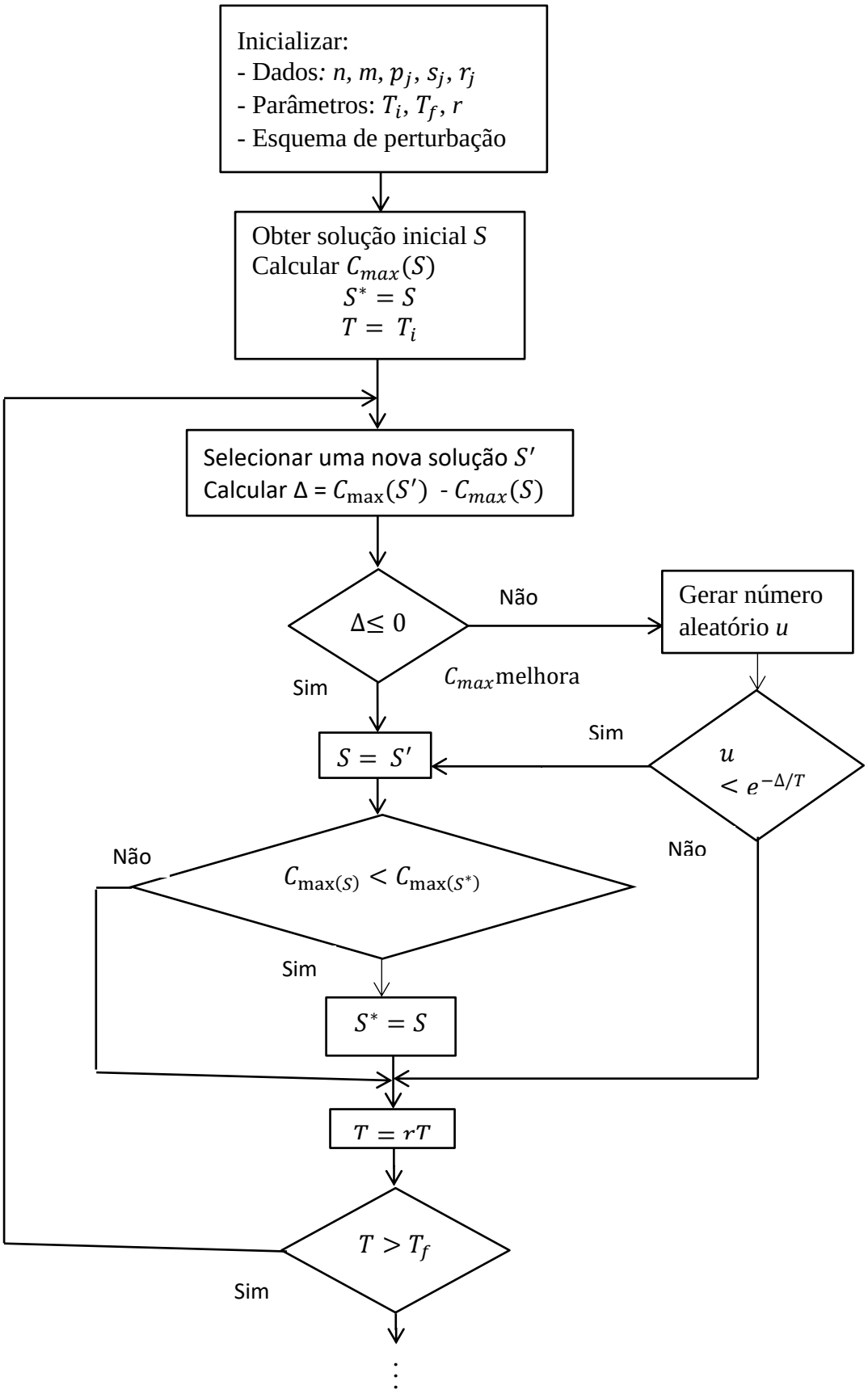
Passo 5.2 Se $\Delta < 0$, então faça $S = S'$.
Se $C_{max}(S) < C_{max}(S^*)$, faça $S^* = S$ e volte ao ***Passo 5.1***.
Senão, vá para o ***Passo 5.3***.

Passo 5.3 Retorne S^* e $C_{max}(S^*)$. FIM.

Nota se que foi inserida uma etapa de melhoria no último passo do algoritmo, em que se verifica se ao final do resfriamento, na programação resultante as máquinas estão com as cargas relativamente equilibradas.

A referência foi a carga de trabalho da última tarefa na máquina de maior carga, ou seja, a soma dos tempos de processamento e de *setup* desta tarefa. Se a diferença entre a carga da máquina mais carregada e a da máquina menos carregada for maior do que esta soma dos tempos de processamento e de *setup* da última tarefa, então é feito o seu deslocamento para a máquina de menor carga. Se o resultado for melhor, então se repete o teste até que não haja mais melhoria.

O fluxograma do algoritmo *Simulated Annealing* proposto segue abaixo, para auxiliar na melhor compreensão do funcionamento da meta-heurística.


Inicializar:
- Dados: n, m, p_j, s_j, r_j
- Parâmetros: T_i, T_f, r
- Esquema de perturbação
Obter solução inicial S
Calcular $C_{max}(S)$
$S^* = S$
$T = T_i$
Selecionar uma nova solução S'
Calcular Δ = $C_{\max}(S')$ - $C_{max}(S)$
Δ≤ 0
Não
Gerar número aleatório u
Sim
C_{max}melhora
$S = S'$
Sim
$u < e^{-\Delta/T}$
Não
Não
$C_{\max(S)} < C_{\max(S^*)}$
Sim
$S^* = S$
$T = rT$
$T > T_f$
Sim

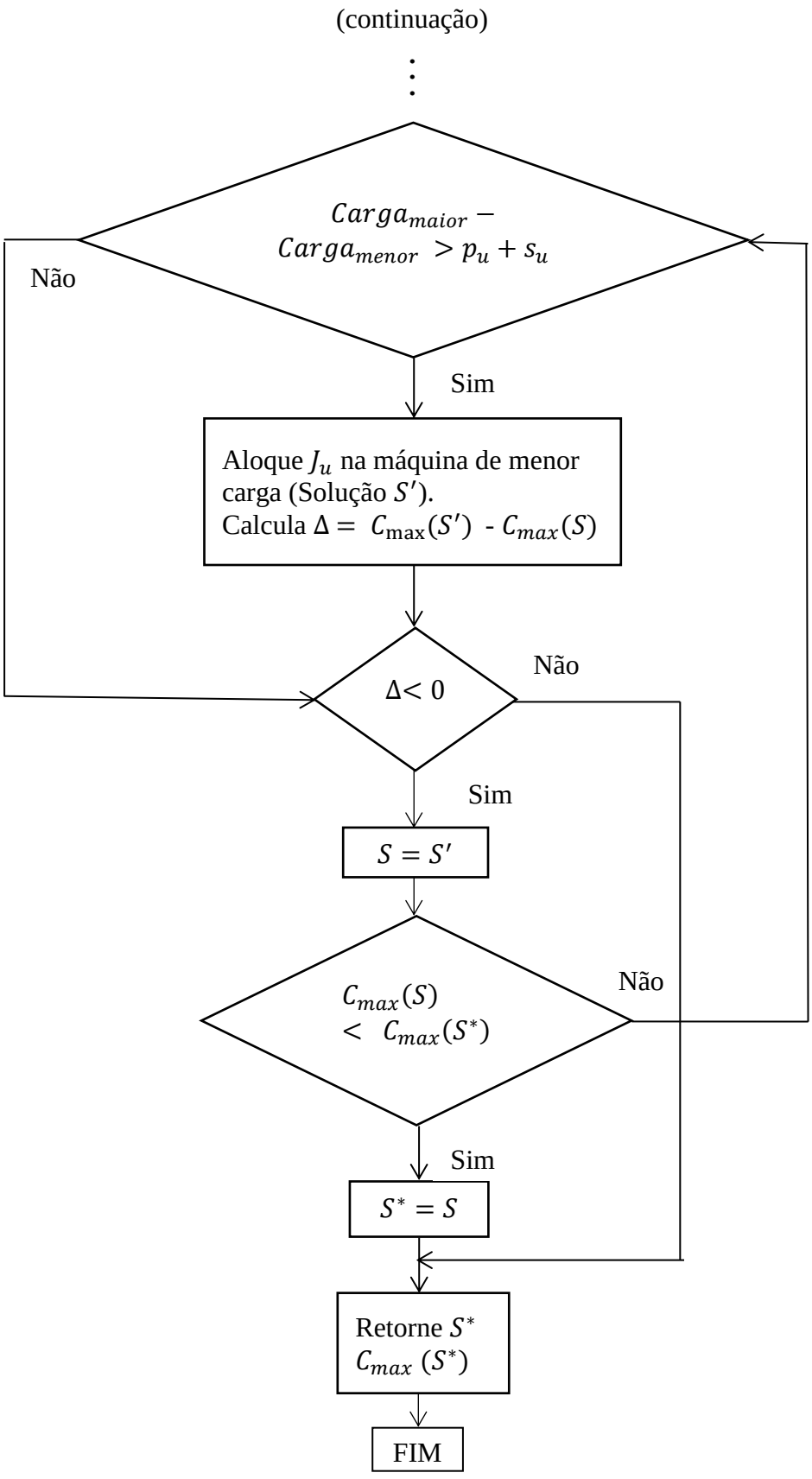


FIGURA 3.11 – Fluxograma do algoritmo *Simulated Annealing* proposto

3.5 Limitantes inferiores propostos

Para comparar os resultados do algoritmo *Simulated Annealing*, foram propostos limitantes inferiores para o *makespan*, que avaliam a qualidade da solução heurística.

O limitante inferior ou *lower bound* (LB) é um valor de referência calculado de forma independente da programação, garantindo-se que seja menor ou igual ao *makespan* ótimo ($LB \leq C^*_{max}$). Assim, quanto maior o valor do limitante inferior melhor (mais acurado), conforme a ilustração da Figura 3.12.

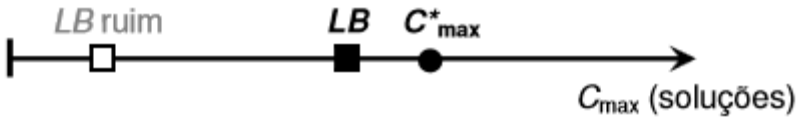


FIGURA 3.12 – Ilustração do limitante inferior para *makespan*

Pela definição, quanto mais próximo do valor do *makespan* ótimo, melhor o limitante inferior. Porém, de antemão não se conhece o valor do *makespan* ótimo, pois é justamente em seu lugar que será utilizado o limitante inferior. Como a definição do limitante inferior garante que seu valor será no máximo igual ao *makespan* ótimo, ou seja, estará sempre à sua esquerda na reta dos valores possíveis para o *makespan* representada na Figura 3.12. Então, quanto mais à direita o valor do limitante inferior, mais próximo estará do *makespan* ótimo, e conforme salientado, quanto maior o seu valor, melhor a referência.

Os métodos de solução podem ser avaliados quanto a sua eficácia em relação a qualidade da solução e quanto a sua eficiência em relação ao custo computacional exigido. Em geral, as heurísticas, sejam construtivas, melhorativas ou meta-heurísticas, não são restritivas em relação a eficiência computacional.

Quanto mais próxima da solução ótima, mais eficaz é a solução heurística. Então foram definidos limitantes inferiores para avaliar a eficácia da meta-heurística proposta. E por não terem sido encontrados na literatura limitantes inferiores para o mesmo problema tratado, as expressões foram propostas com base na estrutura do problema considerado e também em trabalhos clássicos da área.

De acordo com Baker e Trietsch (2009), considerando a relação do problema de máquinas paralelas idênticas, permitindo-se a interrupção de tarefas, o mínimo *makespan* é obtido pela seguinte expressão:

$$LB = \max\left\{\sum_{j=1}^{n} \frac{p_j}{m}, \max_j p_j\right\} \quad (3.1)$$

Acrescentando mais um termo à expressão, Chen (2004) apresenta ainda uma forma tradicional e trivial de se calcular limitante inferior para o *makespan* no problema de máquinas paralelas idênticas. Fazendo a ordenação das tarefas pela regra LPT (*Longest Processing Time*), o limitante inferior é definido por:

$$LB = \max\left\{\sum_{j=1}^{n} \frac{p_j}{m}, p_{[1]}, p_{[m]} + p_{[m+1]}\right\} \quad (3.2)$$

Onde:

$p_{[j]}$ representa o tempo de processamento da j-ésima tarefa da sequência.

Com base nestas expressões e nas restrições especificamente impostas neste trabalho, como a existência de datas de liberação diferentes de zero e tempos de *setup*, foram propostos os limitantes inferiores descritos a seguir.

O primeiro caso (LB_1) é aquele em que existe uma tarefa com carga de trabalho muito grande, ou seja, o *makespan* é definido simplesmente pela soma dos tempos de processamento e de *setup* desta tarefa. Assim, a primeira referência para a composição do limitante inferior para o problema tratado é:

$$LB_1 = \max_j (s_j + p_j) \quad (3.3)$$

O caso do (LB_1) é ilustrado na Figura 3.13. Observe que existe uma tarefa com a carga de trabalho desproporcionalmente maior do que as outras, definindo o valor do *makespan*, mesmo programando todas as demais tarefas na outra máquina.

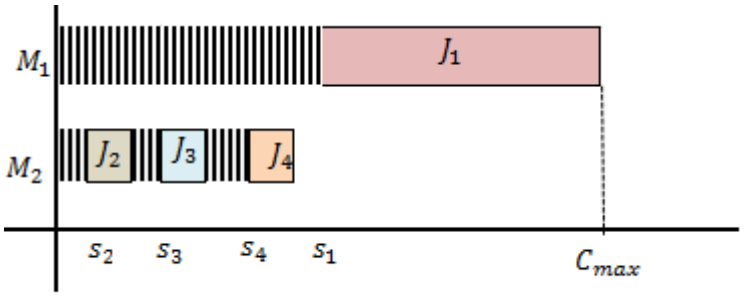


FIGURA 3.13 – Ilustração do LB_1

O segundo caso (LB_2) ainda se refere uma possível tarefa com o maior tempo de processamento, porém tendo a data de liberação maior do que o seu tempo de *setup*. Assim, a sua data de término é definida pela soma da data de liberação e do tempo de processamento, uma vez que o *setup* é antecipado. A sua expressão é dada por:

$$LB_2 = \max_j(r_j + p_j) \tag{3.4}$$

A ilustração do LB_2 é apresentada na Figura 3.14.

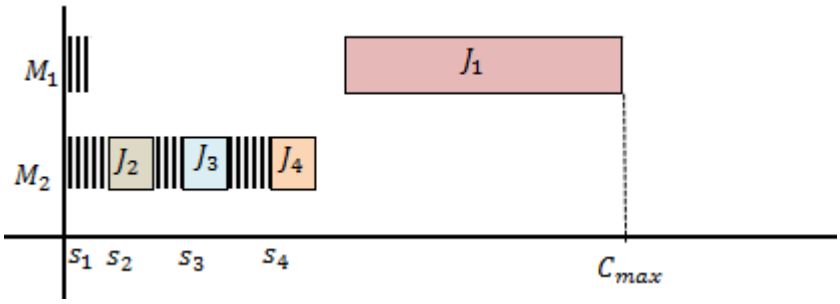


FIGURA 3.14 – Ilustração do LB_2

O terceiro caso (LB_3) é aquele em que se considera a relaxação da restrição de não interrupção das tarefas, ou seja, soma-se a carga de trabalho das tarefas (tempos de processamento e de *setup*) e divide-se pelo número de máquinas. Tendo o cálculo da expressão de LB_3 por:

$$LB_3 = \frac{\sum_{j=1}^{n}(s_j + p_j)}{m} \tag{3.5}$$

A Figura 3.15 apresenta a ilustração deste limitante inferior.

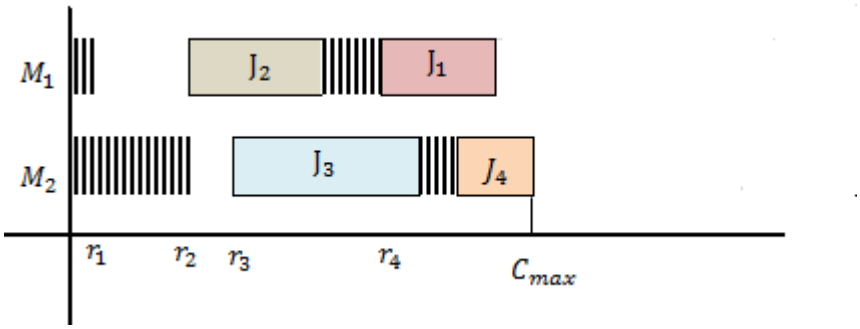


FIGURA 3.15 – Ilustração do LB_3

O último caso (LB_4), utilizando o limitante inferior proposto por Chen (2004) Fazendo a ordenação das tarefas pela regra LPT (*Longest Processing Time*), o limitante inferior é definido por:

$$LB_4 = \max\{p_{[m]} + p_{[m+1]}\} \tag{3.6}$$

A Figura 3.16 apresenta a ilustração deste limitante inferior.

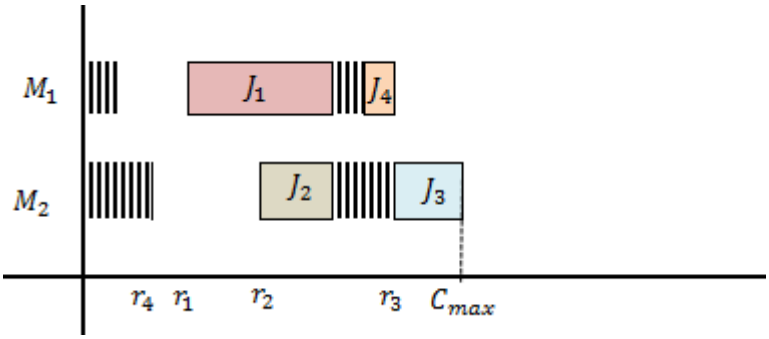


FIGURA 3.16 – Ilustração do LB_4

Para se contemplar todas as situações possíveis, conforme descritas nestes quatro casos, o limitante inferior final considerado será:

$$LB = (\max(\max(\max(LB_1, LB_2), LB_3), LB_4) \tag{3.7}$$

4 EXPERIMENTAÇÃO COMPUTACIONAL E RESULTADOS

4.1 Definição da amostragem

Os parâmetros de expêrimentação foram baseados dos trabalhos publicados mais próximos da literatura. Como o artigo de Kurz e Askin (2001) é mais antigo optou-se em utilizar o artigo de Motoya *et al*. (2010) que é mais recente. Entretanto como no trabalho os autores consideraram intervalos de data de liberação muito pequenos e como o *setup* é antecipado, as liberações acabariam não tendo efeito. Então os parâmetros de experimentação foram adaptados do artigo de Motoya *et al*., 2010 para o problema tratado.

A seguir a Tabela 4.1 com os parâmetros retirados dos artigos citados na revisão bibliográfica que não utilizaram o *Simulated Annealing* como método de resolução. E posteriormente a Tabela 4.2 os parâmetros dos artigos da revisão bibliográfica que utilizaram o *Simulated Annealing*.

TABELA 4.1 – Parâmetros de trabalhos publicados sem utilizar o *Simulated Annealing*

Problema/ Autor(s)/ ano	**Número de máquinas**	**Número de tarefas**	**Intervalo de s_{ij}**	**Intervalo de r_j**	**Intervalo de p_j**
$P_m \mid\mid C_{max}$ Lee e Massey (1988)	2, 3, 4, 5, 6, 8,10;	10, 30, 50, 100	–	–	–
$Pm \mid s_j \mid C_{max}$ Tang (1990)	5, 8, 12	5, 10, 15	U(20,50), U(50,80), U(80,160)	–	U(20,50), U(50, 80)
$P_m \mid\mid C_{max}$ Blocher e Chand (1991)	2,3,4,5,6	4,10,18,28	–	–	U(20,50), U(30,80)
$Pm \mid s_{ij} \mid C_{max}$ Rajgopal e Bidanda (1991)	5,8,12	5,10,15	U(20,50), U(30,80)	–	–
$Pm \mid r_j, s_{ij} \mid C_{max}$ Kurz e Askin (2001)	2, 10	3, 10	U(450-$\sqrt{3}\sigma_s$. $450 + \sqrt{3}\sigma_s$) U(385, 515); U(320, 580) Onde: $\sigma_s =$ $\frac{1}{2}\frac{1.5x450}{9}; \frac{1.5x450}{9}$	0 ;U(0, $(4.950n_m - 4.950)$ Onde: n_m: número de tarefas por máquinas	U(4.23, 4.77); U(1.8, 7.2)
$Pm \mid s_j, P_b \mid C_{max}$ Damodaran e Chang (2008)	2,4	10, 25, 50	U(1,10), U(2,4), U(4,8)	–	U(1,10), U(1,20)
$Pm \mid r_j, s_{ij} \mid C_{max}$ Motoya-Tores *et al.* (2010)	3, 5	10,20, 50, 100	U (0, minp_j)	0, 6, 12, 30, 60; 15, 30, 75, 150; 30, 60, 150,300;	U (1,100)

TABELA 4.2 – Dados para experimentação computacional com o *Simulated Annealin*

Problema/ Autor(s)/ano	**Número de máquinas**	**Número de tarefas**	**Intervalo de s_j ou s_{ij}**	**Intervalo de p_j**	**Temperatura inicial**	**Temperatura final**	**Número de iterações**	**Fator de resfriamento**
$Pm\|s_j\|C_{max}$ Chang *et al.* (2004)	2, 4	10, 25, 50	U(1,10); U(2,4); U(4,8)	U(1,10); U(1,20)	(não cita)	(não cita)	10.000	r = 0.9
$Pm\|\|C_{max}$ Lee *et al*. (2006)	3; 4; 5; 2, 3, 4, 6, 8,10; 3, 5, 8, 10; 2; 3	6, 9,15; 6,9,15; 8,12,20; 8,12,20; 10,15,25; 10,15,25; 10, 30, 50, 100; 10, 11,13,14,16,17; 16,17,21, 22,26,27; 25,26,33,34,41,42; 31,32,41,42,51,52; 9; 10;	–	U(1,20), U(20,50); U (100, 800); U (1,100), U (100, 200), U (100, 800); U (1, 20), U (20, 50), U(1,100), U(50,100), U(100, 200), U(100,800)	(não cita)	Solução for igual aos *lower bounds* ou 30n	12.000	U (0,1)
$Pm\|b\|C_{max}$ Kashan *et al.* (2008)	2, 4	10, 20, 50, 100	–	U(1,10); U(1, 20)	(não cita)	(n, 50); (2n,100)	15.000	U (0,1)
$Pm\|s_{ij}\|C_{max}$ Behnamian *et al.* (2009)	1,2,10,U(1,4); U(1,10)	6, 30,100	$0.5, 0.6, U(0.5,1)$ $0.1, 0.01, U(0, 0.1$	U(50,70); U(20,100)	$0.5, 0.6, U(0.5,1)$	$(m^2x(n+1)/2)$	(não cita)	$0.1, 0.01, U(0, 0.1)$
$Pm\|\|C_{max}$ Laha (2012)	2, 3,4,6,8; 10; 5,10, 20	30, 50,100; 2,3; 200,500	–	U(100,800); U(1,100);U(100,800)	1000	Solução for igual aos *lower bounds* ou 30n	(não cita)	r = 0.9

é a restrição de nunca o número de máquinas ser maior que o número de tarefas, pois se não existiria um problema de alocação de tarefas nas máquinas.

Na experimentação computacional do algoritmo proposto, foram testados e avaliados 720 problemas, definidos pelo número de tarefas (*n*) {10, 20, 30, 50, 80,100}, número de máquinas (*m*) {2, 3, 5, 8}, intervalos de tempos de liberação das tarefas (*r*) {1, 49}; {1,99}; {50,149}, intervalos de tempos de *setup* (*s*) {1, 49} e intervalos de tempos de processamento (*p*) {1,99}. Assim, 6 (alternativas de número de tarefas) x 4 (alternativas de número de máquinas) x 3 (alternativas de intervalos de liberação) x 1 (alternativa de intervalo de *setup*) x 1 (alternativa de intervalo de processamento) = 72 classes. Para cada classe, foram gerados aleatoriamente 10 problemas totalizando 720. A quantidade de problemas gerados em cada classe objetiva reduzir o erro amostral.

4.2 Obtenção dos dados

De acordo com parâmetros definidos, todos os problemas foram gerados aleatórios por meio de um software construido especificamente para essa finalidade, chamado “Gerador de arquivos de dados”.

Foi utilizada a liguagem de programação Delphi e o sistema operacional Windows 8. As configurações da máquinas são a seguintes: processador Pentiun Core i5 da Intel com 1.80GHz de frequência e 6 GB de memória RAM.

4.3 Processo de análise

Os resultados obtidos na experimentação computacional foram analisados por meio do desvio relativo percentual em relação a solução inicial e ao limitante inferior, tempo médio computacional e a ferramente estátistica análise de variância. Foram desenvolvidas 6 métodos para cada classe de problema.

O desvio relativo percentual mede a variação correspondente à melhor solução obtida pelo maior LB. Quando o desvio relativo percentual da solução de um problema é igual a zero para um determinado LB, significa que a duração total da programação fornecida é menor, ou seja o algoritmo apresentou a melhor programação.

O desvio relativo percentual (RPD) é assim calculado,

$$RPD = \left(\frac{C_{max}^{heur} - LB}{LB}\right).100 \qquad (4.1)$$

Onde,

C_{max}^{heur} é o makespan fornecido pelo algoritmo *Simulated annealing* e LB é o melhor limitante inferior fornecido pela meta-heuristica.

O tempo médio de computação de um método é calculado pela soma dos tempos de computação de cada problema dividida pelo número total de problemas resolvidos (média aritmética dos tempos de computação). Na experimentação computacional, o tempo médio de computação foi médido em milissegundos(*ms*).

Como não foram encontrados na literatura métodos heuristicos para programação do ambiente tratado neste trabalho. Por isso adaptou-se diferentes perturbações e a criação de diferentes casos possiveis de limites inferiores a fim de verificar a qualidade da solução heuristica.

4.4 Análise dos Resultados das Perturbações

A Tabela 4.1 expressa o RPD global das perturbações em relação ao limitante inferior.

TABELA 4.3 - RPD global em relação ao limitante inferior

	PS1	**PS2**	**PS3**	**PS4**	**PS5**	**PS6**
sem a etapa de melhoria	12,6	11,2	11,3	7,9	9,5	12,5
com a etapa de melhoria	12,5	10,9	11,0	7,7	9,2	12,3
Diferença	0,1	0,4	0,3	0,2	0,3	0,2

Evidentemente, conforme esperado, os desvios das perturbações com a etapa de melhoria foram sempre menores. Porém, o que se percebe é que a melhoria proporcionada foi relativamente pequena, no máximo de 0,4 pontos percentuais no caso da PS2.

A Figura 4.1 apresenta o comparativo visual dos RPD das perturbações com e sem a etapa de melhoria.

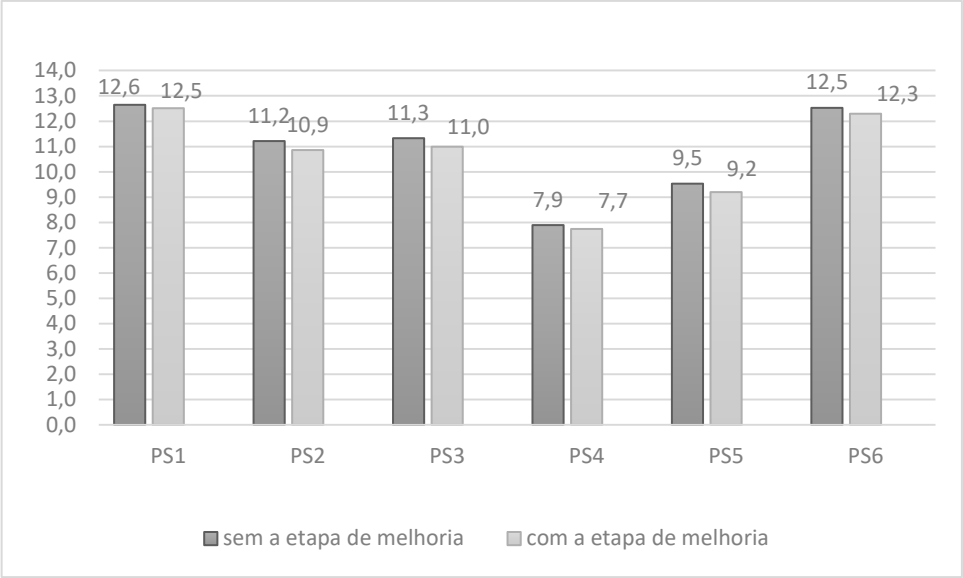


FIGURA 4.1 – Comparativo dos RPD das perturbações com e sem etapa de melhoria

Como pode ser observado na Figura 4.1, a perturbação PS4 obteve os melhores resultados, com RPD médio de 7,7% com a etapa de melhoria. A PS4 é aquela que faz o intercâmbio de tarefas intermáquinas.

Em seguida, ficaram as perturbações PS5, com 9,2%, a PS2, com 10,9% e a PS3, com 11,0% . E por fim, as piores foram a PS6 e a PS1, respectivamente, com 12,3% e 12,5% de desvio. A PS6 faz a realocação de subsequências intermáquinas e a PS1 realiza a troca de tarefas adjacentes intramáquina.

A Tabela 4.2 a seguir apresenta os valores do RPD da solução inicial em relação ao limitante inferior. Antes da análise das perturbações, foram feitos experimentos aplicando a etapa de melhoria na solução inicial.

TABELA 4.4 - RPD global da solução inicial em relação ao limitante inferior

	SI
sem a etapa de melhoria	13,8
com a etapa de melhoria	13,8

Nota-se aqui que não houve variação, ou seja, não houve redução do *makespan* ao aplicar a etapa de melhoria na solução inicial, indicando o equilíbrio das cargas das máquinas nesta solução, pois a melhoria opera e é eficaz quando há desbalanceamento das cargas. Foram verificados os 720 problemas resolvidos e em nenhum deles houve qualquer redução do *makespan* com a etapa de melhoria na solução inicial.

Além disso, não há problema no fato da solução inicial não estar tão próxima do valor do limitante inferior (com 13,8% de desvio), pois o objetivo do trabalho é fazer com que a meta-heurística opere e atinja melhores resultados. E conforme já salientado, o melhor resultado médio obtido foi de 7,7% de RPD em relação ao limitante inferior com a perturbação PS4.

O algoritmo para gerar a solução inicial (SI) foi elaborado com base nas características do problema. Como já salientado, a ideia do SI então seria que a meta-heurística, que aplica perturbações, melhore esta solução inicial. Porém, no caso do SA, ele admite com determinada probabilidade um solução pior do que a anterior (justamente na tentativa de fugir dos ótimos locais). Isto acarreta a possibilidade do SA finalizar com uma solução pior do que a solução inicial.

Então na Tabela 4.3, os desvios positivos (das perturbações em relação à SI) representam os problemas em que o SA forneceu soluções melhores do que a SI. Já os desvios negativos indicam os casos em que o SA forneceu uma solução pior do que a inicial; seguidos de suas respectivas ocorrências.

TABELA 4.5 - RPD das perturbações em relação à solução inicial

	sem melhoria						**com melhoria**					
	PS1	**PS2**	**PS3**	**PS4**	**PS5**	**PS6**	**PS1**	**PS2**	**PS3**	**PS4**	**PS5**	**PS6**
Desvios positivos (melhora)	3,8	3,9	4,0	5,3	4,3	3,6	3,8	4,1	4,2	5,3	4,6	3,7
Número de problemas com melhora	425	523	510	673	618	441	427	534	523	676	631	455
Percentual de problemas com melhora	59%	73%	71%	93%	86%	61%	60%	74%	73%	94%	88%	63%
Desvios negativos (piora)	-3,6	-3,3	-3,3	-2,8	-2,6	-3,4	-3,4	-3,2	-3,3	-2,5	-3,6	-3,6
Número de problemas com piora	295	197	210	47	102	279	293	186	197	44	89	265
Percentual de problemas com piora	41%	27%	29%	6%	14%	39%	41%	26%	27%	6%	12%	37%

A seguir a Figura 4.2 descreve o RPD das perturbações para os desvios positivos (melhora) em relação ao número de problemas.

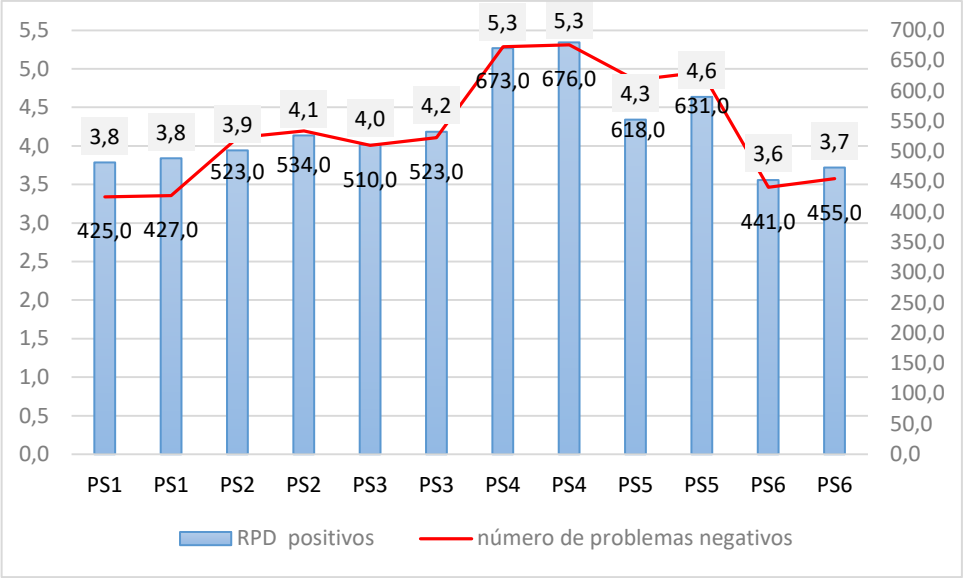


FIGURA 4.2 – RPD das perturbações com desvios positivos pelo número de problemas

As perturbações se mantiveram praticamente constantes sem etapa de melhoria e com etapa de melhoria, na mesma proporção o número de problemas. Com melhores desvios, na etapa de melhoria, a perturbação PS4 (5,3%) e desvios menores as perturbações PS1 (3,8%) e PS6 (3,7%).

A Figura 4.3 apresenta o comparativo visual do RPD das perturbações, para os desvios negativos (piora) em relação ao número de problemas.

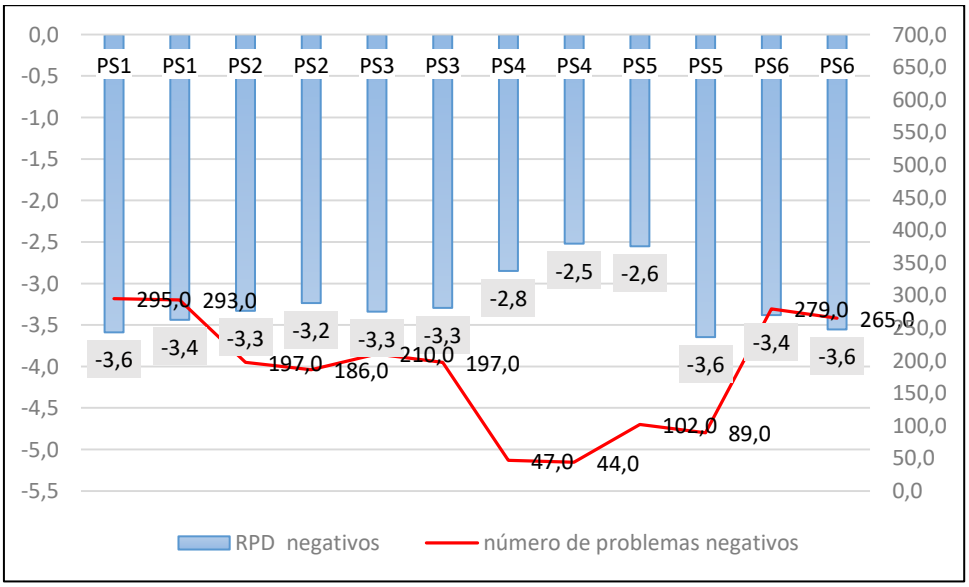


FIGURA 4.3 - RPD das perturbações com desvios negativos pelo número de problemas

Já os desvios negativos ocorrerão em menor quantidade. As perturbações que tiveram variação inferior, na etapa de melhoria, foi a PS4 (-2,5%) e PS5(-2,6%) e que foram as perturbações que tiveram melhores desempenhos globais. As perturbações PS1 (-3,4%) e PS6 (-3,6%) se mantiveram com as maiores diferenças.

Para analisar os resultados das perturbações em relação ao limitante inferior, geraram-se as seguintes tabelas e seus respectivos gráficos. Foram todos separados pelo número de máquinas – 2, 3, 5 e 8 –, e demonstram o desempenho de cada perturbação para cada opção do número de tarefas. Foram considerados os valores obtidos após execução da etapa de melhoria, ou seja, o resultado final da meta-heurística.

A seguir, são apresentados na Tabela 4.4 os desvios das perturbações para os problemas com 2 máquinas.

TABELA 4.6 - RPD das perturbações em problemas com 2 máquinas

n	**PS1**	**PS2**	**PS3**	**PS4**	**PS5**	**PS6**
10	15,6	11,7	11,9	8,2	11,0	13,7
20	7,6	5,3	5,1	3,4	5,2	6,7
30	3,8	3,7	3,3	2,1	3,3	4,4
50	3,2	1,9	2,3	1,2	2,0	2,5
80	2,3	1,4	1,5	0,8	1,2	1,8
100	1,4	1,1	1,3	0,7	1,1	1,5

A seguir a Figura 4.4 descreve o RPD das perturbações em relação ao limitante inferior para 2 máquinas.

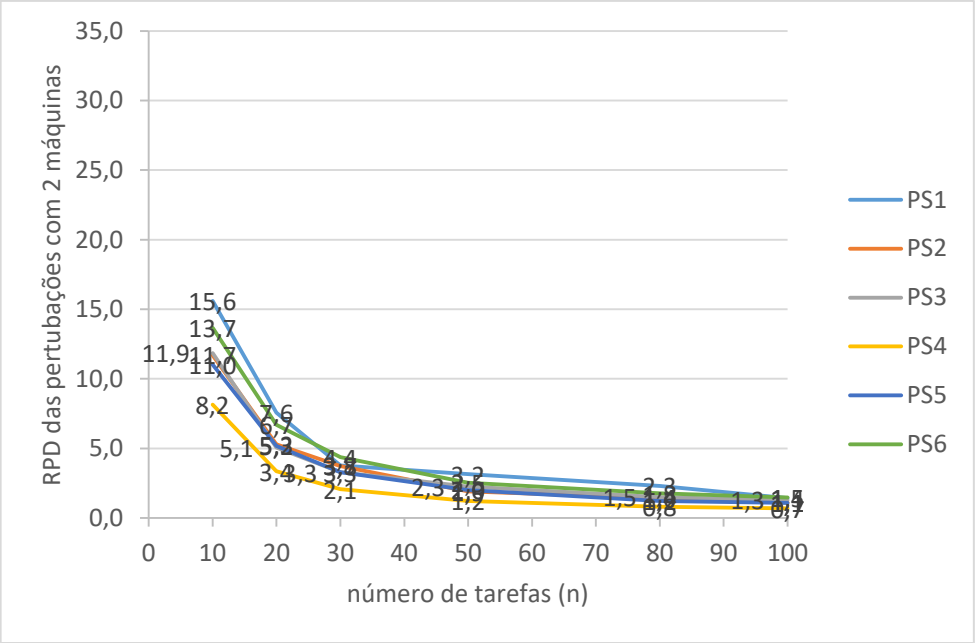


FIGURA 4.4 – RPD das perturbações em problemas com 2 máquinas

Em problemas com duas máquinas, todas as perturbações tiveram desempenho semelhantes. Os maiores valores de RPD ocorreram com 10 tarefas, havendo também a maior amplitude entre a melhor perturbação (PS4, com 8,2%) e a pior (PS1, com 15,6%).

A partir disso, com o aumento do número de tarefas, os desvios relativos tiveram valores decrescentes, chegando a certa estabilidade nas opções de 80 e 100 tarefas.

Nos problemas de maior porte, com 100 tarefas, os RPD de todas as perturbações ficaram muito próximos. Nota-se portanto a maior dispersão dos resultados em problemas de pequeno porte e, quanto maior o porte do problema, menor a variação dos desvios.

TABELA 4.7 - RPD das perturbações em problemas com 3 máquinas

n	PS1	PS2	PS3	PS4	PS5	PS6
10	23,6	22,4	19,5	17,9	18,8	26,4
20	14,4	10,9	10,8	7,1	10,2	13,4
30	8,8	6,1	7,1	4,3	6,2	8,3
50	5,0	4,8	4,6	2,3	3,3	5,4
80	3,1	2,6	2,8	1,8	2,4	3,4
100	2,4	2,2	2,2	1,3	1,7	2,7

Segue a Figura 4.5 que descreve o caso do RPD das perturbações em relação ao limitante inferior para 3 máquinas.

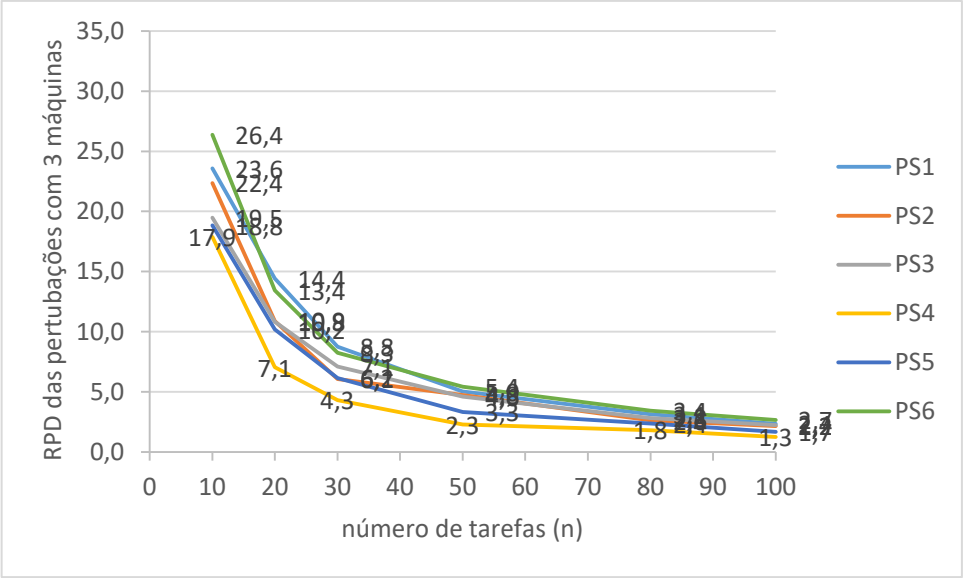


FIGURA 4.5 - RPD das perturbações em problemas com 3 máquinas

No caso dos problemas com três máquinas, o comportamento das curvas das perturbações foi relativamente o mesmo do gráfico anterior. A diferença é que neste caso, os valores dos desvios foram maiores do que com duas máquinas, principalmente em problemas de pequeno e médio porte (10, 20, 30 e 50 tarefas).

TABELA 4.8 - RPD das perturbações em problemas com 5 máquinas

n	PS1	PS2	PS3	PS4	PS5	PS6
10	27,0	25,8	26,0	13,0	18,4	29,8
20	27,4	22,8	22,7	18,0	18,7	24,4
30	18,6	15,6	16,5	12,3	12,5	17,4
50	10,2	9,8	10,1	6,0	7,8	10,5
80	6,8	5,8	6,2	3,6	4,6	6,5
100	5,7	5,0	4,4	3,1	3,6	5,5

Segue a Figura 4.6 que descreve o caso do RPD das perturbações em relação ao limitante inferior para 5 máquinas.

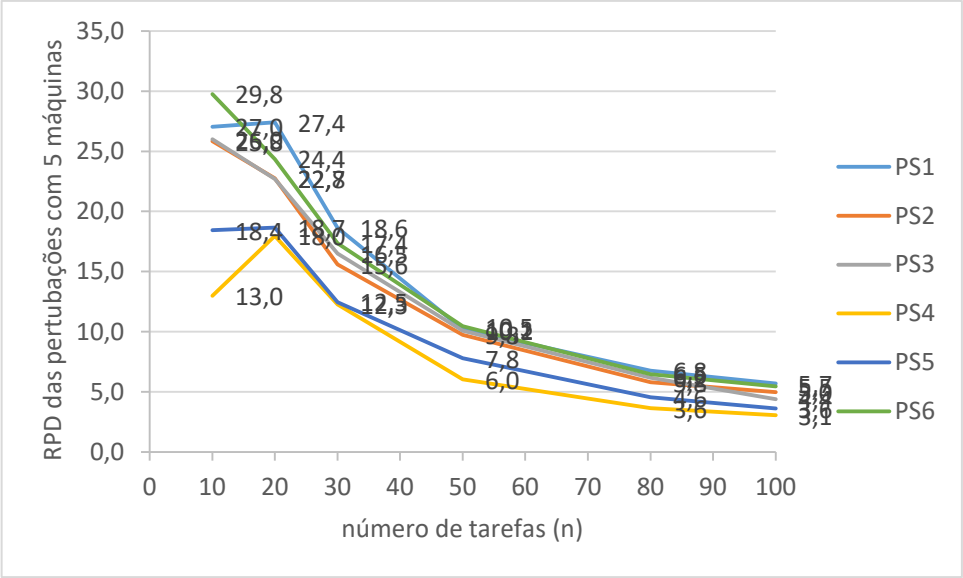


FIGURA 4.6 - RPD das perturbações em problemas com 5 máquinas

Neste gráfico, para cinco máquinas, diferente dos anteriores (2 e 3 máquinas) o comportamento inicial das tarefas, entre 10 e 20, é bastante instável; em metade das perturbações (PS1, PS4 e PS5), os valores para 20 tarefas são maiores que os para 10 tarefas, após isso, elas retornam às tendências das demais. Novamente a perturbação PS4 apresenta os melhores, contudo, há de ressaltar o desempenho da PS5, que se aproxima de PS4.

A seguir os valores da Tabela 4.7 para as perturbações no caso de 8 máquinas com etapa de melhoria.

TABELA 4.9- RPD das perturbações em problemas com 8 máquinas

n	PS1	PS2	PS3	PS4	PS5	PS6
10	5,9	5,7	5,7	2,8	3,8	8,3
20	34,6	30,7	33,3	25,6	30,8	34,1
30	32,2	28,4	29,7	23,3	23,5	30,6
50	18,7	18,0	16,8	12,9	14,5	17,8
80	12,1	10,2	11,1	8,0	9,1	11,0
100	9,9	8,7	8,8	6,2	7,1	9,3

Segue a Figura 4.7 que descreve o caso do RPD das perturbações em relação ao limitante inferior para 8 máquinas.

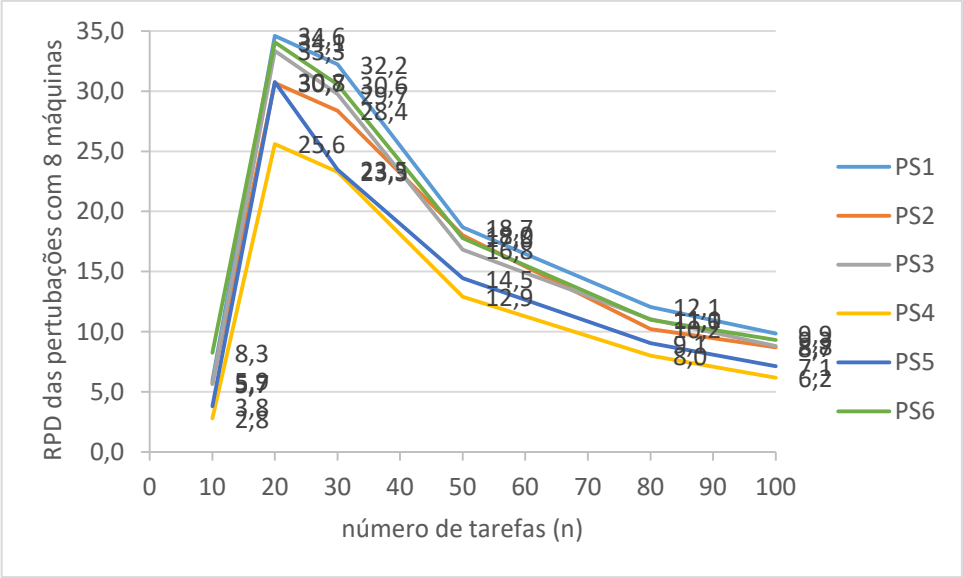


FIGURA 4.7 - RPD das perturbações em problemas com 8 máquinas

Esta representação para problemas com 8 máquinas é a que apresenta os menores desvios para problemas com 10 tarefas (em torno de 5,0%). Porém, já com 20 tarefas, os desvios aumentam substancialmente, passando a ficar em torno de 30,0%, e voltando ao comportamento decrescente para as demais opções do número de tarefas.

Comparados às demais opções do número de máquinas, os desvios aqui ficaram sempre com valores maiores. E novamente, a perturbação PS4 teve sempre os melhores resultados.

Com 8 máquinas e 10 tarefas, temos praticamente uma tarefa por máquina, isso causou a redução do RPD. A medida que aumentou o número de tarefas para 20 o comportamento foi o mesmo para o caso com 2, 3 e 5 máquinas. E os desvios decresceram na medida em que cresceu o número de tarefas, novamente igual aos casos com 2, 3 e 5 máquinas.

Nesta análise por número de máquinas, em todas as opções do número de tarefas, a PS4 sempre obteve o melhor desempenho dentre os esquemas de perturbação; assim vimos que o que foi apresentado na análise global prevaleceu nos demais gráficos. Além disso, a perturbação PS5 foi a que mais se aproximou dos resultados da PS4 em todos os casos, entretanto não chegou nem a empatar em nenhuma. Já os piores resultados foram alternandos entre as perturbações PS1 e PS6. Isto também mantém a coerência com a análise global (apresentado na Figura 4.1).

A maior diferença no desempenho dos métodos foi verificada em problemas com 10 e 20 tarefas, não havendo um padrão de comportamento das curvas nestas opções. Todavia, em todos os casos, com o aumento do número de tarefas, existe uma contínua melhoria

(decaimento dos desvios) e os melhores resultados ocorrem com 100 tarefas , com exceção dos problemas com 8 máquinas (neste caso, os problemas com 10 tarefas apresentam resultados melhores que as outras opções do número de tarefas).

Agora será analizado o desempenho das perturbações em relação ao limitante inferior para os diferentes intervalos de datas de liberação. a seguir a Tabela 4.8 apresenta os desvios para o intervalo de liberação [1,49].

TABELA 4.10 - RPD das perturbações em problemas com intervalo de liberação [1,49]

N	PS1	PS2	PS3	PS4	PS5	PS6
10	15,5	16,6	14,1	5,7	8,5	16,3
20	14,6	12,4	14,5	7,9	10,7	12,0
30	9,7	9,3	9,1	4,7	6,8	9,4
50	6,0	5,6	5,2	2,7	4,1	5,5
80	3,7	3,2	4,0	1,7	2,5	3,3
100	3,2	2,9	2,7	1,2	1,7	2,7

Segue a Figura 4.8 que descreve os valores do RPD das perturbações em relação ao limitante inferior para o intervalo de liberação [1,49].

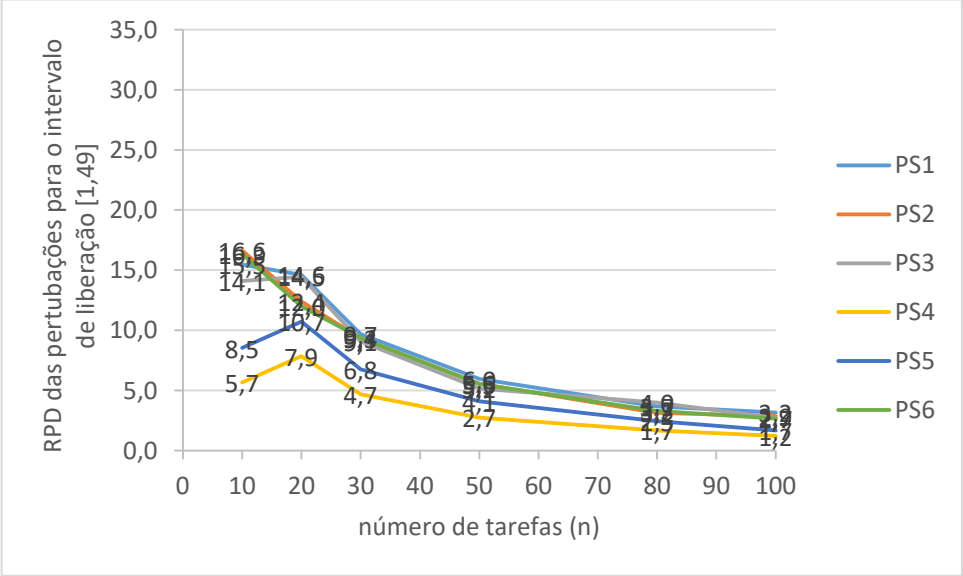


FIGURA 4.8 - RPD das perturbações em problemas com intervalo de liberação [1,49]

Quando o intervalo das datas de liberação variou entre 1 e 49, o comportamento das perturbações para 10 e 20 tarefas é bastante instável. Os piores resultados são das perturbações PS2, PS6, PS1 e PS3. Os melhores resultados são das perturbações PS4 e PS5.

Diferentemente das outras opções de intervalos, como se verá logo adiante, aqui as perturbações PS2 e PS6 apresentaram desvios sempre decrescentes, enquanto as demais tiveram um aumento do desvio quando se passa de 10 para 20 tarefas e em seguinda o decrescimento contínuo.

A seguir os valores da Tabela 4.9 para intervalo de liberação [1,99].

TABELA 4.11 - RPD das perturbações em problemas com intervalo de liberação [1,99]

n	PS1	PS2	PS3	PS4	PS5	PS6
10	16,4	13,9	13,0	9,7	13,2	18,4
20	19,4	15,8	16,3	12,6	15,1	19,8
30	15,6	12,0	12,5	9,4	10,9	13,1
50	7,5	7,6	6,8	4,1	5,6	8,1
80	5,4	4,3	4,4	3,0	3,6	5,5
100	4,3	3,7	3,8	2,2	3,0	4,3

Segue a Figura 4.9 que descreve o caso do RPD das perturbações em relação ao limitante inferior para o intervalo de liberação [1,99].

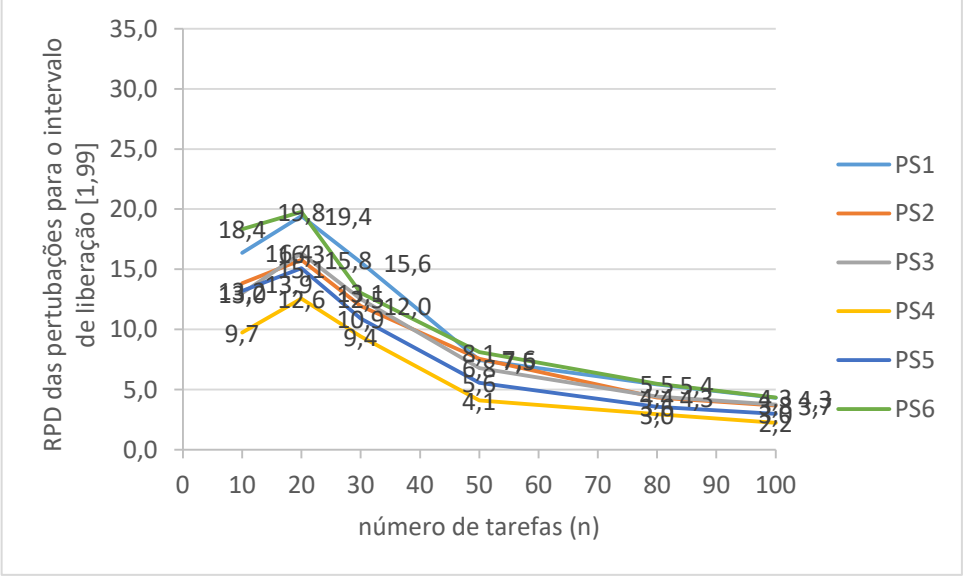


FIGURA 4.9 - RPD das perturbações em problemas com intervalo de liberação [1,99]

Com a ampliação do intervalo de data de liberação das tarefas, entre 1 e 99, o comportamento se mantem instável para 10 e 20 tarefas e a medida que se aumenta o número de tarefas obtem-se melhores resultados. Os piores resultados são as perturbações PS6 e PS1, e os retornam os melhores, PS4 e PS5.

Todas as perturbações tiveram desvios menores com 10 tarefas em relação aos problemas com 20 tarefas e em seguida, com o aumento do número de tarefas, os desvios foram sempre decrescentes.

A seguir os valores da Tabela 4.10 para intervalo de liberação [50,149].

TABELA 4.12 - RPD das perturbações em problemas com intervalo de liberação [50,149]

n	PS1	PS2	PS3	PS4	PS5	PS6
10	22,3	20,0	20,8	16,1	17,6	23,3
20	29,3	24,0	23,4	20,2	23,6	27,5
30	22,3	19,3	20,9	17,6	17,1	22,1
50	14,4	12,9	13,4	10,1	11,5	13,6
80	9,1	7,6	7,8	6,2	7,1	8,3
100	7,0	6,2	6,1	5,0	5,5	7,2

Segue a Figura 4.10 que descreve o caso do RPD das perturbações em relação ao limitante inferior para o intervalo de liberação [50,149].

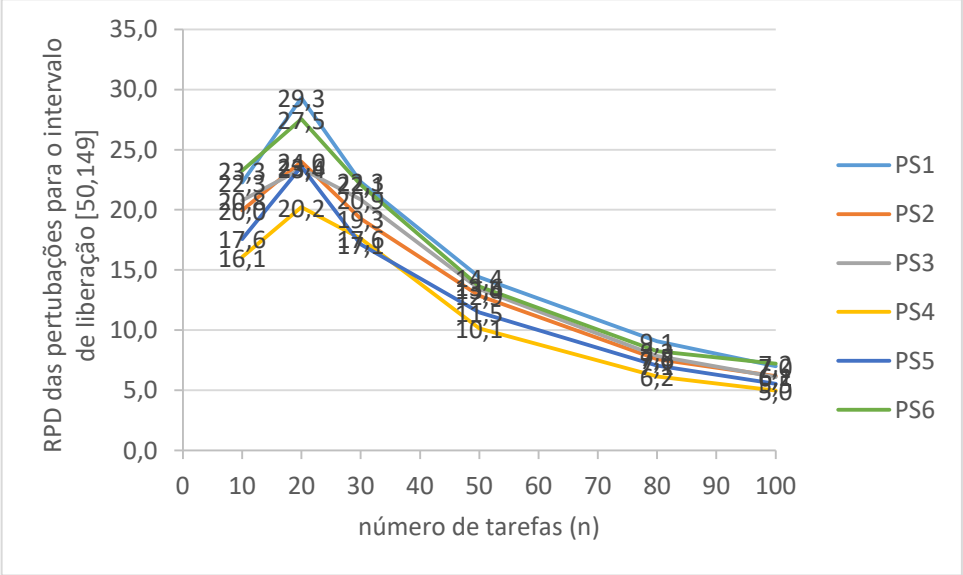


FIGURA 4.10 - RPD das perturbações em problemas com intervalo de liberação [50,149]

Na opção do intervalo de datas de liberação entre 50 e 149, o comportamento das curvas é parecido com os gráficos anteriores, com um aumento dos desvios quando se passa de 10 para 20 tarefas e em seguida, valores decrescentes. Porém, nesta opção de intervalos datas de liberação, os valores dos desvios foram sempre superiores às demais.

O tempo médio de CPU despendido na resolução dos problemas foi, em média, de 0,07 milissegundos, o que não compromete a eficiência computacional dos algoritmos.

Demostra-se, assim, que o algoritmo *Simulated Annealing* é eficiente na solução do problema de máquinas paralelas idênticas com datas de liberação e *setup* independente.

Outra ferramenta para a análise dos resultados foi da Análise de Variância (ANOVA) cujo nível de significância considerado foi de 0,05. Segue o fator e seus respectivos testes de hipóteses:

Fator: esquema de perturbação.

H_0: Não há diferença significativa entre os esquemas de perturbação (PS1, PS2, PS3, PS4, PS5 e PS6) pelo resultado do *makespan*

H_1: Existe diferença significativa entre os esquemas de perturbação (PS1, PS2, PS3, PS4, PS5 e PS6), para pelo menos um dos algoritmos, pelo resultado do *makespan*.

O resultado da tabela ANOVA é apresentado na Tabela 4.11 a seguir.

TABELA 4.13- Resultados da tabela ANOVA para o fator esquema de perturbação

Fonte da variação	**SQ**	**GL**	**MQ**	**F(calculado)**	**P-valor**	**F crítico**
Entre perturbações	406817	5	81363,39	0,092045	0,993503	2,216172
Erro	$3{,}81.e^{+9}$	4314	883952,1			
Total	$3{,}81.e^{+9}$	4319				

No esquema apresentado, o valor da estatística teste "F" foi 0,0993503, que é menor que o "F crítico": 2,216172; para o nível de significância igual a 0,05. Daqui, infere-se que a hipótese nula restrita a este fator é aceita, ou seja, não existe diferença significativa entre os esquemas de perturbação pelo resultado do *makespan*. É importante observar ds perturbações dependem nesse caso de fatores, tais como: número de máquinas, tarefas, data de liberação, tempo de processamento e *setup*.

Entretanto, pelo resultado da tabela ANOVA, essa diferença de desempenho entre os esquemas não chega a ser relevante. Sugere-se portanto a resolução do problema tratado utilizando-se o esquema PS4.

5 CONCLUSÕES

A revisão bibliográfica contribuiu com um exame minucioso da literatura e apresentou o estado da arte dos artigos encontrados até o momento, descreveu a evolução do problema tratado organizando a literatura de forma visual, e também realizou uma análise das publicações e a esquematização os artigos em forma de árvore.

Após a contribuição teórica, foram desenvolvidos: o método de solução inicial, a meta-heurística proposta, os esquemas de perturbação e os limitantes inferiores que foram usados na comparação dos resultados.

Neste estudo foram propostos seis algoritmos *Simulated Annealing* com seus respectivos esquemas de perturbação, para o problema de programação em máquinas paralelas idênticas com tempos de processamento, *setup* indendente e datas de liberação. A função objetivo é o *makespan* e os tempos de *setup* são independentes da sequência de tarefas. Este problema é denotado por $P_m \left| s_j, r_j \right| C_{max}$.

Com base na análise dos resultados obtidos na experimentação computacional verifica-se que em geral a perturbação PS4, forneceu resultados superiores tanto para problemas de pequeno porte (10 a 30 tarefas) com para de grande porte (80 a 100 tarefas) em relação a todas as outras perturbações. Nos intervalos das datas de liberação com diferentes amplitudes {1,49}, {1,99} e {50,149} a perturbações que se destacaram novamente foi a PS4 seguida da PS5. Com piores desempenhos a perturbação PS1 e PS6.

Os algoritmos mostraram-se eficientes computacionalmente, com tempo de computação bastante aceitável. Foi proposto também um algoritmo para geração de sequência inicial, baseado nas estruturas específicas do problema. Os resultados da Análise de Variância revelaram que não existe diferença significativa entre os esquemas de perturbação em termos do *makespan*. Portanto, sugere-se a utilização do esquema PS4, que faz o intercâmbio de tarefas intermáquinas, que forneceu o melhor resultado em relação ao limitante inferior.

Para novas pesquisas seria interessante propor novos métodos de solução para os problemas com outros tipos de restrições, como prazos de entrega e tempos de *setup* dependentes. Como continuidade desta pesquisa, pode-se realizar adaptações nos algoritmos de Kurz e Askin (2001) e Montoya-Torres *et al*. (2010) e comparar os resultados com os obtidos neste trabalho. Além disso, em novas pesquisas, pode-se propor novas meta-heurísticas baseadas em outros paradigmas, como *Particle Swarm Optimization* e Colônia de

Formigas, entre outros, e também incluir diferentes restrições no problema, como prazos de entrega e tempos de setup dependentes da sequência.

Constata-se pela análise do RPD em relação a solução inicial que o algoritmo SA proposto pode ser melhorado se for incluída uma verificação de se considerar a SI como solução final, no caso do SA não fornecer uma solução melhor. Uma sugestão também seria de mais testes computacionais partindo de uma solução inicial aleatória.

REFERÊNCIAS

ALCAN P.; BASLIGIL H. A Literature review for the simulated annealing method on the parallel machines since 2003. **14th International Research/Expert Conference "Trends in the Development of Machinery and Associated Technology"** TMT 2010, Yildiz Technical University, Mechanical Faculty, The Department of Industrial Engineering, Mediterranean Cruise, 2010, p.11-30.

ALLAHVERDI, A.; GUPTA, J. N. D.; ALDOWIASAN, T. A. A review of scheduling research involving setup considerations. **OMEGA** 27, p.219-239, 1999.

BASSANEZI, R. C. **Ensino-aprendizagem com modelagem matemática: uma nova estratégia.** 3. ed. São Paulo: Contexto, 2010.

BAKER, K. R. **Introduction to Sequencing and Scheduling.** John Wileys & Sons, Inc., New York, 1974.

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

BEHNAMIAN, J.; ZANDIEH, M.; CHOMI, S. M. T. Parallel-machine scheduling problems with sequence-dependent setup times using ACO, SA and VNS hybrid algorithm. **Expert Systems with Applications**, 2009. p. 9637–9644.

BLOCHER, J. D.; CHAND, S. Scheduling of parallel processors: a posterior bound on LPT sequencing and a two-step algorithm. **Naval Research Logistics**, 1991. p. 273-291.

CHANG, T. Y.; CHOU, F. D.; LEE, C. E. A heuristic algorithm to minimize total weighted tardiness on a single machine with release dates and sequence-dependen t setup times. **Journal of the Chinese Institute of Industrial Engineering**, 2004. p. 289-304

CHEN, B.; LEUNG, J. Y. T. Handbook of scheduling: algorithms, models, and performance analysis. **Boa Raton: Champan & Hall/CRC**, 2004.

COFFMAN, JR. E. G.; GAREY, M. R.; JOHNSON, D. S. An application of bin-packing to multiprocessor scheduling. **SIAM Journal on Computing**, 1978.

CORRÊA, R. C.; FERREIRA, A.; REBREYEND, P. Scheduling multiprocessor tasks with genetic algorithms. **IEEE Transactions on Parallel and Distributed Systems**, 1999. p.825-840

CORREA, H. L.; GIANESI, I. G. N.; CAON, M. Planejamento, Programação e Controle da Produção. São Paulo: **Atlas**, 2006

DAMODARAN, P.; CHANG, P. Y. Heuristics to minimize makespan of parallel batch processing machines, **The International Journal of Advanced Manufacturing Technology**, 2008. p.1005-1020.

FARBER, G.; COVES, A. Overview on sequencing in mixed model flowshop production line

with static and dynamic context. **Universitat Politècnica de Catalunya**, 2005.

FATEMI, G. S. M. T.; JOLAI G. F. A pairwise interchange algorithm for parallel machine scheduling. **Production Planning & Control**, 1998 p.685–689.

FRIESEN, D. K. Tighter bounds for the MULTIFIT processor scheduling algorithm. **SIAM Journal on Computing**, 1984. p.170-185

FUCHIGAMI, H. Y.; MOCCELLIN, J. V. Regras de prioridade para programação em sistemas *flexible flow line* com tempos de *setup* dependentes da sequência. In: **Anais** SIMPÓSIO DE ENGENHARIA DE PRODUÇÃO (XVI SIMPEP), 2009, Bauru, SP.

FUCHIGAMI, H.Y. **Introdução ao Sequenciamento da Produção**. Catalão: UFG, 2013. Material didático. Versão 5.0.

GRAHAM, R. L. Bounds on multiprocessing timing anomalies. **SIAM Journal on Applied Mathematics**, 1969. p.416-421.

GUIGNARD, M. Solving makespan minimization problems with lagrangean decomposition. **Discrete Applied Mathematics**, 1993. p.17-30.

GUPTA, J. N. D.; RUIZ-TORRES, A. J. A LISTFIT heuristic for minimizing makespan on identical parallel machines. **Production Planning & Control**, 2001 p.28–36.

HAX, A. C.; CANDEA, D. **Production and inventory management**. New Jersey: Prentice – Hall, 1984.

HOU, E.; ANSARI, N.; REN, H. A genetic algorithm for multiprocessor scheduling. **IEEE Transactions on Parallel and Distributed Systems**, 1994. p.113-125.

HU, T. C. Parallel sequencing and assembly line problems. **Operations Research**, 1961. p.841-855.

KASHAN, A. H.; KARIMI, B. K.; JENABI M. A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes. **Computers & Operations Research**, 2008. p.1084-1099.

KIRKPATRICK, S.; GELATT, C. D. JR.; VECCHI, M. P. **Optimization by Simulated Annealing**. Science, v. 220, 1983. p.671-675.

KURZ, M. E.; ASKIN, R. G. Heuristic scheduling of parallel machines with sequence dependent setup times. **International Journal of Production Research**, 2001. p.3747-3765

LAHA, D. A simulated annealing heuristic for minimizing makespan in parallel machine scheduling, **Springer Verlag Berlin Heidelberg**, 2012.

LAWLER, E. Employee Involvement in America: A Study of Contemporary Practice. **Houston: American Productivity and Quality Conter**, 1989

LEE, C. Y.; MASSEY, J. D. Multiprocessor scheduling: combining LPT and MULTIFIT. **Discrete Applied Mathematics**, 1988, p.233-245.

LEE, W. C.; WU, C. C.; CHEN, P. Simulated annealing approach to makespan minimization on identical parallel machines. **The International Journal of Advanced Manufacturing Technology**, 2006. p.328-345.

MacCARTHY, B. L.; LIU, J. Y. Addressing a gap in scheduling research – a review of optimization and heuristic methods in production scheduling. **International Journal of Production Research**, London, v.31, 1993. p.59-63.

MARTINS, R. A. Abordagens Quantitativa e Qualitativa. In: MIGUEL, P.A.C. (Org.). **Metodologia de pesquisa em Engenharia de Produção e Gestão de Operações**. Rio de Janeiro: Elsevier, 2010. p. 45-105.

MASTERS, T. **Practical Neural Network Recipes in C**++, Academic Press, 1993

McNAUGHTON, R. Scheduling with deadlines and loss functions. **Management Science**, 1959.

METROPOLIS, N.; *GELATT JR., C. D.; KYPARISIS, G. J.; VECCHI, M. P.* Equation of state calculations by fast computing machines. **The Journal of Chemical Physics**, 1953, p.1087-1099

MIN, L.; CHENG, W. Identical parallel machine scheduling problem for minimizing the makespan using genetic algorithm combined by simulated annealing. **Chinese Journal of Electronics**, 1998. p. 317-332.

MONMA, C. L.; POTTS, C. N. On the complexity of scheduling with batch setup times. **Operations Research**, 1989, p.798-801,

MONMA, C. L.; POTTS, C. N. Analysis of heuristics for preemptive parallel machine scheduling with batch setup times. **Operations Research**, 1993, p.981-995

MONTOYA-TORRES, J. R.; SOTO-FERNANDO, M.; GONZÁLEZ-SOLANO. F. Production scheduling with sequence-dependent setups and job release times **Dyna**, year 77, 2010. p.260-275 Medellin.

MUNTZ, R. R.; COFFMAN, E. G. Optimal preemptive scheduling on two-processor systems. **IEEE Transactions on Computers**, 1969 p.1014-1027.

MUNTZ, R. R.; COFFMAN, E. G. Preemptive scheduling of real-time tasks on multiprocessor systems. **Journal of the ACM**, 1970, p.324-340.

OSMAN, L.H.; POTTS, C.N. Simulated annealing for permutation flow-shop scheduling. Omega – **The International Journal of Management Science**, Oxford, v.17, n.6, p. 551-557.

OVACIK, I. M.; UZSOY, R. Worst-case error bounds for parallel machine scheduling problems with bounded sequence-dependent setup times. **Operations Research Letters**, 1993, p.251-265.

PINEDO, M. L. **Scheduling: Theory, Algorithms, and Systems**. New York: Springer, 2012.

RAJGOPAL, J.; BIDANDA, B. On scheduling parallel machines with two setup classes. **International Journal of Production Research**, 1991. p.2443-2457.

SARAMAGO, S. F. P.; STEFFEN, Jr. V. Dynamic Optimization for the Trajectory Planning of Robot Manipulators in the Presence of Obstacles. **Journal of the Brazilian Society of Mechanical Sciences**, Brazil, v. XXI, n. 3, 2004. p. 371-398.

SENTHILKUMAR, P.; NARAYANAN, S. Simulated Annealing algorithm to minimize makespan in single machine scheduling problem with uniform parallel machines. **Intelligent Information Management**. Vol.3, 2011. p.22-50.

SLACK, N.*;* RAVETTI, M. G.; TAHERI, F.; PARDALOS, P. M. **Administração da produção**. 1ed. São Paulo: Atlas. Edição: compacta, 1999.

SOUZA, A. B. D.; MOCCELLIN, J. V. Meta-heurística híbrida Algoritmo Genético e Busca Tabu, para a programação de operações *flow shop,* **Anais** Simpósio Brasileiro de Pesquisa Operacional, 2000, Viçosa, MG.

TANG, C. S. Scheduling batches on parallel machines with major and minor set-ups. **European Journal of Operational Research**, 1990, p.28-40

ZACCARELLI, S. B. **Programação e Controle da Produção**. 8a.edição. São Paulo: Pioneira, 1987.

GLOSSÁRIO

$\text{carga}_{\text{maior}}$ – carga da máquina de maior carga.

$\text{carga}_{\text{menor}}$ – carga da máquina de menor carga.

CPLEX – é um software de otimização que utiliza de métodos de resolução da programação matemática para resolver modelos e facilitar a análise de sensibilidade dessas funções, implementado em linguagem C.

lower bound – valor calculado de forma independente da programação, garantindo se que seja menor ou igual ao *makespan* ótimo ($LB \leq C^*_{max}$).

makespan – duração total da programação.

máquinas paralelas – todas as máquinas são iguais e realizam o mesmo procedimento.

release date – data de liberação de uma tarefa, ou seja, ponto onde possa começar a ser executada.

setup time – tempo de preparação da máquina.

Simulated Annealing – algoritmo baseado no processo de resfriamento térmico dos metais.

APÊNDICE A

EXEMPLO ILUSTRATIVO DO ALGORITMO *SIMULATED ANNEALING* PROPOSTO

Para melhor compreensão e apresentação do algoritmo *Simulated Annealing* proposto, será desenvolvido um exemplo numérico ilustrativo do seu funcionamento, considerando um problema com duas máquinas paralelas idênticas e quatro tarefas, cujos dados são fornecidos na Tabela A.1 a seguir.

TABELA A.0.1 – Dados para experimentação computacional

Tarefas	J_1	J_2	J_3	J_4
Tempo de processamento (p_j)	4	7	10	2
Tempo de *setup* independente (s_j)	2	5	1	6
Data de liberação (r_j)	11	3	8	9

Conforme a descrição do algoritmo apresentada na seção 3.4, a sua execução ilustrativa segue o seguinte procedimento:

Passo 1

- Dados*: n* = 4, *m* = 2
- Tempos de processamento e de *setup* e datas de liberação conforme a Tabela A.1.
- Parâmetros: $T_i = 60,\ T_f = 0.01,\ r = 0.85$
- Esquema de perturbação: *PS4* (Intercâmbio de tarefas intermáquinas)

Os parâmetros e o esquema de perturbação foram escolhidos arbitrariamente, apenas para efeito de ilustração.

ITERAÇÃO 1

Passo 2

Utilizando o Algoritmo para solução inicial descrito na seção 3.2 e os valores da Tabela A.1, obtém-se a sequência *S*. O algoritmo faz a ordenação crescente das tarefas pelos respectivos valores de $\frac{r_j}{p_j+s_j}$, conforme apresentado na Tabela A.2.

TABELA A.0.2 – Cálculo do valor para ordenação inicial

Tarefas	J_1	J_2	J_3	J_4
Valor $\frac{r_j}{p_j+s_j}$	1.8	0.2	0.7	1.1

Assim, a solução inicial S fica $J_2 - J_3 - J_4 - J_1$. Em seguida, ainda pelo Algoritmo para solução inicial, aloca-se uma tarefa de cada vez à máquina de menor carga, chegando à programação mostrada no gráfico de Gantt da figura A.1.

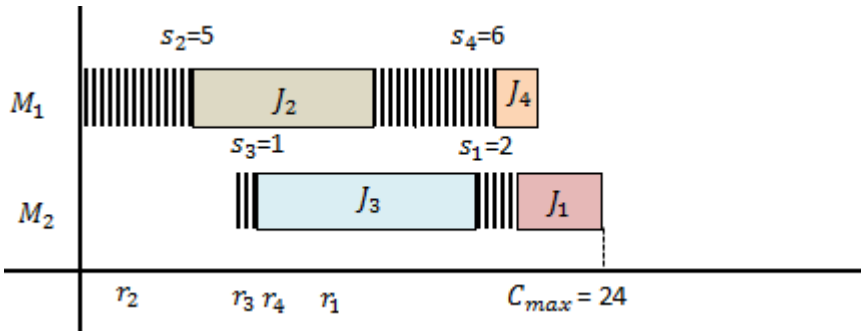


FIGURA A.1 – Programação da solução inicial

Assim, o *makespan* desta sequência S é $C_{max}(S) = 24$.

O Passo 2 do algoritmo encerra fazendo $S^* = S = J_2 - J_3 - J_4 - J_1$, sendo S^* a melhor sequência encontrada até o momento, e $T = T_i = 60$, que é a temperatura inicial.

Passo 3

Para encontrar uma nova solução S' na vizinhança de S, utiliza-se do esquema de perturbação inter-máquinas *PS4*, que seleciona aleatoriamente duas tarefas que sejam de duas máquinas diferentes, também escolhidas aleatoriamente, e em seguida troca-as.

Como o problema tem apenas duas máquinas, é suficiente que se selecione uma tarefa de cada máquina. Assim, por sorteio, foram selecionadas as tarefas J_4 da máquina M_1 e J_3 da máquina M_2. Fazendo a troca das duas, tem-se a programação apresentada na Figura A.2.

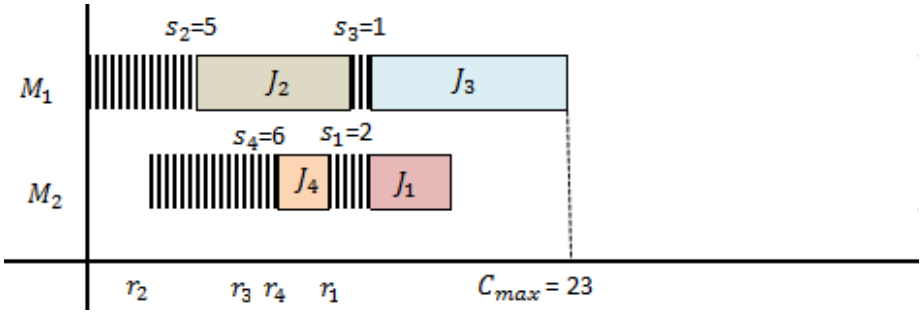


FIGURA A.2 – Programação após a perturbação PS4

O *makespan* da nova sequência *S'* é $C_{max}(S') = 23$.

Calcula-se a diferença entre o *makespan* obtido e o anterior: Δ = 23 – 24= – 1.

Como $\Delta \leq 0$, então faça $S = J_2 - J_4 - J_3 - J_1$. Se $C_{max}(S) < C_{max}(S^*)$, ou seja $23 < 24$, então $S^* = J_2 - J_4 - J_3 - J_1$. Vai para o Passo 4.

Passo 4

A temperatura inicial é reduzida pelo fator de resfriamento ($r = 0.85$):

$T = 0.85x60 \rightarrow T = 51$.

Verifica-se se atingiu o critério de parada, que é a temperatura final ($T_f = 0.01$):

Como $T = 51 > T_f = 0.01$, não atingiu o critério de parada. Portanto, volta ao Passo 3.

ITERAÇÃO 2

Passo 3

Uma nova solução vizinha *S'* é encontrada pelo mesmo esquema de perturbação *PS4*. Assim, por sorteio, foram selecionadas as tarefas J_2 da máquina M_1 e J_1 da máquina M_2. Fazendo a troca das duas, tem-se a programação apresentada na Figura A.3.

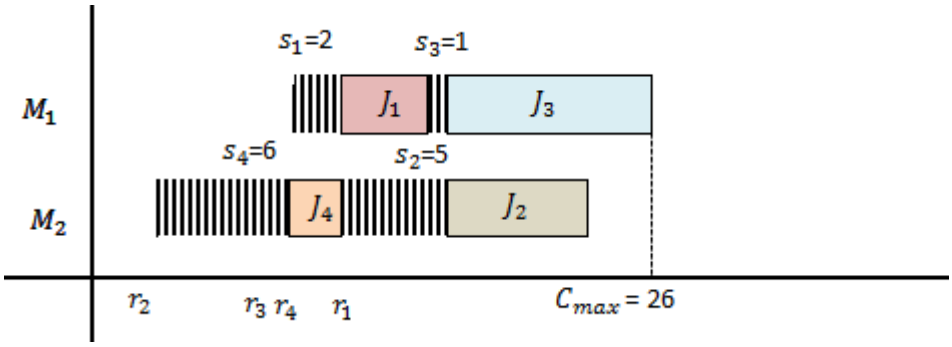


FIGURA A.3 – Programação da nova solução após a perturbação PS4

O *makespan* da nova sequência *S'* é $C_{max}(S') = 26$.

Calcula-se a diferença entre o *makespan* obtido e o anterior: Δ = 26 – 23= 3.

Como $\Delta > 0$, então gera um número aleatório u uniformemente distribuído entre 0 e 1 para aceitar ou não a nova solução obtida. Se $u \leq e^{-\Delta/T}$, então a nova solução é aceita (mesmo sendo pior que a anterior). Foi gerado o número aleatório u = 1. Neste caso, como $e^{-\Delta/T}$=0.94, então $u > e^{-\Delta/T}$ e a nova solução é aceita.

Passo 4

A temperatura é novamente reduzida pelo fator de resfriamento ($r = 0.85$):

$T = 0{,}85x51 \rightarrow T = 43{,}4$.

Verifica-se se atingiu o critério de parada, que é a temperatura final ($T_f = 0.01$): Como $T = 43{,}4 > T_f = 0.01$, não atingiu o critério de parada. Portanto, volta ao Passo 3.

ITERAÇÃO 3

O procedimento continua iterativamente até que se atinja a temperatura final. Como este exemplo é apenas a título de ilustração do funcionamento do algoritmo, as iterações seguintes não serão apresentadas.

O algoritmo armazena em *S** a melhor solução encontrada em todas as iterações e em $C_{\max}(S^*)$ o seu *makespan*. O último passo do algoritmo executa uma etapa de melhoria, conforme ilustrado a seguir.

Passo 5

O objetivo da etapa de melhoria é equilibrar a carga das máquinas, de acordo com o seguinte critério: $carga_{maior} - carga_{menor} > p_u + s_u$. Ou seja, se a diferença a carga da máquina de maior carga ($carga_{maior}$) e carga da máquina de menor carga ($carga_{menor}$) for maior do que os tempos de processamento e de *setup* ($p_u + s_u$) da última tarefa da máquina de maior carga (J_u), então é possível deslocar a tarefa J_u para a máquina de menor carga, em busca da melhoria do *makespan*. Havendo de fato a redução, a melhor solução é considerada e repete-se o procedimento de verificação das cargas da etapa de melhoria até que nenhuma redução do *makespan* possa ser obtida.

Considerando a melhor solução encontrada $S^* = J_2 - J_4 - J_3 - J_1$ (da Figura A.2), como a última tarefa da carga de maior carga é a J_3 e $carga_{maior} - carga_{menor} < p_3 + s_3$, então não é possível efetuar melhoria no *makespan* e o algoritmo finaliza.

Entretanto, para ilustrar a etapa de melhoria, considere, por exemplo, a programação representada na Figura A.4 a seguir.

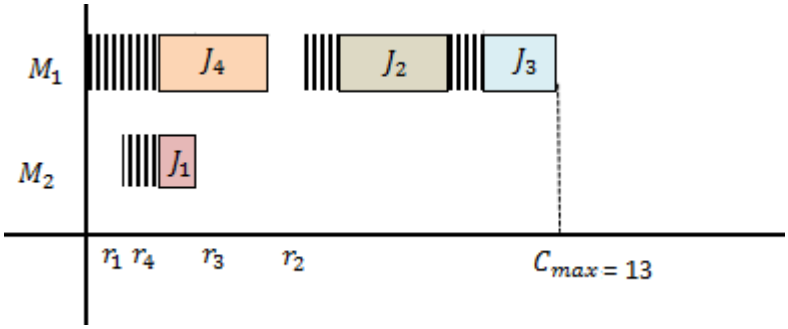


FIGURA A.4 – Representação da solução em que a etapa de melhoria será aplicada

Passo 5.1

Neste caso, como a última tarefa da máquina de maior carga é J_3 e $carga_{maior} - carga_{menor} > p_3 + s_3$, então o algoritmo aloca a tarefa J_3 na máquina M_2, fazendo $S' = J_4 - J_1 - J_3 - J_2$ a nova programação, conforme a Figura A.5.

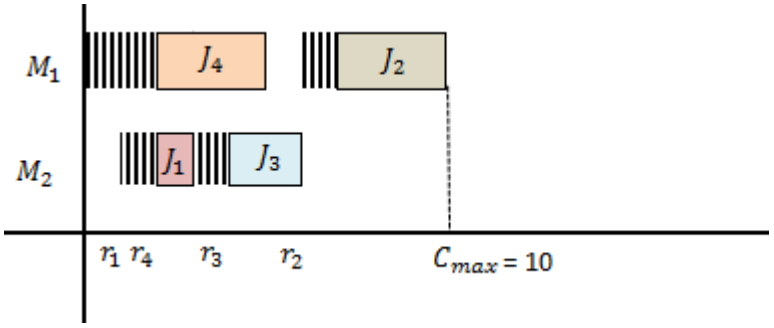


FIGURA A.5 – Representação da solução com a etapa de melhoria

Passo 5.2

Se a nova solução *S'* for melhor do que a solução *S**, então a nova solução é considerada e volta-se ao Passo 5.1. Senão, segue ao Passo 5.3.

Passo 5.1

O retorno a este passo verifica que como $carga_{maior} - carga_{menor} < p_2 + s_2$, nenhuma melhoria pode ser feita, levando ao Passo 5.3.

Passo 5.3

O algoritmo finaliza, retornando a melhor solução encontrada (*S**) e o *makespan* ($C_{max}(S^*)$).

Conforme descrito na seção 3.4, foram propostos limitantes inferiores do *makespan* para avaliar a qualidade da solução fornecida pelo algoritmo *Simulated Annealing*.

Cálculo dos limitantes inferiores

Utilizando as expressões dos limitantes inferiores propostos e os dados do problema exemplo da Tabela A.1, obteve-se:

$$LB_1 = \max_j(s_j + p_j) \rightarrow LB_1 = \max_j(6 + 10) = 16$$

$$LB_2 = \max_j(r_j + p_j) \rightarrow LB_2 = \max_j(11 + 10) = 21$$

$$LB_3 = \frac{\sum_{j=1}^{m}\{\max(r_{[j]}, s_{[j]}) + p_{[j]}\} + \sum_{j=m+1}^{n}(s_{[j]} + p_{[j]})}{m} \rightarrow$$

$$LB_3 = \frac{\sum_{j=1}^{2}\{\max(11,2) + 4 + \max(3,5) + 7\} + \sum_{j=3}^{4}(1 + 10) + (6 + 2)}{2} = 23$$

$$LB_4 = max(s_{[m]}, r_{[m]}) + p_{[m]} + s_{[m+1]} + p_{[m+1]} \rightarrow LB_4 = max(5,3) + 7 + 1 + 10 = 23$$

$$LB = \max(LB_1, LB_2, LB_3, LB_4) \rightarrow LB = \max(16, 21, 23,23) = 23$$

Portanto, LB = 23 e o *makespan* obtido pelo algoritmo é $C_{\max}$ = 23.

Por definição, a expressão do cálculo do limitante inferior deve garantir que o seu valor seja no máximo igual ao *makespan* ótimo ($LB \leq C^*_{max}$). Se o valor do LB foi igual ao $C_{\max}$, sabe-se que o algoritmo forneceu o *makespan* ótimo do problema.

APÊNDICE B

TABELA B.1 – RPD de cada classe de problema

			sem melhoria							com melhoria						
n	m	rj	SI	PS1	PS2	PS3	PS4	PS5	PS6	SI	PS1	PS2	PS3	PS4	PS5	PS6
10	2	[1,49]	8,9	5,4	5,4	6,2	1,7	3,7	5,4	8,9	5,4	5,4	6,2	1,7	3,7	5,4
10	3	[1,49]	15,5	14,7	15,8	12,1	6,7	8,5	15,1	15,5	14,7	14,8	12,1	6,3	8,5	15,1
10	5	[1,49]	25,5	27,6	30,6	29,1	10,7	17,8	33,9	25,5	27,6	28,8	26,6	10,6	17,1	33,9
10	8	[1,49]	12,8	14,3	14,7	8,9	3,7	4,2	11,0	12,8	14,3	12,6	8,9	3,7	4,0	13,4
10	2	[1,99]	14,9	12,5	8,4	7,9	5,0	8,5	10,5	14,9	12,5	8,4	7,9	5,0	8,5	10,5
10	3	[1,99]	30,7	20,0	16,7	14,8	16,3	16,3	25,2	30,7	20,0	16,7	14,8	16,3	15,6	24,3
10	5	[1,99]	34,5	37,0	27,5	29,3	15,2	24,3	33,7	34,5	33,0	26,9	26,5	15,2	21,6	31,1
10	8	[1,99]	4,4	2,5	5,9	2,9	2,4	2,7	6,0	4,4	0,0	3,4	2,9	2,4	7,1	7,5
10	2	[50,149]	33,1	29,0	21,3	22,0	18,3	20,9	27,4	33,1	29,0	21,3	21,4	17,8	20,9	25,1
10	3	[50,149]	47,5	36,0	38,5	32,8	31,1	35,8	39,7	47,5	36,0	35,6	31,6	31,1	32,4	39,7
10	5	[50,149]	32,5	20,6	24,5	26,2	13,2	17,4	25,0	32,5	20,5	21,7	24,9	13,2	16,6	24,3
10	8	[50,149]	4,2	3,6	1,2	5,1	2,2	0,6	3,9	4,2	3,6	1,2	5,1	2,2	0,4	3,9
20	2	[1,49]	4,6	3,5	3,0	2,7	0,7	2,2	2,3	4,6	3,5	3,0	2,7	0,7	2,2	2,3
20	3	[1,49]	10,7	9,3	6,1	6,2	2,2	5,1	6,6	10,7	9,3	6,1	5,3	2,2	5,1	6,6
20	5	[1,49]	14,4	15,1	14,3	16,4	8,4	10,8	13,7	14,4	13,7	14,3	16,2	7,9	10,2	12,2
20	8	[1,49]	35,9	30,5	26,2	32,6	20,2	24,8	25,4	35,9	30,5	26,2	32,6	20,1	22,2	25,4
20	2	[1,99]	5,7	6,5	4,8	3,9	2,2	4,3	5,8	5,7	6,5	4,7	3,9	2,2	4,2	5,6
20	3	[1,99]	12,2	11,6	8,4	10,3	6,2	8,4	13,2	12,2	11,6	8,3	10,0	6,2	8,4	13,2
20	5	[1,99]	24,3	23,9	20,6	20,0	16,9	16,8	24,9	24,3	23,9	20,0	20,0	16,7	14,7	24,2
20	8	[1,99]	36,2	35,8	32,3	32,1	25,4	28,3	37,4	36,2	35,8	30,2	31,4	25,2	33,0	36,1
20	2	[50,149]	11,7	12,7	8,2	8,9	7,2	9,1	13,6	11,7	12,7	8,2	8,6	7,2	9,1	12,2
20	3	[50,149]	24,4	22,4	18,4	18,5	12,8	17,2	21,3	24,4	22,4	18,2	17,2	12,8	17,1	20,6
20	5	[50,149]	49,0	44,7	34,7	32,5	29,9	31,5	38,4	49,0	44,7	34,0	31,9	29,3	31,1	36,7
20	8	[50,149]	42,1	37,4	36,0	37,8	32,2	37,6	40,7	42,1	37,4	35,6	36,0	31,5	37,1	40,7
30	2	[1,49]	3,6	2,9	1,9	1,9	0,2	1,4	2,3	3,6	2,6	1,9	1,9	0,2	1,3	2,3
30	3	[1,49]	4,7	5,8	4,6	4,5	1,1	2,3	4,9	4,7	5,8	4,6	4,4	1,1	2,3	4,8
30	5	[1,49]	13,1	10,1	10,0	10,9	5,4	6,6	11,1	13,1	10,1	9,5	10,9	5,1	5,9	11,0
30	8	[1,49]	23,6	20,0	20,9	19,1	12,0	16,9	19,1	23,6	20,0	20,4	19,1	11,3	14,4	23,0
30	2	[1,99]	5,9	1,8	2,8	2,7	0,9	2,4	2,4	5,9	1,8	2,8	2,6	0,9	2,4	2,4
30	3	[1,99]	10,2	9,4	4,6	6,4	3,0	5,4	7,9	10,2	9,2	3,9	5,4	3,0	5,4	7,5
30	5	[1,99]	16,4	18,1	15,4	14,3	11,5	11,6	15,8	16,4	18,1	15,2	14,3	11,0	11,6	15,8
30	8	[1,99]	30,5	33,4	27,3	29,4	24,2	26,7	28,4	30,5	33,4	26,0	27,9	22,8	24,3	26,6
30	2	[50,149]	7,5	6,9	6,5	5,4	5,1	6,2	8,6	7,5	6,9	6,5	5,4	5,1	6,2	8,4
30	3	[50,149]	13,1	11,4	9,9	11,8	8,9	11,0	12,5	13,1	11,4	9,7	11,5	8,9	10,8	12,5
30	5	[50,149]	27,5	27,8	22,1	25,2	20,8	20,5	25,9	27,5	27,8	22,1	24,3	20,8	19,9	25,3
30	8	[50,149]	47,7	43,9	39,4	42,9	36,5	35,2	44,2	47,7	43,3	38,8	42,2	35,7	31,7	42,1
50	2	[1,49]	1,6	1,5	1,1	0,7	0,3	0,7	1,3	1,6	1,5	1,1	0,7	0,3	0,7	1,3
50	3	[1,49]	3,7	2,9	2,3	2,0	0,9	1,7	2,8	3,7	2,9	2,3	2,0	0,9	1,7	2,8

50	5	[1,49]	7,1	6,7	7,1	7,9	2,5	4,3	6,8	7,1	6,7	7,1	7,9	2,5	4,3	6,8
50	8	[1,49]	13,0	12,7	11,7	10,1	7,2	9,7	11,3	13,0	12,7	11,2	9,9	6,6	8,0	10,7
50	2	[1,99]	3,4	2,5	1,3	1,7	0,5	1,6	1,9	3,4	2,5	1,3	1,5	0,5	1,6	1,8
50	3	[1,99]	4,8	4,2	4,5	3,6	1,6	2,6	4,7	4,8	4,0	4,5	3,2	1,5	2,5	4,7
50	5	[1,99]	10,2	7,8	8,8	7,8	4,3	6,3	10,2	10,2	7,8	8,7	7,6	4,3	6,2	9,6
50	8	[1,99]	16,6	15,5	16,2	15,2	10,9	13,2	17,0	16,6	15,5	15,7	14,9	10,1	11,9	16,5
50	2	[50,149]	5,1	5,4	3,3	4,5	2,9	3,7	4,7	5,1	5,4	3,3	4,5	2,9	3,7	4,5
50	3	[50,149]	8,9	8,3	7,6	8,6	4,5	5,8	8,9	8,9	8,2	7,5	8,6	4,5	5,8	8,9
50	5	[50,149]	16,0	16,2	13,9	14,8	11,4	13,0	15,5	16,0	16,2	13,4	14,8	11,2	13,0	15,1
50	8	[50,149]	29,1	27,8	27,7	26,5	22,7	24,2	26,2	29,1	27,8	27,3	25,7	21,9	23,4	26,2
80	2	[1,49]	1,3	1,1	0,8	1,1	0,2	0,6	0,6	1,3	1,1	0,8	1,1	0,2	0,6	0,6
80	3	[1,49]	2,6	2,1	1,4	1,5	0,6	0,9	1,5	2,6	2,1	1,4	1,3	0,6	0,9	1,5
80	5	[1,49]	4,5	3,9	3,5	5,0	1,5	2,4	4,3	4,5	3,9	3,5	4,9	1,5	2,2	4,0
80	8	[1,49]	8,8	7,6	6,9	8,3	4,4	6,0	6,9	8,8	7,6	6,9	8,3	4,2	5,4	6,9
80	2	[1,99]	2,0	2,3	1,3	1,2	0,4	1,0	1,4	2,0	2,3	1,3	1,0	0,4	1,0	1,4
80	3	[1,99]	3,2	3,0	2,7	2,6	1,4	1,9	3,5	3,2	3,0	2,6	2,5	1,4	1,9	3,5
80	5	[1,99]	6,1	5,9	5,1	5,7	3,2	4,0	6,1	6,1	5,9	4,7	5,4	3,1	4,0	6,1
80	8	[1,99]	9,6	10,7	9,0	9,2	6,9	8,1	11,2	9,6	10,7	8,8	8,9	6,9	7,3	10,9
80	2	[50,149]	3,7	3,5	2,1	2,4	1,8	2,1	3,3	3,7	3,5	2,1	2,4	1,8	2,1	3,3
80	3	[50,149]	5,7	4,4	3,9	5,0	3,5	4,3	5,2	5,7	4,4	3,9	4,7	3,5	4,3	5,2
80	5	[50,149]	9,6	10,5	9,4	8,7	6,4	7,5	9,3	9,6	10,5	9,3	8,3	6,3	7,5	9,3
80	8	[50,149]	16,5	18,1	15,2	16,1	13,7	15,5	15,7	16,5	17,9	15,0	16,0	13,0	14,4	15,2
100	2	[1,49]	0,7	0,5	0,5	0,6	0,2	0,2	0,6	0,7	0,5	0,5	0,6	0,2	0,2	0,6
100	3	[1,49]	1,5	1,6	1,5	1,5	0,4	0,9	1,3	1,5	1,6	1,5	1,5	0,4	0,9	1,3
100	5	[1,49]	3,3	4,0	3,1	2,9	1,1	1,8	3,3	3,3	4,0	3,1	2,9	1,1	1,7	3,3
100	8	[1,49]	7,0	6,6	6,4	5,9	3,2	3,8	5,5	7,0	6,6	6,4	5,9	3,1	3,7	5,4
100	2	[1,99]	1,7	1,3	0,9	1,3	0,5	0,9	1,2	1,7	1,3	0,9	1,3	0,5	0,9	1,2
100	3	[1,99]	2,5	2,0	1,5	1,8	1,1	1,5	2,5	2,5	2,0	1,5	1,6	1,1	1,5	2,5
100	5	[1,99]	5,4	5,0	4,4	4,3	2,3	3,4	5,2	5,4	5,0	4,4	4,1	2,3	3,4	5,2
100	8	[1,99]	9,5	8,9	8,0	8,0	5,5	6,7	8,4	9,5	8,9	8,0	8,0	5,1	6,2	8,4
100	2	[50,149]	2,9	2,5	2,0	2,1	1,4	2,1	2,7	2,9	2,5	2,0	2,0	1,4	2,1	2,7
100	3	[50,149]	4,7	3,5	3,6	3,7	2,3	2,7	4,5	4,7	3,4	3,5	3,6	2,3	2,7	4,2
100	5	[50,149]	7,9	8,0	7,5	6,6	5,9	6,0	8,5	7,9	8,0	7,5	6,3	5,9	5,9	7,9
100	8	[50,149]	13,7	14,1	11,8	12,9	10,5	11,6	14,6	13,7	14,1	11,6	12,6	10,4	11,5	14,1