UNIVERSIDADE FEDERAL DE GOIÁS – REGIONAL CATALÃO
UNIDADE ACADÊMICA ESPECIAL DE MATEMÁTICA E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM E OTIMIZAÇÃO

Jeferson Silva Martins

UM ESTUDO COMPARATIVO ENTRE SOLUÇÕES APLICADAS A UM PROBLEMA DE FLOW LINE MISTO

DISSERTAÇÃO DE MESTRADO

CATALÃO – GO, 2018

sistema de bibliotecas ufg

PRPG
PRÓ-REITORIA DE PÓS-GRADUAÇÃO

UFG

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES E DISSERTAÇÕES 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. Identificação do material bibliográfico: **[X] Dissertação** **[ ] Tese**

2. Identificação da Tese ou Dissertação:

Nome completo do autor: JEFERSON SILVA MARTINS

Título do trabalho: UM ESTUDO COMPARATIVO ENTRE SOLUÇÕES APLICADAS A UM PROBLEMA DE FLOW LINE MISTO

3. Informações de acesso ao documento:

Concorda com a liberação total do documento [X] SIM [ ] NÃO

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio dos arquivos em formato digital PDF da tese ou dissertação.

Jeferson Silva Martins

Jeferson Silva Martins

Ciente e de acordo:

José dos Reis V. de Moura Júnior

Orientador: José dos Reis Vieira de Moura Júnior

Data: 12/04/2018

JEFERSON SILVA MARTINS

UM ESTUDO COMPARATIVO ENTRE SOLUÇÕES APLICADAS A UM PROBLEMA DE FLOW LINE MISTO

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre em Modelagem e Otimização pela Universidade Federal de Goiás – Regional Catalão.

Orientador:
José dos Reis Vieira de Moura Junior

CATALÃO – GO

2018

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

Silva Martins, Jeferson
Um Estudo Comparativo entre Soluções Aplicadas a um Problema de Flow Line Misto [manuscrito] / Jeferson Silva Martins. - 2018.
CCXXIX, 229 f.

Orientador: Prof. José dos Reis Vieira de Moura Júnior.
Dissertação (Mestrado) - Universidade Federal de Goiás, Unidade Acadêmica Especial de Matemática e Tecnologia, Catalão, Programa de Pós-Graduação em Modelagem e Otimização, Catalão, 2018.
Bibliografia. Anexos. Apêndice.
Inclui siglas, símbolos, gráfico, tabelas, algoritmos, lista de figuras, lista de tabelas.

1. Scheduling. 2. Programação Inteira. 3. Otimização. 4. Flow Line Misto. 5. Algoritmo Genético. I. Vieira de Moura Júnior, José dos Reis , orient. II. Título.

CDU 658.5

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
REGIONAL CATALÃO
Programa de Pós-Graduação em Modelagem e Otimização - PPGMO

Defesa Nº 34

Ata de Defesa Pública – Dissertação de Mestrado

Aos DEZESSETE dias do mês de MAIO do ano de 2018, às 14 h: 00 min, reuniram-se os componentes da banca examinadora, professores Dr. JOSÉ DOS REIS VIEIRA DE MOURA JÚNIOR (presidente e Orientador), Dr. HELIO YOCHIHIRO FUCHIGAMI, Dr. THIAGO ALVES DE QUEIROZ para, em sessão pública realizada na Sala de Reuniões (Centro Integrado de Pesquisa), da Regional Catalão (RC), da Universidade Federal de Goiás (UFG), procederem com a avaliação do trabalho intitulado: "UM ESTUDO COMPARATIVO ENTRE SOLUÇÕES APLICADAS A UM PROBLEMA DE FLOW LINE MISTO" em nível de Mestrado, área de concentração *Modelagem e Otimização*, de autoria de JEFERSON SILVA MARTINS, discente do Programa de Pós-Graduação em Modelagem e Otimização (PPGMO) da UFG/RC. A sessão foi aberta pelo presidente da banca, que fez a apresentação formal dos membros da banca. A seguir, a palavra foi concedida ao discente que, dentro do tempo regulamentar, procedeu a apresentação de seu trabalho. Terminada a apresentação, cada membro da banca arguiu o candidato, tendo-se adotado o sistema de diálogo sequencial. Terminada a fase de arguição, procedeu-se a avaliação do trabalho. Os membros da banca consideraram o trabalho final: **(X) Aprovado** ou **( ) Reprovado**. Cumpridas as formalidades de pauta, às 16 h: 20 min a presidência da mesa encerrou a sessão e para constar, eu Dr. JOSÉ DOS REIS VIEIRA DE MOURA JÚNIOR, lavrei a presente Ata que, depois de lida e aprovada, segue assinada pelos membros da banca examinadora e pelo discente e, posteriormente, será homologada pelo Colegiado do PPGMO.

Catalão-GO, 17 de Maio de 2017.

José dos Reis Vieira de Moura Júnior
Prof. Dr.: José dos Reis Vieira de Moura Júnior
Programa de Pós-Graduação em Modelagem e Otimização, UFG/RC.
(Presidente da Banca)

Prof. Dr.: Hélio Yochihiro Fuchigami
Faculdade de Ciência e Tecnologia, UFG/RAG

Thiago Alves de Queiroz
Prof. Dr.: Thiago Alves de Queiroz
Programa de Pós-Graduação em Modelagem e Otimização, UFG/RC

Jeferson Silva Martins
Discente: Jeferson Silva Martins
Programa de Pós-Graduação em Modelagem e Otimização, UFG/RC.

Aos meus Avós Francisco e Deny Osmar e Nelita

Agradecimentos

Agradeço a Deus pelo trabalho realizado nesses últimos dois anos e também pelas oportunidades que esse tempo me proporcionou, pela colheita de sabedoria e dignidade.

Agradeço aos meus pais e irmãos pelos pelo apoio direto e indireto que eles me deram durante esse período e também pelos dias em que eles compreenderam o quanto eu estava insuportável, estressado e anti-ético em casa.

Agradeço aos meus amigos, pela força, pelos risos e pelo acalento que eles me deram quando eu os abandonei e mesmo assim eles se fizeram presente em minha vida.

Agradeço aos avós maternos Deny (*in memorian*) e Francisco (*in memorian*) e aos meus avós paternos Nelita e Osmar (*in memorian*) pela torcida que eles deram em prol da educação dos netos e que mesmo alguns deles não estando aqui na terra, a conquista também é deles pelo apoio que me deram em vida.

Agradeço aos Mestres e Doutores da Universidade Federal de Goiás, em especial ao meu orientador José do Reis, pela paciência e dedicação em me proporcionar sempre o melhor, com recursos humanos e tecnológicos para que essa pesquisa fosse concluída.

Agradeço a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro em contribuição para essa pesquisa.

Enfim, agradeço a todos os elementos humanos que, de certa forma, rezaram e praguejaram pelo meu bem ou pelo meu mal.

Um Obrigado a Todos.

"Not a whit. We defy augury. There's a special providence in the fall of a sparrow." -Hamlet (Shakespeare)

RESUMO

MARTINS, J. S.. *Um Estudo Comparativo entre Soluções Aplicadas a um Problema de Flow Line Misto*. 2018. 229 f. Dissertação (Mestrado em Modelagem e Otimização) – Unidade Acadêmica Especial de Matemática e Tecnologia, Universidade Federal de Goiás – Regional Catalão, Catalão – GO.

Este trabalho apresenta o problema de *Flow Line* Misto para a Programação da Produção em *Scheduling*. Problemas da produção tem sido vastamente estudados dada sua usabilidade em todo o tipo de linha de montagem e produção na indústria mundial. O problema em estudo considera o processamento de tarefas em conjuntos por meio de estações de máquinas buscando um bom sequenciamento das tarefas a serem produzidas melhorando o tempo de processamento (*Makespan*). Neste trabalho, uma revisão bibliográfica foi realizada para fundamentar o estudo encontrando diferentes métodos e problemas que englobam o *Flow Line* Misto mostrando a importância deste estudo no meio industrial e científico. O problema do *Flow Line* Misto foi resolvido por meio de técnicas de otimização, sendo elas um modelo de programação linear inteira resolvido pelo GUROBI, no Lagrangeano Relaxado e no Algoritmo Genético com tempos de processamento e demandas gerados aleatoriamente para as instâncias criadas. Por fim, os resultados obtidos nas técnicas analisadas para o problema foram comparados entre si.

Palavras-chaves: *Scheduling*, Programação Inteira, Otimização, *Flow Line* Misto, Algoritmo Genético.

ABSTRACT

MARTINS, J. S.. *A Comparative Study Between Techniques Applied to a Mixed Flow Line Problem*. 2018. 229 f. Master Thesis in Modelling and Optimization – Unidade Acadêmica Especial de Matemática e Tecnologia, Universidade Federal de Goiás – Regional Catalão, Catalão – GO.

This paper presents the Mixed-Model Flow Line problem for production Scheduling. Production problems have been vastly studied given their usability in all kind of assembly lines and its production in the industry worldwide. The problem processes jobs grouped by families in stations of machines seeking for the best sequencing the families can provide to be produced in the processing time (Makespan). In this work, a bibliographic review was performed to support the study, looking different methods and problems that encompass the Mixed-Model Flow Line in the literature showing its importance towards the industrial and scientific environment. The Mixed-Model Flow Line problem was solved through optimization techniques: An integer lienar programming in GUROBI, Lagrangian Relaxation and Genetic Algorithm. The processing times and demands are randomly generated for each instance. Finally, the results obtained in the techniques used for the problem were compared to each other.

Keywords: Scheduling, Integer Programming, Optimization, Mixed-Model Flow Line, Genetic Algorithm.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ABC — Algoritmo Colônia de Formigas

ACO — Algoritmo Colônia de Formigas

AG — Algoritmo Genético

B&B — *Branch and Bound*

BB — *Bottleneck-Based*

BL — Busca Local

BOV — Busca de Operadores Por Vizinhança

CPLEX — *Software* de otimização da *International Business Machine Corporation*(IBM)

CPU — *Central Processing Unit*

EDD — *Earliest Due Date*

ERD — *Earliest Release Date First*

FCFS — *First Come, First Serve*

FFL — *Flexible Flow Line*

FLM — *Flow Line* Misto

FR&FO — *Fix and Relax and Opt Heuristics*

GUROBI — *Software* de otimização Linear

HI — Heurísticas Insertivas

ICA — Algoritmo Competitivo Imperialista

LINGO — *Software* de otimização linear para Programação da Produção

LPT — *Longest Processing Time*

LR — Lagrangeano Relaxado

MA — Algoritmo Memetico

PCP — Planejamento e Controle da Produção

PIM — Programação Inteira Mista

PN — Rede *Petri-Net*

PP — Programação da Produção

PSO — Enxame de Partículas

RP — Regras de Prioridade

RS — Recozimento Simulado

SIRO — *Service in Random Order*

SM — Algoritmo *Silver-Meal*

SPT — *Sortest Processing Time*

TSP — Problema do Caixeiro Viajante

WLPT — *Weighted Longest Processing Time*

WSPT — *Weighted Shortest Processing Time*

LISTA DE SÍMBOLOS

© — Direito Autoral ou Direito de Autor

® — Marca Registrada

p_{ij} — Tempo de processamento da tarefa j na máquina i

r_j — Data de liberação da tarefa j

d_j — Data máxima de entrega da tarefa j

w_j — Peso da tarefa j

α — Caraterização para o Ambiente de máquinas

β — Caracterização para Restrições de Ambiente de Produção

γ — Caracterização para objetivo do Ambiente de Produção

ø — Caracterização de Valor Nulo ou vazio

s_{jk} — Tempo de preparação de Máquinas

C_{max} — Makespan (Tempo de término das tarefas)

$\bar{F}$ — Tempo médio de fluxo das tarefas

L_{max} — Tempo máximo de atraso das tarefas em relação ao tempo de entrega

T_{max} — Maior tempo de atraso das tarefas

w_jC_j — Tempo de término das tarefas em relação ao peso w_j

w_jT_j — Tempo de término em relação ao tempo de atraso T_j

$\sum$ — Somatório

$\leqslant$ — Operador de comparação de expressão menor ou igual

$\geqslant$ — Operador de comparação de expressão maior ou igual

$\in$ — Operador pertence a

$\forall$ — Operador para todo

θ — Atualizador do Passo do Lagrangeano Relaxado

ϕ — Valor de parâmetro para o atualizador de passo do Lagrangeano Relaxado

$\lceil \rceil$ — Menor inteiro maior ou igual ao número fracionário

Lista de Algoritmos

SUMÁRIO

Capítulo 1

Introdução

Este trabalho aborda e estuda o problema do *Flow Line* Misto (FLM) com conjunto de tarefas para o sequenciamento da produção. O problema FLM é apresentado por Eliiyi e Özlen (2008) com M estações de trabalho, com R conjuntos de produtos associados a N tarefas. Cada tarefa deve pertencer a um conjunto. O processamento das tarefas p_{rm} deve ser completado atendendo a demanda de processamento de cada estação, tendo os conjuntos de tarefas um começo, meio e fim para cada ciclo. O problema FLM é um derivado do clássico *Flow Shop* ou *Flexible Flow Line*, sendo eles, problemas matematicamente desafiadores quando consideradas instâncias para $m \geqslant 3$. No entanto, para o problema FLM, ainda é aberta sua classificação quanto a complexidade NP-Difícil (ELIIYI; ÖZLEN, 2008).

O problema FLM considera um ambiente de produção também conhecido como *Hybrid Flow Shop*, no qual existe um número específico de máquinas em paralelo separadas em estações ou ciclos de processamento. Os conjuntos de tarefas entram na primeira estação e passam por todas as estações disponíveis até a última, completando uma sequência de manufatura. Em cada estação, uma máquina deve ser selecionada para processar o conjunto de tarefas na fila de produção. Desta forma, uma quantidade maior de produtos podem ser processados ao mesmo tempo reduzindo o tempo de processamento. O FLM pode ser encontrado no mundo real em uma variedade de aplicações no meio industrial, incluindo na manufatura de eletrônicos (WITTROCK, 1988), na indústria automotiva (AGNETIS *et al.*, 1997), na indústria de semicondutores (QUADT; KUHN, 2007), entre outros.

Ambientes de produção são problemas que surgiram dentro da área de *Scheduling* como processo de decisão na indústria de manufatura (PINEDO, 2016). O Planejamento e Controle da Produção (PCP) é definido por Fernandes e Filho (2010) em três principais atividades:

- **(1)** A liberação final do produto;
- **(2)** A programação de operações;

- **(3)** O apontamento da produção.

Em linhas gerais, o PCP compreende a demanda que um produto ou tarefa precisa para ser produzido, definindo necessidades de produção como a capacidade total que se pode produzir do item e a disponibilidade de matéria-prima requisitada para a produção do produto. No PCP, o gerente de produção dever ter o controle da produção em suas mãos sempre acompanhando e monitorando as atividades dentro da estação de trabalho. O gerente de produção também deve estar sempre observando os indicadores de produção de medidas de desempenhos empregados nos produtos e, quando necessário, realocando e realimentando a linha de produção de acordo com a demanda exigida.

Pinedo (2016) comenta que a Programação da Produção (PP) é uma parte do PCP que introduz determinado problema dentro de uma unidade de processamento, e, neste sentido, a produção é totalmente conectada a uma visão suportada como tomadora de decisões, visando sempre altos lucros e baixos custos. Os estudos de ambientes de produção que englobam estratégias capazes de satisfazer as regras do negócio devem ser explorados tendo sempre como base diferentes recursos para atender todas as demandas requisitadas.

Dentro da PP, existe o processo responsável por definir o sequenciamento de tarefas com suas respectivas demandas. No ambiente da produção, toda tarefa possui um momento de início (entrada), meio (processamento) e fim (saída) de acordo com sua demanda (SLACK; BRANDON-JONES; JOHNSTON, 2014). Buscando uma forma eficiente de atender os requisitos de administração da produção, surge a necessidade de explorar e identificar diferentes ambientes para aplicar métodos de sequenciamento e manufatura capazes de obter excelência na produção. Porém, problemas de *Scheduling* como o apresentado neste trabalho, são, na maioria das vezes, matematicamente desafiadores e difíceis de resolver devido sua complexidade de solução (GAREY; JOHNSON; SETHI, 1976).

No âmbito do sequenciamento de tarefas, ambientes de produção como o *Flow Line* Misto buscam na maior parte dos trabalhos presentes na literatura, a minimização da duração total da programação das tarefas nas máquinas. Esta minimização é conhecida como *Makespan* (C_{max}). Porém, é possível também encontrar trabalhos que buscaram minimizar o atraso máximo (*tardiness* ou *lateness*), o tempo médio de fluxo (*Mean Flow Time*), entre várias outras.

Basicamente, o FLM é apresentado como um problema matemático com restrições de alocação de tarefas e recursos a serem processados. No decorrer dos anos, o FLM foi sendo estudado e utilizado na prática em diferentes regiões do globo. O estudo de um problema como este é importante, pois, ao longo do tempo, problemas matemáticos cada vez mais elaborados e próximos do mundo real foram introduzidos para se adequar a todo tipo de ramo na indústria visando sempre atender de forma eficiente a programação da produção, reduzindo custos e aumentando lucros.

Dos trabalhos desenvolvidos com a problemática do FLM, métodos de solução capazes de resolver o problema variam entre os métodos exatos de programação inteira mista (SAWIK, 2000), métodos heurísticos (COSTANTINO *et al.*, 2014) e regras de prioridades para o *Scheduling* (ALEXANDROS; CHRISSOLEON, 2009).

Neste trabalho estuda-se o problema do FLM a partir de um modelo matemático desenvolvido em trabalhos presentes na literatura. O problema de Eliiyi e Özlen (2008) aqui investigado é um problema *Flow Line* Misto, em que diferentes tarefas associadas a um conjunto de produtos estão sendo processados na mesma linha. No problema é considerado que as máquinas não têm tempo de manutenção ou de quebra e todos os conjuntos de tarefas estão disponíveis no tempo zero para serem processados. Para que ocorra a transferência de conjuntos entre as estações é necessário primeiro que todos os conjuntos processados na estação vigente sejam terminados para que a transferência ocorra de forma simultânea. O C_j é o tempo de ciclo do processamento dos conjuntos de tarefas nas estações, sendo que o problema busca a minimização do C_{max}.

O FLM é um problema bastante complexo e pode existir variações de características de alocação e processamento que foram adicionados no decorrer dos anos. Na literatura, algumas pesquisas do FLM consideram tempo de preparação das máquinas (*Setup*), janelas de tempo, conjunto de tarefas, rejeição de tarefas, *no-wait* e manutenção preventiva.

1.1 Metodologia, Estratégia Empregada e Objetivos

O presente trabalho tem como objetivo de estudo e análise o FLM por meio do método Algoritmo Genético, um sistema Bio-Inspirado capaz de encontrar soluções que satisfaçam o problema FLM, e o método exato presente no pacote de otimização *GUROBI © Optimizer* na versão 7.5.2. Para isto, foi estudado uma versão do problema de FLM caracterizado por um modelo de Programação Inteira Mista (PIM) através de abordagens da literatura, buscando compreender melhor o funcionamento deste ambiente de produção. O modelo proposto foi codificado em linguagem C++ e também foi codificada em C++ uma heurística Lagrangeana com o propósito de relaxar restrições de difícil resolução, reduzindo assim o tempo computacional durante a resolução de instâncias de grande porte. Para o Algoritmo Genético, o problema foi implementado utilizando o método da penalidade Exterior e sua codificação está no *software* MatLab®.

Para chegar ao problema em questão foi desenvolvido um mapeamento sistemático da literatura buscando por artigos que tratavam do assunto nas bases indexadoras. Todos eles foram acessados entre Janeiro e Março de 2017 no sítio da Universidade Federal de Goiás, Regional Catalão. Após a busca pelos artigos, foi feita uma seleção por títulos com o objetivo de reduzir a quantidade da amostra buscando conteúdos próximos ao tema. A etapa final do estudo foi a leitura e análise dos itens selecionados, classificando-os por ano

de publicação, periódico publicado, métodos de solução e países nos quais os estudos foram desenvolvidos. Por fim, com o mapeamento sistemático, a revisão bibliográfica pôde ser desenvolvida a partir do número final de artigos selecionados bem como a escolha do tema afunilado.

Durante a pesquisa nas bases foi possível observar que a evolução das publicações de artigos com o tema *Flexible Flow Line* ou *Flow Line* Misto ocorreu no decorrer dos anos de 1967 a 2017. Foram considerados três conjuntos de palavras chaves combinadas na pesquisa (a) *Heuristics and Flexible Flow Line*, (b) *Scheduling and Flexible Flow Line*, (c) *Modeling and Flexible Flow Line*.

Após a leitura dos artigos, foi possível delinear os métodos de solução de cada trabalho e pôde-se perceber que diferentes métodos de solução foram desenvolvidos e utilizados no decorrer dos anos. Em linhas gerais, os métodos mais utilizados são os métodos baseados em heurísticas. O método de solução de um problema é crucial para determinar sua eficiência durante a solução do problema e, através disso, optou-se por usar em um primeiro momento o estudo de um método heurístico devido ao tempo computacional gasto. Porém, heurísticas não garantem soluções ótimas e algumas são difíceis de implementar. No segundo momento, pensando na eficácia de solução do problema, optou por utilizar um método exato para nortear os resultados em busca de soluções ótimas e, em um último momento, o método LR foi utilizado na busca de um tempo computacional mais aceitável em relação ao método exato.

Ao revisar a literatura sobre o FLM, pode-se observar que métodos utilizados para resolver o problema começaram com métodos exatos e com regras de prioridade na área de *Scheduling*. No decorrer dos anos, as heurísticas foram sendo introduzidas para solucionar o problema em tempo polinomial aceitável nas instâncias de grande porte. Atualmente, a maioria dos artigos encontrados nas bases acadêmicas são baseados em métodos heurísticos com muitos deles utilizando o Algoritmo Genético.

O presente trabalho utiliza duas linguagens diferentes de programação para a realização dos experimentos. A primeira é do *software* Matlab®, um interpretador numérico que executa códigos fontes de fácil entendimento para os experimentos no Algoritmo Genético. A segunda é a linguagem C++, compilável, que permite uma execução mais rápida de programas por ser executável, para o método exato e do Lagrangeano Relaxado. Ambas as linguagens são capazes de ler e escrever dados de arquivos.

Testes computacionais foram realizados para testar a qualidade de execução da implementação dos métodos usando instâncias geradas aleatoriamente seguindo os padrões propostos por Eliiyi e Özlen (2008). A comparação de resultados também foi realizada para analisar fatores como soluções ótimas alcançadas, tempo computacional gasto e diferença de resultados entre os métodos.

Este trabalho estuda o problema de *Flow Line* Misto com conjunto de tarefas e propõe três diferentes métodos de solução: um algoritmo genético, a resolução de um modelo de PIM e uma relaxação Lagrangeana. Em todos os métodos, teve-se a busca da minimização do *Makespan* (c_{max}) do FLM. O ambiente de FLM, em específico, tem sido pouco explorado no decorrer dos últimos dez anos conforme mostra o capítulo 2. Portanto, a motivação para estudar um problema pouco explorado se torna maior com objetivo de obter resultados que venha a contribuir com a melhoria da programação da produção na indústria da manufatura.

Por se tratar de um ambiente tão corriqueiro na área industrial e ser tão pouco explorado, tem-se uma visão mais ampla na busca por soluções no mundo real para esse ambiente. Além disso, na atualidade, recursos tecnológicos e computacionais são adaptados a fim de otimizar e encontrar melhores resultados para os problemas de produção existentes. Inicialmente, este ambiente era resolvido com soluções que empregavam regras de prioridade na programação da produção e gradualmente, trabalhos que utilizavam o método exato do tipo *Branch & Bound* foram sendo desenvolvidos. Posteriormente, métodos como heurísticas foram empregados sobre o problema.

O objetivo principal deste trabalho é explorar o problema FLM proposto por Eliiyi e Özlen (2008) e analisar os resultados obtidos no LR com os do Algoritmo Genético e com o do método exato do GUROBI em termos de eficácia da solução encontrada e tempo gasto para encontrar o C_{max}. Os objetivos específicos estabelecem a análise de pontos do problema quando inseridos em grandes instâncias, muitas delas, semelhantes a instâncias do mundo real, sendo esta, uma das principais motivações para a realização do trabalho.

1.2 Estrutura do Trabalho

Este texto está estruturado em capítulos. O capítulo 1 introduziu o problema com suas abordagens e porque ele é importante. Os demais capítulos estão organizados como segue abaixo:

- **Capítulo 2:** Descreve uma revisão bibliográfica do problema FLM junto à definição de *Scheduling*, métodos de solução por regras de prioridade da programação da produção e metodologias de soluções alternativas para problemas do FLM e FFL. A revisão também inclui detalhes de trabalhos desenvolvidos e a evolução do FFL e FML na indústria e na ciência, como países de publicação, métodos de otimização propostos e implementados.

- **Capítulo 3:** Mostra a definição do problema FLM com a alocação de tarefas em conjuntos e a busca do menor tempo de conclusão do projeto, além de ciclo de processamento. Este capítulo também apresenta o problema FLM com a Relaxação Lagrange-

ana proposta por Eliiyi e Özlen (2008). O capítulo também apresenta informações da implementação do método do Lagrangeano Relaxado para o GUROBI.

- **Capítulo 4:** Apresenta detalhes do Algoritmo Genético desde sua história até a forma que ele foi desenvolvido, com detalhes de seu funcionamento por meio da seleção natural proposta por Darwin.
- **Capítulo 5:** Neste Capítulo é apresentado os resultados obtidos para o problema FLM considerando instâncias geradas aleatoriamente conforme os parâmetros do problema proposto por Eliiyi e Özlen (2008). Nas comparações, usou-se o Lagrangeano relaxado, Algoritmo Genético e o método exato do GUROBI para analisar a qualidade de resposta do problema para as instâncias utilizadas.
- **Capítulo 6:** Mostram-se as conclusões das pesquisas obtidas até o presente momento, além de trazer propostas para projetos futuros.

Capítulo 2

Conceitos e Revisão sobre o *Flexible Flow Line* e o *Flow Line* Misto

Neste capítulo, faz-se a revisão bibliográfica do ambiente *Flexible Flow Line* e suas variantes com as demais definições da Programação da Produção. Também é descrito a revisão do *Flow Line* Misto com características e restrições empregados no ambiente em questão.

2.1 Definição de *Scheduling*

A PP é um processo de decisão que define todo e qualquer tipo de serviço na indústria de manufatura (PINEDO, 2016). A PP lida com uma série de recursos disponíveis em uma linha de produção e pode ser apresentado em diferentes formas na indústria.

Basicamente, ela lida com tarefas que devem ser processadas alocando recursos para a execução destas tarefas, tendo elas, certos níveis de prioridade; datas de entregas; tempo máximo e mínimo de processamento, sendo muitas destas tarefas associadas com máquinas que podem necessitar de tempo de preparação para recebê-la. De acordo com Pinedo (2016), a PP é o processo mais importante no sistema de produção de determinados ambientes na indústria de manufatura por definir o processo de tomada de decisão.

A programação da produção tem seus objetivos divididos em dois princípios: a PP em manufatura e em serviços. Ambos princípios são sistemas que devem interagir com outras funções do processo. Basicamente, estes processos vêm acoplados em *softwares* de controle que ajudam o gerente de produção a decidir o andamento das tarefas.

Na PP, as ordens que vêm direcionadas ao departamento de produção são transformadas em atividades que utilizam desses recursos. As máquinas de uma fábrica, trabalhadores de uma linha de produção, uma pista de aeroporto e até um CPU (*Central Processing Unit*) são considerados recursos de serviços. Por meio dos recursos de serviços têm-se, então, a minimização de custos de produção e associações a ela como data limite de entrega, mi-

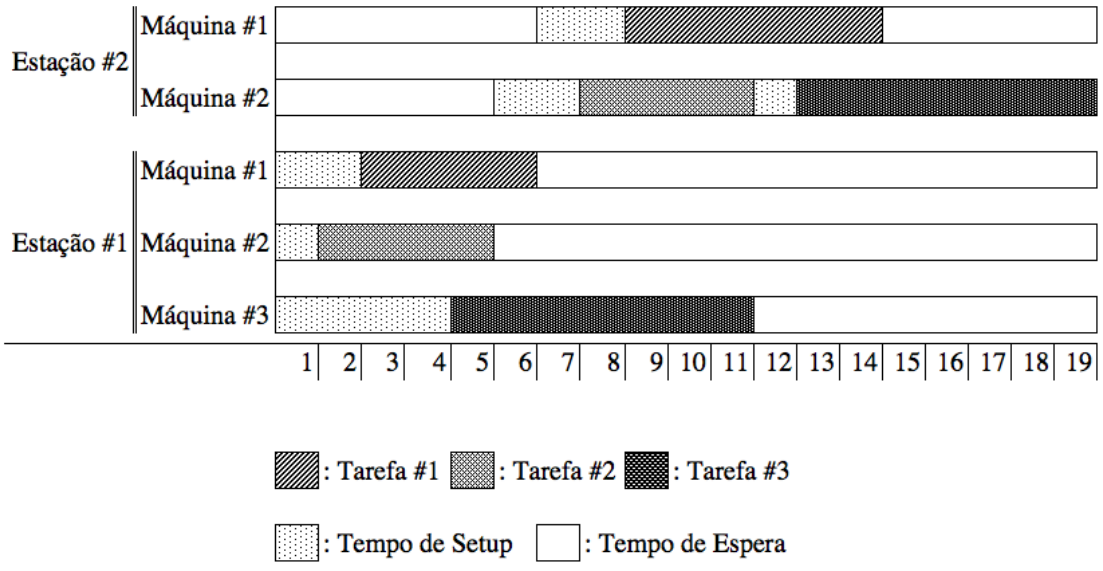
nimização do tempo de processamento, minimização do tempo médio de preparação de máquinas. Tais associações são considerados PP em manufatura.

Usualmente, as etapas da programação consistem em **Alocação**, **Sequenciamento**, e **Programação** (FUCHIGAMI, 2015). Estas etapas definem de forma clássica como as tarefas são processadas e organizadas em sequências de processamento. No mundo real, tarefas podem atrasar, máquinas podem quebrar ou entrar em manutenção e, é importante que o tomador de decisões saiba quais procedimentos devem ser tomados para minimizar o atraso nas entregas ou na linha de produção. É importante que esse tomador de decisões tenha o conhecimento das etapas que regem a programação da produção.

Slack, Brandon-Jones e Johnston (2014, p. 245) mostram que todo o processo de produção com eventos previsíveis e não previsíveis são o teor da PP. Elas são atividades complexas que necessitam de diferentes recursos de natureza combinatória, ou seja, para n tarefas a serem executadas, existem $n!$ sequências possíveis de sequenciamento. Para n tarefas com m> 1 máquinas, existem então $(n!)^m$ possíveis sequenciamentos.

Uma importante ferramenta que auxilia a Programação da Produção é o Gráfico de Gantt. Ele foi inventado por Henry Gantt (1861-1919) e representa de forma visual o progresso das sequências de tarefas com as etapas da programação. A Figura 2.1 mostra o gráfico de Gantt para o *Flexible Flow Line*. A vantagem da representação inventada por Gantt é seu visual simples e de fácil entendimento, mostrando cada operação do processo desde o início até o fim.

Figura 2.1 – Exemplo do Gráfico de Gantt.



Fonte: o autor.

2.1.1 Ambientes de Produção

Dentro da PP, as linhas de produção nas indústrias seguem uma série de operações e são classificadas conforme as mesmas. Usualmente, estas operações seguem uma linha de execução conforme define a planta de produção. Este tipo de classificação de operações que seguem passos ou etapas de execução para a produção é conhecido no mundo da PP como Ambientes de Produção.

Em muitas plantas de produção, os trabalhos devem seguir a linha de processamento e passar por todas as máquinas até serem totalmente processadas, implicando que todas as tarefas passem pela linha de processamento conforme define a planta de produção. No decorrer dos anos, diferentes capacidades de produção das máquinas e limitações em outras tornaram os ambientes mais complexos sendo necessário modelar diferentes ambientes cujo objetivo era obter melhores desempenho de processamento da produção para a demanda requisitada. Os ambientes de produção mais comuns na PP são:

- **Máquina Única**: Processamento singular, independente e linear;
- **Máquinas Paralelas**: Processamento de fila única, seguindo a linha de processamento paralelos;
- ***Flow Shop***: Processamento em série, operação sequencial de fabricação;
- ***Flexible Flow Shop*** ou ***Flexible Flow Line***: Estações de equipamentos com processamento em série, fabricação em família de tarefas ou lotes;
- ***Job Shop***: Processamento individual com fluxo independente;
- ***Open Shop***: Estações de equipamentos com processamento diversificado entre as estações, podendo saltar e voltar tarefas nas máquinas.

Para o modelo em estudo, sua classificação é caracterizada como *Flow Line* Misto. O modelo *Flow Line* Misto possui estações de trabalho contendo m máquinas capazes de processar n tarefas de acordo com a ordem. Este ambiente é bastante comum em linhas de montagem em veículos (AGNETIS *et al.*, 1997), manufatura de eletrônicos (WITTROCK, 1988), na indústria química (SALVADOR, 1973), entre outros. Seu objetivo comum entre tantos na literatura é a minimização do *Makespan* (tempo de conclusão total de todas as tarefas), pois o interesse prático é diminuir o tempo de produção e assim produzir mais visando aumento de lucros. Porém, em outros cenários, é também interessante buscar pela minimização do atraso de tarefas e também pela violação de janelas de tempo entre os estágios de processamento (*no-wait*).

2.1.2 Notação dos Parâmetros da Produção

Desde o início dos estudos sobre a PP, variados tipos de pesquisa foram sendo introduzidos e a quantidade de modelos existentes é vasta. Nesse processo, uma notação capaz de descrever os problemas foi criada para tentar capturar em uma linguagem universal sobre o conceito da PP (PINEDO, 2016).

Na parametrização dos modelos, o número de tarefas e máquinas são finitos e determinados de acordo com o ambiente. Para descrever a quantidade de tarefas, usa-se n e para descrever a quantidade de máquinas, usa-se m. para índices, usa-se j para referir a identificação de uma tarefa e i para referir a identificação de uma máquina.

Quando os valores de i, j estão subscritos, eles denotam que determinada notação está assinalada a máquina i com a tarefa j. Seus parâmetros de acordo com Pinedo (2016) são:

- p_{ij}: Tempo de processamento da tarefa j na máquina i;
- r_j: Data de liberação da tarefa j para ser processada;
- d_j: Data máxima de entrega da tarefa j;
- w_j: Peso da tarefa j como referência de prioridade de processamento;

2.1.3 Notação dos Problemas da Produção

Na programação da produção, os problemas são descritos com a notação de três campos $\alpha \mid \beta \mid \gamma$ (PINEDO, 2016). O primeiro campo, α, descreve o ambiente das máquinas e ele contém apenas uma entrada de parâmetro. O segundo campo, β, caracteriza detalhes das restrições do ambiente e eles podem ter nenhum, um ou vários parâmetros. O último campo, γ, descreve o objetivo do ambiente de produção a ser minimizado ou maximizado, geralmente esse campo tem apenas um único parâmetro.

Os ambientes mais comuns de especificação na entrada do campo α de acordo com Pinedo (2016) podem ser:

- 1: Ambiente de máquina única;
- P_m: Ambiente com m máquinas idênticas em paralelo;
- Q_m: Ambiente com m máquinas em paralelo, porém com diferentes velocidades de processamento;
- R_m: Ambiente com m máquinas paralelas desconexas ou sem relação entre elas;
- F_m: Ambiente de *Flow Shop* com m máquinas em série;

- FF_m: Ambiente de *Flexible Flow Shop* ou *Flexible Flow Line* com m grupos de máquinas paralelas (estações) trabalhando com fluxo em série;
- J_m: Ambiente de *Job Shop* com m máquinas que trabalham de forma independente entre elas;
- FJ_m: Ambiente de *Flexible Job Shop* com m grupos de máquinas paralelas (estações) em série. Cada estação trabalha de forma independente entre elas;
- O_m: Ambiente de *Open Shop* com m máquinas com processamento único de tarefas nas máquinas.

No campo β, das restrições ou recursos das tarefas, as restrições mais comuns são descritas abaixo, de acordo com Pinedo (2016):

- ø: Descreve a abstenção de restrições no ambiente;
- r_j: Restrição que identifica a presença de data de liberação de tarefas;
- $prmp$: Restrição que determina que tarefas não precisam ser totalmente completadas no ciclo, ou seja, a tarefa pode ser interrompida na máquina e posteriormente retornar para ser completada;
- $prec$: Restrição que determina precedências de tarefas. Determinadas tarefas só podem iniciar na máquina m quando elas tiverem sido executadas em uma outra máquina predecessora;
- s_{jk}: Restrição que representa que as tarefas j na máquina k têm tempos de preparação antes de serem processadas ou entre processamentos no fluxo;
- $fmls$: Restrição que representa que n tarefas pertencem a mesma família e tais tarefas podem ou não ter tempos de processamento distintos;
- $brkdwn$: Restrição que representa que nem sempre determinada máquina estará disponível para processar tarefas no ciclo;
- nwt: Restrição que determina que tarefas não podem entrar em estado ocioso entre uma máquina e a outra.

As restrições do campo γ determinam a medida de desempenho a ser minimizada. Os objetivos mais comuns para esse campo, de acordo Pinedo (2016) e Fuchigami (2015), são:

- C_{max}: Conhecido como o *Makespan*, é a medida que determina a contagem de tempo de processamento até a última tarefa deixar o fluxo. Quanto menor o *Makespan*, melhor é o tempo de aproveitamento das máquinas no ambiente;

- $\bar{F}$: Medida que determina o tempo médio de fluxo das tarefas no processamento;
- L_{max}: Medida que calcula o tempo máximo de atraso das tarefas em relação ao tempo de entrega de cada uma;
- T_{max}: Medida que calcula o maior tempo de atraso das tarefas;
- $\sum w_j C_j$: Medida que calcula a soma ponderada do tempo de término em relação ao peso w_j de cada tarefa;
- $\sum w_j T_j$: Medida que calcula a soma ponderada do tempo de término em relação ao tempo de atraso T_j de cada tarefa;

2.1.4 Regras de Sequenciamento

As regras de prioridade vêm sendo estudadas desde o início da teoria de sequenciamento de produção e elas são classificadas de diferentes formas. A principal diferença entre elas são as prioridades estáticas e as prioridades dinâmicas, uma vez que as prioridades estáticas não dependem do tempo. Existem diferentes tipos de regras e muitas delas podem ser adaptáveis ou híbridas. A todo momento novas regras podem ser criadas para agregar à PP.

Fernandes e Filho (2010) e Pinedo (2016) descrevem as principais e as mais conhecidas regras de sequenciamento geralmente classificados por prioridade para determinar o sequenciamento das linhas de produção:

- **SIRO** - *Service in Random Order*: Regra que seleciona aleatoriamente quaisquer tarefas sem identificar nenhuma otimização de processamento;
- **ERD** - *Earliest Release Date First*: Regra que seleciona o menor tempo de liberação das tarefas visando reduzir o tempo de espera de processamento;
- **EDD** - *Earliest Due Date*: Regra que seleciona a tarefa com tempo de processamento mais curto para ser processado primeiro;
- **FCFS** - *First Come, First Serve*: Regra que determina que a primeira tarefa a ser alocada na fila é a primeira a ser processada;
- **SPT** - *Shortest Processing Time*: Regra que seleciona as tarefas com menor tempo de processamento para serem processadas primeiro;
- **LPT** - *Longest Processing Time*: Regra que seleciona as tarefas com o maior tempo de processamento para serem processadas primeiro;
- **WSPT** - *Weighted Shortest Processing Time*: Regra que seleciona as tarefas com o menor peso de prioridade de processamento para serem processadas primeiro;

- **WLPT** - *Weighted Longest Processing Time*: Regra que seleciona as tarefas com o maior peso de prioridade de processamento para serem processadas primeiro.

Na PP, tem-se também meios de solucionar os diferentes ambientes de produção por meio de métodos exatos e heurísticas como o Recozimento Simulado (MUSHARAVATI; HAMOUDA, 2015), busca tabu (KOCHHAR; MORRIS; WONG, 1988) e algoritmo genético (ZANDIEH; MOZAFFARI; GHOLAMI, 2010) que é o estudo desta pesquisa. Há outros estudos que utilizam heurísticas baseadas em sistema bio-inspirados, como Colônia de Formigas (ROSSI; LANZETTA, 2013) e Enxame de Partículas (GOHARI; SALMASI, 2015). Heurísticas podem ser construtivas e na PP elas trabalham de forma instrutiva, adicionando as tarefas na forma de alocação e realocação, trabalhando uma tarefa por vez (PINEDO, 2016). Por se tratar de métodos que trabalham de forma iterativa, nem sempre será possível encontrar uma solução ótima, porém, eles geralmente retornam resultados satisfatórios em tempo polinomial aceitável (PIRAMUTHU; RAMAN; SHAW, 1994).

Outro método de solução de problemas de PP é através da resolução de modelos de Programação Inteira Mista, que utiliza métodos exatos para encontrar soluções para os modelos propostos. Algoritmos como o *Branch-and-Bound* foram largamente usados nos primórdios dos estudos de sequenciamento da produção (PARK; STEUDEL, 1991). Posteriormente, outros métodos de solução exata foram introduzidos.

2.2 Mapeamento Sistemático

Para a revisão da literatura foi desenvolvido primeiramente um mapeamento sistemático conforme feito na tese de doutorado de Paula (2014), posto que a busca por estudos e trabalhos desenvolvidos do tema é feita nas bases de dados ao redor do mundo com palavras e expressões chaves sobre o tema.

Neste estudo, as bases de dados utilizadas foram os indexadores do Portal de Periódicos da Coordenação de Aperfeiçoamento de Nível Superior (CAPES), o *Scopus* e o *Science Direct*. Todos eles foram acessados entre novembro 2016 e março de 2017 no sítio da Universidade Federal de Goiás, Regional Catalão.

Após a busca pelos artigos na base, fez-se uma seleção por meio de títulos com o objetivo de reduzir a quantidade de amostras, buscando conteúdos mais próximos ao tema. A etapa final do estudo foi a leitura e análise dos itens selecionados, classificando-os por ano de publicação, periódicos publicados, país de publicação, tipos e métodos de otimização nos quais os estudos foram desenvolvidos. Por fim, com o mapeamento sistemático, a revisão bibliográfica pôde ser desenvolvida a partir do número final de artigos selecionados e uma evolução sobre o tema estudado foi traçada.

É possível observar através das tabelas e figuras apresentadas, a evolução das publicações de artigos em revistas de periódicos com o tema FFL no decorrer dos anos de 1981 a 2016 e do tema de FLM no decorrer dos anos de 1967 a 2014.

Foram considerados três conjuntos de expressões na busca (a) *Heuristics and Flexible Flow Line*, (b) *Scheduling and Flexible Flow Line*, (c) *Modeling and Flexible Flow Line*. Foram encontrados 1.396 artigos e apenas 49 artigos foram considerados como importantes no tema do FFL. Dos 49, 13 foram considerados no tema do FLM e contribuintes para o mapeamento da pesquisa. O tema FLM foi encontrado por meio da busca das expressões descritas anteriormente e se mostrou promissor durante o mapeamento para seleção do tema final dado sua escassez de estudos científicos e interesse prático na indústria.

Inicialmente, dos 1.396 artigos, 395 são relacionados com o tema (a), e destes 395 apenas 39 tinham o título aderente. No tema (b) foram encontrados 555 artigos, e apenas 37 foram apontados como título aderente. Já com o tema (c) foram obtidos 446 artigos, e após a leitura dos títulos, apenas 25 eram aderentes ao tema.

Após esta primeira seleção dos artigos, o próximo passo foi a leitura dos resumos e busca por repetições entre os indexadores. Do total de 101 artigos com títulos aderentes, apenas 73 foram selecionados na amostragem do mapeamento e por fim, 49 deles foram preferidos para a leitura completa. A Tabela 2.1 mostra o resumo dos resultados obtidos através das buscas nos indexadores da base de dados.

Tabela 2.1 – Resumo dos resultados obtidos nas pesquisas nas bases de dados.

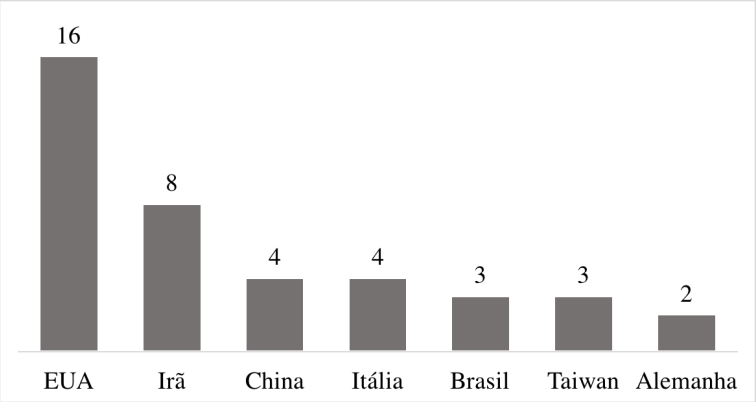
Base de Dados	Palavras-chaves					
	Heuristics e FFL		*Scheduling* e FFL		*Modeling* e FFL	
	Número de Artigos					
	Inicial	Título Aderente	Inicial	Título Aderente	Inicial	Título Aderente
Periódico CAPES	76	24	311	31	184	22
Scopus	60	2	115	1	116	1
Science Direct	259	13	129	5	146	2
TOTAL	395	39	555	37	446	25
Artigos com Resumo Aderente e sem Repetições	32		27		14	
Artigos *Sampling*	73					
Artigos Selecionados do FFL	49					
Artigos Relacionados ao FLM	13					

Fonte: o autor.

Desses 49 artigos preferidos para leitura é possível distinguir uma predominância grande de pesquisas desenvolvidas nos EUA e Irã, com 16 e 8 artigos, respectivamente. Países como; China, Itália e Taiwan também possuem uma quantidade significativa de artigos com o tema FFL. Já com o tema do FLM, a predominância de publicações está na Turquia e Taiwan. Três trabalhos com o tema do FFL foram desenvolvidos no Brasil. De forma geral, a Figura 2.2 e a Figura 2.3 apresentam a quantidade de artigos por país. O critério de seleção dos países dos artigos foi por meio do país de origem do autor principal.

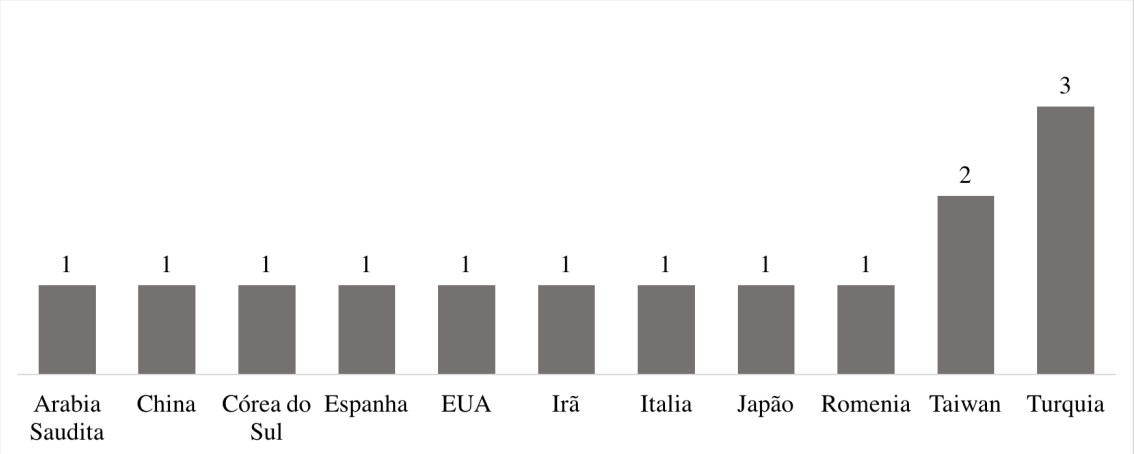
Também foi evidenciado, após a leitura dos artigos, os tipos de métodos de otimização

Figura 2.2 – Artigos publicados por Países com o tema *Flexible Flow Line*



Fonte: o autor.

Figura 2.3 – Artigos publicados por Países com o tema *Flow Line* Misto



Fonte: o autor.

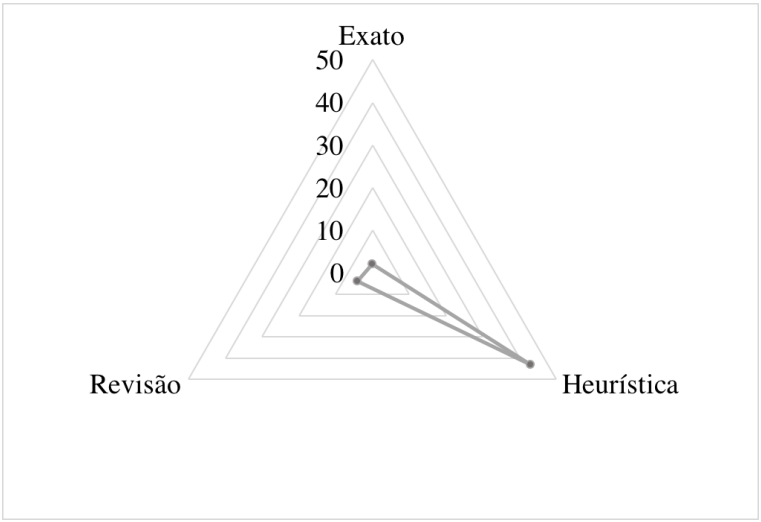
em cada pesquisa. Vários métodos de pesquisa foram desenvolvidos no decorrer dos anos e, em linhas gerais, eles têm um papel importante nos resultados das pesquisas desenvolvidas. O método de solução de um problema é crucial para determinar sua eficácia e performance em termos de otimização processual.

Atualmente, os métodos são classificados entre métodos exatos e métodos heurísticos. Nos métodos exatos está incluído o *Branch & Bound* (BOLAT; SAVSAR; AL-FAWZAN, 1994), rede *Petri Net* (LEWIS; HORNE; ABDALLAH, 2000), entre outros. Nos métodos heurísticos estão incluídos sistemas de otimização como Algoritmo Genético (SIKORA; CHHAJED; SHAW, 1996), Enxame de Partículas (GOHARI; SALMASI, 2015), Recozimento Simulado (HENDIZADEH *et al.*, 2008), entre outros.

Durante a leitura foi possível categorizar os títulos entre tipos de metodologia empregados na pesquisa: Exato, Heurística, Revisão. Dentre eles, o maior número de artigos por categoria no FFL é a heurística com 43 artigos, seguido por revisão e exato com 4 artigos e

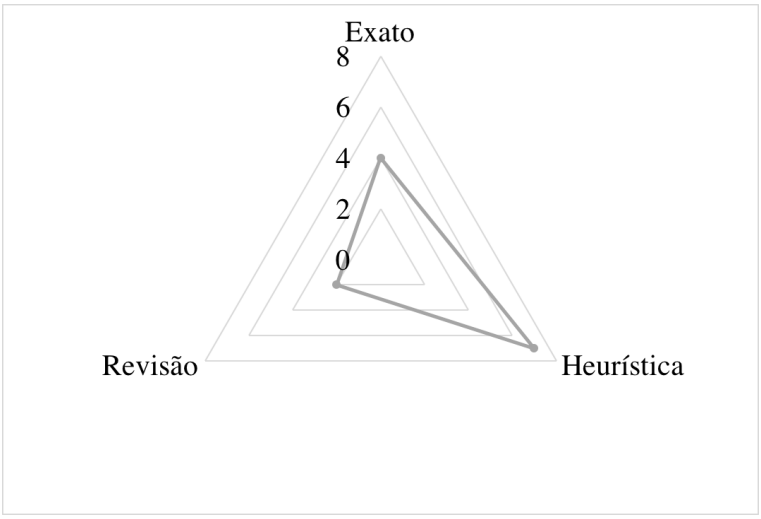
exato com 2 artigos. Já com o tema FLM, o tema heurística tem 7 artigos, seguido por 4 artigos com métodos exatos e 2 artigos de revisão. As Figuras 2.4 e 2.5 mostram as principais metodologias empregadas para o problema de FFL e FLM respectivamente.

Figura 2.4 – Artigos publicados por metodologia com o tema *Flexible Flow Line*



Fonte: o autor.

Figura 2.5 – Artigos publicados por metodologia com o tema *Flow Line* Misto

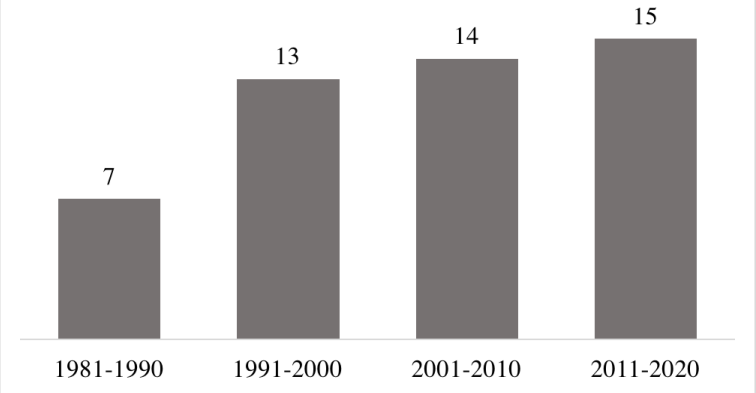


Fonte: o autor.

Na análise temporal de publicações sobre o FFL, pode-se observar que dos 49 artigos da amostragem, cerca de 30% foram publicados nos últimos 10 anos, e na primeira década do século XXI, a porcentagem de publicação ficou próxima dos 14%, com apenas 7 artigos publicados. Isto demonstra que o desenvolvimento nesta área vem crescendo, principalmente nas duas últimas décadas. Com o tema FLM é possível observar que nas décadas de 1960 a 1990 apenas 2 artigos foram publicados e este valor aumentou para 6 na década

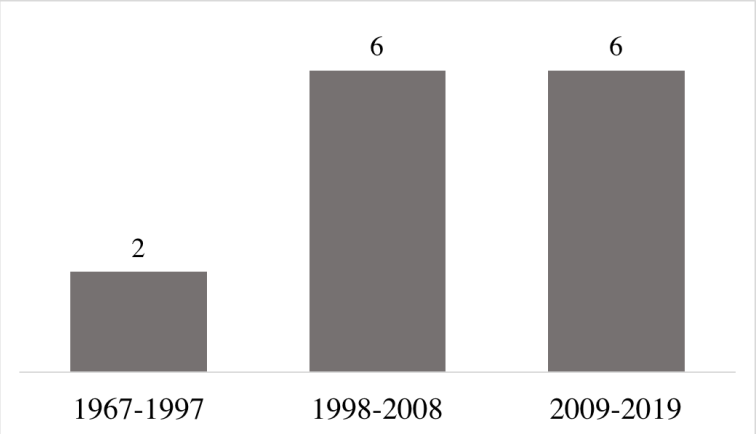
seguinte e manteve-se em 6 na década atual. As Figura 2.6 e 2.7 mostram graficamente a tendência de aumento de trabalhos na área.

Figura 2.6 – Artigos publicados no decorrer dos anos com o tema *Flexible Flow Line*



Fonte: o autor.

Figura 2.7 – Artigos publicados no decorrer dos anos com o tema *Flow Line* Misto



Fonte: o autor.

Dos artigos selecionados, 31 deles estão publicados em 6 importantes periódicos conforme mostra a Tabela 2.2. É importante considerar o periódico do qual o artigo foi publicado durante o mapeamento do tema para identificar através do foco e escopo da revista uma visão mais dilatada das tecnologias e metodologias de estudo empregadas, norteando a revisão da literatura. Observa-se na Tabela 2.2 que dos 31 artigos mencionado, 9 são da revista *European Journal of Operational Research*, uma revista voltada para a área do estudo apresentado nesta dissertação.

Por fim, durante o mapeamento foi possível identificar também os métodos de otimização e resolução empregados nos trabalhos mapeados para delinear o tipo de método que os autores utilizaram durante as pesquisas. Mapear o método de otimização é um fator importante para definir o escopo de uma pesquisa durante a revisão bibliográfica.

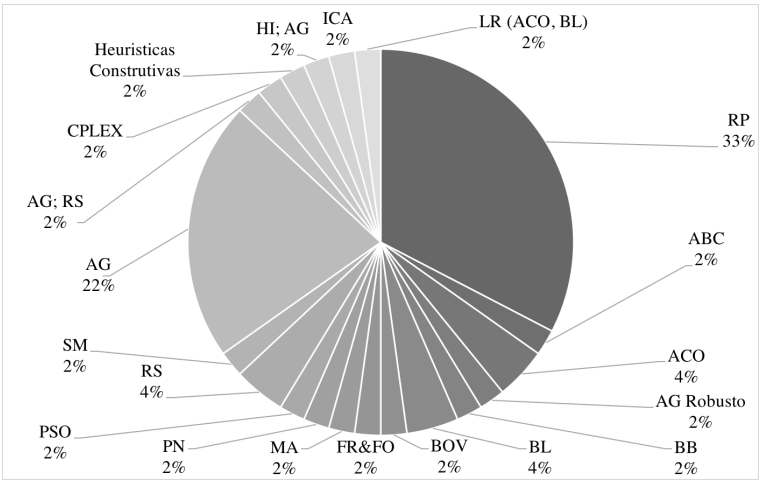
Tabela 2.2 – Periódicos dos artigos publicados com o tema FFL e FLM

Periódico	Nº de artigos
Annals of Operations Research	4
European Journal of Operational Research	9
Journal of Intelligent Manufacturing	6
Journal of Manufacturing Systems	4
Journal of Operations Management	2
The International Journal of Advanced Manufacturing Technology	6

Fonte: o autor.

Dos 49 artigos sobre o tema FFL, foram encontrados 40 diferentes métodos de otimização. Dentre eles, 33% utiliza Regras de Prioridade com algum algoritmo Heurístico, seguidos por 22% dos artigos utilizando o Algoritmo Genético. Dos artigos com o tema FLM, 19% deles usam o Lagrangeano Relaxado por meio do CPLEX como metodologia de otimização, seguido por Regras de Prioridade e Recozimento Simulado com representação de ambas em 18% dos artigos. As Figuras 2.8 e 2.9 mostra graficamente todos os métodos de otimização encontrados os trabalhos mapeados.

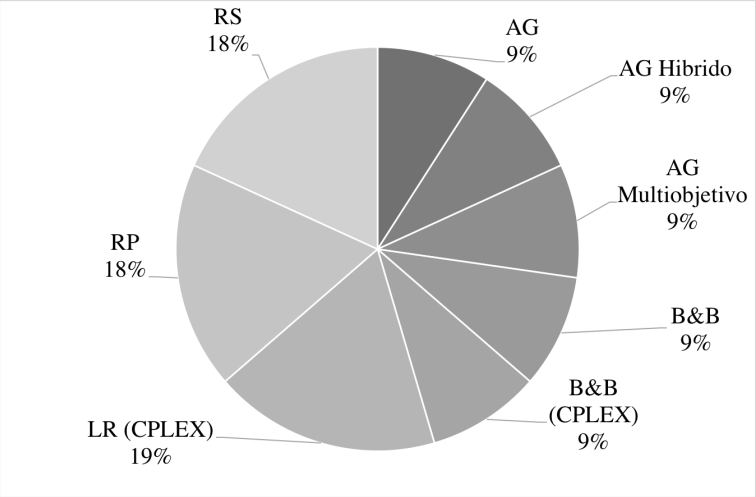
Figura 2.8 – Artigos publicados conforme a técnica de otimização com o tema *Flexible Flow Line*



Fonte: o autor.

Com mapeamento do tema FFL e FLM, a síntese dos artigos publicados selecionados até a atualidade foi feita, sendo possível visualizar os principais trabalhos e impacto deles desde sua publicação. É importante enfatizar que após o mapeamento, a leitura e revisão dos artigos selecionados é descrita para sumarizar os tópicos pertinentes como técnicas exploradas e lacunas que possam existir no decorrer dos anos.

Figura 2.9 – Artigos publicados conforme a técnica de otimização com o tema *Flow Line* Misto



Fonte: o autor.

2.3 Trabalhos da Literatura sobre o *Flexible Flow Line*

A revisão desta seção aborda de forma breve os estudos para o *Flexible Flow Line*. No mapeamento apresentado anteriormente, foi possível descrever a síntese dos artigos publicados até a atualidade, sendo possível visualizar os principais trabalhos. A Tabela 2.3 resumem os trabalhos encontrados na literatura para o *Flexible Flow Line* e ela traz detalhes sobre restrições dos problemas, medidas de desempenho (Função objetivo) que os autores buscavam na pesquisa e métodos de solução que eles utilizaram.

Wittrock (1985) introduziu o FFL com um certo tipo de ambiente de manufatura e com base em sua descrição, ele nomeou o problema como *Flexible Flow Line* (FFL). Sua definição consistia em identificar estações de processamento com máquinas alocadas em série com tarefas sendo executados em cada máquina. Tarefas podiam pular, ou não, máquinas. As tarefas seguiam uma sequência da esquerda para a direita formando uma linha. A princípio, as máquinas não tinham tempo de *setup* e em determinadas máquinas haviam *Buffering* (Filas) que garantiam que os trabalhos fossem operados em ordem.

As restrições do problema definido por Wittrock (1985) era de carregamento e de sequência, ponto que no carregamento o objetivo era minimizar o tempo das filas e na sequência era minimizar a fila de espera para o processamento da tarefa. O autor utilizou de um algoritmo que executava três passos: (1) alocar tarefas pela regra LTP. (2) sequenciar as tarefas pela heurística de balanceamento dinâmico. (3) computar o tempo de processamento das tarefas nas máquinas. Foram testadas instâncias próximas ao mundo real e o autor mostrou em seus resultados que o algoritmo proposto melhorou em aproximadamente 3 horas o tempo de produção gasto na linha de produção.

A heurística da Busca Local para o FFL foi usado por Kochhar e Morris (1987) sendo

Tabela 2.3 – Trabalhos encontrados na literatura sobre FFL e suas variantes.

Tipo	Autor(es)	Restrições	Função Objetivo	Método de Resolução	País
Exato	Lewis, Horne e Abdallah (2000)	–	Makespan	PN	EUA
Heurística	Wittrock (1985)	–	Makespan; Minimização do Fluxo de Estoque	RP	EUA
	Kochhar e Morris (1987)	Setup; Finite Buffers; Interrupção	Makespan; Minimização do Fluxo de Estoque	BL	EUA
	Kochhar, Morris e Wong (1988)	Setup; Carregamento Finito; Interrupção	Makespan	BL	EUA
	Wittrock (1988)	–	Makespan; Minimização do Fluxo de Estoque	RP	EUA
	Melloy, Soyster e Chandra (1990)	Carregamento; Interrupção de Máquinas	Minimização do Tempo de Fluxo	RP	Holanda
	Park e Steudel (1991)	Tempos de Setup	Makespan	RP	EUA
	Ding e Kittichartphayak (1994)	–	Makespan	RP	EUA
	Sikora, Chhajed e Shaw (1996)	Sequência dependente; Tempos de Setup; Carregamento; Data de Entrega	Minimização do Fluxo de Estoque	AG	EUA
	Sikora (1996)	Sequência dependente; Tempos de Setup; Carregamento; Data de Entrega	Minimização do Fluxo de Estoque	SM	EUA
	Agnetis *et al.* (1997)	–	Makespan	RP	Itália
	Lee, Sikora e Shaw (1997)	Dimensionamento de Lotes	Makespan	AG	EUA
	Lucertini, Pacciarelli e Pacifici (1998)	–	Makespan	RP	Itália
	Fanti *et al.* (1998)	Tempos de Setup; Tempos de Transporte	Makespan	RP	Itália
	Costa e Ferreira (1999)	Família de Tarefas	Makespan	RP	Portugal
	Sawik (2000)	Carregamento Finito; Interrupção	Makespan	RP	Polônia
	Buzzo e Moccellin (2000)	–	Makespan	AG; RS	Brasil
	Armentano e Mazzini (2000)	Tempos de Setup; Data de Entrega	Makespan	AG	Brasil
	Leu e Hwang (2002)	–	Makespan	AG	Taiwan
	Kurz e Askin (2003)	Tempos de Setup; Sequência Dependente	Minimização do Tempo de Fluxo	Heurísticas Construtivas	EUA
	Kurz e Askin (2004)	Sequência Dependente	Makespan	HI; AG	EUA
	Guo *et al.* (2008)	–	Minimização da Soma Ponderada do Atraso	AG	China
	Hendizadeh *et al.* (2008)	Família de Tarefas; Tempos de Setup; Sequência Dependente	Makespan	RS	Canadá
	Alexandros e Chrissoleon (2009)	Buffers Intermediário	Makespan	RP	Grécia
	Chen e Chen (2009a)	–	Makespan	RP	Taiwan
	Jolai *et al.* (2009)	No-wait; Janelas de Processamento; Rejeição de Tarefas	Makespan	AG	Irã
	Tavakkoli-Moghaddam, Safaei e Sassani (2009)	Interrupção de Precedência	Makespan	MA	Irã
	Chen e Chen (2009b)	–	Minimização do Atraso	BB	Taiwan
	Naderi, Zandieh e Ghomi (2008)	Sequência dependente; Tempos de Setup; Manutenção Preventiva	Makespan	BOV	Irã
	Zandieh, Mozaffari e Gholami (2010)	Sequência dependente; Tempos de Setup; Liberação de Máquinas	Makespan	AG Robusto	Irã
	Palaniappan (2010)	Sequência Dependente	Makespan	RP	Índia
	Palaniappan e Jawahar (2011)	–	Makespan	AG	Índia
	An e Yan (2011)	–	Makespan	ACO	China
	Zade, A e Bagher (2013)	–	Makespan	AG	Irã
	Stadtler e Sahling (2013)	Dimensionamento de Lotes; Atraso de Tarefas	Makespan	FR&FO	Alemanha
	Sheikh (2013)	Due Window; Atraso processamento; Rejeição de Tarefas	Makespan	AG	EUA
	Rossi e Lanzetta (2013)	Buffers	Makespan	ACO	Itália
	Gohari e Salmasi (2015)	Data de Entrega; Data de Liberação	Makespan	PSO	Irã
	Saif *et al.* (2014)	–	Minimização do Tempo de Fluxo	ABC	China
	Khalili e Naderi (2015)	Tempo de Setup; Sequência Dependente	Makespan; Minimização do Atraso	ICA	Irã
	Musharavati e Hamouda (2015)	–	Makespan	RS	Catar
	Fuchigami, Moccellin e Ruiz (2015)	Tempo de Setup	Makespan	RP	Brasil
	Yu e Seif (2016)	–	Minimização do Atraso e de Manutenção	AG	EUA
	An e Yan (2016)	–	Minimização do Atraso	LR (ACO, BL)	China
	Kia, Ghodsypour e Davoudpour (2017)	Tempos de Setup; Sequência Dependente	Minimização do Atraso; Minimização do Tempo de Fluxo	RP	Irã
Revisão	Fisher (1981)	Revisão Sobre Lagrangeano	–	–	EUA
	Piramuthu, Raman e Shaw (1994)	Revisão da Literatura	–	–	EUA
	Quadt e Kuhn (2007)	Revisão da Literatura	–	–	Alemanha

Fonte: o autor.

esta técnica de otimização testada em um problema real. No estudo, os autores definiram FFL como uma linha de produção que continha estações com máquinas idênticas em cada estação. As restrições consideradas no problema foram tempo de *setup*, *Buffer* finito, e potenciais máquinas quebradas durante o processo. Os autores deste trabalho apresentaram a existência de diferentes técnicas de otimização, mas consideraram a Busca Local no estudo, dada seu sucesso no problema em específico.

Basicamente, a Busca Local de acordo com Kochhar e Morris (1987) mostrada na técnica de solução para resolver o FFL, funciona com a geração de n sequências com soluções factíveis e então, novas sequências são geradas por meio da solução anterior até chegar em uma solução melhor. Na Busca Local, a técnica pode chegar em soluções consideradas ótimas, porém trata-se de ótimos locais quando a sensibilidade da busca não é melhorada durante as iterações. Os resultados foram de formas práticas razoáveis e próximo do real.

Os resultados obtidos por Kochhar e Morris (1987) foram divididos em três partes. Na primeira, eles criaram instâncias onde era possível utilizar a Busca Local para analisar as restrições do problema. Na segunda, eles delinearam os métodos rodando testes por 10 dias em cada instância para analisar como o método correspondia em cada instância. Na terceira parte, eles criaram instâncias próximas ao mundo real com aproximadamente 16 máquinas divididas em 7 estações capazes de escalonar 45 tarefas na linha de produção.

Os Kochhar e Morris (1987) consideraram os resultados promissores quando chegaram aos resultados próximos do ótimo ao usar a Busca Local. Eles enfatizaram que, apesar dos resultados se mostrarem satisfatórios ainda é necessário aprimorar o método para o problema em questão, pois, em algumas aplicações, o método se mostrou ineficaz.

Sikora, Chhajed e Shaw (1996) apresentaram a variação de um FFL consistindo de múltiplas máquinas em cada estação, sendo cada máquina com uma capacidade diferente, tempo de *Setup*, sequência dependente e *Buffering*. O objetivo dos autores era minimizar o *Makespan* sem violar as restrições do problema e, ainda sim minimizar os custos de inventário de produção. Para solucionar este modelo, os autores usaram a heurística *Silver-Meal*, avaliando a performance que o método de otimização poderia proporcionar.

Em linhas gerais, Sikora, Chhajed e Shaw (1996) apresentaram um problema do mundo real por meio de um sistema de manufatura que produz placas de circuitos. Os autores mostraram problemas ao implementar o SM, uma vez que eles modificaram o método para melhor adaptação às restrições que o problema apresentava. Nos experimentos, os autores consideraram dois grupos de problemas com 10 e 20 tarefas, o tamanho dos lotes dos produtos a serem manufaturados foram gerados aleatoriamente entre 50 e 200 e, por fim, foram considerados 3 tipos de *layout* de máquinas com 5, 20 e 35.

Os autores mostraram que os resultados das instâncias em todos os cenários demonstraram uma grande melhora em relação a outros métodos comparados com autores já refe-

renciados. Foi observado por eles que ao integrar um maior número de lotes o *Makespan* aumenta ligeiramente, porém, os valores não são significantes mantendo assim a qualidade da melhora ao usar o RS para otimizar o problema apresentado.

No trabalho desenvolvido por Agnetis *et al.* (1997), os autores analisaram um sistema de manufatura em larga escala projetada para fabricar automóveis em uma planta de processamento localizada na Itália. O estudo analisou todo o processo de produção em busca de melhorar a performance de fabricação de veículos na planta. Era conhecido que na planta ocorria a fabricação de dois modelos de veículos com diferentes versões finais.

Agnetis *et al.* (1997) consideraram um modelo de manutenção em fluxo e propôs a solução do modelo por meio de regras de prioridade. A Planta estudada segue um sistema automatizado de fabricação e o caminho de fabricação seguiam fluxo contínuo, porém, existia muitos gargalos na fabricação, como falha de encontro entre datas de entrega e reposição de matéria-prima.

Após a análise completa da linha de produção, Agnetis *et al.* (1997) consideraram dois tipos básicos de regras de despacho: *Puxe* e *Empurre*. Basicamente, na regra "Puxe", era selecionado o veículo no processo anterior que tem o menor tempo de espera e então o veículo era alocado no próximo estágio da fabricação, ou seja, o sistema seleciona o melhor candidato para ser movido para à próxima fabricação garantindo um menor tempo de fabricação do produto. Na regra "Empurre", o veículo é simplesmente empurrado para o próximo estágio disponível assim que ele for liberado do estágio anterior.

Agnetis *et al.* (1997) obtiveram resultados satisfatórios nas regras implementadas e cada regra necessitou de mínimas mudanças no sistema para que se fosse melhorado o sistema da planta de fabricação analisada. Foi possível observar também que em uma planta de fabricação em que a larga escala de utilização é considerada a regra de despacho de Puxe é mais atraente para obter melhores resultados (Aumento de 1.200 para 1.300 veículos) em redução de tempo de fabricação.

Na análise da complexidade de um modelo com ciclos e reentradas no sistema de manufatura classificado com FFL, Lewis, Horne e Abdallah (2000) abordaram a solução desse problema usando PN. Os autores chamaram o problema de Sistema de Manufatura Flexível e o modelo era capaz de determinar o roteamento de tarefas, a seleção dessas tarefas e os recursos associados na manufatura das mesmas. Tal estratégia foi classificada como um subtipo do *Job Shop*. Os autores mostraram que esse tipo de modelo é, geralmente, solucionado por meio do B&B e também é extensivamente resolvido por PN.

O problema do estudo em questão é classificado por Lewis, Horne e Abdallah (2000) como NP-Completo e ele usa um conjunto de variáveis capazes de serem selecionadas para alocação e então analisadas para serem roteadas ou associadas ao próximo estágio da manufatura. Lewis, Horne e Abdallah (2000) apresentaram resultados satisfatórios para provar a

complexidade computacional por meio da Rede PN, mostrando sua complexidade por meio de demonstrações e teoremas.

Com o objetivo de explorar regras de sequenciamento dependente com tempos de *Setup* no FFL, Kurz e Askin (2003) selecionaram três tipos de heurísticas (Problema do Caixeiro Viajante, algoritmo de Johnson, e regras de sequenciamento). Os autores compararam a performance dos métodos no FFL seguindo práticas comuns, em que toda tarefa deve ser processada em todas as estações. Nas estações, as tarefas podem saltar máquinas, todas as máquinas estão disponíveis no tempo 0 e as tarefas consideram o tempo de programação (*Setup*) no início de cada processo em cada máquina na estação.

No estudo, a heurística que considera o TSP sofreu algumas modificações para o método melhor se adaptar ao problema e, na modificação, o tempo de *setup* dependente à máquina se mostrou irrelevante no experimento. Diferente do TSP, a heurística baseada na regra de sequenciamento do algoritmo de Johnson considerou os tempos de *setup* da sequência de forma dependente à máquina. Nos experimentos, Kurz e Askin (2003) geraram conjuntos de instâncias com combinações de 10 máquinas em cada estação e tempos de *setup* distribuídos de forma uniforme entre as máquinas, com uma variação uniforme entre 20% e 40%.

Os autores apresentaram nos resultados, a comparação entre os métodos e salientaram que houve uma melhora média no *Makespan* de 6% em relação ao *Lower Bound* do problema quando utilizado a heurística baseada no TSP. Por fim, Kurz e Askin (2003) puderam concluir que a heurística baseada na regra de Johnson oferece excelentes alternativas, porém, a heurística baseada no TSP consegue resultados relativamente melhores. Em todos os resultados das heurísticas, a performance manteve-se a mesma para qualquer tamanho de instância, exceto quando foi colocado mais de 100 tarefas.

Quadt e Kuhn (2007) fizeram uma revisão sistemática do tema FFL especificando-o como um *Flow Shop* híbrido de máquinas paralelas, sendo que as tarefas passariam por cada uma das máquinas até completar o procedimento, quando caminhado para todas as estações. Os autores definiram uma taxonomia para classificar a metodologia do FFL. Quadt e Kuhn (2007) apresentaram diversos trabalhos desenvolvidos no decorrer dos anos mostrando métodos ótimos e heurísticos capazes de solucionar o FFL.

Nos métodos heurísticos os Quadt e Kuhn (2007) apresentaram dois tipos de aproximações: a Holística e a de decomposição. Heurísticas, não retornam necessariamente, a solução ótima para o problema, porém, em problemas que requerem tempo polinomial para serem resolvidos, as heurísticas se mostram excelentes candidatas para obter boas soluções. Nos procedimentos ótimos, onde a maioria deles buscam pelo *makespan*, o FFL tem se mostrado interessante ao uso como metodologia de solução, porém ele não se mostra eficaz quando considerado grandes instâncias.

Por fim, Quadt e Kuhn (2007) concluíram que a revisão taxonômica da literatura do *Flexible Flow Line* categoriza os métodos de solução em geral para o problema. Os autores enunciaram que o estudo sistemático ajuda a compreender a existência de métodos de solução e também de analisar a introdução de novos procedimentos para guiar futuros trabalhos. Foi também observado que existe pouca pesquisa relacionada ao FFL em procedimentos integrados ao mundo real e tal ponto é importante considerar ao selecionar temas em trabalhos futuros.

Uma proposta de solução do FFL com tarefas agrupadas em família dependente de sequência por Busca Tabu foi estudada por Hendizadeh *et al.* (2008) em busca do *Makespan*. No trabalho, os autores mediram a eficiência e efetividade da Busca Tabu quando comparada com meta-heurísticas que resolveram o mesmo problema apresentado pelos autores. Nos experimentos, os autores mostraram que a Busca Tabu pode solucionar, em alguns casos, chegar a resultados ótimos. A Busca Tabu proposto utiliza operadores de vizinhança, que são acionados a cada iteração para melhorar a sequência inicial de tarefas alocadas na linha de manufatura, buscando melhores resultados por meio da exploração da vizinhança.

As instâncias consideradas por Hendizadeh *et al.* (2008) foram geradas por meio de uma distribuição uniforme entre 1 e 20 para os tempos de *Setup* das famílias de tarefas, e o número de famílias e máquinas em cada instância era entre 3 e 10, o número de tarefas ficou entre 1 e 10. Os experimentos foram realizados em aproximadamente 900 instâncias e os autores apresentaram resultados na melhora do tempo computacional (0,9 segundos em média) em relação aos demais algoritmos da literatura para o problema em questão.

O estudo de Jolai *et al.* (2009) resolveu um modelo de programação inteira mista para o FFL com predeterminadas janelas de tempo, rejeição de tarefas e *no-wait*. Os autores propuseram um Algoritmo Genético modificado para maximizar os lucros de produção. No problema, as máquinas estão alocadas em estações e cada tarefa pode ser processada em qualquer máquina da estação, mas cada máquina pode apenas processar uma tarefa por vez e todas as máquinas estão disponíveis no tempo zero.

Nos experimentos, Jolai *et al.* (2009) apresentaram o Algoritmo genético, um método de solução baseado na teoria da evolução. Jolai *et al.* (2009) propuseram um AG modificado para o problema apresentado, sem alterar os principais passos do AG original e, então, comparou com as soluções ótimas do LINGO 8 para as mesmas instâncias com 5, 10, 15, 20 e 50 tarefas. Ao todo, os autores testaram 60 instâncias e o AG apresentou excelentes resultados com erros máximos de 1,02% e 65% de instâncias com resultado ótimo encontrado em comparação com o LINGO 8.

Tavakkoli-Moghaddam, Safaei e Sassani (2009) executaram o estudo da eficiência do Algoritmo Memético combinado com busca local para solucionar o FFL com bloqueio de processos e *buffers* de máquinas. Basicamente, os autores apresentaram o problema capaz de produzir diferentes tipos de produtos processados em estações de produção com diferen-

tes máquinas em cada estação. Por se tratar de instâncias de grande porte, principalmente quando considerado o mundo real, obter resultados ótimos para tais instâncias se torna difícil quando considerado métodos tradicionais de otimização, levando os autores a adotar o algoritmo memético para os experimentos do estudo.

Tavakkoli-Moghaddam, Safaei e Sassani (2009) mostraram que o algoritmo memético é composto por uma estrutura de vetores e matrizes que representam o tempo de processamento de cada tarefa e a ordem de execução na linha de processamento. O método usa de operadores de mutação e busca vizinhança para encontrar melhores soluções baseando-se na solução da iteração. Nos experimentos, os autores utilizaram de instâncias da literatura com 10, 20, 30 e 40 tarefas e 2, 3, 4, 5 e 6 máquinas, sendo que cada tarefa possuía um tempo de processamento gerado a partir de uma distribuição uniforme entre 1 e 10. Os resultados foram testados no AG clássico e no AM para quesitos de comparação e o AM se mostrou em média 22,42% mais eficiente que o AG.

Uma proposta de um problema com multi-objetivo para aumento de lucros e minimização do atraso do FFL com janelas de tempo entre as tarefas foi proposta como programação inteira mista por Sheikh (2013). O autor mostrou que o problema era da categoria determinística polinomial, com tempos computacionais ineficientes para métodos exatos quando considerado grandes instâncias. Então eles selecionaram o Algoritmo Genético.

Nos experimentos, Sheikh (2013) mostrou que o AG considera cada solução por meio da população de cromossomos e esses cromossomos representam o ponto de busca no espaço para recombinar e reproduzir melhores soluções. Para comparar a eficiência dos resultados do AG, o autor utilizou o LINGO 8 em instâncias com 5, 6, 7, 10 e 20 tarefas. Os resultados mostraram um erro máximo de 1,33% para instâncias da classe 2 e 1,83% para instâncias da classe 5, mostrando que o AG consegue resolver o problema de forma eficiente, principalmente, para instâncias em que o tempo computacional excede 2 horas no método exato. Nos experimentos, o AG conseguiu encontrar o resultado ótimo de 251 instâncias das 390 testadas.

Com o objetivo de chegar próximo ao real das linhas de produção industrial, Khalili e Naderi (2015) apresentaram um modelo de programação inteira mista para o FFL com sequência dependente com tempos de *Setup*. Os autores consideraram o problema em questão, pois eles identificaram que muitos problemas da literatura não consideram tempos de *Setup* e, indústrias como a têxtil, química e automotiva usam de tempo de *setup* para preparar a máquina antes de processar as tarefas.

Nos experimentos, Khalili e Naderi (2015) utilizaram uma heurística que simula o comportamento imperialista (ICA). Basicamente, o método funciona com a simulação de três processos sócio-políticos dentro da hierarquia de poder; o imperialismo, as colônias de produção e as colônias independentes. Khalili e Naderi (2015) usaram o Algoritmo Genético (NSGA II) para comparar os resultados obtidos em um conjunto de instâncias com 8,

10 e 12, 20, 50, 80, 120 tarefas associadas com máquinas que variam entre 2 a 5. Em termos de tempo computacional, os autores mostraram que o ICA para instâncias pequenas demorou em média 0,01 s e o NSGA II demorou cerca de 7,21 s para as mesmas instâncias. Em geral, o ICA se mostrou 0,08 segundos em média mais eficiente que o NSGA II. Os autores também apontaram que trabalhos futuros com mais restrições e objetivos devem ser introduzido para atender a demanda crescente das linhas reais de produção, cada vez mais sofisticadas.

Avaliando de forma geral os trabalhos, foi possível identificar que a evolução do FFL desde o primeiro caso, com apenas um objetivo, tornou-se multi-objetivo com diferentes critérios dentro do ambiente da produção. Diversos autores introduziram no ambiente FFL características que vêm sendo aproximadas ao real dentro da indústria atualmente. O uso de heurística para solucionar os modelos propostos são, na maioria das vezes, o melhor método de solução. Métodos exatos são esporadicamente estudados devido à complexidade do problema que em todas as ocasiões são classificados com NP-Difícil. Nesses casos, o uso de regras de sequenciamento e heurísticas podem ser os melhores métodos existentes para a solução do problema FFL.

2.4 Trabalhos da Literatura sobre o *Flow Line* Misto

A revisão desta seção aborda de os estudos para o *Flow Line* Misto. Embora o FLM ainda não seja tão estudado através dos tempos, existe um número satisfatório de pesquisadores que abordam o FLM empregando uma grande variação de restrições e metodologia de solução, o que em linhas gerais, engrandece o alcance do FLM nos estudos atuais. A Tabela 2.4 resume os trabalhos encontrados na literatura para o *Flow Line* Misto e ela também traz detalhes sobre restrições dos problemas, medidas de desempenho (Função objetivo) e tamanho das instâncias que os autores buscavam na pesquisa e métodos de solução que eles utilizaram.

Flow Line misto é a linha de produção que tem mais de um único tipo de produto em uma determinada linha de produção conforme aponta Thomopoulos (1967). Em seu estudo, ele descreve um procedimento adaptado de um problema que requeria a fabricação de diferentes modelos de um único produto na linha de produção. O autor descreve que o *Flow Line* Misto (FLM) é um problema mais sofisticado que requer uma fabricação mista alternando em máquinas, diferente dos produtos fabricados em lotes com inventário previsível.

Thomopoulos (1967) ainda em seu estudo faz uma análise para definir e discutir termos praticados na linha de produção na época. Termos como: grupo de trabalhadores, turno de tarefas, sequenciamento de produção, operadores de máquina e ciclo de processamento foram usados em seus modelos para desenhar o arranjo físico ideal para a minimi-

Tabela 2.4 – Trabalhos encontrados na literatura sobre *Flow Line* Misto

Tipo	Autor(es)	Restrições	Função Objetivo	Método de Resolução	Pais
Exato	Bolat, Savsar e Al-Fawzan (1994)	–	Makespan	B&B	Arábia Saudita
	Eliiyi e Özlen (2008)	Conjunto de tarefas	Makespan	LR (CPLEX)	Turquia
	Lin e Chu (2013)	Capacidade de Armazém	Makespan	B&B (CPLEX)	Taiwan
	Lin e Chu (2014)	Capacidade de Armazém; Custo de Inventário	Makespan	LR (CPLEX)	Taiwan
	Fleszar (2017)	–	Makespan	CPLEX	Líbano
Heurística	Thomopoulos (1967)	Conjunto de tarefas	Makespan	RP	EUA
	Karabatı e Sayın (2003)	Precedência de Tarefas	Makespan	RP	Turquia
	Cho *et al.* (2005)	Tempos de Setup	Makespan	RS	Córea do Sul
	Kara (2008)	–	Minimização de sobrecarga	RS	Turquia
	Hashem e Aryanezhad (2009)	–	Redução do tempo de Espera	AG Hibrido	Irã
	Shao *et al.* (2010)	Buffers Intermediário	Makespan; Maximização Inventário	AG Multi-objetivo	China
	Costantino *et al.* (2014)	Capacidade de linha	Makespan	AG	Itália
Revisão	Copaceanu (2006)	Revisão da Literatura	–	–	Romênia
	Jong (2009)	Revisão da Literatura	–	–	Japão

Fonte: o autor.

zação do tempo de ciclo de processamento de tarefas nas máquinas alocadas de forma mista em estações. Nos experimentos, o autor usou de métodos heurísticos baseado na teoria da PP.

Thomopoulos (1967) apresentou em suas conclusões que os resultados obtidos nos procedimentos experimentados melhoraram em 0.6% em relação a sequência atual de programação. Os procedimentos mostraram ser interessantes para definir novos parâmetros na linha de produção do FLM.

Bolat, Savsar e Al-Fawzan (1994) mostraram o problema de FLM como um desafio ao se considerar experimentos em busca da sequência ótima, quando considerado no problema a variedade de tarefas a serem executadas. Um exemplo dado em seu estudo foi sobre uma indústria automotiva em que os autores buscavam a minimização do custo de fabricação por meio da sequência ótima de produção. Foi considerado no problema tempos de *Setup*

Nos experimentos, Bolat, Savsar e Al-Fawzan (1994) utilizaram o B&B para encontrar soluções ótimas para o problema em estudo, porém o B&B requer alto custo computacional mesmo em instâncias pequenas (10-100 tarefas) e, por razão disso, outros dois algoritmos heurísticos foram implementados para resolver instâncias de tamanho real. As instâncias reais foram obtidas por meio de uma indústria de veículos automotivos com cerca de 1000 veículos fabricados ao dia e os resultados heurísticos apresentaram desvio médio de 3% quando comparadas com o B&B.

Karabatı e Sayın (2003) propuseram um estudo de *Flow Line* Misto que processa tarefas em ciclos com transferências simultâneas entre as estações. No estudo, Karabatı e Sayın (2003) desenvolveram uma heurística simples para a solução do problema. O FLM, conforme mostra os autores, permite a manufatura de diferentes tipos de produtos na mesma linha de produção e no problema de Karabatı e Sayın (2003), a busca pela minimização do tempo de ciclo através de uma formulação de sequência fixa é considerado. Como o problema mostrou-se difícil de encontrar solução, os autores desenvolveram um procedimento heurístico baseado nas regras de prioridades por meio do peso de tarefas.

Karabatı e Sayın (2003) testaram instâncias com 6 e 8 tarefas e o tamanho dos conjuntos eram gerados aleatoriamente para cada grupo de tarefas na instância. O tempo médio de solução foi de 0,13 segundos para instâncias com 6 tarefas e 1,07 segundos para 8 tarefas. Por fim eles apresentaram resultados de melhora no tempo de processamento do método heurístico nas instâncias de difícil solução em cerca de 20% melhor que o exato.

A investigação do RS para solucionar o *Flow Line* Misto foi investigada por Cho *et al.* (2005) como meios de determinar a melhor parametrização do algoritmo em questão a fim de otimizar as soluções das instâncias de grande porte. Os autores buscavam minimizar a taxa de uso da linha de produção e, consequentemente, a redução do tempo de *Setup* usado

nas máquinas antes do processamento. O trabalho se deu por experimentos computacionais em instâncias caracterizadas sob três níveis de programação, sendo eles com tamanho de 20, 100 e 500 tarefas. Os autores concluíram que o RS foi bastante eficaz para solucionar o problema do FLM por meios dos parâmetros testados. Cho *et al.* (2005) também mostraram que ainda é necessário executar outros procedimentos experimentais para garantir a efetividade do estudo quando considerado certos tamanhos de instâncias.

O estudo de uma revisão da literatura do FLM com diferentes abordagens e variações de layout foi apresentado por Copaceanu (2006). O principal objetivo foi encontrar novas aplicações e oportunidades de desenvolvimento do ambiente FLM por meio de avanços da tecnologia. Copaceanu (2006) dividiu o trabalho em dois blocos, o primeiro abordou o sequenciamento de tarefas que apresenta um sistema operante por meio de linhas em U e máquinas paralelas. A segunda, apresentou o problema por meio de processos estocásticos na indústria.

Copaceanu (2006) apresentou a importância da performance dos métodos nos trabalhos referenciados e os métodos heurísticos se mostraram bons para resolução dos procedimentos práticos em quase todos os trabalhos que consideram grandes instâncias próximas do real. O Autor ainda apresentou a variabilidade de procedimentos estocásticos onde é possível a análise de máquinas quebradas e adição de demanda continua nas linhas de produção, porém, os trabalhos estudados são, na maioria, de forma determinísticas. Finalmente, mostrou-se a importância deste modelo na indústria moderna, cujo ambiente tem apresentado grande interesse ao público mercantil dada sua flexibilidade e autonomia em encontrar saídas práticas para as fabricas ao customizar diferentes produtos sem altos custos.

Um problema de *Flow Line* misto caracterizado em ciclos com a fabricação de produtos com customização foi proposto por Eliiyi e Özlen (2008). O objetivo dos autores foi obter o *Makespan* dos ciclos total de processamento por meio de uma heurística Lagrangeana. O Problema é composto por N tarefas agrupadas em R grupos de famílias que tem por objetivo serem processados em estações síncronas para as transferências entre os ciclos. No trabalho, Eliiyi e Özlen (2008) caracterizou o FLM como NP-difícil, porém ainda há necessidade de estudos que compõem sua complexidade.

Eliiyi e Özlen (2008) propuseram uma Relaxação Lagrangeana capaz de quebrar as restrições de difícil solução do modelo como forma de melhorar o tempo computacional gasto no processamento de instâncias de grande porte e, então, utilizaram o solver CPLEX 8.1.1 como coadjuvante do LR. Foram criadas instâncias com tarefas iguais a 10, 15, 20, 30, 60 e o tempo de processamento foi gerado aleatoriamente com valores entre 1 e 20 unidades.

Nos resultados, Eliiyi e Özlen (2008) conseguiram resolver na sua otimalidade 17 das 54 instâncias geradas sendo que para instâncias com mais de 20 tarefas o tempo de processamento foi de mais de 50.000 segundos. Para instâncias com mais de 30 tarefas os autores utilizaram apenas o LR para encontrar os limitantes da solução e então aplicaram algoritmo

heurístico para buscar pelos resultados. Por fim, Eliiyi e Özlen (2008) mostraram que o LR pode obter boas soluções em um baixo tempo computacional e sua variação média de resultados chegou a 5,7% em relação ao resultado ótimo buscado. Os autores apresentaram também uma proposta de desenvolvimento de uma pesquisa baseada no B&B para melhorar os limitantes do LR buscando a redução do tempo computacional no processamento e até a otimalidade de instâncias maiores.

Com o objetivo de encontrar as necessidades das preferências dos consumidores, Hashem e Aryanezhad (2009) estudaram um sistema de produção real em grande escala e suas propostas consistiram em analisar um problema capaz de atender a demanda da indústria, buscando a instalação em sub-rotinas capazes de desafogar a principal linha do FLM. O objetivo desse estudo foi manter uma taxa constante de alimentação de produtos na linha de produção, minimizar o tempo de quebra de produção nas linhas e a minimização do tempo de preparação entre as estações durante a troca de material para produção.

Nos experimentos, Hashem e Aryanezhad (2009) utilizaram o algoritmo genético híbrido para executar os testes computacionais do problema. Os resultados se mostraram eficientes em 2,4%, em relação ao resultado ótimo conhecido na literatura das instâncias utilizadas. Os autores salientaram que uma investigação ainda é necessária para entender melhor as pesquisas que usam meta-heurísticas para solucionar problemas no ambiente que eles abordaram.

Shao *et al.* (2010) propuseram o estudo de um sistema de produção composto por uma linha de fabricação mista com carregamento intermediário de produtos e eles buscavam dois objetivos: Minimização do inventário de reposição e minimização do *Makespan*. Por esse problema ser considerado NP-difícil, os autores não consideraram um método exato para resolução das instâncias, e então apresentaram um algoritmo genético multi-objetivo com uma análise de melhora em relação a outros dois métodos analisados por meio da regra de Pareto.

Nos experimentos, Shao *et al.* (2010) desenvolveram instâncias com até 15 estações de trabalhos, capacidade máxima do inventário de até 200 peças e tarefas com 2 e 6 unidades para o plano de demanda. O Algoritmo genético conseguiu obter bons resultados com tempos de processamento de até 2,894 segundos em instâncias com até 6 tarefas. A regra do Pareto mostrou que o método funcionou bem quase sempre todas as instâncias escapando dos ótimos locais indo sempre em direção de bons resultados.

Shao *et al.* (2010) enfatizaram que ainda é importante realizar estudos para o Algoritmo Genético multi-objetivo e também considerar o estudo de outros métodos multi-objetivo como enxame de partículas e recozimento simulado, pois a problema é importante na indústria onde o tomador de decisões necessita otimizar e reduzir diferentes processos dentro da linha de produção.

Lin e Chu (2014) apresentaram um problema FLM real baseado em uma fábrica localizada em Taiwan cujo objetivo era minimizar o custo de fabricação atendendo a demanda dos consumidores. Os autores apresentaram a complexidade do problema e, então, foi proposto a heurística do Lagrangeano Relaxado para minimizar o custo computacional nos testes realizados nas instâncias. Os autores compararam os resultados com programas computacionais disponíveis no mercado para resolver modelos de PIM. Nos experimentos foi criado um estudo de caso da fábrica em questão e a base de dados coletada no estudo ficou em 38,439 restrições com cerca de 78,737 variáveis de decisão. O solver comercial CPLEX levou cerca de 31,462 segundos para resolver a base de caso e o LR foi 15 vezes mais rápido em termos computacionais para apresentar uma solução para o mesmo estudo de caso.

Lin e Chu (2014) apresentaram a importância da heurística LR para a solução de instâncias de grande porte, pois o método é capaz de relaxar restrições rígidas, garantindo uma melhor eficiência.

Fleszar (2017) considerou um problema de programação linear inteira mista para a busca da minimização do tempo de ciclo de processamento em estações de manufatura com múltiplas máquinas em cada estação caracterizado como FLM. Os autores afirmaram que no estudo apresentado, o problema apresenta um modelo da programação linear mista mais simples do que os demais problemas da literatura.

Ao realizar os experimentos, Fleszar (2017) testou 1200 instâncias com variados parâmetros na quantidade de estações, números de máquinas e de tarefas. Os resultados obtidos por meio do CPLEX 12.2 foram comparados com heurísticas híbridas de outros estudos anteriores. Entre as 1200 instâncias, cerca de 81% delas foi possível encontrar a solução ótima em trabalhos anteriores, e o autor conseguiu obter solução ótima para todas elas, exceto duas. O autor ressaltou também que o *Gap* médio das instâncias analisadas ficou entre 0 e 21,30 % quando comparadas com o melhor *Lower Bound* conhecido da instância.

Avaliando os trabalhos realizados do FLM, é possível observar que se trata de um ambiente difundido na indústria, porém ainda há muito o que ser feito. O uso de heurísticas para solucionar os modelos propostos são, na maioria das vezes, o melhor método de solução. Os métodos exatos são esporadicamente estudados e quando estudados, são, quase todos utilizados por meio do Lagrangeano Relaxado. A adição de restrições é um importante fator dentro dos casos do FLM, pois tais restrições torna o ambiente similar aos problemas enfrentados no dia-a-dia da indústria, e é interessante agregar em um único ambiente, o maior número de restrições que aproximam a atualidade.

Capítulo 3

Problema do *Flow Line* Misto com conjunto de Tarefas

Este capítulo apresenta o modelo de programação inteira mista para o ambiente FLM. É apresentada a formulação para obtenção do *Makespan* com restrições do FLM. Também são apresentadas todas as restrições de processamento para um melhor entendimento do problema.

3.1 O Ambiente *Flow Line* Misto

Quando se fala de processamento em linhas de produção com customização de produtos surgem questões de como isto pode ser feito em um tempo favorável para as indústrias, sem perdas de processamento ou de recursos. Por meio deste questionamento surgem as linhas de processamento de produtos criadas no início do século XX, como é a linha de montagem de veículos produzida por Henry Ford em seu clássico método de fabricação do Modelo Ford T (JONG, 2009). No entanto, em sua primeira linha de montagem, a produção era engessada e produzia apenas um único tipo de veículo sendo esta linha chamada de *Single-model Assembly Line* (linha de montagem de modelo único). Por meio do *Single-model*, a demanda das linhas de montagem foram requerendo diferentes especificações através dos tempos, onde-se teve a necessidade de desenvolver métodos capazes de satisfazer tais especificações que já poderiam ser chamadas de customização de produto.

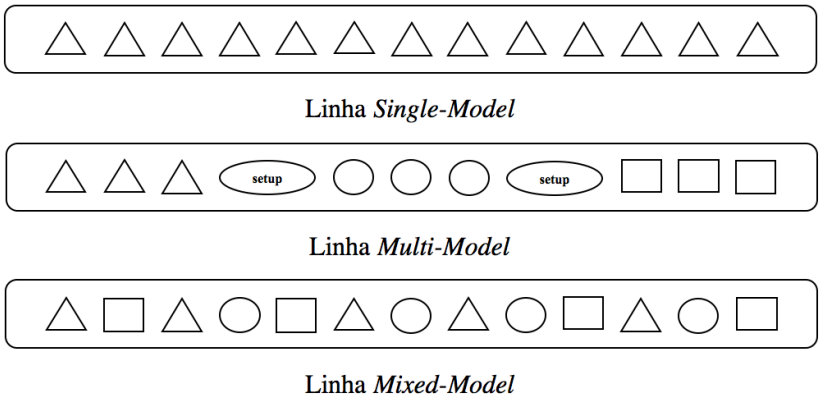
Com o surgimento de novas ideias de fabricação através do aprimoramento da linha de produção *Single-model*, surgiu um projeto de linha de montagem bastante similar à linha de fabricação de um único produto. Com este novo modelo de linha de montagem, chamado de *Multi-model Assembly Line*, era possível fabricar e produzir em lotes. Cada lote poderia ter um tamanho particular de produção de acordo com a demanda e a projeção de venda de cada produto fabricado (JONG, 2009). Porém, esse ambiente de produção tinha uma grande desvantagem ao serem confeccionados produtos em lotes, pois ela demandava um

extenso e pesado inventário de matéria-prima para fabricação e, posteriormente, espaço para estocagem dos lotes fabricados.

Thomopoulos (1967) definiu que *Mixed-Model Assembly Line* é a prática de fabricar ou produzir uma gama de produtos distintos na mesma linha de montagem, sem distinguir ou alterar a sequência de fabricação dado a caracterização específica do modelo, tornando, assim a linha de montagem mais amigável e de fácil manuseio durante o processo. Esse modelo de montagem surgiu através da evolução do *Single-model Assembly Line*, passando pela linha *Multi-model Assembly Line*, até chegar no *Mixed-Model Assembly Line*. Hoje, esse modelo é vastamente usado em inúmeras linhas de montagem para a fabricação de produtos em diferentes categorias de indústrias (QUADT; KUHN, 2007).

A Figura 3.1 mostra as características e variâncias do modelo misto de linhas de produção comentados.

Figura 3.1 – Diferença entre *Single-model*, *Multi-model* e *Mixed-Model Assembly Line*



Fonte: Adaptado de Jong (2009).

Ao longo da linha na Figura 3.1 pode-se perceber os diferentes tipos de produtos pelas diferentes formas geométricas além dos tempos de preparação (*Setup*) entre diferentes Lotes. Sistemas de produção em massa podem proporcionar aos clientes um *feedback* satisfatório em relação à variedade de customização e variedade de cada produto, sem alterar tempo do processo de fabricação, viabilizando melhores preços. Tal modelo se torna, então, atrativo aos olhos dos empresários que sempre buscam aumentar os lucros. Para que o *Mixed-Model* funcionasse, foi preciso que as empresas primeiro compreendessem o que os usuários dos produtos queriam, e então, criar as personalizações padrões dos produtos. Estas personalizações são agrupadas em famílias ou grupo de produtos fazendo com que cada família ou grupo tenha seu custo de produção e a soma do custo de todas as famílias não seja diferente do custo da produção daquele único produto sem personalização.

3.2 Definição do Problema *Flow Line* Misto

O problema de *Flow Line* Misto é definido em um ambiente configurado com M estações contendo uma ou mais máquinas disponíveis para processarem N tarefas, uma vez que cada tarefa é associada a um conjunto de produtos R de tamanho D_r (número máximo de tarefas por conjunto). Assim, existem R conjuntos que resultam N tarefas $\{n_1, n_2, n_3...\}$ sendo que cada conjunto de tarefas contém um subgrupo para serem processadas em M estações. O tempo de processamento p_{rm} é o conjunto $r \in R$ na estação m e para cada estação existe um tempo diferente de processamento do conjunto. Neste problema é assumido que cada tarefa n tem o mesmo tempo de processamento dentro do conjunto R e que cada conjunto é processado em todas as estações. De forma sequencial, cada conjunto R deve ser totalmente processado na primeira estação e indo sequencialmente para a próxima, até que seja processado em todas as estações. C_j é o tempo de duração em cada ciclo e a determinação do ciclo consiste no maior tempo dentre o conjunto de produtos processado no ciclo.

Pode-se definir que o problema de *Flow Line* Misto para o sequenciamento de produção aqui tratado é definido como:

$$FF_m|fmls|C_{max} \tag{3.1}$$

Trata-se de um problema de programação mista derivado do *Flexible Flow Line* com tarefas agrupadas por conjuntos e o objetivo principal é a obtenção do *Makespan*. Matematicamente, o problema pode ser definido como:

Sejam as variáveis de decisão $X_{r,n}$, sendo 1 se a tarefa n do conjunto de produtos R ocupa a $n-ésima$ posição da programação, 0 senão.

$$\text{Min z} = \sum_{j=1}^{M+N-1} C_j \tag{3.2}$$

Sujeito à:

$$\sum_{r=1}^{R} X_{r,n} = 1, \ \forall \ n = 1,...,N \tag{3.3}$$

$$\sum_{n=1}^{N} X_{r,n} = D_r, \ \forall \ r = 1,...,R \tag{3.4}$$

$$C_j \geqslant \sum_{r=1}^{R} p_{r,k} X_{r,(j-k+1)}, \forall \ j,k \in JK = \left\{ \begin{array}{ll} j = 1,...,M-1, & k = 1,...,j \\ j = M,...,N, & k = 1,...,M \\ j = N+1,...,N+M-1, & k = j-M+1,...,M \end{array} \right\} \tag{3.5}$$

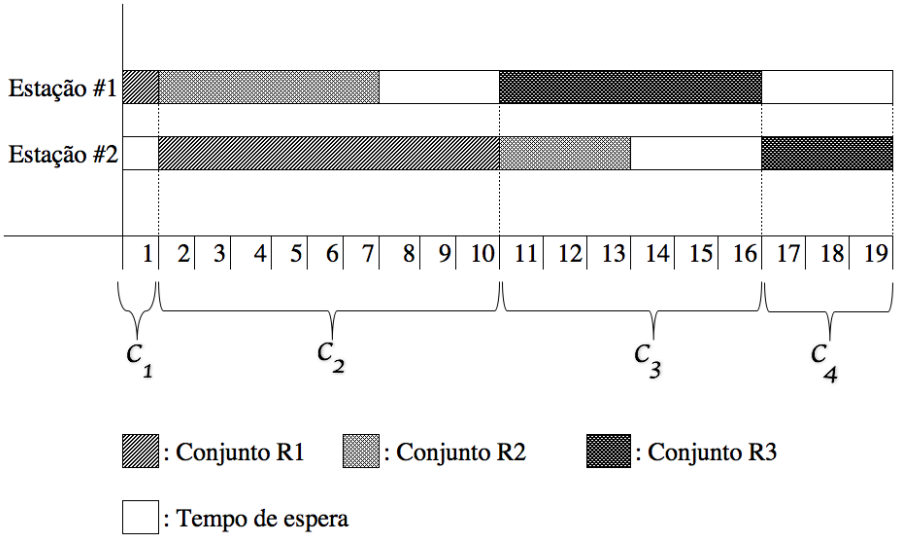
$$X_{r,n} \in 0,1, \ \forall \ n = 1,...,N, \ r = 1,...,R \tag{3.6}$$

A função objetivo (3.2) busca por minimizar o tempo de ciclo total do conjunto (*Makespan*). As restrições descritas em (3.3) garantem que em cada posição do sequenciamento na estação m é ocupada apenas por um único conjunto R associado a n tarefas. As restrições em (3.4) atribuem as tarefas $\{n_1, n_2, n_3...\}$ aos conjuntos $\{R_1, R_2, R_3, R_4...\}$ de acordo com a capacidade de demanda de tarefas D_r que cada conjunto de produtos pode ter, sendo que cada tarefa pertence a exatamente um único conjunto. As restrições descritas em (3.5) identificam o tempo máximo de cada ciclo. Note que p_{rm} é o tempo de processamento do produto $r \in R$ quando processado na estação m, e D_r é a demanda de tarefas possíveis dentro de cada conjunto r. C_j é o tempo entre $(j-1)$ e j no processamento de cada conjunto. A quantidade de ciclos possíveis no modelo é definida por $N+M-1$. O objetivo do problema é minimizar $C_{max} = \sum_{j=1}^{M+N-1} C_j$.

As máquinas do primeiro ciclo são ocupadas pelas primeiras tarefas e as demais máquinas do ciclo permanecem em espera até que a alocação utilize todas as máquinas durante o tempo de processamento das tarefas do produto $r \in R$. As tarefas devem seguir a fila para serem processadas e a transferência de tarefas entre as estações são permitidos quando finaliza seu ciclo na máquina atual. Como o tempo de preparação das máquinas é independente para cada tarefa, ele é, consequentemente, incluído no tempo de processamento.

O tempo de cada ciclo no FLM é definido pelo maior tempo de processamento p_{rm} do conjunto $r \in R$ na máquina m. A Figura 3.2 mostra o gráfico de Gantt com o processamento dos R_1, R_2, R_3 com *Makespan* igual a 19.

Figura 3.2 – Exemplo do Gráfico de Gantt para o *Flow Line* Misto



Fonte: o autor.

Note que apenas uma estação está em uso no primeiro ciclo processando o conjunto R_1, enquanto a segunda estação permanece em estado de espera até que o conjunto de tarefas R_1 seja processado. Ao finalizar o processamento no primeiro ciclo então o conjunto

de tarefas R_1 é iniciado no segundo ciclo e um novo conjunto de tarefas R_2 é carregado na primeira estação simultaneamente. Isto acontece até que todos os conjuntos de tarefas sejam processados em todos os ciclos de todas as estações. Para a obter do C_{max}, a soma de todos os ciclos é realizada considerando apenas o valor do conjunto R com maior tempo de processamento entre todos os conjuntos processados no ciclo.

O modelo FLM busca encontrar o melhor sequenciamento do conjunto de produtos como forma de minimização do tempo total de processamento nas estações. O problema FLM assume que não há a possibilidade de máquinas indisponíveis para manutenção ou máquinas quebradas durante o processamento. Todos os conjuntos de tarefas estão disponíveis para serem processados a qualquer momento a partir do momento 0, quando se inicia as estações. A transferência dos conjuntos de tarefas entre as estações é síncrona, ou seja, elas ocorrem de forma sucessiva e simultânea com todos os conjuntos disponíveis em processamento. Este tipo de transferência acontece em sistemas automatizados de manufatura no qual braços robóticos ou esteiras de transporte transferem produtos de uma estação para a outra.

3.3 *Flow Line* Misto com o Lagrangeano Relaxado

O problema do FLM com relaxação Lagrangeana foi proposto por Eliiyi e Özlen (2008) como forma de relaxar restrições do problema de difícil resolução buscando um tempo computacional aceitável durante o processamento. O procedimento do LR consiste em relaxar uma ou mais restrições de um problema de forma em que é possível encontrar limitantes de soluções para o problema, melhorando o esforço computacional na solução do modelo matemático relaxado.

3.3.1 Definição do Lagrangeano Relaxado

A relaxação Lagrangeana começou a aparecer em artigos como uma ideia de grande usabilidade na década de 70, descrevendo os procedimentos básicos para descrever uma relaxação Lagrangeana, baseando-se em um problema de otimização combinatória inteira. A dualização que ocorre na relaxação Lagrangeana ocorre de forma que uma das restrições é acrescida à função objetivo por meio de um multiplicador de Lagrange, gerando um problema Lagrangeano de mais fácil solução (FISHER, 1981). O valor ótimo encontrando no Lagrangeano relaxado é um limitante inferior na busca do valor ótimo do problema original quanto este é de minimização. Com base nisso, a obtenção de limitante inferior do problema original é capaz de reduzir o campo de busca pela melhor solução reduzindo, assim, o tempo de processamento.

A Relaxação Lagrangeana consiste basicamente em quebrar restrições de difícil solução do problema de otimização combinatória inteira para melhor o tempo de busca pelo

mínimo do problema. Seja o modelo descrito em (3.7-3.10), de forma que:

$$\text{Z=Min} f(x) \tag{3.7}$$

Sujeito à:

$$g(x) = b \tag{3.8}$$

$$h(x) \leqslant e \tag{3.9}$$

$$x \geqslant 0 \tag{3.10}$$

em que x é uma matriz ou vetor, e é um vetor $k \times 1$, b é um vetor $b \times 1$. Fisher (1981) mostrou a construção básica da relaxação fazendo a restrição (3.8) do problema ser integralmente relaxada como parte da função objetivo. Para cada restrição em (3.8), existe um $\lambda \geq 0$ associado a restrição, chamado de multiplicador de Lagrange. Então, o problema relaxado primal é expresso na forma (3.11-3.13):

$$Min\ F(x,\lambda) = f(x) + \lambda(g(x) - b) \tag{3.11}$$

Sujeito à:

$$h(x) \leqslant e \tag{3.12}$$

$$x \geqslant 0 \tag{3.13}$$

Desta forma, o problema primal relaxado pode executar a busca do limitante inferior que satisfaça $\lambda \in R_+^m$ e seja capaz de computar (3.14)

$$L(\lambda) = \min_{x \in e}\ F(x,\lambda) \tag{3.14}$$

Assim, tem-se a definição do problema dual para a relaxação das restrições do problema conforme as equações (3.15) e (3.16) (FISHER, 1981):

$$\max_{\lambda \in R_+^m}\ L(\lambda) \tag{3.15}$$

$$\max_{\lambda \in R_+^m} \min_{x \in e}\ F(x,\lambda) \tag{3.16}$$

Vale ressaltar que a parte dual da relaxação Lagrangeana tem um papel importante por ser a parte do problema relaxado que determina o limitante inferior. Para $\lambda \in R_+^m$ tem-se $L(\lambda) \leqslant f(x)$. Para a atualização do λ durante a busca pelo limitante inferior, usa-se o método do subgradiente para a restrição relaxada como forma de ir a caminho da direção de $g(x)$ sem ocorrer violações.

3.3.2 Relaxação Lagrangeana do *Flow Line* Misto

A Relaxação Lagrangeana do FLM, ocorre relaxando a restrição da equação (3.5) do modelo. Note que esta restrição define o tempo de ciclo de cada transferência de conjunto entre as estações durante o processamento dos conjuntos de tarefas. O modelo relaxado de Eliiyi e Özlen (2008) pode ser expresso (3.17-3.21):

$$\text{Min LR(z)} = \sum_{j=1}^{M+N-1} C_j - \sum_{j,k\in JK} \lambda_{jk} C_j + \sum_{j,k\in JK} \lambda_{jk} \sum_{r=1}^{R} p_{r,k} X_{r,(j-k+1)} \tag{3.17}$$

Sujeito à:

$$\sum_{r=1}^{R} X_{r,n} = 1, \ \forall n = 1,...,N \tag{3.18}$$

$$\sum_{n=1}^{N} X_{r,n} = D_r, \ \forall r = 1,...,R \tag{3.19}$$

$$X_{r,n} \in 0,1, \ \forall \ n = 1,...,N, \ r = 1,...,R \tag{3.20}$$

$$\lambda_{j,k} \geqslant 0 \ \forall \ j,k \in JK \tag{3.21}$$

Por meio deste problema, tem-se os valores de $\lambda_{j,k}$ para a busca da solução. Após escrever a relaxação Lagrangeana do problema original, Eliiyi e Özlen (2008) subdividiu o problema em dois para reduzir o custo computacional. O primeiro problema, LR_1, representa um problema de alocação de tarefas e o LR_2 representa um problema de alocação cíclica.

O LR_1, o modelo da relaxação Lagrangeana é definido em (3.22-3.25):

$$\text{Min } LR_1\text{(z)} = \sum_{j,k\in JK} \lambda_{jk} \sum_{r=1}^{R} p_{r,k} X_{r,(j-k+1)} \tag{3.22}$$

Sujeito a:

$$\sum_{r=1}^{R} X_{r,n} = 1, \ \forall n = 1,...,N \tag{3.23}$$

$$\sum_{n=1}^{N} X_{r,n} = D_r, \ \forall r = 1,...,R \tag{3.24}$$

$$0 \leqslant X_{r,n} \leqslant 1,\ \forall n = 1,...,N,\ r = 1,...,R \tag{3.25}$$

Nas restrições (3.25), observe que o problema deixa de ser binário, assim, melhorando a performance durante a resolução do problema de alocação. Conforme dito, o LR_2, é um problema cíclico que determina os limitantes superior e inferior de cada tempo de ciclo C_j. Ele pode ser definido por (3.26-3.28):

$$\text{Min } LR_2(z) = \sum_{j=1}^{M+N-1} C_j - \sum_{j,k \in JK} \lambda_{jk} C_j \tag{3.26}$$

Sujeito à:

$$C_j \leqslant \max_k\{\max_r\{p_{r,k}\}\},\ r = 1,...,R,\ \forall\ j,k \in JK = \left\{ \begin{array}{ll} j = 1,...,M-1, & k = 1,...,j \\ j = M,...,N, & k = 1,...,M \\ j = N+1,...,N+M-1, & k = j-M+1,...,M \end{array} \right\}. \tag{3.27}$$

$$C_j \geqslant \max_k\{\min_r\{p_{r,k}\}\},\ r = 1,...,R,\ \forall\ j,k \in JK = \left\{ \begin{array}{ll} j = 1,...,M-1, & k = 1,...,j \\ j = M,...,N, & k = 1,...,M \\ j = N+1,...,N+M-1, & k = j-M+1,...,M \end{array} \right\}. \tag{3.28}$$

Observe que as restrições (3.27) e (3.28) foram adicionadas como conjuntos de desigualdades para mostrar que o tempo de ciclo é dependente do tempo de processamento do conjunto R na estação M. Assim, as restrições são os limitantes do valor máximo definido por (3.27) e valor mínimo definido por (3.28) que cada ciclo pode assumir. Eliiyi e Özlen (2008) ainda mostra que o limitante superior do LR_2 e tempo máximo que as estações podem assumir para aquele ciclo.

O valor da função objetivo do problema relaxado é o limitante inferior que o problema original pode assumir na busca do resultado ótimo do problema, sendo que as somas dos dois subproblemas compõem $LR(z) = LR_1(z) + LR_2(z)$. Após conhecer a solução $X_{r,n}$ de $LR(z)$, é possível, por meio da restrição (3.5) do problema original, calcular o limitante superior.

3.3.3 Procedimento Lagrangeano Relaxado

No Lagrangeano Relaxado, o principal objetivo é otimizar o problema reduzindo o custo computacional. O método pode ser usado como um coadjuvante por meios de outros métodos de otimização, como o *Branch & Bound*. No caso aqui apresentado, o LR é usado com o GUROBI e depois associado a uma heurística para a obtenção do limitante superior.

Ainda é bem pouco compreendido os fatores que garantem a convergência do problema, e diferentes métodos são encontrados para definir o valor dos multiplicadores associados a restrição relaxada (BALAS; CARREIRA, 1996).

Pensando no contexto de buscar os limitantes superior e inferior no problema FLM com a relaxação Lagrangeana, o procedimento mostrado no Algoritmo 3.1 foi desenvolvido por Eliiyi e Özlen (2008) e então implementado para o FLM para utilização nesse estudo. O procedimento retorna o limitante inferior do problema, enquanto a heurística Lagrangeano Relaxado mostrado no Algoritmo 3.2 retorna resultados do limitante superior resolvendo as restrições (3.5).

Algoritmo 3.1: Procedimento Lagrangeano Relaxado

1 Inicialize $t = 1$;
2 $\lambda_{jk}^{t} = \frac{1-0.01}{M}$;
3 $\alpha^{t} = 0.5$;
4 **repita**
5 Resolva $LR_1(z)$ e $LR_2(z)$ com os multiplicadores de lagrange λ_{jk}^{t};
6 Encontre as variáveis ótimas de $X_{r,n}^{t}$ e c_j^{t};
7 Faça $LB^{t} = LR1^{t}(z) + LR2^{t}(z) = LR^{t}(z)$;
8 Encontre o UB^{t} utilizando a heurística Lagrangeano Relaxado;
9 Faça $UB = min\{UB, UB^{t}\}$;
10 **se** $\lceil LB^{t} \rceil = UB$ **então**
11 Solução ótima encontrada;
12 Pare;
13 **senão**
14 Determine o passo θ^{t};
15 Atualize λ_{jk}^{t+1};
16 **se** *LB não melhorar até* $NumIt$ **então**
17 $\phi^{t+1} = \phi^{t} * multiplicador$;
18 $t = t + 1$
19 **até** $t > Maxit$;
20 **se** $t = Maxit$ **então**
21 Resolva o problema original acrescentando restrições de limitante inferior e superior;
22 Pare;

Na linha 8 do Algoritmo 3.1, a determinação do passo é definida pelo subgradiente (FISHER, 1981) da restrição relaxada em (3.29) e (3.30):

$$\theta^{t} = \frac{\phi^{t}(UB - LB^{t})}{\sum_{j,k \in JK}(\sum_{r=1}^{R} p_{r,k} X_{r,(j-k+1)}^{t} - C_j^{t})^2} \tag{3.29}$$

$$\lambda_{jk}^{t+1} = \max\left\{0, \lambda_{jk}^{t} + \theta^{t}(\sum_{r=1}^{R} p_{r,k} X_{r,(j-k+1)}^{t} - C_j^{t})\right\} \tag{3.30}$$

A Heurística Lagrangeano Relaxado é importante para calcular o limitante superior do problema. Observe que as restrições (3.5) determinam o tempo de ciclo de cada conjunto na estação. Conhecendo os valores ótimos de $X_{r,n}$, é possível computar os valores de C_j em cada ciclo por meio de (3.2) no Algoritmo 3.1.

Algoritmo 3.2: Heurística Lagrangeano Relaxado

1 Encontre o valor C_j para cada ciclo usando as restrições (3.5) com a solução de $LR_1(z)$;
2 Calcule o valor do UB por meio da função objetivo (3.2) usando os valores encontrados de C_j;

Capítulo 4

Algoritmo Genético

Este capítulo apresenta o Algoritmo Genético (AG) que foi utilizado para resolver o FLM, com a função da penalidade exterior, sendo usada para inserir o modelo de programação inteira mista e, assim, avaliar e busca de soluções do FLM.

4.1 Algoritmo Genético

O AG é uma heurística bio-inspirada que se fundamenta em um mecanismo da teoria da seleção natural por meio da genética de população. Utilizando a teoria de seleção natural idealizada por Darwin, foi possível imitar seus princípios básicos de genética populacional e viabilidade de indivíduos. De acordo com Lobato (2008), foi John Holland que, em 1975, desenvolveu as primeiras pesquisas envolvendo princípios genéticos como imitação para procedimentos computacionais.

O AG é considerado o primeiro algoritmo evolutivo que não utiliza informações do gradiente da função objetivo e das restrições para atualizar o caminho de convergência da solução do problema e, consequentemente, ele é capaz de escapar dos ótimos locais das funções, uma vez que métodos que utilizam gradiente raramente conseguem escapar dos ótimos locais das funções (LOBATO, 2008). Basicamente, o AG gera candidatos à solução da função de forma aleatória, explorando os candidatos em contorno da função para encontrar novos candidatos à solução do problema, sendo eles cada vez mais próximos dos resultados globais do problema. Portanto, o AG sempre impõe limites no campo de geração dos candidatos para que as soluções no domínio das variáveis de projeto não violem as restrições do problema.

O Algoritmo Genético é muito utilizado nos estudos heurísticos atualmente, e ele pode ser aplicado em diferentes áreas de pesquisa, incluindo os estudos da PP. Yu e Seif (2016) utilizaram o método para minimizar o tempo de atraso com manutenção de máquinas, Palaniappan e Jawahar (2011) utilizaram o AG para otimizar produção em lotes de uma

linha de montagem em *Flow Line*. O AG também foi utilizado por Costantino *et al.* (2014) para otimizar o tempo de fluxo de uma linha *Flow Line* Misto na fabricação de motocicletas, bastante similar ao problema presente nesta dissertação.

O funcionamento do AG geralmente se dá pela manutenção e efetivação de candidatos em forma de população que são representados com os cromossomos cuja aptidão é avaliada dentro da função objetivo. Cada cromossomo é composto de uma sequência de elementos capazes de chegar a uma possível solução para o problema.

Como qualquer outro tipo de algoritmo bio-inspirado, o AG tem um conjunto de parâmetros cujos valores afetam o desempenho e a qualidade das soluções obtidas. Este conjunto de parâmetros inclui o tamanho da população ($nPop$), que é o número de cromossomos a cada iteração. O percentual probabilístico de cruzamento dos cromossomos (p_c), o percentual probabilístico de mutação dos cromossomos (p_m), e a condição de parada ($maxIt$), que é o número de iterações transcorridas até que o algoritmo pare. A capacidade do método em encontrar bons resultados está conectada aos parâmetros p_c e p_m e $nPop$, então, estes parâmetros afetam os candidatos e as soluções da função (SARAMAGO, 1999). O Algoritmo 4.1 representa o funcionamento do Algoritmo Genético com os passos de execução de cada etapa.

Algoritmo 4.1: Estrutura do Algoritmo Genético

Entrada: Arquivos de leitura, parâmetros $nPop$, p_m, p_c, $maxIt$
1 Leia os arquivos de instância;
2 Aleatoriamente gere a população Inicial;
3 **repita**
4 Aplique o operador de reprodução;
5 Aplique os operadores de cruzamento e mutação;
6 Calcule a função objetivo com a população atual;
7 **se** *a função satisfez as restrições sem violações* **então**
  Saída: Solução da função objetivo do problema
8 Pare;
9 **até** $t > MaxIt$;
Saída: Solução da função objetivo após t atingir $MaxIt$

O AG usa de uma nomenclatura que é referida na literatura como forma de se relacionar com a linguagem natural de seleção proposta por Darwin. Através dessa analogia é possível compreender melhor o funcionamento do AG. A Tabela 4.1 apresenta essa analogia entre a linguagem de seleção natural e o AG.

Conforme descreve, a solução é composta por cromossomos que compõem um vetor de tamanho n com as variáveis da função objetivo a serem avaliadas no AG. Cada gene do cromossomo designa uma variável candidata da função do problema e os genes dos cromossomos são expressos de forma binária. Nos dias atuais, existem versões que utilizam conversores de binário para real, mas para o problema estudado neste trabalho, a versão

Linguagem Natural	Algoritimo Genético
Cromossomo	Variáveis da Função
Gene	Unidade de cromossomo
Alelo	Valor do Gene
População	Candidatos Cromossomos
Geração	Iteração
Aptidão	Função Objetivo

Fonte: Adaptado de Lobato (2008)

Quadro 4.1 – Linguagem Natural X Algoritmo Genético

binária é a mais adequada.

O AG é um método iterativo que durante sua execução, a população é modificada (reproduzida) de acordo com sua mutação e cruzamento, tornando possível que o operador de reprodução copie a cadeia de cromossomos e reproduza-a levando em consideração o valor da função objetivo atual. Já o operador de cruzamento é responsável por combinar as partes de dois cromossomos para gerar um novo descendente e, por fim, o operador de mutação é capaz de modificar de forma aleatória e ocasional, o valor dos alelos da cadeia em busca de inserir diversidade na população (SARAMAGO, 1999). Em linhas gerais, pode-se dizer que o AG trabalha com três operadores básicos: **Reprodução** ou **Seleção**, **Cruzamento** e **Mutação**.

Seguindo a mesma linha de raciocínio dos algoritmos evolutivos, o AG também inicia seus candidatos iniciais da população aleatoriamente seguindo a delimitação do domínio especificado. Para que isso aconteça, deve-se ter definido o tamanho da população e também o número de variáveis da função objetivo e o respectivo domínio. A seguir, todos os indivíduos da população são replicados e reproduzidos de acordo com os operadores básicos.

4.1.1 Seleção

No operador de reprodução, o candidato/cromossomo que apresentar a maior aptidão, ou seja, aquele que tem mais chances de sobreviver, tem maiores chances de contribuir com a geração seguinte com pelo menos um gene apto bom. Desta forma, o operador copia as melhores soluções representadas pelos candidatos/cromossomos sobreviverem na reprodução, contribuindo com seus genes para a geração de cromossomos candidatos (SARAMAGO, 1999).

Para o operador de reprodução, a equação (4.1) é definida para que o candidato de baixa adaptabilidade tenha uma alta probabilidade de se extinguir da população. Dessa forma, os candidatos com alta adaptabilidade conseguem permanecer na população para

garantir uma boa reprodução dos cromossomos.

$$p_i = \frac{f(x_i)}{\sum_{j=1}^{n} f_j(x)}, i = 1, 2..., n \tag{4.1}$$

onde i representa a iteração a cada reprodução do algoritmo.

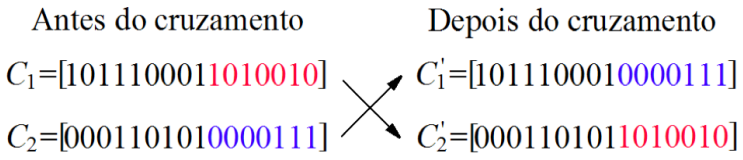
Vale ressaltar que os cromossomos de alta adaptabilidade podem ser selecionados mais de uma vez no decorrer das iterações e somente os melhores conseguem permanecer por mais tempo.

4.1.2 Cruzamento

O operador de cruzamento é simplesmente a troca de material genético entre os candidatos para modificar a população de cromossomos candidatos por meio de filhos. Existem diversas formas de promover o cruzamento dos cromossomos para a troca do material genético e, no algoritmo em questão, a troca é feita utilizando o fator de cruzamento p_c. A cadeia de genes no cromossomo é quebrada em dois pontos aleatórios e a troca de informações genéticas acontece entre os cromossomos i e j selecionados.

O processo de escolha pode ser feito em uniformidade de pares, com pontos de um único alelo e com pontos duplos por pares de alelos e, se não for possível formar pares nos genes do cromossomo, a escolha aleatória é realizada até que o cruzamento aconteça em busca de obter os pares necessários para um cruzamento efetivo. A Figura 4.1 mostra como funciona o processo de cruzamento dos cromossomos.

Figura 4.1 – Representação do Operador de Cruzamento



Fonte: Adaptado de Lobato (2008)

O valor da probabilidade do cruzamento p_c é definido manualmente e deve ser calibrada para seu uso de acordo com o comportamento do AG sobre a função de aptidão do problema a ser avaliado.

4.1.3 Mutação

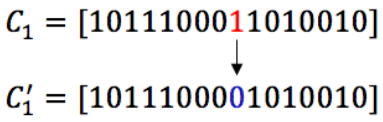
O operador de mutação tem a funcionalidade de manter a diversidade, ou seja, modificar o cromossomo através da troca de alelos dos genes na cadeia de cromossomos para diversificar a população. A troca acontece de forma que o alelo (0) passa a ser o alelo (1) ou vice-versa. A forma básica de funcionamento desse operador é selecionar um ou mais

alelos aleatórios em posições diferentes na cadeia dos cromossomos e então permutarem os mesmos alelos entre os cromossomos.

A aplicabilidade deste operador consiste em selecionar alelos baseando-se na probabilidade de mutação p_m para modificá-los. O fator de mutação p_m garante que durante a aplicação dos operadores de reprodução e mutação, o material genético com forte adaptabilidade não seja perdido no processo. Sendo assim, o operador de mutação protege o AG de perder bons cromossomos (SARAMAGO, 1999).

A Figura 4.2 mostra o funcionamento da mutação com a troca de um alelo no cromossomo com o objetivo de manter o material genético forte no AG a cada avaliação da função objetivo do problema.

Figura 4.2 – Representação do Operador de Mutação



Fonte: Adaptado de Saramago (1999)

4.1.4 Critérios de Parada

Pode-se dizer que existe algumas formas de finalizar um procedimento iterativo evolutivo como o AG. Os mais comuns, conforme aponta Lobato (2008) são:

- (1) Número máximo de Iterações;
- (2) Valor da função objetivo menor que o valor referenciado;
- (3) Erro absoluto ou erro relativo;
- (4) Tempo de processamento;
- (5) Intervenção do usuário.

Vale ressaltar que os itens 1 e 4 são os mais utilizados como critério de parada, porém, o critério depende do problema e da área de aplicação. Para o estudo do FLM, o critério utilizado no Algoritmo Genético foi o do item 1, uma vez que na análise dos resultados é dado ênfase na qualidade do resultado obtido por meio da avaliação da função objetivo sem violações de restrições.

4.1.5 Restrições de Igualdade e Desigualdade

O AG é um método evolutivo contido na literatura que não lida com problemas de otimização restrita e assim como inúmeros outros algoritmos são necessários adaptações para otimizar tais problemas. Usualmente, os problemas de otimização são construídos com restrições de igualdade e/ou de desigualdade. Para essas adaptações, como a do problema aqui exposto que contém restrições, pode-se usar o método da Penalidade Exterior, descrito em Vanderplaats (1999).

Basicamente, o método consiste em definir uma função chamada de Pseudo-Objetivo $\phi(x, r_p)$, que em definição, é o problema original com suas restrições associado à função objetivo, tornando o problema sem restrições, porém, com penalidade de violação de restrição. Esta função é definida por (4.2), conforme mostra Vanderplaats (1999):

$$\phi(x, r_p) = f(x) + r_p P(x) \tag{4.2}$$

onde $P(x)$ é a função de penalidade (as restrições), $f(x)$ é a função objetivo original e r_p é o parâmetro de penalidade para garantir a não violação das restrições. A função de penalidade $P(x)$ possui diferentes formas para restrições de igualdade e restrições de desigualdade, sendo definidas em (4.3-4.4):

$$P(x) = \sum_{j=1}^{m} (max[0, g_j(x)])^2 \tag{4.3}$$

$$P(x) = \sum_{k=1}^{l} (h_k(x))^2 \tag{4.4}$$

na qual a (4.3) é a penalidade para restrições de desigualdade $g_j(x) \geqslant 0$, e (4.4) é a penalidade para restrições de igualdade $h_k(x) = 0$. É interessante afirmar que $g(x)$ deve ser menor ou igual a zero para que a equação (4.3) seja aplicada e quando equações onde $g(x)$ for maior ou igual a zero, deve-se primeiro converter a equação para menor ou igual a zero.

Na equação (4.2), o parâmetro r_p é ponderado de forma que ele aumenta seu valor quando as restrições são violadas, assim como a potência pondera em $P(x)$ nas restrições $g(x)$ e $h(x)$. Desta forma, toda e qualquer restrição que for violada durante o procedimento reflete como uma penalização dentro da função pseudo-objetivo. O r_p tem um papel importante na convergência das funções, pois quando ele tende ao infinito, o valor numérico da função pesudo-objetivo tende ao valor da função objetivo original com as restrições satisfeitas (VANDERPLAATS, 1999).

Capítulo 5

Experimentos Computacionais

Este capítulo apresenta os resultados computacionais para a resolução das instâncias geradas aleatoriamente do problema *Flow Line* Misto através da proposta de Eliiyi e Özlen (2008). Os resultados foram obtidos por meio da resolução do modelo (3.2-3.6) no GUROBI, Lagrangeano relaxado e o Algoritmo Genético. Os resultados foram comparados entre cada técnica de resolução.

5.1 Características de Implementação

No estudo em busca dos resultados e da experimentação do problema, optou-se pela implementação do modelo (3.2-3.6) FLM no GUROBI e do Lagrangeano Relaxado no Algoritmo 3.1. Os dois métodos mencionados foram implementados na linguagem C++ e ambos usaram o *framework* de otimização do GUROBI na versão 7.5.2. A implementação do Algoritmo Genético foi extraída de Heris, S. M. K. (2015). Para que fosse possível o uso do algoritmo genético implementado, foi necessário adaptar o problema pelo método da penalidade exterior e, então, integrar a implementação do problema à implementação de Heris, S. M. K. (2015).

Para a geração das instâncias, foi gerou-se aleatoriamente diferentes problemas-teste de acordo com os tamanhos das instâncias de Eliiyi e Özlen (2008). As quantidades de tarefas foram $N = 10,\ 15,\ 20$ para o grupo de instâncias de pequeno porte e $N = 30,\ 60,\ 120$ para o grupo de instâncias de grande porte. O número de conjuntos R associados às tarefas segue a fórmula de $R = \lceil N/4 \rceil$, $\lceil N/2 \rceil$ e N conjuntos, criando uma combinação de diferentes conjuntos para cada quantidade máxima de tarefas. O número de estações foi fixado em $M = 3,\ 5,\ 8$. O tempo de processamento p_{rm} foi gerado através de uma distribuição uniforme entre 1 e 20 tempos de unidade e as demandas D_r de cada conjunto foram geradas aleatoriamente usando uma distribuição uniforme entre 1 e a quantidade de tarefas existentes.

5.1.1 Calibração dos Parâmetros do Lagrangeano Relaxado

Os testes computacionais do Lagrangeano Relaxado, do Algoritmo Genético e do GUROBI foram realizados em um computador com processador Intel ® Core i5-3570K de 3.40 GHz, 8 GB de memória RAM e sistema operacional Linux Ubuntu 16.04 LTS. Os parâmetros de calibração para o Lagrangeano Relaxado foram testados de forma que garantisse que o valor de $\lambda_{j,k}$ estabilizasse na busca dos limitantes sem violar a restrição relaxada. Eliiyi e Özlen (2008) definiram os parâmetros de forma que o procedimento tivesse um bom desempenho com baixo custo operacional e durante os testes de calibração da implementação desse estudo, foi possível utilizar quase todos os mesmos parâmetros utilizados na proposta de Eliiyi e Özlen (2008).

Na execução do Lagrangeano Relaxado, os valores de λ não afetam significavelmente no valor final do problema relaxado. Porém, a escolha do valor inicial desempenha um papel fundamental nos limitantes finais encontrados no método (FISHER, 1981). Na implementação, o número de iterações é definido conforme o número de tarefas de cada instância; instâncias de até 30 tarefas, o limite de iterações é de 400, em instâncias entre 31 e 60 tarefas, o limite de iterações é de 800, em instâncias de 61 até 120 tarefas, o limite de iterações é de 1200.

Na linha 16 do Algoritmo 3.1, no qual descreve a penalidade quando não ocorre a melhora do LB, o parâmetro $NumIt$ também depende do número de tarefas das instâncias, sendo o limite de 20, 30 e 40 para instâncias com tarefas entre 0-30, 31-60, 61-120 respectivamente. O valor de ϕ inicialmente foi definido com $0,5$ e o valor do $atualizador$ de penalidade por violação foi definido em $0,75$. Diferente de Eliiyi e Özlen (2008), o desempenho do procedimento em todas as instâncias funcionou melhor com o atualizador definido em $0,75$ e não $0,5$. O valor do multiplicador de Lagrange foi definido em $\lambda_{j,k} = (1-0,01)/M$ para todo j e k, um valor pequeno dependente do número de estações de cada instância, para garantir que seu valor inicial não ficasse fora da região de busca.

Os critérios de parada do Lagrangeano são dois: Se a próxima parte inteira do limitante inferior $\lceil LB \rceil$ for igual ao limitante superior (UB), então existe uma solução encontrada e o método para, caso contrário, após o método atingir o número máximo de iterações, o problema é, então, resolvido pelo GUROBI com a adição de duas restrições (5.1-5.2):

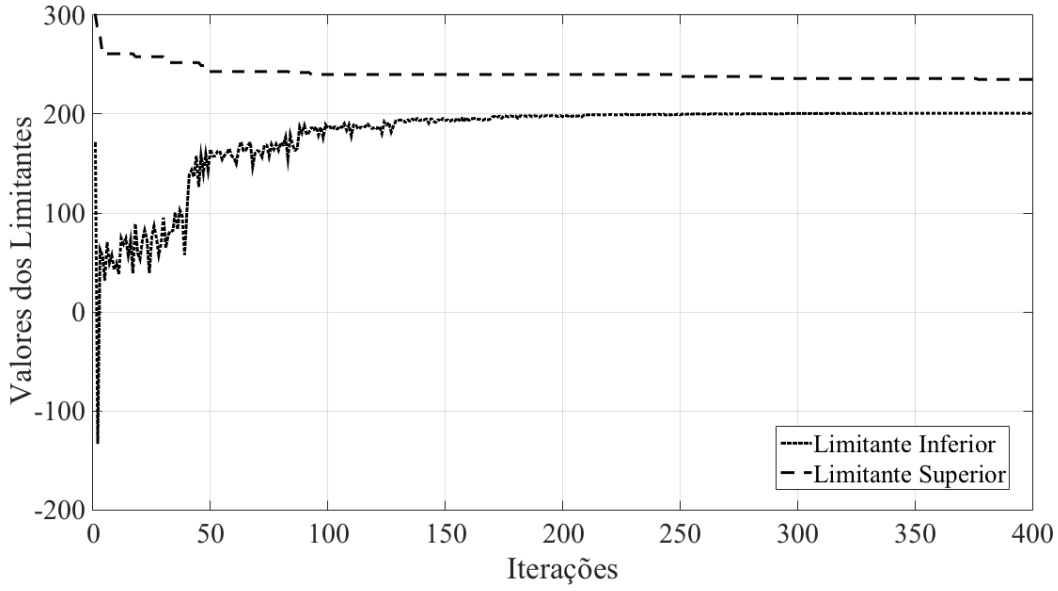
$$\sum_{j=1}^{M+N-1} C_j \geqslant LB \tag{5.1}$$

$$\sum_{j=1}^{M+N-1} C_j \leqslant UB \tag{5.2}$$

As restrições (5.1) e (5.2) são os limitantes encontrados no LR capazes de delimitar o campo de busca do problema reduzindo para apenas o intervalo indicado. As Figuras 5.1

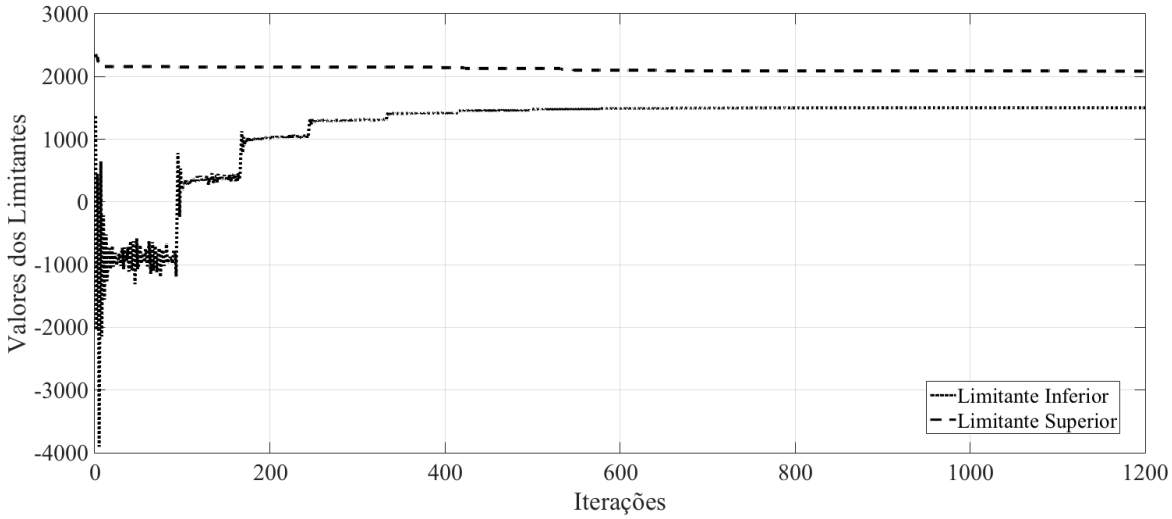
e 5.2 mostram o comportamento do LR na redução do campo de busca por meio dos limitantes encontrados no método do Lagrangeano Relaxado. Observe que em nenhuma das duas instâncias o valor do limitante inferior alcançou o valor do limitante superior, levando o método a usar o GUROBI para buscar pelo resultado.

Figura 5.1 – Valores dos Limitantes no Lagrangeano Relaxado para uma Instância de Pequeno Porte



Fonte: O autor

Figura 5.2 – Valores dos Limitantes no Lagrangeano Relaxado para uma Instância de Grande Porte



Fonte: O autor

Utilizando os parâmetros de Eliiyi e Özlen (2008), com testes realizados fazendo a mudança de ϕ, $NumIt$ e o $atualizador$, foi possível identificar que os melhores parâmetros são os dos autores, exceto o $atualizador$ que funcionou melhor em $0,75$. Os resultados

para todas as instâncias testadas pelo método do Lagrangeano Relaxado são apresentados na seção 5.2.

5.1.2 Calibração dos Parâmetros do Algoritmo Genético

O algoritmo genético foi implementado no Matlab® por Heris, S. M. K. (2015) e então, adaptado para uso no problema deste estudo com o método da penalidade exterior para o tratamento das restrições.

Os parâmetros utilizados no algoritmo genético foram selecionados por meio de calibração dos parâmetros p_c e p_m testados em duas instâncias consideradas pequenas no grupo de pequeno porte e de grande porte. O número de população ($nPop$) depende do número de tarefas de cada instância sendo 500 para 10 tarefas, 1000 para 15 tarefas, 1500 para 20, 2000 para 30, 2500 para 60 e por fim 3000 para 120 tarefas. A estratégia em aumentar o número da população de acordo com o tamanho de N veio ao identificar que as instâncias maiores necessitavam de mais iterações para encontrar um bom resultado.

Foram testados três grupos de parâmetros no algoritmo genético [$p_c = 0,8$ e $p_m = 0,3$], [$p_c = 0,9$ e $p_m = 0,1$] e [$p_c = 0,5$ e $p_m = 0,4$] e para os testes, cada instância de calibração criada foi submetida a uma análise de sensibilidade realizado por meio da execução de 10 vezes em cada um dos grupos. Um gráfico de *BoxPlot* foi criado para identificar de forma visual a variação alcançada, pois ele utiliza de dados estatísticos para identificar a qualidade dos resultados obtidos. Nele, é mostrado o resultado máximo, mínimo, o primeiro e terceiro quartil e a mediana. O objetivo por trás da distribuição do *BoxPlot* é selecionar o grupo que apresenta o centro dos dados entre os quartis menores e uma simetria nos resultados.

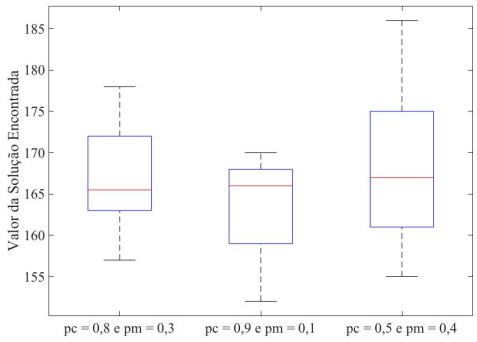
A Figura 5.3 mostra a variabilidade dos resultados obtidos com os grupos de parâmetros [$p_c = 0,8$ e $p_m = 0,3$], [$p_c = 0,9$ e $p_m = 0,1$] e [$p_c = 0,5$ e $p_m = 0,4$] na instância FLM31010 gerada aleatoriamente para este fim de calibração. Observa-se a diferença dos quartis nos três grupos e que o primeiro grupo possui a menor mediana e também uma boa simetria no primeiro e terceiro quartil.

A Figura 5.4 mostra a variabilidade dos resultados obtidos com os grupos de parâmetros [$p_c = 0,8$ e $p_m = 0,3$], [$p_c = 0,9$ e $p_m = 0,1$] e [$p_c = 0,5$ e $p_m = 0,4$] na instância FLM3308 gerada aleatoriamente para a calibração. Observa-se que também para esse grupo, a parametrização [$p_c = 0,8$ e $p_m = 0,3$] possui o menor conjunto de quartis com uma boa simetria, além da mediana estar mais próxima do primeiro quartil.

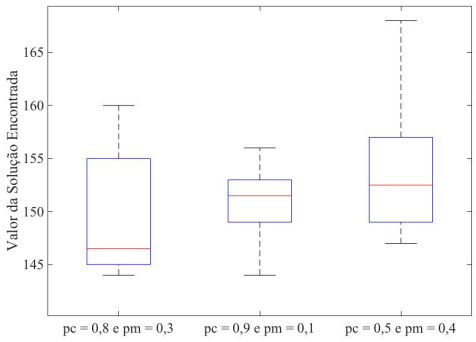
O número de iterações ($maxIt$) foi definido em 800, uma vez que é possível observar, conforme mostram as Figuras 5.5 e 5.6, a convergência da função objetivo entre as iterações 200 e 600. Também, é possível observar, a convergência dos resultados por meio dos parâmetros testados e verificar mais uma vez que a melhor convergência se deu por meio dos parâmetros em $p_c = 0,8$ e $p_m = 0,3$.

Figura 5.3 – Análise dos parâmetros para [$p_c = 0,8$ e $p_m = 0,3$] e [$p_c = 0,9$ e $p_m = 0,1$] e [$p_c = 0,5$ e $p_m = 0,4$] na Instância FLM31010

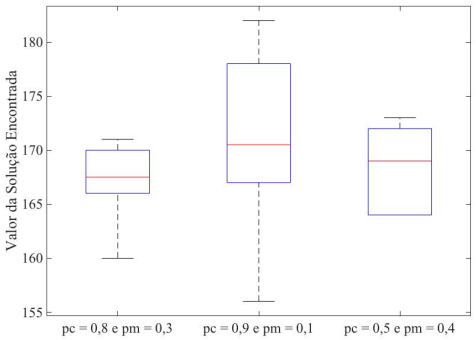
(a) FLM31010.1



(b) FLM31010.2



(c) FLM31010.3



Fonte: O autor

Tendo por base os testes realizados, pode-se concluir que o Algoritmo genético é capaz de retornar bons resultados usando a seguinte configuração: $p_c = 0,8$ e $p_m = 0,3$, $MaxIt = 800$ e $nPop$ variando de acordo com a quantidade de tarefas de cada instância. É importante dizer que o AG é um método heurístico, que não é capaz de retornar resultados ótimos em todas as vezes que ele é executado e, por meio disso, cada instância neste estudo foi rodada 10 vezes, sendo a menor solução dentre elas foi selecionada.

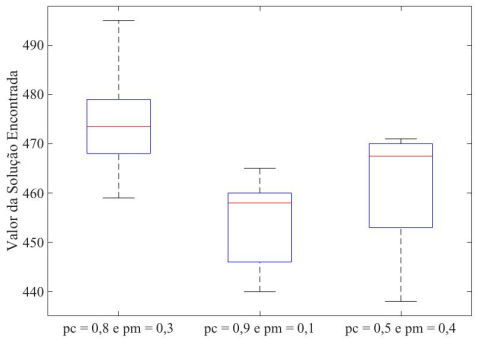
5.2 Resultados Obtidos

As Tabelas 5.1 e 5.2 desta seção mostram os resultados obtidos pelo GUROBI, Lagrangeano Relaxado e Algoritmo Genético nas instâncias geradas aleatoriamente. As comparações entre os métodos também são apresentadas nas tabelas e a comparação de resultados encontrados neste estudo não foi comparada com as do Eliiyi e Özlen (2008) devido as instâncias serem geradas de forma aleatória e os autores não disponibilizaram suas instâncias.

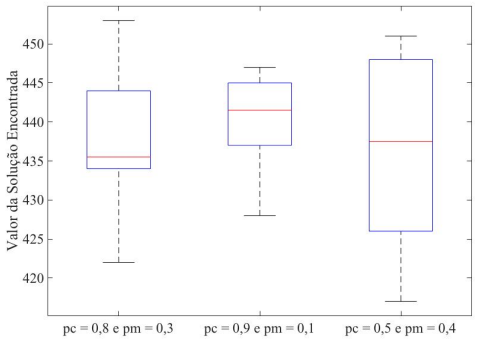
As informações das tabelas seguem da seguinte forma: na primeira coluna, tem-se o nome da instância; na segunda coluna há os parâmetros de tamanho de cada instância (M

Figura 5.4 – Análise dos parâmetros para [$p_c = 0,8$ e $p_m = 0,3$] e [$p_c = 0,9$ e $p_m = 0,1$] e [$p_c = 0,5$ e $p_m = 0,4$] na Instância FLM3308

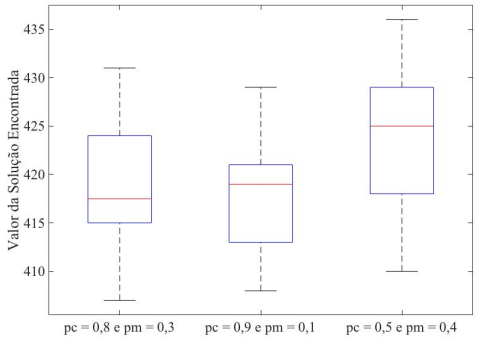
(a) FLM3308.1



(b) FLM3308.2

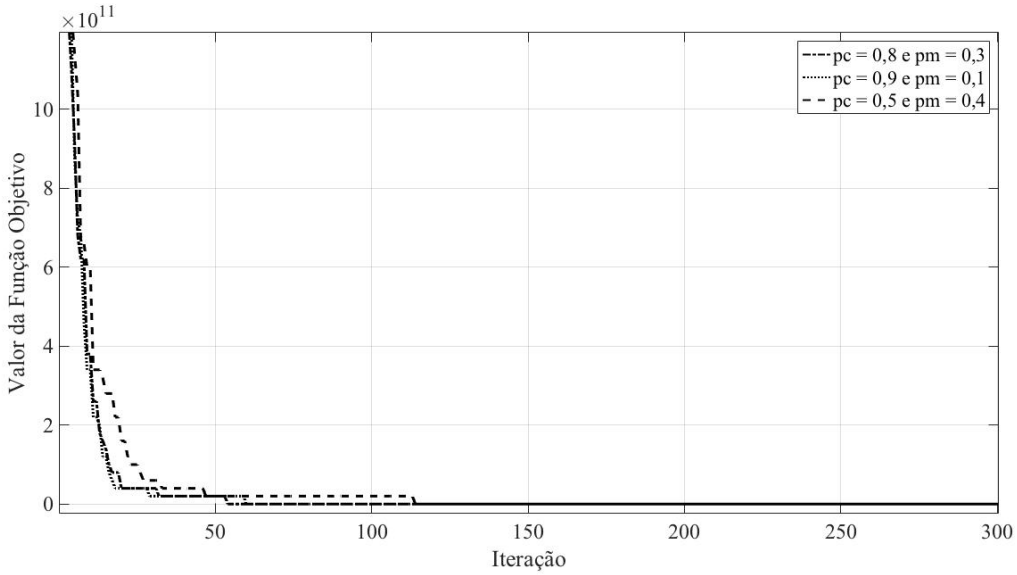


(c) FLM3308.3



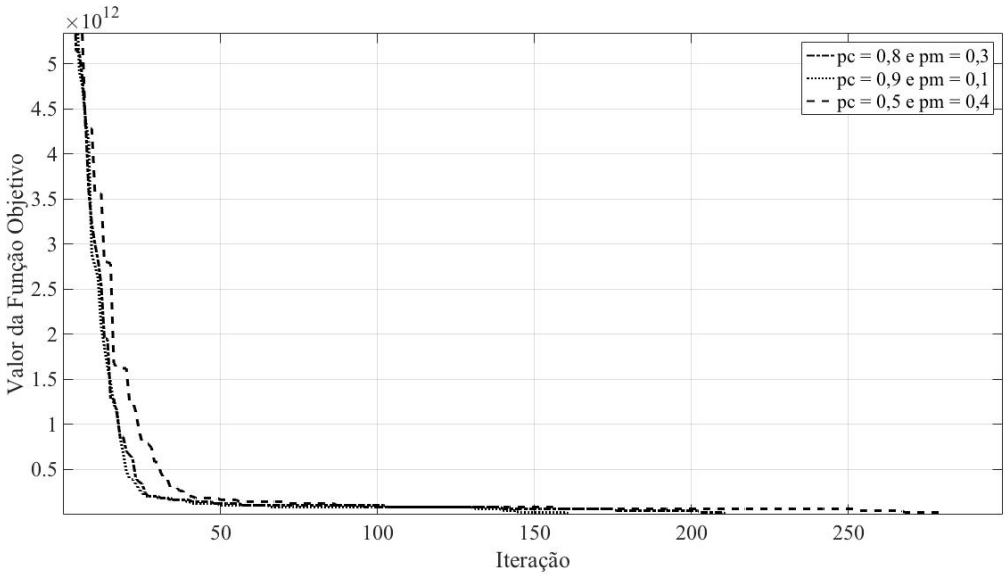
Fonte: O autor

Figura 5.5 – Convergência da instância FLM31010 nos parâmetros [$p_c = 0,8$ e $p_m = 0.3$] e [$p_c = 0,9$ e $p_m = 0,1$] e [$p_c = 0,5$ e $p_m = 0,4$]



Fonte: O autor

Figura 5.6 – Convergência da instância FLM3308 nos parâmetros [$p_c = 0,8$ e $p_m = 0.3$] e [$p_c = 0,9$ e $p_m = 0,1$] e [$p_c = 0,5$ e $p_m = 0,4$]



Fonte: O autor

para o número de estações, N para o número de tarefas e R para o número de conjuntos). Após esses dados, seguem os resultados do Lagrangeano relaxado com o tempo do procedimento e depois os valores encontrados dos limitantes inferior (LB) e superior (UB) e após, o tempo computacional do GUROBI com restrições de limitantes adicionados ao problema original. Na coluna seguinte seguem os resultados do modelo do problema original resolvido pelo GUROBI e, por fim, os resultados do Algoritmo Genético com o tempo computacional e a diferença percentual ($Dif = (AG - GU)/GU$) do resultado do AG com resultado do GUROBI.

A Tabela 5.1 apresenta os resultados das instâncias de pequeno porte. Das 27 instâncias apresentadas, o método do GUROBI conseguiu encontrar solução ótima (GAP igual a zero) para 26 instâncias com um tempo médio de CPU durante a execução de 405,76 segundos e um GAP igual a 5,99% na Instância FLM82020. No Lagrangeano Relaxado, 26 das 27 instâncias retornaram resultado ótimo com GAP igual a zero e a instânciasFLM82020 obteve um GAP igual a 3,92% bem próximo da solução ótima. O tempo computacional médio foi de 705,99 segundos no Lagrangeano, cerca de 42,5% mais lento que o GUROBI.

O Algoritmo Genético mostrou-se o método com o melhor tempo médio computacional de 32,64 segundos, e em 4 instâncias (FLM3103, FLM3105, FLM5103 e FLM8103) ele foi capaz de encontrar o resultado ótimo para o problema. A diferença média dos resultados obtidos no AG com o GUROBI ficou em 8,1% dentre todas as 27 instâncias. A Figura 5.7 mostra de forma gráfica os resultados obtidos nos métodos. É possível observar um ligeiro aumento nos resultados quando comparado o AG com o LR e o GUROBI. No LR e o GUROBI, por terem obtido em 26 das 27 instâncias GAP igual a zero, os marcadores se sobrepuseram.

Tabela 5.1 – Resultados das Instâncias de Pequeno Porte com até 20 Tarefas

Instância	M	N	R	Lagrangeano Relaxado						GUROBI			Algoritmo Genético		
				Tempo	LB	UB	Tempo	Solução	Gap	Tempo	Solução	Gap	Tempo	Solução	Dif AG-GU
FLM3103	3	10	3	0,53	168	168	–	–	0,00%	0,00	168	0,00%	0,64	168	0,0%
FLM3105	3	10	5	1,38	142	159	0,09	157	0,00%	0,06	157	0,00%	2,81	157	0,0%
FLM31010	3	10	10	2,78	146	156	0,54	152	0,00%	0,10	152	0,00%	8,74	160	5,3%
FLM3154	3	15	4	0,13	203	203	–	–	0,00%	0,01	203	0,00%	2,78	205	1,0%
FLM3158	3	15	8	2,54	174	205	4,52	201	0,00%	2,33	201	0,00%	8,75	228	13,4%
FLM31515	3	15	15	3,15	170	188	41,29	181	0,00%	15,86	181	0,00%	27,09	212	17,1%
FLM3205	3	20	5	0,98	269	300	0,35	296	0,00%	1,25	296	0,00%	4,80	320	8,1%
FLM32010	3	20	10	2,56	280	305	36,66	292	0,00%	10,11	292	0,00%	15,50	324	11,0%
FLM32020	3	20	20	3,78	226	256	1.188,85	243	0,00%	319,30	243	0,00%	51,06	301	23,9%
FLM5103	5	10	3	1,98	171	178	0,03	178	0,00%	0,02	178	0,00%	4,64	178	0,0%
FLM5105	5	10	5	1,82	147	167	1,16	164	0,00%	0,04	164	0,00%	5,42	167	1,8%
FLM51010	5	10	10	2,07	156	171	0,38	171	0,00%	0,36	171	0,00%	22,30	183	7,0%
FLM5154	5	15	4	2,72	212	267	0,56	264	0,00%	0,23	264	0,00%	4,25	281	6,4%
FLM5158	5	15	8	1,69	238	271	11,17	261	0,00%	2,68	261	0,00%	21,37	275	5,4%
FLM51515	5	15	15	3,36	201	321	45,86	225	0,00%	12,83	225	0,00%	43,19	259	15,1%
FLM5205	5	20	5	2,18	388	396	0,09	390	0,00%	0,10	390	0,00%	11,22	399	2,3%
FLM52010	5	20	10	2,63	280	330	1.368,69	316	0,00%	830,59	316	0,00%	34,82	353	11,7%
FLM52020	5	20	20	3,73	258	307	894,36	291	0,00%	916,75	291	0,00%	92,50	358	23,0%
FLM8103	8	10	3	3,05	243	272	0,05	271	0,00%	0,02	271	0,00%	7,09	271	0,0%
FLM8105	8	10	5	2,07	176	220	0,39	216	0,00%	0,20	216	0,00%	6,33	225	4,2%
FLM81010	8	10	10	1,93	169	196	0,52	196	0,00%	0,17	196	0,00%	23,25	211	7,7%
FLM8154	8	15	4	0,90	273	293	0,12	292	0,00%	0,09	292	0,00%	9,11	299	2,4%
FLM8158	8	15	8	2,49	237	308	17,26	297	0,00%	8,70	297	0,00%	39,56	324	9,1%
FLM81515	8	15	15	2,65	259	328	380,00	312	0,00%	141,23	312	0,00%	114,91	342	9,6%
FLM8205	8	20	5	2,17	306	407	9,13	393	0,00%	7,83	393	0,00%	25,09	425	8,1%
FLM82010	8	20	10	2,89	379	433	6.447,74	413	0,00%	1.484,62	413	0,00%	78,56	452	9,4%
FLM82020	8	20	20	4,62	277	378	7.200,00	358	3,92%	7.200,00	358	5,99%	215,53	414	16,9%
Média				2,33			705,99		0,1%	405,76		0,2%	32,64		8,1%

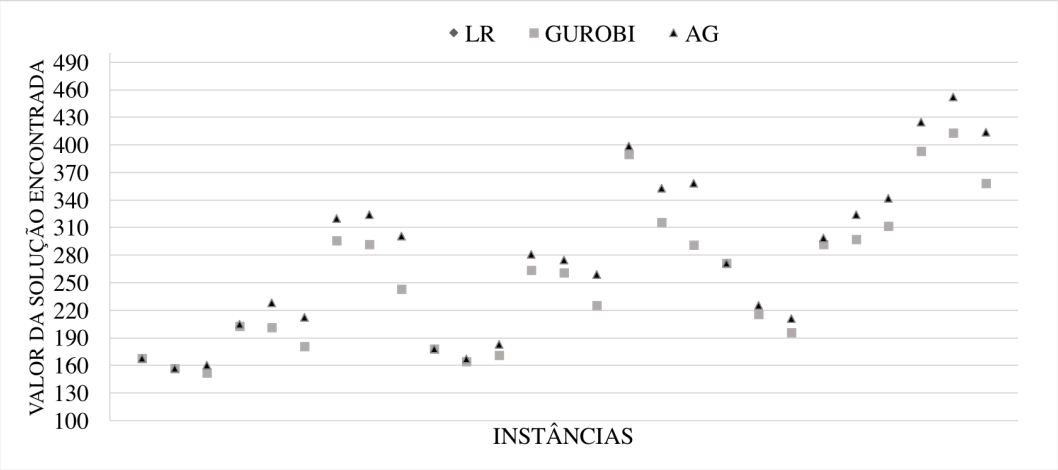
Fonte: o autor.

Tabela 5.2 – Resultados das Instâncias de Grande Porte com até 120 Tarefas

Instância	M	N	R	Lagrangeano Relaxado						GUROBI			Algoritmo Genético		
				Tempo	LB	UB	Tempo	Solução	Gap	Tempo	Solução	Gap	Tempo	Solução	Dif AG-GU
FLM3308	3	30	8	2,93	378	421	2.759,06	398	**0,00%**	525,00	398	0,00%	83,97	459	15,3%
FLM33015	3	30	15	5,98	340	403	7.200,00	378	**2,38%**	7.200,00	379	3,12%	100,18	455	20,1%
FLM33030	3	30	30	6,39	342	402	7.200,00	382	**2,88%**	7.200,00	382	2,88%	153,44	483	26,4%
FLM36015	3	60	15	16,57	754	827	7.200,00	792	**4,80%**	7.200,00	792	4,80%	242,10	885	11,7%
FLM36030	3	60	30	13,40	758	844	7.200,00	796	**4,77%**	7.200,00	799	5,13%	406,27	958	19,9%
FLM36060	3	60	60	54,67	658	770	7.200,00	711	**6,75%**	7.200,00	718	7,66%	685,19	881	22,7%
FLM312030	3	120	30	64,31	1.356	1.628	7.200,00	1.491	8,72%	7.200,00	1.448	8,53%	1.113,40	1.843	27,3%
FLM312060	3	120	60	120,25	1.354	1.628	7.200,00	1.481	7,70%	7.200,00	1.474	7,26%	2.296,38	1.813	23,0%
FLM3120120	3	120	120	264,36	1.326	1.564	7.200,00	1.437	**4,53%**	7.200,00	1.442	4,85%	4.393,80	1.780	23,4%
FLM5308	5	30	8	1,97	442	520	7.200,00	492	1,07%	7.200,00	492	1,02%	91,79	560	13,8%
FLM53015	5	30	15	5,25	395	487	7.200,00	451	**4,66%**	7.200,00	453	5,96%	106,51	527	16,3%
FLM53030	5	30	30	6,22	407	483	7.200,00	460	**6,83%**	7.200,00	464	7,54%	182,53	552	19,0%
FLM56015	5	60	15	12,56	698	912	7.200,00	850	11,41%	7.200,00	848	8,84%	325,56	1.000	17,9%
FLM56030	5	60	30	22,10	726	895	7.200,00	844	13,74%	7.200,00	840	13,33%	439,14	1.020	21,4%
FLM56060	5	60	60	55,90	713	867	7.200,00	810	11,98%	7.200,00	808	11,76%	949,62	1.009	24,9%
FLM512030	5	120	30	71,57	1.340	1.670	7.200,00	1.584	14,84%	7.200,00	1.567	13,91%	1.391,04	1.885	20,3%
FLM512060	5	120	60	122,52	1.293	1.650	7.200,00	1.587	18,34%	7.200,00	1.579	17,92%	2.655,40	1.929	22,2%
FLM5120120	5	120	120	291,66	1.441	1.762	7.200,00	1.668	**13,50%**	7.200,00	1.703	15,32%	5.740,95	2.040	19,8%
FLM8308	8	30	8	2,45	393	534	7.200,00	489	3,48%	7.200,00	487	2,46%	105,22	574	17,9%
FLM83015	8	30	15	3,96	398	500	7.200,00	474	**6,75%**	7.200,00	479	7,72%	132,59	555	15,9%
FLM83030	8	30	30	7,85	404	533	7.200,00	506	**13,90%**	7.200,00	512	14,84%	204,10	617	20,5%
FLM86015	8	60	15	14,23	949	1.127	7.200,00	1.048	**7,16%**	7.200,00	1.050	8,48%	436,35	1.186	13,0%
FLM86030	8	60	30	27,26	761	1.018	7.200,00	957	**15,99%**	7.200,00	950	19,05%	620,56	1.107	16,5%
FLM86060	8	60	60	51,02	737	1.016	7.200,00	967	**22,96%**	7.200,00	983	24,01%	1.067,43	1.147	16,7%
FLM812030	8	120	30	80,73	1.503	2.102	7.200,00	2.005	22,89%	7.200,00	1.977	19,68%	1.668,21	2.304	16,5%
FLM812060	8	120	60	147,00	1.632	2.011	7.200,00	1.947	15,71%	7.200,00	1.933	15,21%	3.188,08	2.266	17,2%
FLM8120120	8	120	120	300,38	1.328	1.989	7.200,00	1.998	33,28%	7.200,00	1.919	30,38%	7.040,65	2.218	15,6%
Média				65,69			7.035,52		10,41%	6.952,78		10,43%	1.326,68		19,1%

Fonte: o autor.

Figura 5.7 – Resultados obtidos nas instâncias de Pequeno Porte.



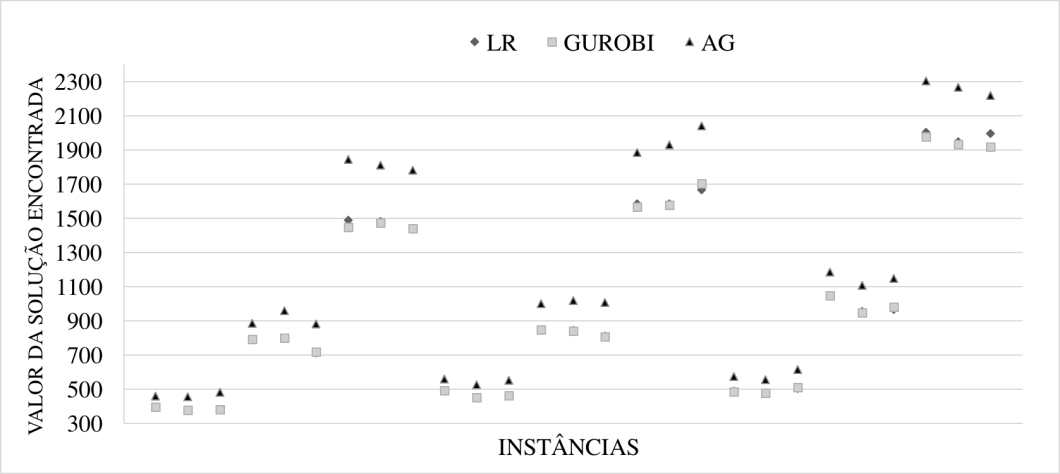
Fonte: O autor

A Tabela 5.2 mostra os resultados das instâncias de grande porte para os três métodos comentados nas seções anteriores. Das 27 instâncias apresentadas, o GUROBI conseguiu um GAP médio de 10,43%, o Lagrangeano Relaxado alcançou um GAP médio de 10,41% e o Algoritmo genético, a diferença média entre o GUROBI fechou com 19,01%. Em termos de tempo computacional, o Lagrangeano Relaxado encontrou os limitantes das instâncias com uma média de 65,69 segundos e a segunda parte foi interrompida por critério de parada, ao atingir 7.200 segundos. O Algoritmo Genético teve uma média computacional de 1.326,68 segundos, cerca de 19% mais rápido que os demais métodos.

No Lagrangeano Relaxado, cujo procedimento é de reduzir o campo de busca por meio dos limitantes e, então, aplicar o GUROBI nas instâncias, o mesmo apresentou uma melhora de soluções em 15 das 27 instâncias testadas, apesar da diferença de GAP entre eles serem de apenas 0,02%. A Figura 5.8 mostra, de forma gráfica, os resultados obtidos entre os três métodos utilizados. Nela, é possível ver a diferença de resultados entre o LR, GUROBI e o AG, onde o AG apresenta resultados com GAP mais distantes, porém, com tempo computacional menor.

A Figura 5.9 mostra o GAP obtido no LR e o GUROBI e a diferença de solução do AG com o GUROBI. No Gráfico 5.9a, há a diferença do AG/GU nas instâncias de pequeno porte e o GAP do GUROBI e Lagrangeano Relaxado. Veja que neste gráfico o GAP do LR e do GUROBI apresenta apenas 1 instâncias com GAP entre 0,00 e 5%. Os demais GAP são a variação de resultados obtidas no AG. No Gráfico 5.9b é apresentado o GAP obtido nas instâncias de Grande Porte. O método que obteve a maior quantidade de resultados com GAP inferior a 5% foi o LR, seguido pelo GUROBI, enquanto o AG não obteve nenhuma instância com esses limites de GAP. Para o GAP entre 5.01% e 10%, o GUROBI conseguiu obter mais instâncias seguido pelo LR. Observe-se que o AG foi apenas aparecer quando a diferença de resultados acima de 10,01%, mostrando que sua margem de resultados perto do ótimo é um pouco

Figura 5.8 – Resultados obtidos nas instâncias de Grande Porte

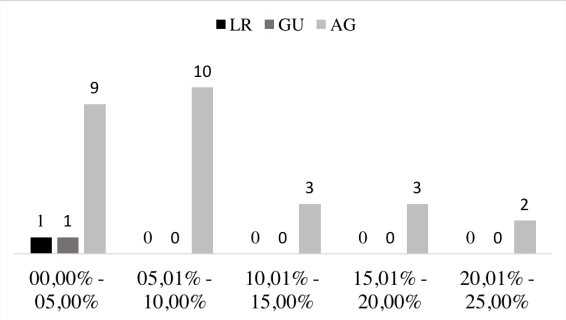


Fonte: O autor

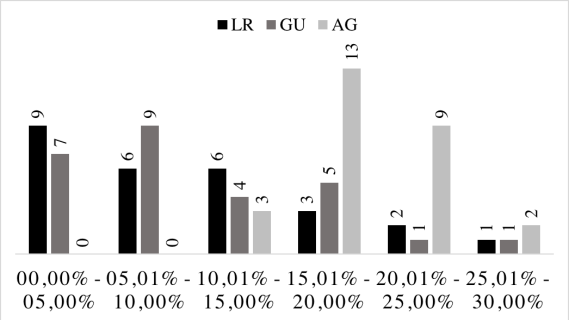
maior que os outros dois métodos.

Figura 5.9 – GAP médio em porcentagem obtido para o FLM

(a) GAP médio das instâncias de Pequeno Porte



(b) GAP médio das instâncias de Grande Porte



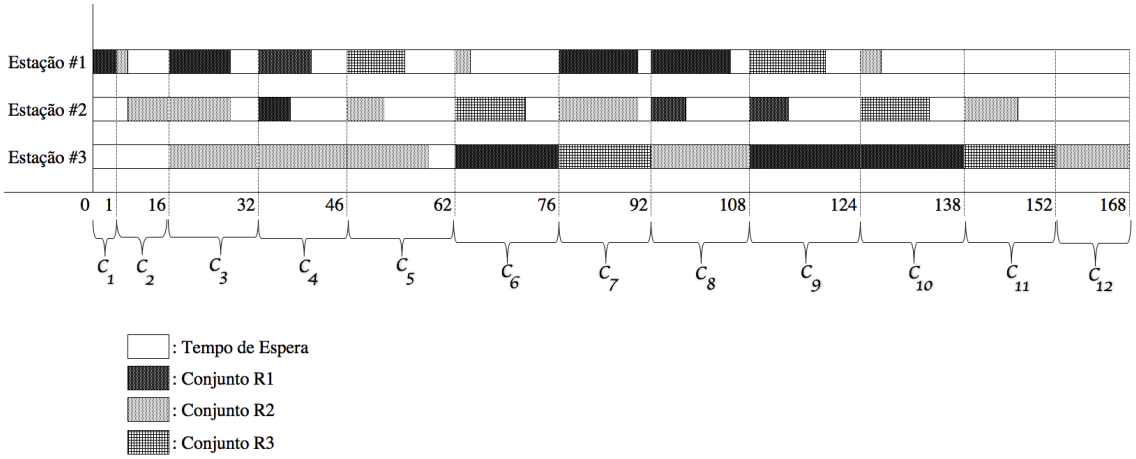
Fonte: O autor

Para fins de exemplificação de um caso, foi selecionada uma instância de pequeno porte com 10 tarefas e 3 estações para apresentar a solução Lagrangeano Relaxado através do gráfico de Gantt. Em problemas de sequenciamento de produção, a busca pelo *Makespan* vem juntamente com a sequência das ordens das tarefas a serem processadas, mostrando de forma visual, todas as etapas de processamento, tendo elas, conforme dito anteriormente, tarefas com inicio, meio e fim. A Figura 5.10 mostra a instância com 3 estações, 10 tarefas e 3 conjuntos. Observe-se que no gráfico é apresentado apenas o conjunto de tarefas e, como a quantidade de estações e de conjuntos são iguais ($M = N = 3$), estas tarefas podem ser processadas em estações distintas em um único ciclo.

Na instância FLM3103 apresentada na Figura 5.10, o C_{max} é de 168. Observa-se que no primeiro ciclo, as estações 2 e 3 permanecem em tempo de espera, e apenas no segundo ciclo é que elas entram em utilização, sendo que nos últimos ciclos, as estações (1) e (2) não

estão sendo utilizadas.

Figura 5.10 – Gráfico de Gantt para a Instância FLM3103



Fonte: O autor

Capítulo 6

Conclusões

Neste trabalho foi apresentado o problema *Flow Line* Misto com o conjunto de tarefas a serem sequenciadas em máquinas (ELIIYI; ÖZLEN, 2008). O *Flow Line* Misto descreve o sequenciamento de um conjunto de tarefas em estações de máquinas com a possibilidade de manufatura mista no qual é possível trabalhar com diferentes conjuntos de produtos ao mesmo tempo, uma vez que são modificados apenas por características, como cor, acabamento e detalhes. Esse tipo de personalização é bastante comum na indústria automotiva conforme aponta Agnetis *et al.* (1997) e na indústria de semicondutores descrita por Quadt e Kuhn (2007).

O problema FLM deste trabalho considerou N tarefas e M estações de máquinas possíveis, sendo que por via de regra, cada conjunto R de tarefas deve passar por todos os ciclos até finalizar a execução de todas as estações. Foram utilizados três métodos de otimização para resolver 54 instâncias geradas aleatoriamente conforme a literatura do problema.

No primeiro método, que usa o *framework* do GUROBI, o modelo do problema foi implementado na forma original de Eliiyi e Özlen (2008), sendo analisado o tempo computacional e a melhor solução encontrada por meio do GAP dos resultados. No segundo método, o Lagrangeano Relaxado, foi estudada a busca de uma melhora no tempo de processamento e a melhora de resultados para as instâncias com N igual 30, 60 ou 120 tarefas em relação aos resultados do GUROBI. Por fim, foi utilizado um Algoritmo Genético para as instâncias geradas e uma comparação dos resultados com o GUROBI foi feita nos mesmo termos do Lagrangeano Relaxado.

Os resultados do GUROBI tinham como objetivo principal a busca pela solução ótima e o método obteve sucesso nas instâncias de até 30 tarefas. Já nas instâncias de 30 a 120 tarefas, o método retornou um GAP médio de $10,43\%$ e caso não houvesse critério de parada em $7,200$ segundos, o método continuaria a busca pela solução ótima do problema.

O Lagrangeano Relaxado se mostrou bastante efetivo na busca pelos limitantes superior e inferior e ao integrar os limitantes no problema original por meio de restrições laterais.

Com isso a melhora dos resultados se mostrou efetiva em 55% das instâncias com 30, 60 ou 120 tarefas. Já nas instâncias com 10, 15 ou 20 tarefas, o Lagrangeano ficou com mesmo desempenho do GUROBI na qualidade dos resultados obtidos.

O Algoritmo genético obteve resultados inferiores com GAPs altos se comparado com o do GURUBI, porém o AG é um excelente método para encontrar soluções viáveis em baixo tempo computacional. Um exemplo disso está nas instâncias com tarefas entre 60 e 120. Observa-se que no GUROBI ou no LR, o tempo de processamento foi em média de 6.952 segundos, já no AG o tempo foi em média de 1.326 segundos. O GAP entre os métodos ficou em 19, 1%, que pode ser considerado, em diferentes circunstâncias, grande para o tomador de decisões que precisa de bons resultados em tempo reduzido.

O Algoritmo Genético requer bastante atenção durante a calibração dos parâmetros para garantir que as instâncias tenham bom desempenho. Uma forma de encontrar os parâmetros ideais foi selecionar diferentes parâmetros e testá-los nas instâncias consideradas pequenas, analisando as iterações realizadas até que o método encontrasse resultados que não violassem as restrições pela penalidade exterior. Outro fator interessante no AG é o tamanho da população selecionada para abranger o campo de busca de bons resultados. A melhor forma de satisfazer o tamanho da população para a variação de tarefas foi definir um tamanho diferente de população para cada instância, de acordo com a quantidade de N tarefas.

Ao analisar o tempo computacional, o AG foi aproximadamente 48 vezes mais rápido que o GUROBI e 22 vezes mais rápido que o Lagrangeano Relaxado. Isto mostra uma grande importância para indústrias que almejam rápidas análises em pouco tempo computacional tomar decisões importantes no sequenciamento de produção.

Pode-se concluir com este estudo que os três métodos estudados foram satisfatórios para o problema. Porém, é importante analisar duas realidades entre os métodos: a primeira realidade é nos métodos exatos onde a busca pelo resultado ótimo é encontrada em instâncias com poucas tarefas, e o custo computacional é baixo. Isto ficou evidente em instâncias como a FLM3103, FLM3105, FLM31010 que não gastaram um segundo para encontrar o resultado ótimo com o GUROBI. Já em instâncias maiores, a partir de 20 tarefas, a realidade se transforma e o GUROBI não são foi capaz de encontrar um resultado satisfatório em um tempo computacional pequeno; a segunda realidade é a referente ao AG, que não retornou bons resultados para todas as instâncias, porém seu tempo computacional é satisfatório para análises rápidas a curto prazo cuja importância é relevante no mundo da indústria.

Neste estudo, os critérios de parada para os métodos foram o tempo máximo de execução e o número máximo de iterações. O tempo máximo de execução do GUROBI foi determinado em 7, 200 segundos, porém, o método permite ir mais além do que isto até que se encontre a solução ótima do problema. Portanto o tempo de 7, 200 foi definido para o GUROBI como critério de parada, tendo o método encontrado um resultado ótimo ou não.

No Lagrangeano Relaxado, a primeira parte do método utilizou o critério de parada por meio do número máximo de iterações, sendo eles de 400, 800 e 1200, de acordo com o tamanho da instância. A segunda parte do método seguiu os mesmos critérios do GUROBI. O AG também utilizou o número máximo de iterações, sendo definido em 800 para todas as iterações e os demais parâmetros do AG foram calibrados conforme está no capítulo 5.

O trabalho utilizou do estudo proposto por Eliiyi e Özlen (2008) no intuito de obter uma melhor exploração do FLM que se mostrou pouco estudado nos últimos anos pela literatura, embora seja de grande interesse industrial. Os resultados obtidos se mostraram satisfatórios com 26 instâncias resolvidas em sua otimalidade e coerentes com os principais objetivos do presente estudo, porém não foi possível comparar os resultados aqui apresentados com os do Eliiyi e Özlen (2008), pois as instâncias foram geradas aleatoriamente e, consequentemente, os resultados do *Makespan* foram diferentes, impossibilitando a análise. No entanto, Eliiyi e Özlen (2008) informou que em seu estudo apenas 17 das 54 instâncias puderam ser resolvido em sua otimalidade, mostrando que o estudo aqui apresentando alcançou 50% de eficiênca em contra partida de 31% de Eliiyi e Özlen (2008).

Em termos de tempo computacional, também não foi possível comparar o Lagrangeano Relaxado de Eliiyi e Özlen (2008) com o deste estudo, pois os recursos computacionais tiveram atualizações tecnológicas impossibilitando tal relação. É importante enfatizar que em instâncias pequenas, o trabalho de Eliiyi e Özlen (2008) gastou mais de 50.000 segundos e não conseguiu encontrar uma solução aproximada do ótimo, e no estudo desta dissertação, foi possivel obter resultados com GAPs menor que 5% nas instâncias com as mesmas configurações com um tempo máximo de 7.200 segundos.

Trabalhos futuros para este problema incluem a investigação de formas de reduzir o tempo de espera das estações ociosas entre os ciclos, bem como a possibilidade de considerar máquinas indisponíveis ou quebradas durante o processo. Essas restrições podem ser interessantes para que o problema se aproxime cada vez mais do que de fato acontece na indústria de manufatura. Porém, faz-se necessário saber que novas restrições no problema requerem maior dificuldade de implementação e de modelagem.

Na parte de otimização, trabalhos futuros incluem a consideração de um outro método que possa se beneficiar de regras sequenciamento da PP. Outros *solvers* como LINGO, CPLEX ou até a proposta de um algoritmo exato adaptado para o FLM podem ser considerados.

O Lagrangeano Relaxado pode ter um estudo aprofundado na análise do gradiente para uma melhora dos valores do atualizador λ, como forma de reduzir as violações dos limitantes, melhorando o tempo de execução. O gradiente e o valor de λ tem papel fundamental no desempenho do LR.

No método AG, a proposta de um método genético para o FLM pode ser considerada.

Uma proposta de um AG para o FLM bastante comum na área da PP e bastante coerente com o estudo do FLM pode ser considerada, pois é possível encontrar na literatura trabalhos que fizeram tais adaptações com problemas similares e a performance do AG permitiu melhoras significativas. Um exemplo disso é no trabalho de Sheikh (2013), que, em seus resultados, os autores chegaram a encontrar soluções ótimas nas instâncias testadas usando um AG proposto para o FFL.

REFERÊNCIAS

AGNETIS, A. *et al.* Scheduling of flexible flow lines in an automobile assembly plant. *European Journal of Operational Research*, v. 97, n. 2, p. 348 – 362, 1997. ISSN 0377-2217. Citado 5 vezes nas páginas 29, 37, 48, 50 e 89.

ALEXANDROS, D. C.; CHRISSOLEON, P. T. Exact analysis of a two-workstation one-buffer flow line with parallel unreliable machines. *European Journal of Operational Research*, v. 197, n. 2, p. 572 – 580, 2009. ISSN 0377-2217. Citado 2 vezes nas páginas 31 e 48.

AN, Y.; YAN, H. An iterative approach based on hybrid optimization for simultaneous optimization of production planning and scheduling. In: *2011 Fourth International Symposium on Computational Intelligence and Design*. [S.l.: s.n.], 2011. v. 1, p. 244–249. Citado na página 48.

AN, Y.-W.; YAN, H.-S. Lagrangean relaxation approach to joint optimization for production planning and scheduling of synchronous assembly lines. *International Journal of Production Research*, v. 54, n. 22, p. 6718–6735, 2016. Citado na página 48.

ARMENTANO, V. A.; MAZZINI, R. A genetic algorithm for scheduling on a single machine with set-up times and due dates. *Production Planning and Control*, v. 11, n. 7, p. 713–720, 2000. Citado na página 48.

BALAS, E.; CARREIRA, M. C. A dynamic subgradient-based branch-and-bound procedure for set covering. *Operations Research*, v. 44, n. 6, p. 875 – 890, 1996. Citado na página 69.

BOLAT, A.; SAVSAR, M.; AL-FAWZAN, M. A. Algorithms for real-time scheduling of jobs on mixed model assembly lines. *Computers and Operations Research*, v. 21, n. 5, p. 487 – 498, 1994. ISSN 0305-0548. Citado 3 vezes nas páginas 43, 55 e 56.

BUZZO, W. R.; MOCCELLIN, J. V. Programação da produção em sistemas flow shop utilizando um método heurístico híbrido algoritmo genético-simulated annealing. *Gestão e Produção*, scielo, v. 7, p. 364 – 377, 12 2000. ISSN 0104-530X. Citado na página 48.

CHEN, C.-L.; CHEN, C.-L. A bottleneck-based heuristic for minimizing makespan in a flexible flow line with unrelated parallel machines. *Computers and Operations Research*, v. 36, n. 11, p. 3073 – 3081, 2009. ISSN 0305-0548. Citado na página 48.

____. Bottleneck-based heuristics to minimize total tardiness for the flexible flow line with unrelated parallel machines. *Computers and Industrial Engineering*, v. 56, n. 4, p. 1393 – 1401, 2009. ISSN 0360-8352. Citado na página 48.

CHO, H.-S. *et al.* A robust design of simulated annealing approach for mixed-model sequencing. *Computers and Industrial Engineering*, v. 48, n. 4, p. 753 – 764, 2005. ISSN 0360-8352. Selected Papers from The 30th International Conference on Computers and Industrial Engineering. Citado 3 vezes nas páginas 55, 56 e 57.

COPACEANU, C. Mixed-model assembly line balancing problem: variants and solving techniques. In: *International Conference on Multimodal Interaction*. [S.l.: s.n.], 2006. v. 45, p. 337–360. Citado 2 vezes nas páginas 55 e 57.

COSTA, M. T.; FERREIRA, J. S. A simulation analysis of sequencing rules in a flexible flowline. *European Journal of Operational Research*, v. 119, n. 2, p. 440 – 450, 1999. ISSN 0377-2217. Citado na página 48.

COSTANTINO, F. *et al.* Scheduling mixed-model production on multiple assembly lines with shared resources using genetic algorithms: The case study of a motorbike company. *Advances in Decision Sciences*, 2014. Citado 3 vezes nas páginas 31, 55 e 72.

DING, F.-Y.; KITTICHARTPHAYAK, D. Heuristics for scheduling flexible flow lines. *Computers and Industrial Engineering*, v. 26, n. 1, p. 27 – 34, 1994. ISSN 0360-8352. Citado na página 48.

ELIIYI, D. T.; ÖZLEN, M. A lagrangean relaxation approach for the mixed-model flow line sequencing problem. *Computers and Operations Research*, v. 35, n. 3, p. 933 – 943, 2008. ISSN 0305-0548. Part Special Issue: New Trends in Locational Analysis. Citado 18 vezes nas páginas 29, 31, 32, 33, 34, 55, 57, 58, 65, 67, 68, 69, 77, 78, 79, 81, 89 e 91.

FANTI, M. *et al.* Genetic multi-criteria approach to flexible line scheduling. *International Journal of Approximate Reasoning*, v. 19, n. 1, p. 5 – 21, 1998. ISSN 0888-613X. Citado na página 48.

FERNANDES, F.; FILHO, M. G. *Planejamento e controle da produção: dos fundamentos ao essencial*. São Paulo, SP: Atlas, 2010. ISBN 9788522458714. Citado 2 vezes nas páginas 29 e 40.

FISHER, M. L. The lagrangian relaxation method for solving integer programming problems. *Management Science*, v. 50, n. 12, p. 1861–1871, 1981. Citado 5 vezes nas páginas 48, 65, 66, 69 e 78.

FLESZAR, K. A new milp model for the accessibility windows assembly line balancing problem level 2 (awalbp-l2). *European Journal of Operational Research*, v. 259, n. 1, p. 169 – 174, 2017. ISSN 0377-2217. Citado 2 vezes nas páginas 55 e 59.

FUCHIGAMI, H. Y. *Sequenciamento da Produção*. Catalão, GO: Editora Cegraf, Coleção Lavor, 2015. ISBN 978-85-68359-37-2. Citado 2 vezes nas páginas 36 e 39.

FUCHIGAMI, H. Y.; MOCCELLIN, J. V.; RUIZ, R. Novas regras de prioridade para programação em flexible flow line com tempos de setup explícitos. *Production*, scielo, v. 25, p. 779 – 790, 12 2015. ISSN 0103-6513. Citado na página 48.

GAREY, M. R.; JOHNSON, D. S.; SETHI, R. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, v. 1, n. 2, p. 117–129, 1976. ISSN 0364-765X. Citado na página 30.

GOHARI, S.; SALMASI, N. Flexible flowline scheduling problem with constraints for the beginning and terminating time of processing of jobs at stages. *International Journal of Computer Integrated Manufacturing*, v. 28, n. 10, p. 1092–1105, 2015. Citado 3 vezes nas páginas 41, 43 e 48.

GUO, Z. X. *et al.* A genetic-algorithm-based optimization model for scheduling flexible assembly lines. *The International Journal of Advanced Manufacturing Technology*, v. 36, n. 1, p. 156–168, 2008. ISSN 1433-3015. Citado na página 48.

HASHEM, S. M. A. e; ARYANEZHAD, M. An efficient method to solve a mixed-model assembly line sequencing problem considering a sub-line. *World Applied Sciences Journal*, v. 6, n. 2, p. 168 – 181, 2009. ISSN 1818-4952. Citado 2 vezes nas páginas 55 e 58.

HENDIZADEH, S. H. *et al.* Meta-heuristics for scheduling a flowline manufacturing cell with sequence dependent family setup times. *International Journal of Production Economics*, v. 111, n. 2, p. 593 – 605, 2008. ISSN 0925-5273. Special Section on Sustainable Supply Chain. Citado 3 vezes nas páginas 43, 48 e 52.

Heris, S. M. K. *Implementation of Binary Genetic Algorithm in MATLAB*. 2015. Https://www.yarpiz.com. Online; accesso 15 Março 2017. Citado 2 vezes nas páginas 77 e 80.

JOLAI, F. *et al.* A genetic algorithm for solving no-wait flexible flow lines with due window and job rejection. *The International Journal of Advanced Manufacturing Technology*, v. 42, n. 5, p. 523, 2009. ISSN 1433-3015. Citado 2 vezes nas páginas 48 e 52.

JONG, R. *Designing Mixed-Model Assembly Lines: A Literature Review*. Dissertação (Mestrado) — Delft University of Technology, Osaka University, Japan, 2009. Citado 3 vezes nas páginas 55, 61 e 62.

KARA, Y. Line balancing and model sequencing to reduce work overload in mixed-model u-line production environments. *Engineering Optimization*, v. 40, n. 7, p. 669–684, 2008. Citado na página 55.

KARABATı, S.; SAYıN, S. Assembly line balancing in a mixed-model sequencing environment with synchronous transfers. *European Journal of Operational Research*, v. 149, n. 2, p. 417 – 429, 2003. ISSN 0377-2217. Sequencing and Scheduling. Citado 2 vezes nas páginas 55 e 56.

KHALILI, M.; NADERI, B. A bi-objective imperialist competitive algorithm for no-wait flexible flow lines with sequence dependent setup times. *The International Journal of Advanced Manufacturing Technology*, v. 76, n. 1, p. 461–469, 2015. ISSN 1433-3015. Citado 2 vezes nas páginas 48 e 53.

KIA, H.; GHODSYPOUR, S. H.; DAVOUDPOUR, H. New scheduling rules for a dynamic flexible flow line problem with sequence-dependent setup times. *Journal of Industrial Engineering International*, v. 13, n. 3, p. 297–306, Sep 2017. ISSN 2251-712X. Disponível em: <https://doi.org/10.1007/s40092-017-0185-y>. Citado na página 48.

KOCHHAR, S.; MORRIS, R. J. Heuristic methods for flexible flow line scheduling. *Journal of Manufacturing Systems*, v. 6, n. 4, p. 299 – 314, 1987. ISSN 0278-6125. Citado 3 vezes nas páginas 47, 48 e 49.

KOCHHAR, S.; MORRIS, R. J.; WONG, W. S. The local search approach to flexible flow line scheduling. *Engineering Costs and Production Economics*, v. 14, n. 1, p. 25 – 37, 1988. ISSN 0167-188X. Citado 2 vezes nas páginas 41 e 48.

KURZ, M. E.; ASKIN, R. G. Comparing scheduling rules for flexible flow lines. *International Journal of Production Economics*, v. 85, n. 3, p. 371 – 388, 2003. ISSN 0925-5273. Structuring and Planning Operations. Citado 2 vezes nas páginas 48 e 51.

____. Scheduling flexible flow lines with sequence-dependent setup times. *European Journal of Operational Research*, v. 159, n. 1, p. 66 – 82, 2004. ISSN 0377-2217. Citado na página 48.

LEE, I.; SIKORA, R.; SHAW, M. J. A genetic algorithm-based approach to flexible flow-line scheduling with variable lot sizes. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 27, n. 1, p. 36–54, Feb 1997. ISSN 1083-4419. Citado na página 48.

LEU, S.-S.; HWANG, S.-T. Ga-based resource-constrained flow-shop scheduling model for mixed precast production. *Automation in Construction*, v. 11, n. 4, p. 439 – 452, 2002. ISSN 0926-5805. Citado na página 48.

LEWIS, F. L.; HORNE, B. G.; ABDALLAH, C. T. Computational complexity of determining resource loops in re-entrant flow lines. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, v. 30, n. 2, p. 222–229, 2000. ISSN 1083-4427. Citado 3 vezes nas páginas 43, 48 e 50.

LIN, D.-Y.; CHU, Y.-M. The mixed-product assembly line sequencing problem of a door-lock company in taiwan. *Computers and Industrial Engineering*, v. 64, n. 1, p. 492 – 499, 2013. ISSN 0360-8352. Citado na página 55.

____. A lagrangian relaxation approach to the mixed-product assembly line sequencing problem: A case study of a door-lock company in taiwan. *Applied Mathematical Modelling*, v. 38, n. 17, p. 4493 – 4511, 2014. ISSN 0307-904X. Citado 2 vezes nas páginas 55 e 59.

LOBATO, F. S. *Otimização Multi-objetivo para o projeto de sistemas de engenharia*. Tese (Doutorado) — Universidade Federal de Uberlândia, Uberlândia, 9 2008. Citado 4 vezes nas páginas 71, 73, 74 e 75.

LUCERTINI, M.; PACCIARELLI, D.; PACIFICI, A. Modeling an assembly line for configuration and flow management. *Computer Integrated Manufacturing Systems*, v. 11, n. 1, p. 15 – 24, 1998. ISSN 0951-5240. Citado na página 48.

MELLOY, B. J.; SOYSTER, A. L.; CHANDRA, M. J. An observation on storage allocation in a flow line with a common buffer. *European Journal of Operational Research*, v. 46, n. 3, p. 388 – 392, 1990. ISSN 0377-2217. Citado na página 48.

MUSHARAVATI, F.; HAMOUDA, A. M. S. Multiple parts process planning in serial-parallel flexible flow lines: part i—process plan modeling framework. *The International Journal of Advanced Manufacturing Technology*, v. 78, n. 1, p. 115–137, 2015. ISSN 1433-3015. Citado 2 vezes nas páginas 41 e 48.

NADERI, B.; ZANDIEH, M.; GHOMI, S. M. T. F. A study on integrating sequence dependent setup time flexible flow lines and preventive maintenance scheduling. *Journal of Intelligent Manufacturing*, v. 20, n. 6, p. 683, 2008. ISSN 1572-8145. Citado na página 48.

PALANIAPPAN, N. J. P. Integration of procurement and production scheduling in flexible flow-line assembly. *International Journal of Integrated Supply Management*, v. 5, n. 4, p. 344 – 364, out. 2010. ISSN 0364-765X. Citado na página 48.

PALANIAPPAN, P.; JAWAHAR, N. A genetic algorithm for simultaneous optimisation of lot sizing and scheduling in a flow line assembly. *International Journal of Production Research*, v. 49, n. 2, p. 375–400, 2011. Citado 2 vezes nas páginas 48 e 71.

PARK, T.; STEUDEL, H. J. Analysis of heuristics for job sequencing in manufacturing flow line work-cells. *Computers and Industrial Engineering*, v. 20, n. 1, p. 129–140, 1991. ISSN 0360-8352. Citado 2 vezes nas páginas 41 e 48.

PAULA, H. Martins de. *Uso de Suspensões preparadas com semente de moringa oleifera associada a coagulantes químicos no tratamento da água residuária de usinas de concreto*. Tese (Doutorado) — Faculdade Estadual de Engenharia Cívil, Universidade Estadual de Campinas, 12 2014. Citado na página 41.

PINEDO, M. L. *Scheduling: Theory, Algorithms, and Systems*. 5rd. ed. New York, NY: Springer International Publishing, 2016. ISBN 978-3-319-26580-3. Citado 7 vezes nas páginas 29, 30, 35, 38, 39, 40 e 41.

PIRAMUTHU, S.; RAMAN, N.; SHAW, M. J. Learning-based scheduling in a flexible manufacturing flow line. *IEEE Transactions on Engineering Management*, v. 41, n. 2, p. 172–182, May 1994. ISSN 0018-9391. Citado 2 vezes nas páginas 41 e 48.

QUADT, D.; KUHN, H. A taxonomy of flexible flow line scheduling procedures. *European Journal of Operational Research*, v. 178, n. 3, p. 686 – 698, 2007. ISSN 0377-2217. Citado 6 vezes nas páginas 29, 48, 51, 52, 62 e 89.

ROSSI, A.; LANZETTA, M. Scheduling flow lines with buffers by ant colony digraph. *Expert Systems with Applications*, v. 40, n. 9, p. 3328 – 3340, 2013. ISSN 0957-4174. Citado 2 vezes nas páginas 41 e 48.

SAIF, U. *et al.* Multi-objective artificial bee colony algorithm for simultaneous sequencing and balancing of mixed model assembly line. *International Journal of Advanced Manufacturing Technology*, v. 75, n. 9-12, p. 1809 – 1827, 2014. ISSN 02683768. Citado na página 48.

SALVADOR, M. S. A solution to a special class of flow shop scheduling problems. In: ____. *Symposium on the Theory of Scheduling and Its Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1973. p. 83–91. ISBN 978-3-642-80784-8. Citado na página 37.

SARAMAGO, S. F. P. *Métodos de Otimização Randômica: Algoritmos Genéticos e Simulated Annealing*. São Carlos, SP: Sociedade Brasileira de Matemática Aplicada e Computacional, 1999. v. 6. Citado 3 vezes nas páginas 72, 73 e 75.

SAWIK, T. Mixed integer programming for scheduling flexible flow lines with limited intermediate buffers. *Mathematical and Computer Modelling*, v. 31, n. 13, p. 39 – 52, 2000. ISSN 0895-7177. Citado 2 vezes nas páginas 31 e 48.

SHAO, X. *et al.* Metaheuristic approaches to sequencing mixed-model fabrication/assembly systems with two objectives. *The International Journal of Advanced Manufacturing Technology*, v. 48, n. 9, p. 1159–1171, Jun 2010. ISSN 1433-3015. Citado 2 vezes nas páginas 55 e 58.

SHEIKH, S. Multi-objective flexible flow lines with due window, time lag, and job rejection. *The International Journal of Advanced Manufacturing Technology*, v. 64, n. 9, p. 1423–1433, 2013. ISSN 1433-3015. Citado 3 vezes nas páginas 48, 53 e 92.

SIKORA, R. A genetic algorithm for integrating lot-sizing and sequencing in scheduling a capacitated flow line. *Computers and Industrial Engineering*, v. 30, n. 4, p. 969 – 981, 1996. ISSN 0360-8352. Citado na página 48.

SIKORA, R.; CHHAJED, D.; SHAW, M. J. Integrating the lot-sizing and sequencing decisions for scheduling a capacitated flow line. *Computers and Industrial Engineering*, v. 30, n. 4, p. 659 – 679, 1996. ISSN 0360-8352. Citado 3 vezes nas páginas 43, 48 e 49.

SLACK, N.; BRANDON-JONES, A.; JOHNSTON, R. *Operations Management*. 7th. ed. London, UK: Pearson, 2014. ISBN 9780273776208. Citado 2 vezes nas páginas 30 e 36.

STADTLER, H.; SAHLING, F. A lot-sizing and scheduling model for multi-stage flow lines with zero lead times. *European Journal of Operational Research*, v. 225, n. 3, p. 404 – 419, 2013. ISSN 0377-2217. Citado na página 48.

TAVAKKOLI-MOGHADDAM, R.; SAFAEI, N.; SASSANI, F. A memetic algorithm for the flexible flow line scheduling problem with processor blocking. *Computers and Operations Research*, v. 36, n. 2, p. 402 – 414, 2009. ISSN 0305-0548. Scheduling for Modern Manufacturing, Logistics, and Supply Chains. Citado 3 vezes nas páginas 48, 52 e 53.

THOMOPOULOS, N. T. Line balancing-sequencing for mixed-model assembly. *Management Science*, v. 14, n. 2, p. B–59–B–75, 1967. Citado 4 vezes nas páginas 54, 55, 56 e 62.

VANDERPLAATS, G. *Numerical Optimization Techniques for Engineering Design*. Colorado Springs, CO: Vanderplaats Research and Development, Incorporated, 1999. ISBN 9780944956007. Citado na página 76.

WITTROCK, R. J. Scheduling algorithms for flexible flow lines. *IBM Journal of Research and Development*, v. 29, n. 4, p. 401–412, July 1985. ISSN 0018-8646. Citado 2 vezes nas páginas 47 e 48.

____. An adaptable scheduling algorithm for flexible flow lines. *Operations Research*, v. 36, n. 3, p. 445–453, 1988. Citado 3 vezes nas páginas 29, 37 e 48.

YU, A. J.; SEIF, J. Minimizing tardiness and maintenance costs in flow shop scheduling by a lower-bound-based ga. *Computers and Industrial Engineering*, v. 97, p. 26 – 40, 2016. ISSN 0360-8352. Citado 2 vezes nas páginas 48 e 71.

ZADE, A. E.; A, F.; BAGHER, M. A dynamic genetic algorithm for solving a single machine scheduling problem with periodic maintenance. *ISRN Industrial Engineering*, 2013. Citado na página 48.

ZANDIEH, M.; MOZAFFARI, E.; GHOLAMI, M. A robust genetic algorithm for scheduling realistic hybrid flexible flow line problems. *Journal of Intelligent Manufacturing*, v. 21, n. 6, p. 731–743, 2010. ISSN 1572-8145. Citado 2 vezes nas páginas 41 e 48.

APÊNDICE A

SOLVER GUROBI

```
1  //Copyright (c) 2018, Jeferson Silva Martins
2  //Mestrado em Modelagem e Otimização
3  //
4  //Código do Projeto: GUFLM
5  //Título: Mixed-Model Flow Line with Exact Gurobi Optmization
6  //Publisher: jefersonsilvam
7  //
8  //Developers: S. M., Jeferson (Estudante de Mestrado)
9  //Contato: jsm.ctl@gmail.com
10
11 //1 - incluir a biblioteca do GUROBI com as funções para otimização
12 #include "gurobi_c++.h"
13
14 //2 - incluir bibliotecas do C++ para demais funções: entrada/saída
       ; manipulação de arquivos; etc
15 #include <iostream>//manipulação de entrada e saida
16 #include <iomanip>//formatação de saida de dados
17 #include <fstream>//manipulacao de arquivos
18 #include <sstream>//manipular cadeia de caracteres - strings
19 #include <cstring>//manipula strings no formato da linguagem C
20
21 // evitar usar o nome std ao usar funções da biblioteca padrão
22 using namespace std;
23 //funcao para trasnformar um int em string
24 string itos(int i){
25     stringstream s;
26     s << i;
27     return s.str();
28 }
29
30 //3 - criar a função principal do programa
31 int main(int argc, char* argv[]){
32     try{//tratar/capturar exceções, como divisão por zero, acesso a
```

```
        índice não definido em vetor, etc.
33
34      //4 - declarar as variáveis necessária para o programa
35      int i, j, k, n, m, r; //indice para vetores
36      int min, max;
37      int N; //Quantidade de Tarefas
38      int M; //Quantidade de Estações
39      int R; //Quantidade de Familia de Modelos
40      char nome[50];//nome da instancia MMS
41      int* d; //Vetor de demanda de cada Familia
42      int** p; //Matriz de Processamento de Familia
43
44      //4.1 - Define o ambiente do GUROBI
45      GRBEnv env = GRBEnv();
46
47      //4.2 - Cria a variável que vai armazenar o modelo e a
            inicializa com o ambiente 'env'
48      GRBModel model = GRBModel(env);
49
50      //4.3 - Ler os dados do arquivo de entrada
51      argc--; argv++;
52      if (argc < 1){
53          cerr << "Como usar o programa: ./ex instancia.txt" << endl;
54          return 1;
55      }
56      //arquivo de entrada que vai conter o dados.txt
57      ifstream arq(argv[0], ios::in);
58      //vai abrir MMS3103.txt para o modo read/leitura
59      if (!arq){
60          cerr << "\n**Erro: abrir arquivo de entrada**\n" << endl;
61          exit(1);
62      }
63      //4.4 - Os dados lidos do arquivo são atribuidos as variáveis
            do programa
64      arq >> N; //leitura da 1a linha do arquivo
65      arq >> M; //leitura da 2a linha do arquivo
66      arq >> R; //leitura da 3a linha do arquivo
67
68      //inicializar a variável de Demanda
69      d = new int[R]; // R produtos ok
70
71      //inicializar a matriz de processamento
72      p = new int*[R]; //aloca as linhas da matriz p[r][m]
73      for (i = 0; i < R; i++){
74          p[i] = new int[M]; //aloca cada coluna da matriz p[r][m]
75      }
76
```

```
77      //Leitura da demanda Dr
78      for (j = 0; j < R; j++){
79          arq >> d[j];
80      }
81
82      //leitura do processamento Prm
83      for (i = 0; i < R; i++){
84          for (j = 0; j < M; j++){
85              arq >> p[i][j];
86          }
87      }
88
89      arq >> nome;//leitura do nome na ultima linha do arquivo
90      //printf("%s",nome);
91      arq.close(); //fecha o arquivo, ja que leu todos os dados
92
93      //4.5 - Cria as variáveis do problema de otimização
94      //O método addVar tem os argumentos:
95      //Limitante inferior da variável: qualquer número real,
96      //Limitante superior da variável: qualquer número real maior do
            que o limitante inferior. Para infinito usar GRB_INFINITY
97      //Coeficiente na função objetivo: pode setar com zero, para
            depois definir a Função Objetivo, OU já pode setar o valor.
98      //Tipo da variável: GRB_BINARY = {0, 1}, GRB_INTEGER = valores
            inteiros, GRB_CONTINUOUS = número reais
99      //Nome da variável: qualquer nome que deseja atribuir,
            geralmente o mesmo do modelo
100
101     //variaveis de c[j]
102     GRBVar* c = new GRBVar[N + M - 1];
103     //informa como é cada variavel do problema
104     max=0;
105     min=1000000;
106     for (i = 0; i < R; i++){
107         for (j = 0; j < M; j++){
108             if (max < p[i][j]){
109                 max = p[i][j];
110             }
111             if (min > p[i][j]){
112                 min = p[i][j];
113             }
114         }
115     }
116     //printf("\n min=%d, max=%d\n",min, max);
117     for (j = 0; j < N + M - 1; j++){
118         c[j] = model.addVar(min, max, max, GRB_INTEGER, "c[" + itos
                (j + 1) + "]");
```

```
119         }
120
121         //variaveis do tipo x[r][n]
122         GRBVar** x = new GRBVar*[R];
123         for (i = 0; i < R; i++){
124             x[i] = new GRBVar[N];
125         }
126
127         for (i = 0; i < R; i++){
128             for (j = 0; j < N; j++){
129                 x[i][j] = model.addVar(0.0, 1.0, 0.0, GRB_BINARY, "x["
                        + itos(i + 1) + "][" + itos(j + 1) + "]");
130                 //printf("x[%d][%d]\n",i,j);
131             }
132         }
133
134         //Após definir as variáveis, é preciso atualizar o modelo
                SEMPRE
135         model.update();
136
137         //5 - Define a função objetivo do problema, caso não tenha
                usado valor ZERO no addVar. Tem os argumentos
138         //Expressão da função objetivo propriamente dita,
139         //Critério do objetivo: Maximizar = GRB_MAXIMIZE, para
                Minimizar = GRB_MINIMIZE
140         //criamos uma expressão linear para representar a funcao
                objetivo
141
142
143         //SUM_{j=1 ate N+M-1} c[j]
144         GRBLinExpr obj = 0.0;
145         for (j = 0; j < N + M - 1; j++){
146             obj = obj + c[j];
147             //printf("c%d\n",j);
148         }
149
150         //setar o objetivo
151         model.setObjective(obj, GRB_MINIMIZE);
152
153         //6 - Adiciona as restrições do modelo
154         //6.1 - A 1a conjunto de restrição: SUM_{r=1 ate R} x[rn] = 1,
                n=1,...,N
155         //O primeiro argumento é a expressão da restrição
156         //O segundo argumento é o nome da restrição que você vai dar
157
158         for (n = 0; n < N; n++){
159             GRBLinExpr rest1 = 0.0;
```

```
160             for (r = 0; r < R; r++){
161                 rest1 = rest1 + x[r][n];
162             }
163             model.addConstr(rest1 == 1, "rest1 para n=" + itos(n + 1));
164         }
165 
166 
167         //6.2 - A 2a restrição: SUM_{n=1 ate N} x[rn] = D[r], r=1,...,R
168         for (r = 0; r < R; r++){
169             GRBLinExpr rest2 = 0.0;
170             for (n = 0; n < N; n++){
171                 rest2 = rest2 + x[r][n];
172             }
173             model.addConstr(rest2 == d[r], "rest2 para r=" + itos(r +
                    1));
174         }
175 
176         //6.3 - A 3a restrição: c[j]>=SUM_{r=1 ate R} p[rk]*x[r(j-k+1)
                ], para todo j,k (j=1 até M-1 e k=1 ate j) (j=M ate N e k=1
                 ate M) (j=n+1 até N+m-1 e k=j-N+1 ate M)
177         for (j = 1; j <= M - 1; j++){
178             for (k = 1; k <= j; k++){
179                 GRBLinExpr rest3 = 0.0;
180                 for (r = 1; r <= R; r++){
181                     rest3 = rest3 + (p[r-1][k-1] * x[r-1][j - k]);
182                 }
183                 model.addConstr(c[j-1] >= rest3, "rest3 para j=" + itos
                        (j) + "e k=" + itos(k));
184             }
185         }
186 
187 
188         for (j = M; j <= N; j++){
189             for (k = 1; k <= M; k++){
190                 GRBLinExpr rest3 = 0.0;
191                 for (r = 1; r <= R; r++){
192                     rest3 = rest3 + (p[r-1][k-1] * x[r-1][j - k]);
193                 }
194                 model.addConstr(c[j-1] >= rest3, "rest3 para j=" + itos
                        (j) + "e k=" + itos(k));
195             }
196         }
197 
198         for (j = N+1; j <= N + M - 1; j++){
199             for (k = j - N + 1; k <= M; k++){
200                 GRBLinExpr rest3 = 0.0;
201                 for (r = 1; r <= R; r++){
```

```
202                     rest3 = rest3 + (p[r-1][k-1] * x[r-1][j - k]);
203                 }
204                 model.addConstr(c[j-1] >= rest3, "rest3 para j=" + itos
                        (j) + "e k=" + itos(k));
205             }
206         }
207 //return 0;
208 
209         //Após inserir as restrições, atualizar o modelo
210         model.update();
211 
212         //7 - Chama o método para otimizar o modelo
213         //especifica quanto tempo deverá ser gasto para otimizar - 3600
                s = 1h
214         model.getEnv().set(GRB_DoubleParam_TimeLimit, 7200);
215         //O GUROBI escolhe automaticamente o algoritmo a usar, entre
                eles: primal simplex, dual simplex, etc.
216         model.optimize();
217 
218         //8 - Após a otimização, imprime o resultado encontrado para o
                usuário
219         //Calibra a saida de dados para imprimir reais com 4 casas
                decimais
220         //ios::fixed --  mosta o numero com ponto fixo
221         //ios::showpoint -- mostra a parte fracionária do número
222         //ios::left -- imprime os valores alinhados a esquerda
223         //setprecision(3) -- estabelece o número de casas decimais, que
                é 3 neste caso
224         cout << setiosflags(ios::fixed | ios::showpoint | ios::left) <<
                setprecision(3) << endl << endl;
225 
226         //8.1 - Imprime o valor das variáveis para o usuário na tela
227         //O método .get permite obter as informações dependendo do parâ
                metro usado:
228         //GRB_DoubleAttr_X =  retorna o valor da variável
229 
230         cout <<  "\nRESULTADO DO FLM:  " << nome << endl;
231 
232         //imprime as variaveis cj
233         for (j = 0; j < N + M - 1; j++)
234         {
235             cout << "Ciclo c[" + itos(j + 1) + "]: " << c[j].get(
                    GRB_DoubleAttr_X) << endl;
236         }
237 
238         //imprime as variaveis xrn
239         for (r = 0; r < R; r++)
```

```
240     {
241         for (n = 0; n < N; n++)
242         {
243             cout << "x[" + itos(r + 1) + "][" + itos(n + 1) + "]: "
                    << x[r][n].get(GRB_DoubleAttr_X) << endl;
244         }
245     }
246
247     //imprime a ordem de Execução dos Produtos
248     printf("\n");
249     for (j = 1; j <= M - 1; j++){
250         printf("==================================\n");
251         for (k = 1; k <= j; k++){
252             for (r = 1; r <= R; r++){
253                 if (x[r-1][j-k].get(GRB_DoubleAttr_X) == 1){
254                     printf("No Ciclo C[%d] o produto %d é
                            processado na maquina %d em %d tempos\n",j,
                             r, k, p[r-1][k-1]);
255                 }
256             }
257         }
258     }
259
260
261     for (j = M; j <= N; j++){
262         printf("==================================\n");
263         for (k = 1; k <= M; k++){
264             for (r = 1; r <= R; r++){
265                 if (x[r-1][j-k].get(GRB_DoubleAttr_X) == 1){
266                     printf("No Ciclo C[%d] o produto %d é
                            processado na maquina %d em %d tempos\n",j,
                             r, k, p[r-1][k-1]);
267                 }
268             }
269         }
270     }
271
272     for (j = N+1; j <= N + M - 1; j++){
273         printf("==================================\n");
274         for (k = j - N + 1; k <= M; k++){
275             for (r = 1; r <= R; r++){
276                 if (x[r-1][j-k].get(GRB_DoubleAttr_X) == 1){
277                     printf("No Ciclo C[%d] o produto %d é
                            processado na maquina %d em %d tempos\n",j,
                             r, k, p[r-1][k-1]);
278                 }
279             }
```

```
280         }
281     }
282
283
284     //8.2 - Imprime o valor da função objetivo
285     //GRB_DoubleAttr_ObjVal = retorna o valor da função objetivo
286     cout << "\n\nValor da Funcao Objetivo: " << model.get(
            GRB_DoubleAttr_ObjVal) << endl;
287
288     //9 - Imprimir outras informações do modelo
289     //Para maiores detalhes sobre parâmetros, procure no arquivo "
            refman.pdf"
290     //Para conhecer melhor o funcionamento do GUROBI, procure no
            arquivo "quickstart.pdf"
291     //Para ver outros exemplos usando o GUROBI, veja o arquivo "
            examples.pdf"
292
293     //Para imprimir mais dados do modelo
294     cout << endl << "\n------Mais informacoes sobre o Modelo------\
            n" << endl;
295     cout << "Status da solucao: (2 - otimo) | (3 - inviavel) | (5 -
             ilimitado): "  << model.get(GRB_IntAttr_Status) << endl;
296     cout << "Tempo de otimização: "  << model.get(
            GRB_DoubleAttr_Runtime) << " segundos" << endl;
297     cout << "Numero total de variaveis: "  << model.get(
            GRB_IntAttr_NumVars) << endl;
298                     //cout << "--Numero de variaveis Inteiras: "
                            << model.get(GRB_IntAttr_NumIntVars) <<
                            endl;
299     cout << "--Numero de variaveis Binarias: "  << model.get(
            GRB_IntAttr_NumBinVars) << endl;
300     cout << "Numero de nos da arvore B&B explorados: " <<
            setprecision(0) << model.get(GRB_DoubleAttr_NodeCount) <<
            endl;
301
302     //11 - Libera a memoria alocada
303     delete[] c;
304     for (i = 0; i < N; i++)
305         delete[] x[i];
306     delete[] x;
307     for (j = 0; j < M; j++)
308         delete[] p[j];
309     delete[] p;
310
311     }
312     //Tratamento de excessões, caso ocorra algum erro durante a
            exeução da função main
```

```
313     catch (GRBException e)
314     {
315         cout << "Erro no código = " << e.getErrorCode() << endl;
316         cout << e.getMessage() << endl;
317     }
318     catch (...)
319     {
320         cout << "Exceção durante a otimização!!!" << endl;
321     }
322
323     //Programa finalizou adequadamente
324     return 0;
325     }//fim do int main
```

APÊNDICE B

ALGORTIMO LAGRANGEANO RELAXADO

```
1  //Copyright (c) 2018, Jeferson Silva Martins
2  //Mestrado em Modelagem e Otimização
3  //
4  //Código do Projeto: LRFLM
5  //Título: Mixed-Model Flow Line with Exact Gurobi Optmization
6  //Publisher: jefersonsilvam
7  //
8  //Developers: S. M., Jeferson (Estudante de Mestrado)
9  //Contato: jsm.ctl@gmail.com
10
11 //incluir a biblioteca do GUROBI com as funções para otimização
12 #include "gurobi_c++.h"
13
14 //incluir bibliotecas do C++ para demais funções: entrada/saída;
       manipulação de arquivos; etc
15 #include <iostream>//manipulação de entrada e saida
16 #include <iomanip>//formatação de saida de dados
17 #include <fstream>//manipulacao de arquivos
18 #include <sstream>//manipular cadeia de caracteres - strings
19 #include <cstring>//manipula strings no formato da linguagem C
20 #include <math.h> //manupula o uso do ceil
21 #include <time.h>
22 #include <utime.h>
23 #include <sys/times.h>
24 #include <complex>
25
26 // evitar usar o nome std ao usar funções da biblioteca padrão
27 using namespace std;
28
29 //funcao para trasnformar um int em string
30 string itos(int i){
```

```
31      stringstream s;
32      s << i;
33      return s.str();
34  }
35
36  double** gur1(double** q, double** y, int** p, int* d, int N, int M
        , int R, GRBEnv env);
37  double* gur2(double** y, int** p, double* z, int* d, int N, int M,
        int R, GRBEnv env);
38  double gur3(double* cb, double** xb, int** p, int* d, int N, int M,
         int R, int UB, double lb, GRBEnv env);
39  double upperbound(double** x, int M, int N, int R, int** p);
40
41  double* z;
42  double** q;
43
44  //criar a função principal do programa
45  int main(int argc, char* argv[]){
46
47      //criar o Ambiente Gurobi
48      GRBEnv env = GRBEnv();
49
50      //declarar as variáveis necessária para o programa
51      int t, NumIt, it, num;
52      int* d, *ub;
53      double multiplier, divi, A, UB, s, yjk, divi1, LBO, lb;
54      double* lr2, *lr1, *lr, *a, *LB, *c, *cb;
55      double** y, **Y, **x, **xb;
56      int **p;
57
58      int i, j, k, n, m, r,b; //indice para vetores
59      int min, max;
60      int N; //Quantidade de Tarefas
61      int M; //Quantidade de Estações
62      int R; //Quantidade de Familia de Modelos
63      char nome[50];//nome da instancia MMS
64
65      //marcar o tempo gasto
66      float inicio, fim, tempoT;
67
68      //Ler os dados do arquivo de entrada
69      argc--; argv++;
70      if (argc < 1){
71          cerr << "Como usar o programa: ./ex instancia.txt" << endl;
72          return 1;
73      }
74      //arquivo de entrada que vai conter o dados.txt
```

```
75      ifstream arq(argv[0], ios::in);
76      //vai abrir MMS3103.txt para o modo read/leitura
77      if (!arq){
78          cerr << "\n**Erro: abrir arquivo de entrada**\n" << endl;
79          exit(1);
80      }
81      //Os dados lidos do arquivo são atribuidos as variáveis do
            programa
82      arq >> N; //leitura da 1a linha do arquivo
83      arq >> M; //leitura da 2a linha do arquivo
84      arq >> R; //leitura da 3a linha do arquivo
85
86      //inicializar a variável de Demanda
87      d = new int[R];// R produtos ok
88      c = new double[M+N]; //vetor c
89
90      //Leitura da demanda Dr
91      for (j = 0; j < R; j++){
92          arq >> d[j];
93      }
94
95      //inicializar a matriz de processamento
96      p = new int*[R]; //aloca as linhas da matriz p[r][m]
97      for (i = 0; i < R; i++){
98          p[i] = new int[M]; //aloca cada coluna da matriz p[r][m]
99      }
100
101     //leitura do processamento Prm
102     for (i = 0; i < R; i++){
103         for (j = 0; j < M; j++){
104             arq >> p[i][j];
105         }
106     }
107
108     arq >> nome;//leitura do nome na ultima linha do arquivo
109     //printf("%s\n", nome);
110     arq.close(); //fecha o arquivo, ja que leu todos os dados
111
112     fflush(stdout);
113     // Inicializar o tamanho das iterações
114     if (N <= 30){
115         it = 400;
116     }
117     if (N > 30 && N <= 60){
118         it = 800;
119     }
120     if (N > 60 && N <= 120){
```

```
121             it = 1200;
122         }
123
124         //inicializar x
125         x = new double*[R+1];
126         for (i = 0; i < R+1; i++){
127             x[i] = new double[N];
128         }
129
130         for (i = 0; i < R+1; i++){
131             for (j = 0; j < N; j++){
132                 x[i][j] = 0.0;
133             }
134         }
135
136
137         //inicializar alpha, LR, LR1, LR2
138         a = new double[it];
139         lr = new double[it];
140         lr1 = new double[it];
141         lr2 = new double[it];
142         ub = new int[it];
143         //UB = new double[it];
144         LB = new double[it];
145
146         //inicializar variaveis de retorno
147         q = new double*[R+1];
148         for (i = 0; i < R+1; i++){
149             q[i] = new double[N];
150         }
151
152         for (i = 0; i < R+1; i++){
153             for (j = 0; j < N; j++){
154                 q[i][j]=0.0;
155             }
156         }
157
158         z= new double[M+N];
159
160         for (i = 0; i < M+N; i++){
161             z[i]=0.0;
162         }
163
164
165         //variaveis de valor ótimo
166         cb = new double[N + M - 1];
167
```

```
168         xb = new double*[R];
169         for (i = 0; i < R; i++){
170             xb[i] = new double[N];
171         }
172 
173         //Passo 1
174         //parametros iniciais
175         a[0] = 0.5;
176         UB = 10000;
177         NumIt = 0;
178         multiplier = 0.5;
179         num = 0;
180 
181 
182         //inicializar a matriz y do lamba
183         y = new double*[N+M-1];
184         for (i = 0; i < N+M-1; i++){
185             y[i] = new double[M];
186         }
187 
188 
189         for (i = 0; i < N+M-1; i++){
190             for (j = 0; j < M; j++){
191                 y[i][j]=0.0;
192             }
193         }
194 
195 
196         //Atribuir valor inicial ao lambda
197         for (j = 1; j <= M - 1; j++){
198             for (k = 1; k <= j; k++){
199                 y[j-1][k-1] = (1-0.01)/M;
200                 //y[j-1][k-1] = 0.9;
201             }
202         }
203 
204 
205         for (j = M; j <= N; j++){
206             for (k = 1; k <= M; k++){
207                 y[j-1][k-1] = (1-0.01)/M;
208                 //y[j-1][k-1] = 0.9;
209             }
210         }
211 
212         for (j = N+1; j <= N + M - 1; j++){
213             for (k = j - N + 1; k <= M; k++){
214                 y[j-1][k-1] = (1-0.01)/M;
```

```
215                 //y[j-1][k-1] = 0.9;
216             }
217         }
218
219         //Dispara o cronometro
220         inicio = clock();
221
222         printf("-------------------------------------------------\n");
223         for (t = 0; t < it; t++){
224             printf("######ITERATION %d#########\n", t+1);
225
226             //GUROBI em LR1
227             x = gur1(q, y, p, d, N, M, R, env);
228
229             lr1[t]=x[R][0];
230
231             //GUROBI em LR2
232             c = gur2(y, p, z, d, N, M, R, env);
233
234             lr2[t]=c[M+N-1];
235
236
237             LB[t] = lr1[t] + lr2[t];
238
239             //Passo 3
240             //Passo 3.1
241             ub[t] = upperbound(x, M, N, R, p);
242
243             UB = fmin(ub[t], UB);
244
245
246             //Passo 4
247             if (ceil(LB[t]) == UB){
248
249                 //finaliza o cronometro
250                 fim = clock();
251                 tempoT=(((float)(fim-inicio))/CLOCKS_PER_SEC);
252                 printf("
                        -------------------------------------------------\n"
                    );
253                 printf("RESULTADO DO FLM:  %s\n", nome);
254                 printf("Tempo de Otimização: %.4f segundos\n",tempoT);
255                  printf("LB: %.2f\n",LB[t]);
256                 printf("UB: %.2f\n",UB);
257
258                 //imprime as variaveis Cj
259                 for (j = 1; j <= N + M - 1; j++){
```

```
260                 printf("\nC=[%d]: %.2f", j-1, c[j-1]);
261             }
262             printf("\n");
263 
264             //imprime as variaveis xrn
265             for (r = 0; r < R; r++){
266                 for (n = 0; n < N; n++){
267                     printf("\nx[%d][%d]: %.2f",r + 1, n + 1, x[r][n
                            ]);
268                 }
269             }
270 
271             printf("\n\nCmax: %.2f\n", UB);
272             //imprime a ordem de Execução dos Produtos
273             printf("\n");
274             for (j = 1; j <= M - 1; j++){
275                 printf("==================================\n");
276                 for (k = 1; k <= j; k++){
277                     for (r = 1; r <= R; r++){
278                         if (x[r-1][j-k] == 1){
279                             printf("No Ciclo C[%d] o produto %d é
                                    processado na maquina %d em %d
                                    tempos\n",j, r, k, p[r-1][k-1]);
280                         }
281                     }
282                 }
283             }
284 
285 
286             for (j = M; j <= N; j++){
287                 printf("==================================\n");
288                 for (k = 1; k <= M; k++){
289                     for (r = 1; r <= R; r++){
290                         if (x[r-1][j-k] == 1){
291                             printf("No Ciclo C[%d] o produto %d é
                                    processado na maquina %d em %d
                                    tempos\n",j, r, k, p[r-1][k-1]);
292                         }
293                     }
294                 }
295             }
296 
297             for (j = N+1; j <= N + M - 1; j++){
298                 printf("==================================\n");
299                 for (k = j - N + 1; k <= M; k++){
300                     for (r = 1; r <= R; r++){
301                         if (x[r-1][j-k] == 1){
```

```
302                                 printf("No Ciclo C[%d] o produto %d é
                                        processado na maquina %d em %d
                                        tempos\n",j, r, k, p[r-1][k-1]);
303                             }
304                         }
305                     }
306                 }
307                 exit(0);
308             }
309
310             //Determinação do Passo
311             divi = 0.0;
312             for (j = 1; j <= M - 1; j++){
313                 for (k = 1; k <= j; k++){
314                     A = 0.0;
315                     for (r = 1; r <= R; r++){
316                         A += (p[r-1][k-1] * x[r-1][j-k]);
317                     }
318                     divi += y[j-1][k-1]*(pow((A - c[j-1]), 2));
319                 }
320             }
321
322             for (j = M; j <= N; j++){
323                 for (k = 1; k <= M; k++){
324                     A = 0.0;
325                     for (r = 1; r <= R; r++){
326                         A += (p[r-1][k-1] * x[r-1][j-k]);
327                     }
328                     divi += y[j-1][k-1]*(pow((A - c[j-1]), 2));
329                 }
330             }
331
332             for (j = N+1; j <= N + M - 1; j++){
333                 for (k = j - N + 1; k <= M; k++){
334                     A = 0.0;
335                     for (r = 1; r <= R; r++){
336                         A += (p[r-1][k-1] * x[r-1][j-k]);
337                     }
338                     divi += y[j-1][k-1]*(pow((A - c[j-1]), 2));
339                 }
340             }
341
342             s = a[t] * (UB - LB[t]) / divi;
343
344             //Multipliers Update
345             for (j = 1; j <= M - 1; j++){
346                 for (k = 1; k <= j; k++){
```

```
347                 A = 0.0;
348                 for (r = 1; r <= R; r++){
349                     A = A + (p[r-1][k-1] * x[r-1][j-k]);
350                 }
351                 yjk = y[j-1][k-1] + s * (A - c[j-1]);
352                 y[j-1][k-1] = fmax(0, yjk);
353             }
354         }
355 
356         for (j = M; j <= N; j++){
357             for (k = 1; k <= M; k++){
358                 A = 0.0;
359                 for (r = 1; r <= R; r++){
360                     A = A + (p[r-1][k-1] * x[r-1][j-k]);
361                 }
362                 yjk = y[j-1][k-1] + s * (A - c[j-1]);
363                 y[j-1][k-1] = fmax(0, yjk);
364             }
365         }
366 
367         for (j = N+1; j <= N + M - 1; j++){
368             for (k = j - N + 1; k <= M; k++){
369                 A = 0.0;
370                 for (r = 1; r <= R; r++){
371                     A = A + (p[r-1][k-1] * x[r-1][j-k]);
372                 }
373                 yjk = y[j-1][k-1] + s * (A - c[j-1]);
374                 y[j-1][k-1] = fmax(0, yjk);
375             }
376         }
377 
378 
379         if (t >= 1 && LB[t] <= LB[t-1]){
380             NumIt += 1;
381             num += 1;
382         }
383 
384         if ((NumIt == 20 && N <= 30) || (NumIt == 30 && N > 30 && N
                <= 60) || (NumIt == 40 && N > 60 && N <= 120)){
385             a[t+1] = a[t] * multiplier;
386             NumIt = 0;
387         }
388         else{
389             a[t+1] = a[t];
390         }
391 
392         fim = clock();
```

```
393             tempoT=(((float)(fim-inicio))/CLOCKS_PER_SEC);
394             printf("Tempo de Otimização: %.4f segundos\n",tempoT);
395             if ((t == it-1) && (ceil(LB[t]) != UB)){
396                 //passando os melhores valores das variaveis
397                 for (j = 0; j < N + M - 1; j++){
398                     cb[j] = c[j];
399                 }
400 
401                 for (i = 0; i < R; i++){
402                     for (j = 0; j < N; j++){
403                         xb[i][j] = x[i][j];
404                     }
405                 }
406 
407                 lb=ceil(LB[t]);
408                 printf("LB: %.2f\n",lb);
409                 printf("UB: %.2f\n",UB);
410                 printf("
                        ----------------------------------------------\n"
                    );
411                 printf("RESULTADO DO MMS:  %s\n", nome);
412                 LBO = gur3(cb, xb, p, d, N, M, R, UB, lb, env);
413 
414                 tempoT += LBO;
415                 printf("Tempo de Otimização: %.4f segundos\n",tempoT);
416                 exit(0);
417             }
418             fflush(stdout);
419         }
420 }
421 
422 double** gur1(double** q, double** y, int** p, int* d, int N, int M
        , int R, GRBEnv env){
423         int i, j, k, r, n;
424         GRBLinExpr obj1;
425 
426         for (i = 0; i < R+1; i++){
427             for (j = 0; j < N; j++){
428                 q[i][j]=0.0;
429             }
430         }
431 
432         //4.2 - Cria a variável que vai armazenar o modelo e a
            inicializa com o ambiente 'env'
433         GRBModel model = GRBModel(env);
434 
435         //variaveis do tipo x[r][n]
```

```
436     GRBVar** x = new GRBVar*[R];
437     for (i = 0; i < R; i++){
438         x[i] = new GRBVar[N];
439     }
440 
441     for (i = 0; i < R; i++){
442         for (j = 0; j < N; j++){
443             x[i][j] = model.addVar(0.0, 1.0, 0.0, GRB_INTEGER, "x["
                    + itos(i + 1) + "][" + itos(j + 1) + "]");
444         }
445     }
446 
447     //Após definir as variáveis, é preciso atualizar o modelo
        SEMPRE
448     model.update();
449 
450     GRBLinExpr obj = 0.0;
451     for (j = 1; j <= M - 1; j++){
452         for (k = 1; k <= j; k++){
453             obj1 = 0.0;
454             for (r = 1; r <= R; r++){
455                 obj1 += (p[r-1][k-1] * x[r-1][j - k]);
456             }
457             obj += y[j-1][k-1] * obj1;
458         }
459     }
460 
461     for (j = M; j <= N; j++){
462         for (k = 1; k <= M; k++){
463             obj1 = 0.0;
464             for (r = 1; r <= R; r++){
465                 obj1 += (p[r-1][k-1] * x[r-1][j - k]);
466             }
467             obj += y[j-1][k-1] * obj1;
468         }
469     }
470 
471     for (j = N+1; j <= N + M - 1; j++){
472         for (k = j - N + 1; k <= M; k++){
473             obj1 = 0.0;
474             for (r = 1; r <= R; r++){
475                 obj1 += (p[r-1][k-1] * x[r-1][j - k]);
476             }
477             obj += y[j-1][k-1] * obj1;
478         }
479     }
480 
```

```
481         //setar o objetivo
482         model.setObjective(obj, GRB_MINIMIZE);
483 
484         for (n = 0; n < N; n++){
485             GRBLinExpr rest1 = 0.0;
486             for (r = 0; r < R; r++){
487                 rest1 += x[r][n];
488             }
489             model.addConstr(rest1 == 1, "rest1 para n=" + itos(n + 1));
490         }
491 
492         for (r = 0; r < R; r++){
493             GRBLinExpr rest2 = 0.0;
494             for (n = 0; n < N; n++){
495                 rest2 += x[r][n];
496             }
497             model.addConstr(rest2 == d[r], "rest2 para r=" + itos(r +
                    1));
498         }
499 
500         //6.3 - A 3a restrição: c[j]>=SUM_{r=1 ate R} p[rk]*x[r(j-k+1)
                ], para todo j,k (j=1 até M-1 e k=1 ate j) (j=M ate N e k=1
                 ate M) (j=n+1 até N+m-1 e k=j-N+1 ate M)
501         for (r = 0; r < R; r++){
502             for (n = 0; n < N; n++){
503                 model.addConstr(x[r][n] <= 1, "rest3 para r=" + itos(r
                        + 1) + "e n="+ itos(n + 1));
504             }
505 
506         }
507         model.update();
508 
509         //7 - Chama o método para otimizar o modelo
510         //especifica quanto tempo deverá ser gasto para otimizar - 3600
                 s = 1h
511         model.getEnv().set(GRB_DoubleParam_TimeLimit, 7200);
512         //O GUROBI escolhe automaticamente o algoritmo a usar, entre
                eles: primal simplex, dual simplex, etc.
513         model.optimize();
514 
515         //8 - Após a otimização, imprime o resultado encontrado para o
                usuário
516         cout << setiosflags(ios::fixed | ios::showpoint | ios::left) <<
                 setprecision(3) << endl << endl;
517 
518         for (r = 0; r < R; r++){
519             for (n = 0; n < N; n++){
```

```
520             q[r][n] = x[r][n].get(GRB_DoubleAttr_X);
521             }
522         }
523
524         q[R][0]=model.get(GRB_DoubleAttr_ObjVal);
525
526
527         for (i = 0; i < R; i++){
528             delete[] x[i];
529         }
530         delete[] x;
531
532         return q;
533 }//fim do gurobi 1
534
535 double* gur2(double** y, int** p, double* z, int* d, int N, int M,
        int R, GRBEnv env){
536
537         //4 - declarar as variáveis necessária para o programa
538         int i, j, k, r, n;
539         double minr, min, max, maxk, maxr;
540
541         for (i = 0; i < M+N; i++){
542             z[i]=0.0;
543         }
544
545         //4.2 - Cria a variável que vai armazenar o modelo e a
                inicializa com o ambiente 'env'
546         GRBModel model = GRBModel(env);
547
548         //variaveis de c[j]
549         GRBVar* c = new GRBVar[N + M - 1];
550
551         //informa como é cada variavel do problema
552         max=0;
553         min=1000000;
554         for (i = 0; i < R; i++){
555             for (j = 0; j < M; j++){
556                 if (max < p[i][j]){
557                     max = p[i][j];
558                 }
559                 if (min > p[i][j]){
560                     min = p[i][j];
561                 }
562             }
563         }
564
```

```
565     for (j = 0; j < N + M - 1; j++){
566         c[j] = model.addVar(min, max, max, GRB_INTEGER, "c[" + itos
                (j + 1) + "]");
567     }
568
569     //Após definir as variáveis, é preciso atualizar o modelo
            SEMPRE
570     model.update();
571
572     //SUM_{j=1 ate N+M-1} c[j]
573     GRBLinExpr obj = 0.0;
574     GRBLinExpr obj1 = 0.0;
575     for (j = 1; j <= N + M - 1; j++){
576         obj1 += c[j-1];
577     }
578
579     //SUM_{j,k até Escalonamento} y[j][k]*c[j]
580     GRBLinExpr obj2 = 0.0;
581     for (j = 1; j <= M - 1; j++){
582         for (k = 1; k <= j; k++){
583             obj2 += (y[j-1][k-1] * c[j-1]);
584         }
585     }
586
587
588     for (j = M; j <= N; j++){
589         for (k = 1; k <= M; k++){
590             obj2 += (y[j-1][k-1] * c[j-1]);
591         }
592     }
593
594     for (j = N+1; j <= N + M - 1; j++){
595         for (k = j - N + 1; k <= M; k++){
596             obj2 += (y[j-1][k-1] * c[j-1]);
597         }
598     }
599
600     obj = obj1 - obj2;
601     //setar o objetivo
602     model.setObjective(obj, GRB_MINIMIZE);
603
604     //6 - Adiciona as restrições do modelo
605
606     //6.1 - A 1a conjunto de restrição:
607     for (j = 1; j <= M - 1; j++){
608         maxk = 0.0;
609         for (k = 1; k <= j; k++){
```

```
610             maxr = 0.0;
611             for (r = 1; r <= R; r++){
612                 if (maxr <= p[r-1][k-1]){
613                     maxr = p[r-1][k-1];
614                 }
615             }
616             if (maxk <= maxr){
617                 maxk = maxr;
618             }
619         }
620         model.addConstr(c[j-1] <= maxk, "rest1 para j=" + itos(j-1)
                );
621     }
622
623     for (j = M; j <= N; j++){
624         maxk = 0.0;
625         for (k = 1; k <= M; k++){
626             maxr = 0.0;
627             for (r = 1; r <= R; r++){
628                 if (maxr <= p[r-1][k-1]){
629                     maxr = p[r-1][k-1];
630                 }
631             }
632             if (maxk <= maxr){
633                 maxk = maxr;
634             }
635         }
636         model.addConstr(c[j-1] <= maxk, "rest1 para j=" + itos(j-1)
                );
637     }
638
639     for (j = N+1; j <= N + M - 1; j++){
640         maxk = 0.0;
641         for (k = j - N + 1; k <= M; k++){
642             maxr = 0.0;
643             for (r = 1; r <= R; r++){
644                 if (maxr <= p[r-1][k-1]){
645                     maxr = p[r-1][k-1];
646                 }
647             }
648             if (maxk <= maxr){
649                 maxk = maxr;
650             }
651         }
652         model.addConstr(c[j-1] <= maxk, "rest1 para j=" + itos(j-1)
                );
653     }
```

```
654
655     //6.2 - A 2a conjunto de restrição:
656     for (j = 1; j <= M - 1; j++){
657         maxk = 0.0;
658         for (k = 1; k <= j; k++){
659             minr = 1000.0;
660             for (r = 1; r <= R; r++){
661                 if (minr >= p[r-1][k-1]){
662                     minr = p[r-1][k-1];
663                 }
664             }
665             if (maxk <= minr){
666                 maxk = minr;
667             }
668         }
669         model.addConstr(c[j-1] >= maxk, "rest2 para j=" + itos(j-1)
                );
670     }
671
672
673     for (j = M; j <= N; j++){
674         maxk = 0.0;
675         for (k = 1; k <= M; k++){
676             minr = 1000.0;
677             for (r = 1; r <= R; r++){
678                 if (minr >= p[r-1][k-1]){
679                     minr = p[r-1][k-1];
680                 }
681             }
682             if (maxk <= minr){
683                 maxk = minr;
684             }
685         }
686         model.addConstr(c[j-1] >= maxk, "rest2 para j=" + itos(j-1)
                );
687     }
688
689     for (j = N+1; j <= N + M - 1; j++){
690         maxk = 0.0;
691         for (k = j - N + 1; k <= M; k++){
692             minr = 1000.0;
693             for (r = 1; r <= R; r++){
694                 if (minr >= p[r-1][k-1]){
695                     minr = p[r-1][k-1];
696                 }
697             }
698             if (maxk <= minr){
```

```
699                 maxk = minr;
700             }
701         }
702         model.addConstr(c[j-1] >= maxk, "rest2 para j=" + itos(j-1)
                );
703     }
704     //Após inserir as restrições, atualizar o modelo
705     model.update();
706 
707     //7 - Chama o método para otimizar o modelo
708     //especifica quanto tempo deverá ser gasto para otimizar - 3600
            s = 1h
709     model.getEnv().set(GRB_DoubleParam_TimeLimit, 7200);
710     //O GUROBI escolhe automaticamente o algoritmo a usar, entre
            eles: primal simplex, dual simplex, etc.
711     model.optimize();
712 
713     //8 - Após a otimização, imprime o resultado encontrado para o
            usuário
714     cout << setiosflags(ios::fixed | ios::showpoint | ios::left) <<
            setprecision(3) << endl << endl;
715 
716     for (j = 1; j <= N + M - 1; j++){
717         z[j-1]= c[j-1].get(GRB_DoubleAttr_X);
718     }
719 
720     z[M+N-1] = model.get(GRB_DoubleAttr_ObjVal);
721 
722     delete[] c;
723 
724     return z;
725 }//fim do gurobi 2
726 
727 double gur3(double* cb, double** xb, int** p, int* d, int N, int M,
        int R, int UB, double lb, GRBEnv env){
728     //4 - declarar as variáveis necessária para o programa
729     int i, j, k, n, m, r; //indice para vetores
730     int min, max;
731     double ub, OBJ;
732 
733     ub=UB;
734 
735     GRBModel model = GRBModel(env);
736     //4.5 - Cria as variáveis do problema de otimização
737 
738     //variaveis de c[j]
739     GRBVar* c = new GRBVar[N + M - 1];
```

```
740     //informa como é cada variavel do problema
741     max=0;
742     min=1000000;
743     for (i = 0; i < R; i++){
744         for (j = 0; j < M; j++){
745             if (max < p[i][j]){
746                 max = p[i][j];
747             }
748             if (min > p[i][j]){
749                 min = p[i][j];
750             }
751         }
752     }
753     //printf("\n min=%d, max=%d\n",min, max);
754     for (j = 0; j < N + M - 1; j++){
755         c[j] = model.addVar(min, max, cb[j], GRB_INTEGER, "c[" +
                itos(j + 1) + "]");
756     }
757
758     //variaveis do tipo x[r][n]
759     GRBVar** x = new GRBVar*[R];
760     for (i = 0; i < R; i++){
761         x[i] = new GRBVar[N];
762     }
763
764     for (i = 0; i < R; i++){
765         for (j = 0; j < N; j++){
766             x[i][j] = model.addVar(0.0, 1.0, xb[i][j], GRB_BINARY,
                    "x[" + itos(i + 1) + "][" + itos(j + 1) + "]");
767             //printf("x[%d][%d]\n",i,j);
768         }
769     }
770
771     //Após definir as variáveis, é preciso atualizar o modelo
            SEMPRE
772     model.update();
773
774
775     //SUM_{j=1 ate N+M-1} c[j]
776     GRBLinExpr obj = 0.0;
777     for (j = 0; j < N + M - 1; j++){
778         obj = obj + c[j];
779         //printf("c%d\n",j);
780     }
781
782     //setar o objetivo
783     model.setObjective(obj, GRB_MINIMIZE);
```

```
784
785     for (n = 0; n < N; n++){
786         GRBLinExpr rest1 = 0.0;
787         for (r = 0; r < R; r++){
788             rest1 = rest1 + x[r][n];
789             //printf("x[%d][%d]+",r,n);
790         }
791         model.addConstr(rest1 == 1, "rest1 para n=" + itos(n + 1));
792         //printf("==1\n");
793     }
794
795
796     //6.2 - A 2a restrição: SUM_{n=1 ate N} x[rn] = D[r], r=1,...,R
797     for (r = 0; r < R; r++){
798         GRBLinExpr rest2 = 0.0;
799         for (n = 0; n < N; n++){
800             rest2 += x[r][n];
801         }
802         model.addConstr(rest2 == d[r], "rest2 para r=" + itos(r +
                1));
803     }
804
805     //6.3 - A 3a restrição: c[j]>=SUM_{r=1 ate R} p[rk]*x[r(j-k+1)
            ], para todo j,k (j=1 até M-1 e k=1 ate j) (j=M ate N e k=1
             ate M) (j=n+1 até N+m-1 e k=j-N+1 ate M)
806     for (j = 1; j <= M - 1; j++){
807         for (k = 1; k <= j; k++){
808             GRBLinExpr rest3 = 0.0;
809             for (r = 1; r <= R; r++){
810                 rest3 = rest3 + (p[r-1][k-1] * x[r-1][j - k]);
811             }
812             model.addConstr(c[j-1] >= rest3, "rest3 para j=" + itos
                    (j) + "e k=" + itos(k));
813         }
814     }
815
816
817     for (j = M; j <= N; j++){
818         for (k = 1; k <= M; k++){
819             GRBLinExpr rest3 = 0.0;
820             for (r = 1; r <= R; r++){
821                 rest3 = rest3 + (p[r-1][k-1] * x[r-1][j - k]);
822             }
823             model.addConstr(c[j-1] >= rest3, "rest3 para j=" + itos
                    (j) + "e k=" + itos(k));
824         }
825     }
```

```
826
827     for (j = N+1; j <= N + M - 1; j++){
828         for (k = j - N + 1; k <= M; k++){
829             GRBLinExpr rest3 = 0.0;
830             for (r = 1; r <= R; r++){
831                 rest3 = rest3 + (p[r-1][k-1] * x[r-1][j - k]);
832             }
833             model.addConstr(c[j-1] >= rest3, "rest3 para j=" + itos
                    (j) + "e k=" + itos(k));
834         }
835     }
836
837     //Após inserir as restrições, atualizar o modelo
838     model.update();
839
840     //7 - Chama o método para otimizar o modelo
841     //especifica quanto tempo deverá ser gasto para otimizar - 3600
            s = 1h
842     model.getEnv().set(GRB_DoubleParam_TimeLimit, 7200);
843     //O GUROBI escolhe automaticamente o algoritmo a usar, entre
        eles: primal simplex, dual simplex, etc.
844     model.optimize();
845
846     //8 - Após a otimização, imprime o resultado encontrado para o
        usuário
847     cout << setiosflags(ios::fixed | ios::showpoint | ios::left) <<
            setprecision(3) << endl << endl;
848
849     //8.1 - Imprime o valor das variáveis para o usuário na tela
850
851     //imprime as variaveis cj
852     for (j = 0; j < N + M - 1; j++)
853     {
854         cout << "Ciclo c[" + itos(j + 1) + "]: " << c[j].get(
                GRB_DoubleAttr_X) << endl;
855     }
856
857     //imprime as variaveis xrn
858     for (r = 0; r < R; r++)
859     {
860         for (n = 0; n < N; n++)
861         {
862             cout << "x[" + itos(r + 1) + "][" + itos(n + 1) + "]: "
                    << x[r][n].get(GRB_DoubleAttr_X) << endl;
863         }
864     }
865
```

```
866         //imprime a ordem de Execução dos Produtos
867         printf("\n");
868         for (j = 1; j <= M - 1; j++){
869             printf("================================\n");
870             for (k = 1; k <= j; k++){
871                 for (r = 1; r <= R; r++){
872                     if (x[r-1][j-k].get(GRB_DoubleAttr_X) == 1){
873                         printf("No Ciclo C[%d] o produto %d é
                                processado na maquina %d em %d tempos\n",j,
                                 r, k, p[r-1][k-1]);
874                     }
875                 }
876             }
877         }
878 
879 
880         for (j = M; j <= N; j++){
881             printf("================================\n");
882             for (k = 1; k <= M; k++){
883                 for (r = 1; r <= R; r++){
884                     if (x[r-1][j-k].get(GRB_DoubleAttr_X) == 1){
885                         printf("No Ciclo C[%d] o produto %d é
                                processado na maquina %d em %d tempos\n",j,
                                 r, k, p[r-1][k-1]);
886                     }
887                 }
888             }
889         }
890 
891         for (j = N+1; j <= N + M - 1; j++){
892             printf("================================\n");
893             for (k = j - N + 1; k <= M; k++){
894                 for (r = 1; r <= R; r++){
895                     if (x[r-1][j-k].get(GRB_DoubleAttr_X) == 1){
896                         printf("No Ciclo C[%d] o produto %d é
                                processado na maquina %d em %d tempos\n",j,
                                 r, k, p[r-1][k-1]);
897                     }
898                 }
899             }
900         }
901 
902         cout << "\n\nValor da Funcao Objetivo: " << model.get(
                GRB_DoubleAttr_ObjVal) << endl;
903 
904         //9 - Imprimir outras informações do modelo
905         OBJ=model.get(GRB_DoubleAttr_Runtime);
```

```
906     //Para imprimir mais dados do modelo
907     cout << endl << "\n------Mais informacoes sobre o Modelo------\
            n" << endl;
908     cout << "Status da solucao: (2 - otimo) | (3 - inviavel) | (5 -
             ilimitado): "  << model.get(GRB_IntAttr_Status) << endl;
909     cout << "Tempo de otimização do GUROBI: "  << model.get(
            GRB_DoubleAttr_Runtime) << " segundos" << endl;
910     cout << "Numero total de variaveis: "  << model.get(
            GRB_IntAttr_NumVars) << endl;
911                     //cout << "--Numero de variaveis Inteiras: "
                            << model.get(GRB_IntAttr_NumIntVars) <<
                            endl;
912     cout << "--Numero de variaveis Binarias: "  << model.get(
            GRB_IntAttr_NumBinVars) << endl;
913     cout << "Numero de nos da arvore B&B explorados: " <<
            setprecision(0) << model.get(GRB_DoubleAttr_NodeCount) <<
            endl;
914
915     return OBJ;
916 }//fim do gurobi 3
917
918 double upperbound(double** x, int M, int N, int R, int** p){
919
920     int b, w, q, A, max, i, j, k, r;
921     double UB;
922     double* C;
923
924     C = new double[M+N-1]; //aloca as linhas da matriz p[r][m]
925
926
927     for (j = 1; j <= M - 1; j++){
928         C[j-1] = 0;
929         for (k = 1; k <= j; k++){
930             A = 0;
931             for (r = 1; r <= R; r++){
932                 A = A + (p[r-1][k-1] * x[r-1][j - k]);
933             }
934             if (A > C[j-1]){
935                 C[j-1] = A;
936             }
937         }
938     }
939
940     for (j = M; j <= N; j++){
941         C[j-1] = 0;
942         for (k = 1; k <= M; k++){
943             A = 0;
```

```
944                 for (r = 1; r <= R; r++){
945                     A = A + (p[r-1][k-1] * x[r-1][j - k]);
946                 }
947                 if (A > C[j-1]){
948                     C[j-1] = A;
949                 }
950             }
951         }
952 
953 
954         for (j = N+1; j <= N + M - 1; j++){
955             C[j-1] = 0;
956             for (k = j - N + 1; k <= M; k++){
957                 A = 0;
958                 for (r = 1; r <= R; r++){
959                     A = A + (p[r-1][k-1] * x[r-1][j - k]);
960                 }
961                 if (A > C[j-1]){
962                     C[j-1] = A;
963                 }
964             }
965         }
966 
967         UB = 0.0;
968         for (j = 1; j <= M+N-1; j++){
969             UB = UB + C[j-1];
970         }
971 
972         return UB;
973 }//fim do Upperbound
```

APÊNDICE C

ALGORITMO GENÉTICO

```
%
% Copyright (c) 2015, Yarpiz (www.yarpiz.com)
% All rights reserved. Please read the "license.txt" for license terms.
%
% Project Code: YPEA101
% Project Title: Implementation of Binary Genetic Algorithm in MATLAB
% Publisher: Yarpiz (www.yarpiz.com)
%
% Developer: S. Mostapha Kalami Heris (Member of Yarpiz Team)
%
% Contact Info: sm.kalami@gmail.com, info@yarpiz.com
%

function ga=ga(M,N,R,P,D,rp,result,tim,gama)

CostFunction=@(x) MinOne(x,M,N,R,P,D,rp);     % Cost Function

nVar=R*N;             % Number of Decision Variable
VarSize=[1 nVar];   % Decision Variables Matrix Size

%% GA Parameters

MaxIt=1500;      % Maximum Number of Iterations
if (N == 10)
    nPop=500;    % Population Size
end
if (N == 15)
    nPop=1000;
end
if (N == 20)
    nPop=1500;
```

```
end
if (N == 30)
    nPop=2000;
end
if (N == 60)
    nPop=2500;
end
if (N == 120)
    nPop=3000;
end

pc=0.8;                    % Crossover Percentage
nc=2*round(pc*nPop/2);  % Number of Offsprings (also Parnets)

pm=0.3;                    % Mutation Percentage
nm=round(pm*nPop);       % Number of Mutants
mu=0.1;                  % Mutation Rate

%ANSWER=questdlg('Choose selection method:','Genetic Algorith',...
%    'Roulette Wheel','Tournament','Random','Roulette Wheel');
ANSWER=('Random');
UseRouletteWheelSelection=strcmp(ANSWER,'Roulette Wheel');
UseTournamentSelection=strcmp(ANSWER,'Tournament');
UseRandomSelection=strcmp(ANSWER,'Random');

if UseRouletteWheelSelection
    beta=8;          % Selection Pressure
end

if UseTournamentSelection
    TournamentSize=3;    % Tournamnet Size
end

%pause(0.01); % Needed due to a bug in older versions of MATLAB

%% Initialization

empty_individual.Position=[];
empty_individual.Cost=[];

pop=repmat(empty_individual,nPop,1);

for i=1:nPop

    % Initialize Position
    pop(i).Position=randi([0 1],VarSize);

```

```
79      % Evaluation
80      pop(i).Cost=CostFunction(pop(i).Position);
81
82  end
83
84  % Sort Population
85  Costs=[pop.Cost];
86  [Costs, SortOrder]=sort(Costs);
87  pop=pop(SortOrder);
88
89  % Store Best Solution
90  BestSol=pop(1);
91
92  % Array to Hold Best Cost Values
93  BestCost=zeros(MaxIt,1);
94
95  % Store Cost
96  WorstCost=pop(end).Cost;
97
98  fprintf(result,'\n#########Iteration#########\n');
99  %% Main Loop
100 tic;
101 for it=1:MaxIt
102         rp=gama*rp;
103     % Calculate Selection Probabilities
104     if UseRouletteWheelSelection
105         P=exp(-beta*Costs/WorstCost);
106         P=P/sum(P);
107     end
108
109     % Crossover
110     popc=repmat(empty_individual,nc/2,2);
111     for k=1:nc/2
112
113         % Select Parents Indices
114         if UseRouletteWheelSelection
115             i1=RouletteWheelSelection(P);
116             i2=RouletteWheelSelection(P);
117         end
118         if UseTournamentSelection
119             i1=TournamentSelection(pop,TournamentSize);
120             i2=TournamentSelection(pop,TournamentSize);
121         end
122         if UseRandomSelection
123             i1=randi([1 nPop]);
124             i2=randi([1 nPop]);
125         end
```

```
126
127         % Select Parents
128         p1=pop(i1);
129         p2=pop(i2);
130
131         % Perform Crossover
132         [popc(k,1).Position, popc(k,2).Position]=Crossover(p1.
                Position,p2.Position);
133
134         % Evaluate Offsprings
135         popc(k,1).Cost=CostFunction(popc(k,1).Position);
136         popc(k,2).Cost=CostFunction(popc(k,2).Position);
137
138     end
139     popc=popc(:);
140
141
142     % Mutation
143     popm=repmat(empty_individual,nm,1);
144     for k=1:nm
145
146         % Select Parent
147         i=randi([1 nPop]);
148         p=pop(i);
149
150         % Perform Mutation
151         popm(k).Position=Mutate(p.Position,mu);
152
153         % Evaluate Mutant
154         popm(k).Cost=CostFunction(popm(k).Position);
155
156     end
157
158     % Create Merged Population
159     pop=[pop
160          popc
161          popm]; %#ok
162
163     % Sort Population
164     Costs=[pop.Cost];
165     [Costs, SortOrder]=sort(Costs);
166     pop=pop(SortOrder);
167
168     % Update Worst Cost
169     WorstCost=max(WorstCost,pop(end).Cost);
170
171     % Truncation
```

```
172     pop=pop(1:nPop);
173     Costs=Costs(1:nPop);
174
175     % Store Best Solution Ever Found
176     BestSol=pop(1);
177
178     % Store Best Cost Ever Found
179     BestCost(it)=BestSol.Cost;
180
181     %Calcule the time in the itaration
182     codeelapsed_time = toc;
183     % Show Iteration Information
184     fprintf(result,'\nIteration %d: Best Cost = %d,Time = %.4f', it
            , BestCost(it), codeelapsed_time);
185     fprintf('\nIteration %d: Best Cost = %.4d,Time = %.4f', it,
            BestCost(it), codeelapsed_time);
186     %disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(
            BestCost(it)) ',Time = ' num2str(codeelapsed_time)]);
187 end
188 codeelapsed_time = toc;
189 fprintf(result,'\n\n#########Solution found#########\n');
190 fprintf(result,'####Stop by MaxIt#####\n');
191 fprintf(result,'Iteration: %d\n',it);
192 fprintf(result,'Best X=[ ');
193 for z=1:R*N
194     fprintf(result,'%d ',BestSol.Position(1,z));
195     X(z)=BestSol.Position(1,z);
196 end
197 fprintf(result,']\n');
198
199 b=1;
200 for w=1:R
201     for q=1:N
202         X(w,q)=BestSol.Position(1,b);
203         b=b+1;
204     end
205 end
206
207 UB=upperboundc(X,M,N,R,P);
208 fprintf(result,'Cmax: %d\n', UB);
209 fprintf(result,'Resultado AG: %d\n',BestCost(it));
210 fprintf(result,'Time Spent: %.4f\n',codeelapsed_time);
211
212
213
214 for j=1:M-1
215     fprintf(result,'==================================\n');
```

```
216     for k=1:j
217         for r=1:R
218             if (X(r,j-k+1)==1)
219                 fprintf(result,'No Ciclo C[%d] o produto %d e
                        processado na maquina %d em %d tempos\n',j, r,
                        k, P(r,k));
220             end
221         end
222     end
223 end
224
225 for j=M:N
226     fprintf(result,'================================\n');
227     for k=1:M
228         for r=1:R
229             if (X(r,j-k+1)==1)
230                 fprintf(result,'No Ciclo C[%d] o produto %d e
                        processado na maquina %d em %d tempos\n',j, r,
                        k, P(r,k));
231             end
232         end
233     end
234 end
235
236 for j=N+1:N+M-1
237     fprintf(result,'================================\n');
238     for k=j-N+1:M
239         for r=1:R
240             if (X(r,j-k+1)==1)
241                 fprintf(result,'No Ciclo C[%d] o produto %d e
                        processado na maquina %d em %d tempos\n',j, r,
                        k, P(r,k));
242             end
243         end
244     end
245 end
246
247 fprintf(tim,'%d\t%d\t%.4f\n',BestCost(it),UB,codeelapsed_time);
248 display('end of running');
249 %% Results
250 % hold on
251 % figure(1);
252 % plot(BestCost,'LineWidth',2);
253 % xlabel('Iteration');
254 % ylabel('Cost');
255 % grid on;
256 end
```

```
% Copyright (c) 2015, Yarpiz (www.yarpiz.com)
% All rights reserved. Please read the "license.txt" for license terms.
%
% Project Code: YPEA101
% Project Title: Implementation of Binary Genetic Algorithm in MATLAB
% Publisher: Yarpiz (www.yarpiz.com)
%
% Developer: S. Mostapha Kalami Heris (Member of Yarpiz Team)
%
% Contact Info: sm.kalami@gmail.com, info@yarpiz.com
%

function [y1, y2]=Crossover(x1,x2)

    pSinglePoint=0.1;
    pDoublePoint=0.2;
    pUniform=1-pSinglePoint-pDoublePoint;

    METHOD=RouletteWheelSelection([pSinglePoint pDoublePoint pUniform]);

    switch METHOD
        case 1
            [y1, y2]=SinglePointCrossover(x1,x2);

        case 2
            [y1, y2]=DoublePointCrossover(x1,x2);

        case 3
            [y1, y2]=UniformCrossover(x1,x2);

    end

end
```

```
% Copyright (c) 2017, Jeferson Silva Martins
% All rights reserved. Master in Modeling and Optmization
%
% Project Code: LRFLMWGA
% Project Title: Lagrangean Relaxation for FLM in MATLAB
% Publisher: jefersonsilvam
%
% Developers: S. M., Jeferson (Student of Master Degree)
% Contact Info: jsm.ctl@gmail.com
%
%
```

```
12
13 function UB=upperboundc(X,M,N,R,P)
14     x=X;
15     for j=1:M-1
16         c(j)=0;
17         for k=1:j
18             A=0;
19             for r=1:R
20                 A=A+(P(r,k)*x(r,j-k+1));
21             end
22             if (A > c(j))
23                 c(j) = A;
24             end
25         end
26     end
27
28     for j=M:N
29         c(j)=0;
30         for k=1:M
31             A=0;
32             for r=1:R
33                 A=A+(P(r,k)*x(r,j-k+1));
34             end
35             if (A > c(j))
36                 c(j) = A;
37             end
38         end
39     end
40
41     for j=N+1:N+M-1
42         c(j)=0;
43         for k=j-N+1:M
44             A=0;
45             for r=1:R
46                 A=A+(P(r,k)*x(r,j-k+1));
47             end
48             if (A > c(j))
49                 c(j) = A;
50             end
51         end
52     end
53
54     UB=0;
55     for j=1:M+N-1
56         UB=UB+c(j);
57     end
58 end
```

```
% Copyright (c) 2015, Yarpiz (www.yarpiz.com)
% All rights reserved. Please read the "license.txt" for license terms.
%
% Project Code: YPEA101
% Project Title: Implementation of Binary Genetic Algorithm in MATLAB
% Publisher: Yarpiz (www.yarpiz.com)
%
% Developer: S. Mostapha Kalami Heris (Member of Yarpiz Team)
%
% Contact Info: sm.kalami@gmail.com, info@yarpiz.com
%

function [y1, y2]=DoublePointCrossover(x1,x2)

    nVar=numel(x1);

    cc=randsample(nVar-1,2);
    c1=min(cc);
    c2=max(cc);

    y1=[x1(1:c1) x2(c1+1:c2) x1(c2+1:end)];
    y2=[x2(1:c1) x1(c1+1:c2) x2(c2+1:end)];

end
```

```
% Copyright (c) 2015, Yarpiz (www.yarpiz.com)
% All rights reserved. Please read the "license.txt" for license terms.
%
% Project Code: YPEA101
% Project Title: Implementation of Binary Genetic Algorithm in MATLAB
% Publisher: Yarpiz (www.yarpiz.com)
%
% Developer: S. Mostapha Kalami Heris (Member of Yarpiz Team)
%
% Contact Info: sm.kalami@gmail.com, info@yarpiz.com
%

function y=Mutate(x,mu)

    nVar=numel(x);

    nmu=ceil(mu*nVar);

    j=randsample(nVar,nmu);
```

```
20
21      y=x;
22      y(j)=1-x(j);
23
24  end
```

```
1   % Copyright (c) 2015, Yarpiz (www.yarpiz.com)
2   % All rights reserved. Please read the "license.txt" for license
        terms.
3   %
4   % Project Code: YPEA101
5   % Project Title: Implementation of Binary Genetic Algorithm in
        MATLAB
6   % Publisher: Yarpiz (www.yarpiz.com)
7   %
8   % Developer: S. Mostapha Kalami Heris (Member of Yarpiz Team)
9   %
10  % Contact Info: sm.kalami@gmail.com, info@yarpiz.com
11  %
12
13  function z=MinOne(x,M,N,R,P,D,rp)
14      %% Vector adjustment
15      b=1;
16      for w=1:R
17          for q=1:N
18              X(w,q)=x(b);
19              b=b+1;
20          end
21      end
22      %%clear x
23      %%x=X;
24
25
26     %% UpperBound
27      for j=1:M-1
28          q=1;
29          for k=1:j
30              A=0;
31              for r=1:R
32                  A=A+(P(r,k)*X(r,j-k+1));
33              end
34              c(j,q)=A;
35              q=q+1;
36          end
37      end
38
39      for j=M:N
40          q=1;
```

```
41          for k=1:M
42              A=0;
43              for r=1:R
44                  A=A+(P(r,k)*X(r,j-k+1));
45              end
46              c(j,q)=A;
47              q=q+1;
48          end
49      end
50
51      for j=N+1:N+M-1
52          q=1;
53          for k=j-N+1:M
54              A=0;
55              for r=1:R
56                  A=A+(P(r,k)*X(r,j-k+1));
57              end
58              c(j,q)=A;
59              q=q+1;
60          end
61      end
62
63      for j=1:M+N-1
64          C(j)=max(c(j,:));
65      end
66      l=1;
67      %% Constraint #1
68      for n=1:N
69          T=0;
70          for r=1:R
71              T=T+X(r,n);
72          end
73          g(l)=T-1;
74          l=l+1;
75      end
76
77      %% Constrainst #2
78      for r=1:R
79          T=0;
80          for n=1:N
81              T=T+X(r,n);
82          end
83          g(l)=T-D(r);
84          l=l+1;
85      end
86
87      %% Exterior Penalt
```

```
88      rese=0;
89      for i=1:l-1
90          rese=rese+(g(i)^2);
91      end
92
93      %% Constrainst #3
94      y=l;
95      for j=1:M-1
96          for k=1:j
97              A=0;
98              for r=1:R
99                  A=A+(P(r,k)*X(r,j-k+1));
100             end
101             g(y)=A-C(j);
102             y=y+1;
103         end
104     end
105
106     for j=M:N
107         for k=1:M
108             A=0;
109             for r=1:R
110                 A=A+(P(r,k)*X(r,j-k+1));
111             end
112             g(y)=A-C(j);
113             y=y+1;
114         end
115     end
116
117     for j=N+1:N+M-1
118         for k=j-N+1:M
119             A=0;
120             for r=1:R
121                 A=A+(P(r,k)*X(r,j-k+1));
122             end
123             g(y)=A-C(j);
124             y=y+1;
125         end
126     end
127         resi=0;
128     for i=l:y-1
129         resi=resi+(max(0,g(i)))^2;
130     end
131
132     zc=0;
133     for j=1:N+M-1
134         zc=zc+C(j);
```

```
135         end
136
137         z=zc+((rese+resi)*rp);
138     end
```

ANEXO A

FORMATOS DE SAÍDA DOS RESULTADOS

Saída do Algoritmo Genético:

```
#########Iteration#########

Iteration 1: Best Cost = 40000000199,Time = 0.0639
Iteration 2: Best Cost = 40000000191,Time = 0.0953
Iteration 3: Best Cost = 40000000180,Time = 0.1228
Iteration 4: Best Cost = 40000000176,Time = 0.1505
Iteration 5: Best Cost = 20000000169,Time = 0.1806
Iteration 6: Best Cost = 20000000169,Time = 0.2080
Iteration 7: Best Cost = 20000000169,Time = 0.2351
Iteration 8: Best Cost = 20000000169,Time = 0.2621
Iteration 9: Best Cost = 20000000158,Time = 0.2893
Iteration 10: Best Cost = 20000000158,Time = 0.3159
Iteration 11: Best Cost = 20000000158,Time = 0.3429
Iteration 12: Best Cost = 20000000158,Time = 0.3691
Iteration 13: Best Cost = 170,Time = 0.3951
Iteration 14: Best Cost = 170,Time = 0.4216
Iteration 15: Best Cost = 170,Time = 0.4481
Iteration 16: Best Cost = 170,Time = 0.4743
Iteration 17: Best Cost = 170,Time = 0.5003
Iteration 18: Best Cost = 170,Time = 0.5262
Iteration 19: Best Cost = 170,Time = 0.5517
Iteration 20: Best Cost = 170,Time = 0.5778
Iteration 21: Best Cost = 170,Time = 0.6040
Iteration 22: Best Cost = 170,Time = 0.6303
Iteration 23: Best Cost = 170,Time = 0.6561
Iteration 24: Best Cost = 170,Time = 0.6818
Iteration 25: Best Cost = 170,Time = 0.7077
Iteration 26: Best Cost = 170,Time = 0.7338
Iteration 27: Best Cost = 170,Time = 0.7610
```

```
Iteration 28: Best Cost = 170,Time = 0.7896
Iteration 29: Best Cost = 170,Time = 0.8164
Iteration 30: Best Cost = 170,Time = 0.8423
Iteration 31: Best Cost = 170,Time = 0.8680
Iteration 32: Best Cost = 170,Time = 0.8940
Iteration 33: Best Cost = 170,Time = 0.9196
Iteration 34: Best Cost = 170,Time = 0.9452
Iteration 35: Best Cost = 170,Time = 0.9710
Iteration 36: Best Cost = 170,Time = 0.9966
Iteration 37: Best Cost = 170,Time = 1.0223
Iteration 38: Best Cost = 170,Time = 1.0479
Iteration 39: Best Cost = 170,Time = 1.0735
Iteration 40: Best Cost = 170,Time = 1.0989
Iteration 41: Best Cost = 170,Time = 1.1245
Iteration 42: Best Cost = 170,Time = 1.1500
Iteration 43: Best Cost = 170,Time = 1.1756
Iteration 44: Best Cost = 170,Time = 1.2014
Iteration 45: Best Cost = 170,Time = 1.2270
Iteration 46: Best Cost = 170,Time = 1.2525
Iteration 47: Best Cost = 170,Time = 1.2780
Iteration 48: Best Cost = 170,Time = 1.3034
Iteration 49: Best Cost = 170,Time = 1.3289
Iteration 50: Best Cost = 170,Time = 1.3543
Iteration 51: Best Cost = 170,Time = 1.3798
Iteration 52: Best Cost = 170,Time = 1.4054
Iteration 53: Best Cost = 170,Time = 1.4308
Iteration 54: Best Cost = 170,Time = 1.4562
Iteration 55: Best Cost = 170,Time = 1.4816
Iteration 56: Best Cost = 170,Time = 1.5072
Iteration 57: Best Cost = 170,Time = 1.5327
Iteration 58: Best Cost = 170,Time = 1.5581
Iteration 59: Best Cost = 170,Time = 1.5835
Iteration 60: Best Cost = 170,Time = 1.6090
Iteration 61: Best Cost = 170,Time = 1.6346
Iteration 62: Best Cost = 170,Time = 1.6600
Iteration 63: Best Cost = 170,Time = 1.6853
Iteration 64: Best Cost = 170,Time = 1.7110
Iteration 65: Best Cost = 170,Time = 1.7367
Iteration 66: Best Cost = 170,Time = 1.7627
Iteration 67: Best Cost = 170,Time = 1.7885
Iteration 68: Best Cost = 170,Time = 1.8139
Iteration 69: Best Cost = 170,Time = 1.8394
Iteration 70: Best Cost = 170,Time = 1.8649
Iteration 71: Best Cost = 170,Time = 1.8901
Iteration 72: Best Cost = 170,Time = 1.9154
Iteration 73: Best Cost = 170,Time = 1.9406
Iteration 74: Best Cost = 170,Time = 1.9661
```

```
Iteration 75: Best Cost = 170,Time = 1.9915
Iteration 76: Best Cost = 170,Time = 2.0168
Iteration 77: Best Cost = 170,Time = 2.0422
Iteration 78: Best Cost = 170,Time = 2.0674
Iteration 79: Best Cost = 170,Time = 2.0928
Iteration 80: Best Cost = 170,Time = 2.1182
Iteration 81: Best Cost = 170,Time = 2.1434
Iteration 82: Best Cost = 170,Time = 2.1688
Iteration 83: Best Cost = 170,Time = 2.1941
Iteration 84: Best Cost = 170,Time = 2.2197
Iteration 85: Best Cost = 170,Time = 2.2455
Iteration 86: Best Cost = 170,Time = 2.2710
Iteration 87: Best Cost = 170,Time = 2.2966
Iteration 88: Best Cost = 170,Time = 2.3221
Iteration 89: Best Cost = 170,Time = 2.3476
Iteration 90: Best Cost = 170,Time = 2.3730
Iteration 91: Best Cost = 170,Time = 2.3986
Iteration 92: Best Cost = 170,Time = 2.4242
Iteration 93: Best Cost = 170,Time = 2.4497
Iteration 94: Best Cost = 170,Time = 2.4752
Iteration 95: Best Cost = 170,Time = 2.5008
Iteration 96: Best Cost = 170,Time = 2.5264
Iteration 97: Best Cost = 170,Time = 2.5518
Iteration 98: Best Cost = 170,Time = 2.5773
Iteration 99: Best Cost = 170,Time = 2.6027
Iteration 100: Best Cost = 170,Time = 2.6283
Iteration 101: Best Cost = 170,Time = 2.6539
Iteration 102: Best Cost = 170,Time = 2.6794
Iteration 103: Best Cost = 170,Time = 2.7049
Iteration 104: Best Cost = 170,Time = 2.7305
Iteration 105: Best Cost = 170,Time = 2.7562
Iteration 106: Best Cost = 170,Time = 2.7818
Iteration 107: Best Cost = 170,Time = 2.8073
Iteration 108: Best Cost = 170,Time = 2.8328
Iteration 109: Best Cost = 170,Time = 2.8586
Iteration 110: Best Cost = 170,Time = 2.8842
Iteration 111: Best Cost = 170,Time = 2.9098
Iteration 112: Best Cost = 170,Time = 2.9353
Iteration 113: Best Cost = 170,Time = 2.9607
Iteration 114: Best Cost = 170,Time = 2.9861
Iteration 115: Best Cost = 170,Time = 3.0116
Iteration 116: Best Cost = 170,Time = 3.0371
Iteration 117: Best Cost = 170,Time = 3.0626
Iteration 118: Best Cost = 170,Time = 3.0881
Iteration 119: Best Cost = 170,Time = 3.1137
Iteration 120: Best Cost = 170,Time = 3.1391
Iteration 121: Best Cost = 170,Time = 3.1647
```

```
Iteration 122: Best Cost = 170,Time = 3.1903
Iteration 123: Best Cost = 170,Time = 3.2157
Iteration 124: Best Cost = 170,Time = 3.2415
Iteration 125: Best Cost = 170,Time = 3.2668
Iteration 126: Best Cost = 170,Time = 3.2921
Iteration 127: Best Cost = 170,Time = 3.3178
Iteration 128: Best Cost = 170,Time = 3.3435
Iteration 129: Best Cost = 170,Time = 3.3689
Iteration 130: Best Cost = 170,Time = 3.3944
Iteration 131: Best Cost = 170,Time = 3.4200
Iteration 132: Best Cost = 170,Time = 3.4456
Iteration 133: Best Cost = 170,Time = 3.4711
Iteration 134: Best Cost = 170,Time = 3.4968
Iteration 135: Best Cost = 170,Time = 3.5223
Iteration 136: Best Cost = 170,Time = 3.5482
Iteration 137: Best Cost = 170,Time = 3.5736
Iteration 138: Best Cost = 170,Time = 3.5990
Iteration 139: Best Cost = 170,Time = 3.6244
Iteration 140: Best Cost = 170,Time = 3.6499
Iteration 141: Best Cost = 170,Time = 3.6755
Iteration 142: Best Cost = 170,Time = 3.7011
Iteration 143: Best Cost = 170,Time = 3.7266
Iteration 144: Best Cost = 170,Time = 3.7521
Iteration 145: Best Cost = 170,Time = 3.7775
Iteration 146: Best Cost = 170,Time = 3.8031
Iteration 147: Best Cost = 170,Time = 3.8285
Iteration 148: Best Cost = 170,Time = 3.8542
Iteration 149: Best Cost = 170,Time = 3.8798
Iteration 150: Best Cost = 170,Time = 3.9051
Iteration 151: Best Cost = 170,Time = 3.9307
Iteration 152: Best Cost = 170,Time = 3.9560
Iteration 153: Best Cost = 170,Time = 3.9817
Iteration 154: Best Cost = 170,Time = 4.0072
Iteration 155: Best Cost = 170,Time = 4.0328
Iteration 156: Best Cost = 170,Time = 4.0585
Iteration 157: Best Cost = 170,Time = 4.0839
Iteration 158: Best Cost = 170,Time = 4.1094
Iteration 159: Best Cost = 170,Time = 4.1349
Iteration 160: Best Cost = 170,Time = 4.1603
Iteration 161: Best Cost = 170,Time = 4.1858
Iteration 162: Best Cost = 170,Time = 4.2113
Iteration 163: Best Cost = 170,Time = 4.2369
Iteration 164: Best Cost = 170,Time = 4.2623
Iteration 165: Best Cost = 170,Time = 4.2876
Iteration 166: Best Cost = 170,Time = 4.3130
Iteration 167: Best Cost = 170,Time = 4.3387
Iteration 168: Best Cost = 170,Time = 4.3640
```

```
Iteration 169: Best Cost = 170,Time = 4.3893
Iteration 170: Best Cost = 170,Time = 4.4145
Iteration 171: Best Cost = 170,Time = 4.4400
Iteration 172: Best Cost = 170,Time = 4.4655
Iteration 173: Best Cost = 170,Time = 4.4913
Iteration 174: Best Cost = 170,Time = 4.5167
Iteration 175: Best Cost = 170,Time = 4.5424
Iteration 176: Best Cost = 170,Time = 4.5680
Iteration 177: Best Cost = 170,Time = 4.5934
Iteration 178: Best Cost = 170,Time = 4.6190
Iteration 179: Best Cost = 170,Time = 4.6445
Iteration 180: Best Cost = 170,Time = 4.6703
Iteration 181: Best Cost = 170,Time = 4.6958
Iteration 182: Best Cost = 170,Time = 4.7213
Iteration 183: Best Cost = 170,Time = 4.7470
Iteration 184: Best Cost = 170,Time = 4.7726
Iteration 185: Best Cost = 170,Time = 4.7982
Iteration 186: Best Cost = 170,Time = 4.8238
Iteration 187: Best Cost = 170,Time = 4.8491
Iteration 188: Best Cost = 170,Time = 4.8744
Iteration 189: Best Cost = 170,Time = 4.8999
Iteration 190: Best Cost = 170,Time = 4.9252
Iteration 191: Best Cost = 170,Time = 4.9507
Iteration 192: Best Cost = 170,Time = 4.9764
Iteration 193: Best Cost = 170,Time = 5.0020
Iteration 194: Best Cost = 170,Time = 5.0274
Iteration 195: Best Cost = 170,Time = 5.0531
Iteration 196: Best Cost = 170,Time = 5.0789
Iteration 197: Best Cost = 170,Time = 5.1045
Iteration 198: Best Cost = 170,Time = 5.1300
Iteration 199: Best Cost = 170,Time = 5.1554
Iteration 200: Best Cost = 170,Time = 5.1808

#########Solution found#########
####Stop by MaxIt#####
Iteration: 200
Best X=[ 0 0 1 1 0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1
   0 ]
Cmax: 170
Time Spent: 5.180827e+00
================================
No Ciclo C[1] o produto 2 e processado na maquina 1 em 1 tempos
================================
No Ciclo C[2] o produto 2 e processado na maquina 1 em 1 tempos
No Ciclo C[2] o produto 2 e processado na maquina 2 em 15 tempos
================================
No Ciclo C[3] o produto 1 e processado na maquina 1 em 15 tempos
```

```
No Ciclo C[3] o produto 2 e processado na maquina 2 em 15 tempos
No Ciclo C[3] o produto 2 e processado na maquina 3 em 16 tempos
================================
No Ciclo C[4] o produto 1 e processado na maquina 1 em 15 tempos
No Ciclo C[4] o produto 1 e processado na maquina 2 em 5 tempos
No Ciclo C[4] o produto 2 e processado na maquina 3 em 16 tempos
================================
No Ciclo C[5] o produto 2 e processado na maquina 1 em 1 tempos
No Ciclo C[5] o produto 1 e processado na maquina 2 em 5 tempos
No Ciclo C[5] o produto 1 e processado na maquina 3 em 14 tempos
================================
No Ciclo C[6] o produto 3 e processado na maquina 1 em 8 tempos
No Ciclo C[6] o produto 2 e processado na maquina 2 em 15 tempos
No Ciclo C[6] o produto 1 e processado na maquina 3 em 14 tempos
================================
No Ciclo C[7] o produto 2 e processado na maquina 1 em 1 tempos
No Ciclo C[7] o produto 3 e processado na maquina 2 em 10 tempos
No Ciclo C[7] o produto 2 e processado na maquina 3 em 16 tempos
================================
No Ciclo C[8] o produto 1 e processado na maquina 1 em 15 tempos
No Ciclo C[8] o produto 2 e processado na maquina 2 em 15 tempos
No Ciclo C[8] o produto 3 e processado na maquina 3 em 16 tempos
================================
No Ciclo C[9] o produto 3 e processado na maquina 1 em 8 tempos
No Ciclo C[9] o produto 1 e processado na maquina 2 em 5 tempos
No Ciclo C[9] o produto 2 e processado na maquina 3 em 16 tempos
================================
No Ciclo C[10] o produto 1 e processado na maquina 1 em 15 tempos
No Ciclo C[10] o produto 3 e processado na maquina 2 em 10 tempos
No Ciclo C[10] o produto 1 e processado na maquina 3 em 14 tempos
================================
No Ciclo C[11] o produto 1 e processado na maquina 2 em 5 tempos
No Ciclo C[11] o produto 3 e processado na maquina 3 em 16 tempos
================================
No Ciclo C[12] o produto 1 e processado na maquina 3 em 14 tempos
```

Saída do Gurobi:

```
Optimize a model with 68 rows, 137 columns and 645 nonzeros
Variable types: 0 continuous, 137 integer (120 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 2e+01]
  RHS range        [1e+00, 2e+00]
Found heuristic solution: objective 244
Presolve removed 2 rows and 2 columns
Presolve time: 0.00s
```

```
Presolved: 66 rows, 135 columns, 556 nonzeros
Variable types: 0 continuous, 135 integer (120 binary)

Root relaxation: objective 1.751127e+02, 296 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0  175.11267    0   58  244.00000  175.11267  28.2%     -    0s
H    0     0                       236.0000000  175.11267  25.8%     -    0s
H    0     0                       217.0000000  175.11267  19.3%     -    0s
     0     0  175.27374    0   60  217.00000  175.27374  19.2%     -    0s
     0     0  175.50546    0   60  217.00000  175.50546  19.1%     -    0s
     0     0  175.67336    0   64  217.00000  175.67336  19.0%     -    0s
     0     0  175.67722    0   66  217.00000  175.67722  19.0%     -    0s
     0     0  175.76252    0   70  217.00000  175.76252  19.0%     -    0s
     0     0  175.97625    0   74  217.00000  175.97625  18.9%     -    0s
H    0     0                       207.0000000  175.97625  15.0%     -    0s
     0     0  176.24129    0   72  207.00000  176.24129  14.9%     -    0s
     0     0  176.26071    0   74  207.00000  176.26071  14.8%     -    0s
     0     0  177.02046    0   82  207.00000  177.02046  14.5%     -    0s
     0     0  177.07170    0   86  207.00000  177.07170  14.5%     -    0s
     0     0  177.16269    0   92  207.00000  177.16269  14.4%     -    0s
     0     0  177.38394    0   88  207.00000  177.38394  14.3%     -    0s
     0     0  178.31405    0   82  207.00000  178.31405  13.9%     -    0s
     0     0  178.75544    0   89  207.00000  178.75544  13.6%     -    0s
```

```
     0     0  179.31368    0   88  207.00000  179.31368  13.4%
         -    0s
     0     0  179.63139    0   91  207.00000  179.63139  13.2%
         -    0s
     0     0  180.25206    0   90  207.00000  180.25206  12.9%
         -    0s
     0     0  180.67945    0   95  207.00000  180.67945  12.7%
         -    0s
H    0     0                       205.0000000  180.67945  11.9%
   -    0s
     0     0  181.15340    0   86  205.00000  181.15340  11.6%
         -    0s
     0     0  181.52263    0   85  205.00000  181.52263  11.5%
         -    0s
     0     0  182.25488    0   88  205.00000  182.25488  11.1%
         -    0s
     0     0  182.49393    0   91  205.00000  182.49393  11.0%
         -    0s
     0     0  183.04327    0   88  205.00000  183.04327  10.7%
         -    0s
     0     0  183.40306    0   89  205.00000  183.40306  10.5%
         -    0s
     0     0  183.83054    0   91  205.00000  183.83054  10.3%
         -    0s
     0     0  183.95493    0   93  205.00000  183.95493  10.3%
         -    0s
     0     0  184.36615    0   86  205.00000  184.36615  10.1%
         -    0s
     0     0  184.71669    0   89  205.00000  184.71669  9.89%
         -    0s
     0     0  185.25555    0   86  205.00000  185.25555  9.63%
         -    0s
     0     0  185.41984    0   83  205.00000  185.41984  9.55%
         -    0s
     0     0  185.66858    0   84  205.00000  185.66858  9.43%
         -    0s
     0     0  185.73762    0   86  205.00000  185.73762  9.40%
         -    0s
     0     0  185.86989    0   87  205.00000  185.86989  9.33%
         -    0s
     0     2  185.86989    0   87  205.00000  185.86989  9.33%
         -    0s
H 1246   810                       201.0000000  189.69885  5.62%
   29.8    1s

Cutting planes:
  Gomory: 2
```

```
  MIR: 146
  StrongCG: 1
  Flow cover: 79
  Zero half: 8

Explored 4606 nodes (139611 simplex iterations) in 2.33 seconds
Thread count was 4 (of 4 available processors)

Solution count 6: 201 205 207 ... 244
Pool objective bound 201

Optimal solution found (tolerance 1.00e-04)
Best objective 2.010000000000e+02, best bound 2.010000000000e+02,
   gap 0.0000%



RESULTADO DO FLM:  FLM3158
Ciclo c[1]: 5.000
Ciclo c[2]: 10.000
Ciclo c[3]: 8.000
Ciclo c[4]: 8.000
Ciclo c[5]: 12.000
Ciclo c[6]: 3.000
Ciclo c[7]: 19.000
Ciclo c[8]: 19.000
Ciclo c[9]: 14.000
Ciclo c[10]: 14.000
Ciclo c[11]: 3.000
Ciclo c[12]: 19.000
Ciclo c[13]: 19.000
Ciclo c[14]: 20.000
Ciclo c[15]: 20.000
Ciclo c[16]: 3.000
Ciclo c[17]: 5.000
x[1][1]: 0.000
x[1][2]: 0.000
x[1][3]: -0.000
x[1][4]: -0.000
x[1][5]: 1.000
x[1][6]: 0.000
x[1][7]: 0.000
x[1][8]: -0.000
x[1][9]: -0.000
x[1][10]: 1.000
x[1][11]: 0.000
x[1][12]: 0.000
```

```
x[1][13]:  0.000
x[1][14]:  0.000
x[1][15]:  0.000
x[2][1]:  0.000
x[2][2]:  1.000
x[2][3]:  1.000
x[2][4]:  0.000
x[2][5]:  0.000
x[2][6]:  0.000
x[2][7]:  0.000
x[2][8]:  0.000
x[2][9]:  0.000
x[2][10]:  0.000
x[2][11]:  0.000
x[2][12]:  0.000
x[2][13]:  0.000
x[2][14]:  -0.000
x[2][15]:  0.000
x[3][1]:  -0.000
x[3][2]:  0.000
x[3][3]:  0.000
x[3][4]:  -0.000
x[3][5]:  0.000
x[3][6]:  1.000
x[3][7]:  0.000
x[3][8]:  -0.000
x[3][9]:  -0.000
x[3][10]:  -0.000
x[3][11]:  1.000
x[3][12]:  0.000
x[3][13]:  0.000
x[3][14]:  -0.000
x[3][15]:  0.000
x[4][1]:  0.000
x[4][2]:  0.000
x[4][3]:  0.000
x[4][4]:  0.000
x[4][5]:  0.000
x[4][6]:  0.000
x[4][7]:  0.000
x[4][8]:  -0.000
x[4][9]:  1.000
x[4][10]:  0.000
x[4][11]:  -0.000
x[4][12]:  0.000
x[4][13]:  0.000
x[4][14]:  1.000
```

```
x[4][15]: -0.000
x[5][1]: 0.000
x[5][2]: 0.000
x[5][3]: 0.000
x[5][4]: -0.000
x[5][5]: 0.000
x[5][6]: 0.000
x[5][7]: 1.000
x[5][8]: 1.000
x[5][9]: -0.000
x[5][10]: 0.000
x[5][11]: -0.000
x[5][12]: -0.000
x[5][13]: 0.000
x[5][14]: 0.000
x[5][15]: 0.000
x[6][1]: 1.000
x[6][2]: 0.000
x[6][3]: 0.000
x[6][4]: 1.000
x[6][5]: 0.000
x[6][6]: -0.000
x[6][7]: 0.000
x[6][8]: -0.000
x[6][9]: -0.000
x[6][10]: 0.000
x[6][11]: 0.000
x[6][12]: 0.000
x[6][13]: 0.000
x[6][14]: 0.000
x[6][15]: 0.000
x[7][1]: 0.000
x[7][2]: 0.000
x[7][3]: 0.000
x[7][4]: 0.000
x[7][5]: 0.000
x[7][6]: 0.000
x[7][7]: -0.000
x[7][8]: 0.000
x[7][9]: 0.000
x[7][10]: 0.000
x[7][11]: 0.000
x[7][12]: -0.000
x[7][13]: 0.000
x[7][14]: 0.000
x[7][15]: 1.000
x[8][1]: 0.000
```

```
x[8][2]: 0.000
x[8][3]: 0.000
x[8][4]: 0.000
x[8][5]: -0.000
x[8][6]: 0.000
x[8][7]: -0.000
x[8][8]: 0.000
x[8][9]: 0.000
x[8][10]: -0.000
x[8][11]: 0.000
x[8][12]: 1.000
x[8][13]: 1.000
x[8][14]: 0.000
x[8][15]: 0.000

================================
No Ciclo C[1] o produto 6 é processado na maquina 1 em 5 tempos
================================
No Ciclo C[2] o produto 2 é processado na maquina 1 em 6 tempos
No Ciclo C[2] o produto 6 é processado na maquina 2 em 10 tempos
================================
No Ciclo C[3] o produto 2 é processado na maquina 1 em 6 tempos
No Ciclo C[3] o produto 2 é processado na maquina 2 em 8 tempos
No Ciclo C[3] o produto 6 é processado na maquina 3 em 3 tempos
================================
No Ciclo C[4] o produto 6 é processado na maquina 1 em 5 tempos
No Ciclo C[4] o produto 2 é processado na maquina 2 em 8 tempos
No Ciclo C[4] o produto 2 é processado na maquina 3 em 6 tempos
================================
No Ciclo C[5] o produto 1 é processado na maquina 1 em 12 tempos
No Ciclo C[5] o produto 6 é processado na maquina 2 em 10 tempos
No Ciclo C[5] o produto 2 é processado na maquina 3 em 6 tempos
================================
No Ciclo C[6] o produto 3 é processado na maquina 1 em 1 tempos
No Ciclo C[6] o produto 1 é processado na maquina 2 em 3 tempos
No Ciclo C[6] o produto 6 é processado na maquina 3 em 3 tempos
================================
No Ciclo C[7] o produto 5 é processado na maquina 1 em 14 tempos
No Ciclo C[7] o produto 3 é processado na maquina 2 em 19 tempos
No Ciclo C[7] o produto 1 é processado na maquina 3 em 13 tempos
================================
No Ciclo C[8] o produto 5 é processado na maquina 1 em 14 tempos
No Ciclo C[8] o produto 5 é processado na maquina 2 em 11 tempos
No Ciclo C[8] o produto 3 é processado na maquina 3 em 19 tempos
================================
No Ciclo C[9] o produto 4 é processado na maquina 1 em 14 tempos
No Ciclo C[9] o produto 5 é processado na maquina 2 em 11 tempos
```

```
No Ciclo C[9] o produto 5 é processado na maquina 3 em 14 tempos
================================
No Ciclo C[10] o produto 1 é processado na maquina 1 em 12 tempos
No Ciclo C[10] o produto 4 é processado na maquina 2 em 10 tempos
No Ciclo C[10] o produto 5 é processado na maquina 3 em 14 tempos
================================
No Ciclo C[11] o produto 3 é processado na maquina 1 em 1 tempos
No Ciclo C[11] o produto 1 é processado na maquina 2 em 3 tempos
No Ciclo C[11] o produto 4 é processado na maquina 3 em 3 tempos
================================
No Ciclo C[12] o produto 8 é processado na maquina 1 em 9 tempos
No Ciclo C[12] o produto 3 é processado na maquina 2 em 19 tempos
No Ciclo C[12] o produto 1 é processado na maquina 3 em 13 tempos
================================
No Ciclo C[13] o produto 8 é processado na maquina 1 em 9 tempos
No Ciclo C[13] o produto 8 é processado na maquina 2 em 18 tempos
No Ciclo C[13] o produto 3 é processado na maquina 3 em 19 tempos
================================
No Ciclo C[14] o produto 4 é processado na maquina 1 em 14 tempos
No Ciclo C[14] o produto 8 é processado na maquina 2 em 18 tempos
No Ciclo C[14] o produto 8 é processado na maquina 3 em 20 tempos
================================
No Ciclo C[15] o produto 7 é processado na maquina 1 em 20 tempos
No Ciclo C[15] o produto 4 é processado na maquina 2 em 10 tempos
No Ciclo C[15] o produto 8 é processado na maquina 3 em 20 tempos
================================
No Ciclo C[16] o produto 7 é processado na maquina 2 em 2 tempos
No Ciclo C[16] o produto 4 é processado na maquina 3 em 3 tempos
================================
No Ciclo C[17] o produto 7 é processado na maquina 3 em 5 tempos


Valor da Funcao Objetivo: 201.000


------Mais informacoes sobre o Modelo------

Status da solucao: (2 - otimo) | (3 - inviavel) | (5 - ilimitado):
   2
Tempo de otimização: 2.334 segundos
Numero total de variaveis: 137
--Numero de variaveis Binarias: 120
Numero de nos da arvore B&B explorados: 4606.
```

Saída do Lagrangeano Relaxado:

```
RESULTADO DO FLM:  FLM3103
Tempo de Otimização: 0.1026 segundos
```

```
LB: 167.20
UB: 168.00

C=[0]: 1.00
C=[1]: 15.00
C=[2]: 16.00
C=[3]: 14.00
C=[4]: 16.00
C=[5]: 16.00
C=[6]: 14.00
C=[7]: 16.00
C=[8]: 16.00
C=[9]: 16.00
C=[10]: 14.00
C=[11]: 14.00

x[1][1]: -0.00
x[1][2]: -0.00
x[1][3]: 1.00
x[1][4]: 1.00
x[1][5]: -0.00
x[1][6]: -0.00
x[1][7]: 1.00
x[1][8]: 1.00
x[1][9]: 0.00
x[1][10]: -0.00
x[2][1]: 1.00
x[2][2]: 1.00
x[2][3]: -0.00
x[2][4]: -0.00
x[2][5]: 0.00
x[2][6]: 1.00
x[2][7]: -0.00
x[2][8]: -0.00
x[2][9]: -0.00
x[2][10]: 1.00
x[3][1]: -0.00
x[3][2]: -0.00
x[3][3]: -0.00
x[3][4]: -0.00
x[3][5]: 1.00
x[3][6]: 0.00
x[3][7]: -0.00
x[3][8]: -0.00
x[3][9]: 1.00
x[3][10]: 0.00

```

```
Cmax: 168.00

================================
No Ciclo C[1] o produto 2 é processado na maquina 1 em 1 tempos
================================
No Ciclo C[2] o produto 2 é processado na maquina 1 em 1 tempos
No Ciclo C[2] o produto 2 é processado na maquina 2 em 15 tempos
================================
No Ciclo C[3] o produto 1 é processado na maquina 1 em 15 tempos
No Ciclo C[3] o produto 2 é processado na maquina 2 em 15 tempos
No Ciclo C[3] o produto 2 é processado na maquina 3 em 16 tempos
================================
No Ciclo C[4] o produto 1 é processado na maquina 1 em 15 tempos
No Ciclo C[4] o produto 1 é processado na maquina 2 em 5 tempos
No Ciclo C[4] o produto 2 é processado na maquina 3 em 16 tempos
================================
No Ciclo C[5] o produto 3 é processado na maquina 1 em 8 tempos
No Ciclo C[5] o produto 1 é processado na maquina 2 em 5 tempos
No Ciclo C[5] o produto 1 é processado na maquina 3 em 14 tempos
================================
No Ciclo C[6] o produto 2 é processado na maquina 1 em 1 tempos
No Ciclo C[6] o produto 3 é processado na maquina 2 em 10 tempos
No Ciclo C[6] o produto 1 é processado na maquina 3 em 14 tempos
================================
No Ciclo C[7] o produto 1 é processado na maquina 1 em 15 tempos
No Ciclo C[7] o produto 2 é processado na maquina 2 em 15 tempos
No Ciclo C[7] o produto 3 é processado na maquina 3 em 16 tempos
================================
No Ciclo C[8] o produto 1 é processado na maquina 1 em 15 tempos
No Ciclo C[8] o produto 1 é processado na maquina 2 em 5 tempos
No Ciclo C[8] o produto 2 é processado na maquina 3 em 16 tempos
================================
No Ciclo C[9] o produto 3 é processado na maquina 1 em 8 tempos
No Ciclo C[9] o produto 1 é processado na maquina 2 em 5 tempos
No Ciclo C[9] o produto 1 é processado na maquina 3 em 14 tempos
================================
No Ciclo C[10] o produto 2 é processado na maquina 1 em 1 tempos
No Ciclo C[10] o produto 3 é processado na maquina 2 em 10 tempos
No Ciclo C[10] o produto 1 é processado na maquina 3 em 14 tempos
================================
No Ciclo C[11] o produto 2 é processado na maquina 2 em 15 tempos
No Ciclo C[11] o produto 3 é processado na maquina 3 em 16 tempos
================================
No Ciclo C[12] o produto 2 é processado na maquina 3 em 16 tempos
```

ANEXO B

INSTÂNCIAS DO FLM

O Formato dos arquivos é:

```
Linha 1: Número de Tarefas
Linha 2: Número de Estações
Linha 3: Número de Conjuntos

Vetor de Demanda

Matriz de Processamento do conjunto $r$ na estação $m$

Nome da instância
```

```
10
3
3

4
4
2

15      5       14
1       15      16
8       10      16

FLM3103
```

```
10
3
5

2
2
2
2
```

```
2

8       6       13
15      19      8
11      19      7
19      8       17
7       8       12

FLM3105
```

```
10
3
10

1
1
1
1
1
1
1
1
1
1

20      5       20
7       8       13
15      15      12
16      1       8
11      6       10
20      16      6
8       20      19
13      16      2
10      16      11
16      1       8

FLM31010
```

```
15
3
4

4
4
3
4

15      4       10
```

```
8         3         1
13        3         17
17        4         3

FLM3154
```

```
15
3
8

2
2
2
2
2
2
1
2

12        3         13
6         8         6
1         19        19
14        10        3
14        11        14
5         10        3
20        2         5
9         18        20

FLM3158
```

```
15
3
15

1
1
1
1
1
1
1
1
1
1
1
1
1
1
```

```
1
3       11      20
7       7       12
10      9       19
7       6       6
9       20      12
5       18      3
4       17      5
16      2       12
11      8       13
16      1       6
3       13      8
12      18      18
3       3       17
16      13      8
19      5       6

FLM31515
```

```
20
3
5

4
4
4
4
4

16      11      20
17      2       12
1       11      16
18      5       5
13      3       11

FLM3205
```

```
20
3
10

2
2
2
1
3
2
```

```
1
1
4
2

19      19      14
13      13      5
12      6       17
13      4       15
17      7       10
13      15      20
2       15      17
11      16      11
14      11      9
8       8       11

FLM32010
```

```
20
3
20

1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1

13      1       2
10      7       9
8       4       3
17      16      20
12      7       1
```

```
11      11      16
19      4       17
6       13      18
16      6       2
16      14      8
8       14      6
12      15      17
2       10      9
2       2       19
11      5       4
16      19      6
19      4       3
3       17      3
12      11      18
10      20      12

FLM32020
```

```
10
5
3

4
4
2

13      9       3       9       19
8       2       4       1       10
11      5       5       19      10

FLM5103
```

```
10
5
5

2
2
2
2
2

8       19      8       13      6
5       20      17      15      15
9       12      1       13      4
2       2       1       10      14
3       5       4       11      4

```

```
FLM5105
```

```
10
5
10

1
1
1
1
1
1
1
1
1
1

11      18      5       9       6
11      12      4       3       7
17      13      5       6       9
16      12      9       9       11
13      5       7       12      2
8       7       19      6       6
17      10      9       13      17
11      5       4       15      1
8       17      19      5       19
19      4       20      3       15

FLM51010
```

```
15
5
4

4
4
4
3

20      10      8       19      7
11      13      20      16      14
11      14      1       2       3
5       8       18      6       15

FLM5154
```

```
15
5
```

```
8

4
2
1
2
1
2
1
2

14      13      10      1       14
4       18      4       2       11
1       17      20      11      20
15      12      15      2       13
11      4       11      17      17
10      5       10      17      10
19      18      2       15      9
13      1       14      3       17

FLM5158
```

```
15
5
15

1
1
1
1
1
1
1
1
1
1
1
1
1
1
1

6       14      9       5       19
4       2       10      19      5
6       6       16      6       15
9       5       7       16      5
11      14      16      4       3
10      17      10      6       13
```

```
18        7         1         2         10
11        16        4         12        10
19        14        15        14        14
13        1         10        11        16
20        13        4         9         8
5         8         7         13        14
14        19        13        13        9
6         1         4         14        17
14        10        15        13        17

FLM51515
```

```
20
5
5

4
2
4
6
4

6         10        15        3         16
7         13        11        2         13
3         11        20        9         16
19        13        5         9         19
13        11        3         8         20

FLM5205
```

```
20
5
10

2
2
1
1
4
2
2
2
2
2

15        14        6         8         18
11        8         4         5         20
7         15        17        16        11
```

```
3       8       9       19      18
12      14      18      7       12
6       15      8       14      4
1       9       16      9       4
16      1       8       17      9
5       7       17      16      15
9       9       16      4       17

FLM52010
```

```
20
5
20

1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1

1       18      3       9       4
18      20      5       5       8
13      16      3       19      3
20      12      4       2       1
11      19      1       3       19
10      12      13      3       7
17      1       6       4       6
5       3       11      13      7
10      18      14      12      10
19      10      10      2       13
12      17      11      19      1
17      5       9       15      17
15      12      3       15      12
```

```
12      13      10      2       18
5       1       18      18      7
14      13      18      19      9
2       8       6       20      2
13      1       5       18      4
14      10      12      16      14
15      4       13      11      7

FLM52020
```

```
10
8
3

4
4
2

11      6       16      4       11      5       10      15
15      9       17      8       7       19      19      15
20      10      3       2       4       14      3       12

FLM8103
```

```
10
8
5

2
2
2
2
2

18      1       10      2       13      11      15      11
2       18      3       6       2       3       1       15
5       4       20      1       2       17      1       15
2       2       7       11      16      7       14      16
9       7       6       16      19      6       13      6

FLM8105
```

```
10
8
10

1
1
```

```
1
1
1
1
1
1
1
1

11      14      15      16      10      3       3       14
10      3       5       8       4       16      19      9
18      15      15      5       16      6       4       10
16      3       20      2       3       13      6       13
15      3       18      16      6       20      16      2
2       13      2       5       5       9       10      7
2       7       8       8       11      14      16      16
2       14      8       12      2       16      8       14
16      15      14      5       9       9       6       3
19      12      12      13      3       14      1       3

FLM81010
```

```
15
8
4

6
4
2
3

15      3       4       5       12      19      4       6
11      3       4       6       4       15      13      8
3       2       7       18      5       12      20      2
13      3       7       15      2       7       4       14

FLM8154
```

```
15
8
8

1
2
2
1
4
2
```

```
2
1

13      11      20      4       6       15      15      10
9       13      5       15      19      7       17      5
8       20      14      8       5       14      8       17
17      9       13      17      8       8       13      20
7       2       8       15      2       13      12      1
17      18      3       12      13      1       11      11
16      13      1       4       4       19      6       2
18      8       9       20      1       17      5       17

FLM8158
```

```
15
8
15

1
1
1
1
1
1
1
1
1
1
1
1
1
1

13      9       9       5       18      11      15      10
19      13      5       6       2       9       13      14
8       16      3       14      1       20      17      20
9       12      7       10      12      17      8       7
20      12      15      13      16      10      15      17
19      12      16      5       7       18      17      15
14      11      14      4       4       3       7       20
20      2       1       17      7       8       12      1
16      15      17      16      5       19      20      8
7       20      19      19      11      19      11      14
14      8       16      3       19      15      7       6
5       20      1       4       13      13      13      5
6       7       8       2       3       7       8       15
14      18      15      10      8       19      16      13
```

```
11      10      15      4       2       3       9       12

FLM81515
```

```
20
8
5

4
4
4
4
4

15      17      11      20      12      1       9       4
18      3       10      1       14      2       3       7
6       12      18      20      8       20      6       3
15      8       8       4       13      14      6       18
3       17      9       14      17      5       7       2

FLM8205
```

```
20
8
10

2
2
4
1
1
2
2
2
2
2

7       6       11      1       19      17      9       8
1       11      13      12      11      17      14      10
4       20      7       7       15      8       19      5
15      15      3       19      12      12      17      16
15      17      10      20      1       18      10      18
18      9       8       6       9       19      16      19
12      10      16      17      13      14      9       12
2       12      16      18      11      5       20      12
19      6       14      12      8       14      20      3
17      15      3       18      19      2       18      18

```

```
FLM82010
```

```
20
8
20

1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1

12      11      20      11      8       14      3       19
7       14      12      13      10      11      12      14
10      20      11      17      8       5       1       10
15      15      7       11      16      12      16      17
18      9       9       5       15      3       17      11
15      17      10      10      9       14      19      12
1       3       2       9       14      12      20      14
14      2       18      20      19      2       11      8
9       2       2       13      16      2       6       5
9       4       9       14      15      4       3       12
3       7       17      15      3       1       11      18
17      7       8       7       8       9       12      9
7       1       13      11      12      17      16      3
5       11      17      12      10      13      2       9
7       2       18      4       2       11      14      7
8       3       19      12      5       18      11      9
11      13      4       14      17      2       4       17
12      18      6       9       1       19      19      9
8       20      18      17      18      3       12      8
8       12      12      15      2       11      9       8

```

```
FLM82020

30
3
8

3
4
5
4
3
4
3
4

1       17      14
7       15      9
19      19      15
11      20      7
9       4       15
2       12      13
6       3       2
14      5       4

FLM3308
```

```
30
3
15

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

7       14      8
7       9       15
```

```
13      14      15
2       11      4
15      5       9
19      20      9
18      18      8
15      6       3
18      17      2
1       20      17
9       8       4
5       5       1
8       9       15
10      7       18
16      7       20

FLM33015
```

```
30
3
30

1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
```

```
1
1

3       14      14
5       12      18
11      20      2
3       3       13
20      12      1
2       9       19
6       19      5
13      16      1
20      20      19
7       12      7
1       18      7
14      10      4
13      15      8
17      12      17
7       9       9
17      9       17
16      4       8
6       7       2
6       11      17
15      4       4
14      16      18
9       3       3
16      11      20
5       13      15
19      1       19
8       13      4
1       20      17
11      10      13
11      11      15
5       7       17

FLM33030
```

```
60
3
15

4
4
4
4
4
4
4
4
```

```
4
4
4
4
4
4
4

19        12        9
15        17        17
15        17        1
9         19        18
6         17        1
2         11        10
5         18        13
17        7         7
15        14        17
11        17        19
7         7         2
16        11        10
1         5         2
10        5         1
14        11        10

FLM36015
```

```
60
3
30

2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
```

```
2
2
2
2
2
2
2
2
2
2
2
2

14        16        20
9         6         11
19        9         13
14        11        1
13        20        2
20        9         17
5         15        16
5         13        4
1         15        14
5         17        16
11        17        7
20        16        4
20        17        2
4         5         4
18        9         17
1         9         9
7         8         7
10        12        19
15        16        10
8         19        4
4         16        2
19        7         11
5         10        3
8         11        16
19        19        15
13        8         15
11        8         6
11        12        5
20        12        7
20        16        13

FLM36030
```

```
60
3
```

```
60

1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
```

```
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1

10      3       7
15      19      15
9       9       11
13      8       4
5       17      8
16      4       3
4       2       20
3       15      12
13      6       11
2       8       15
7       14      8
19      18      3
1       14      16
18      1       5
7       2       5
6       10      11
6       5       1
18      20      2
1       10      14
3       14      11
4       10      11
9       5       11
1       11      5
4       20      13
16      9       19
19      19      6
2       18      6
9       2       14
18      16      4
1       12      9
4       4       18
```

```
13        17        11
15        15        16
12        18        18
8         17        12
7         1         4
2         19        10
15        20        19
7         16        9
7         5         6
11        7         17
6         3         13
8         19        1
15        15        3
14        14        19
3         4         19
3         8         19
16        11        9
19        11        14
5         5         16
18        11        2
13        7         20
7         13        3
14        7         3
13        2         3
16        20        17
17        7         18
16        18        3
4         19        16
17        7         16

FLM36060
```

```
120
3
30

4
4
4
4
2
4
2
4
4
6
4
6
```

```
4
2
4
4
4
8
4
2
4
2
4
4
4
4
2
6
4
6

2          19         1
7          9          13
17         20         11
3          7          19
17         6          18
12         7          13
14         18         8
4          7          20
17         9          11
2          17         13
7          6          16
10         2          11
18         6          7
1          11         1
6          5          10
16         15         15
19         6          1
19         8          18
12         7          12
7          11         2
2          12         12
14         8          20
20         4          4
13         20         6
18         16         9
4          17         3
7          3          16
11         12         15
```

```
15       9        15
16       1        6

FLM31203
```

```
120
3
60

2
2
2
2
2
2
2
2
2
1
2
2
2
2
2
2
1
2
2
2
2
2
2
2
2
1
2
2
2
4
2
2
2
1
2
2
2
2
2
```

```
2
2
2
1
2
2
1
2
6
2
2
2
2
2
2
2
2
2
2
2

4       20      3
15      19      13
7       7       7
20      10      15
9       7       9
17      5       14
16      16      15
10      4       14
12      11      2
17      6       7
13      18      16
19      9       1
2       10      16
5       2       8
8       16      12
18      12      7
12      4       18
8       9       10
2       11      19
7       9       3
5       12      20
9       5       6
15      16      5
14      5       11
5       20      2
```

```
11        15        2
16        6         18
8         16        13
4         7         5
15        13        20
2         13        16
3         11        5
20        2         1
8         10        16
3         1         19
5         7         5
20        20        15
11        11        8
2         14        2
19        16        12
4         3         13
4         19        14
17        13        20
12        19        1
20        14        19
6         8         5
5         10        11
12        14        13
11        10        2
15        20        10
17        16        20
12        17        4
15        20        1
18        17        15
12        17        17
5         10        8
20        5         18
18        1         18
3         1         4
15        7         17

FLM312060
```

```
120
3
120

1
1
1
1
1
1
```

11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	1
21	1
22	1
23	1
24	1
25	1
26	1
27	1
28	1
29	1
30	1
31	1
32	1
33	1
34	1
35	1
36	1
37	1
38	1
39	1
40	1
41	1
42	1
43	1
44	1
45	1
46	1
47	1
48	1
49	1
50	1
51	1
52	1
53	1
54	1
55	1
56	1
57	1

58	1
59	1
60	1
61	1
62	1
63	1
64	1
65	1
66	1
67	1
68	1
69	1
70	1
71	1
72	1
73	1
74	1
75	1
76	1
77	1
78	1
79	1
80	1
81	1
82	1
83	1
84	1
85	1
86	1
87	1
88	1
89	1
90	1
91	1
92	1
93	1
94	1
95	1
96	1
97	1
98	1
99	1
100	1
101	1
102	1
103	1
104	1

```
105 1
106 1
107 1
108 1
109 1
110 1
111 1
112 1
113 1
114 1
115 1
116 1
117 1
118 1
119 1
120 1
121 1
122 1
123 1
124 1
125 1
126
127 5          11         3
128 16         16         9
129 2          3          15
130 5          3          9
131 14         14         6
132 4          4          10
133 8          10         11
134 2          13         18
135 18         8          18
136 11         16         18
137 20         17         18
138 11         10         20
139 16         15         18
140 15         16         16
141 5          1          17
142 13         18         14
143 3          17         5
144 8          14         6
145 2          18         8
146 8          12         18
147 16         19         9
148 7          13         2
149 10         14         20
150 14         10         1
151 13         7          14
```

152	18	3	20
153	1	9	8
154	18	2	10
155	14	14	3
156	13	4	2
157	5	19	17
158	7	14	1
159	19	12	4
160	2	18	17
161	13	12	5
162	14	20	17
163	13	17	3
164	17	14	6
165	2	4	5
166	2	10	19
167	11	11	14
168	19	10	20
169	3	7	17
170	16	1	8
171	13	6	1
172	15	19	10
173	8	2	20
174	13	14	11
175	4	16	18
176	3	6	5
177	13	17	19
178	19	5	18
179	1	4	12
180	14	6	4
181	3	11	4
182	4	4	8
183	1	12	1
184	9	14	16
185	5	8	9
186	10	8	2
187	2	1	7
188	14	17	10
189	9	6	18
190	9	13	10
191	10	3	14
192	20	17	20
193	2	17	8
194	4	12	7
195	10	4	12
196	18	11	3
197	6	8	6
198	18	18	5

199	7	3	13
200	15	16	9
201	2	12	8
202	5	10	19
203	20	10	16
204	15	7	16
205	20	19	15
206	12	18	2
207	19	7	8
208	18	7	11
209	8	1	14
210	7	14	14
211	4	13	3
212	18	7	17
213	9	18	2
214	19	1	4
215	14	15	3
216	7	18	13
217	6	13	12
218	12	16	15
219	6	13	12
220	17	18	12
221	10	4	19
222	3	5	5
223	16	14	13
224	17	12	3
225	10	20	7
226	2	14	15
227	16	10	19
228	14	8	12
229	2	4	8
230	20	18	15
231	9	9	7
232	15	12	12
233	14	14	2
234	10	14	16
235	11	14	16
236	20	9	12
237	10	15	17
238	20	15	8
239	14	9	19
240	5	20	18
241	1	3	7
242	4	8	20
243	2	13	11
244	3	9	13
245	7	14	3

```
10        5         16

FLM3120120
```

```
30
5
8

3
4
5
4
3
4
3
4

9         2         20        4         19
11        12        11        12        14
19        10        15        8         20
12        18        20        15        5
19        1         18        10        5
7         4         1         9         3
13        15        16        11        6
11        20        12        19        1

FLM5308
```

```
30
5
15

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

```

```
10      5       11      20      1
18      1       18      17      6
1       14      10      5       14
1       11      13      2       15
10      16      15      4       5
5       11      20      15      18
15      3       16      16      2
8       1       20      16      13
15      13      13      20      12
9       19      13      8       13
17      5       9       19      7
14      13      20      9       1
19      16      12      1       10
13      3       1       4       5
18      8       5       13      3

FLM53015
```

```
30
5
30

1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
```

```
1
1
1
1

20      19      16      19      20
12      5       15      6       10
12      8       18      15      11
12      17      1       20      9
18      18      19      7       10
8       20      19      16      20
6       7       18      8       18
17      11      16      17      4
18      7       4       15      6
17      15      18      11      13
19      3       19      8       8
17      1       5       3       10
14      11      6       12      10
2       18      9       5       8
15      7       2       13      16
4       18      13      4       1
7       3       5       16      9
8       17      19      18      10
18      16      14      11      3
9       19      20      17      17
20      8       5       2       8
15      20      20      20      13
10      1       9       6       7
14      13      19      6       18
15      7       2       13      2
9       16      15      18      4
4       15      5       13      20
18      5       6       16      8
10      19      4       9       19
18      3       19      12      4

FLM53030
```

```
60
5
15

2
4
4
4
6
4
```

```
6
4
1
1
4
4
6
4
6

19      20      16      5       12
11      3       6       11      9
17      7       20      2       1
2       13      13      4       13
2       8       18      1       11
8       2       11      13      4
11      5       12      12      13
1       4       13      10      16
19      17      7       5       12
1       9       18      3       13
20      18      5       4       19
12      15      2       17      17
10      4       9       20      12
19      12      14      17      7
15      8       10      12      10

FLM56015
```

```
60
5
30

2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
```

```
2
2
2
2
2
2
2
2
2
2
2
2
2
2

16      4       1       16      20
8       4       11      14      13
9       7       17      12      1
10      13      10      6       7
17      20      14      19      20
9       12      2       20      18
6       6       7       4       10
11      11      19      4       8
7       8       9       13      8
2       18      1       5       4
13      5       16      14      14
18      3       14      14      3
16      16      16      18      15
5       18      4       4       3
3       8       2       1       9
12      15      17      20      2
18      12      15      14      11
4       8       13      14      8
13      8       12      8       6
5       11      6       16      12
1       6       6       12      7
17      9       12      15      12
16      9       12      4       8
7       2       2       3       12
15      3       9       2       18
13      19      6       16      4
20      20      20      11      13
8       4       15      20      16
4       12      15      19      6
8       14      12      18      15

FLM56030
```

```
60
5
60

1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
```

```
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1

20      5       19      13      17
12      5       14      8       1
11      3       3       8       2
4       20      15      18      12
9       7       8       16      11
13      1       14      11      9
20      15      7       15      6
3       16      7       1       12
15      9       4       4       20
6       8       20      7       6
2       8       20      9       3
3       5       18      12      3
5       4       14      19      15
7       11      3       13      14
5       16      9       14      13
2       4       1       18      19
1       5       7       3       4
8       18      14      10      13
13      13      17      15      13
6       4       11      16      5
11      11      13      4       3
17      20      18      4       7
10      7       20      11      16
15      14      14      18      1
17      13      6       15      8
9       1       3       5       4
16      10      11      19      16
7       11      16      17      11
7       9       11      1       7
```

```
15      9       1       11      8
9       3       3       6       5
19      3       17      6       4
4       16      4       7       15
16      4       16      5       10
10      11      20      2       2
10      15      11      14      1
12      9       4       6       9
18      1       7       18      15
1       16      17      19      13
1       13      9       14      20
4       14      19      13      13
1       10      5       17      14
14      6       10      4       13
2       9       12      6       2
13      16      17      1       1
13      7       2       16      4
14      6       11      6       15
15      20      14      8       7
17      12      19      8       4
3       12      19      2       17
12      7       19      16      5
3       11      19      1       12
13      20      1       17      15
12      14      18      7       19
9       5       15      11      17
15      18      15      15      20
9       14      5       3       17
10      17      15      4       13
17      2       3       12      3
16      1       20      3       2

FLM56060
```

```
120
5
30

4
4
4
4
2
4
2
4
4
6
```

```
4
6
4
2
4
4
4
8
4
2
4
2
4
4
4
4
2
6
4
6

4 18 5 3 15
14 6 7 10 9
6 9 15 12 5
15 11 16 14 1
12 9 15 7 11
2 3 3 14 15
7 12 19 2 8
18 4 19 17 6
13 5 19 15 2
6 19 12 10 3
11 10 17 18 14
4 9 2 9 18
15 10 11 11 5
16 2 11 12 15
18 11 7 5 1
2 13 1 8 17
8 15 7 1 4
12 11 11 8 7
11 1 2 14 5
14 19 12 16 17
15 11 14 2 3
8 10 1 8 4
18 11 18 17 6
3 10 18 5 5
6 3 9 5 4
11 17 13 15 9
```

```
17      15      16      10      8
12      14      10      7       18
11      8       15      5       11
11      10      14      9       1

FLM512030
```

```
120
5
60

2
2
2
2
2
2
2
2
2
1
2
2
2
2
2
2
2
1
2
2
2
2
2
2
2
1
2
2
2
```

```
2
2
2
2
2
1
2
2
1
2
6
2
2
2
2
2
2
2
2
2
2
2

10      8       3       14      2
11      2       6       14      1
1       16      15      4       3
20      19      9       11      9
18      17      2       1       8
5       7       6       20      1
14      19      3       10      18
12      12      20      1       8
3       1       1       2       12
9       15      16      18      5
7       4       10      10      19
18      9       14      20      18
2       10      17      10      5
16      4       10      8       20
7       8       9       7       12
2       3       16      14      1
5       10      1       7       14
5       10      19      20      7
2       15      2       2       13
4       7       1       1       13
2       15      3       14      6
4       14      8       1       20
3       17      10      10      14
```

```
7       5       5       6       7
6       19      9       14      2
13      13      14      13      4
20      4       12      18      4
17      5       11      10      15
19      14      4       8       20
15      11      16      5       19
2       14      16      6       14
5       12      16      4       17
15      16      9       2       9
1       3       18      20      2
19      6       2       6       4
3       13      18      2       8
7       3       13      10      20
11      7       9       16      7
8       2       10      3       12
19      11      20      6       3
15      19      18      15      13
3       6       2       20      4
20      10      19      5       14
11      9       9       7       6
20      10      12      7       12
12      19      14      5       18
12      12      7       19      5
13      12      9       10      14
11      4       3       12      11
6       14      3       15      8
6       12      10      6       16
11      19      5       14      2
9       11      7       5       6
11      12      5       12      17
12      18      3       4       20
17      9       13      16      10
3       17      2       7       10
6       5       12      11      8
9       19      1       2       1
13      8       12      15      13

FLM512060
```

```
120
5
120

1
1
1
1
```

9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	1
21	1
22	1
23	1
24	1
25	1
26	1
27	1
28	1
29	1
30	1
31	1
32	1
33	1
34	1
35	1
36	1
37	1
38	1
39	1
40	1
41	1
42	1
43	1
44	1
45	1
46	1
47	1
48	1
49	1
50	1
51	1
52	1
53	1
54	1
55	1

56	1
57	1
58	1
59	1
60	1
61	1
62	1
63	1
64	1
65	1
66	1
67	1
68	1
69	1
70	1
71	1
72	1
73	1
74	1
75	1
76	1
77	1
78	1
79	1
80	1
81	1
82	1
83	1
84	1
85	1
86	1
87	1
88	1
89	1
90	1
91	1
92	1
93	1
94	1
95	1
96	1
97	1
98	1
99	1
100	1
101	1
102	1

```
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1

12      4       4       14      5
13      17      2       10      10
2       9       16      1       19
4       10      20      14      12
15      6       17      20      13
5       17      4       8       15
20      6       19      13      12
6       20      16      7       3
5       5       3       7       20
12      18      11      14      9
11      20      3       9       13
5       2       16      11      18
20      11      17      19      3
12      20      18      8       16
3       11      9       12      11
14      11      11      6       10
19      20      1       17      20
1       7       4       15      18
15      8       3       5       14
9       16      2       8       3
16      20      17      11      7
10      6       13      9       10
3       20      9       3       5
```

150	7	7	6	10	12
151	9	9	10	9	12
152	8	1	13	8	1
153	20	1	12	19	12
154	3	15	13	5	7
155	7	13	6	16	4
156	9	18	13	13	13
157	19	19	15	17	3
158	3	12	16	18	14
159	5	11	3	16	18
160	4	13	19	16	20
161	15	7	15	12	16
162	13	18	12	11	4
163	14	11	12	18	16
164	12	18	6	2	6
165	19	11	9	15	12
166	2	14	8	12	19
167	12	19	10	17	2
168	6	20	10	19	3
169	13	1	11	19	17
170	7	10	20	6	17
171	20	5	19	5	3
172	16	6	2	7	15
173	20	13	9	19	3
174	2	3	8	15	9
175	1	10	20	17	20
176	20	7	2	6	10
177	7	12	13	5	16
178	3	3	10	20	1
179	17	15	9	3	6
180	13	19	1	7	9
181	9	19	1	14	1
182	5	20	9	13	12
183	19	19	19	9	5
184	8	17	14	8	17
185	9	13	8	17	3
186	15	18	6	20	7
187	3	11	17	19	4
188	20	1	14	5	17
189	6	15	7	15	18
190	10	10	13	19	2
191	17	6	14	6	2
192	20	19	5	11	18
193	1	14	7	3	6
194	20	20	15	5	14
195	18	7	20	3	13
196	15	18	7	12	9

197	14	7	5	19	9
198	20	10	19	12	2
199	8	5	11	20	6
200	4	3	3	14	4
201	9	4	14	5	5
202	11	7	20	7	2
203	7	20	10	7	11
204	17	3	12	14	13
205	11	8	15	4	17
206	9	19	16	19	17
207	17	14	10	2	9
208	2	18	6	19	2
209	14	20	8	8	1
210	12	18	2	19	4
211	10	9	3	1	14
212	13	15	10	6	2
213	8	16	5	12	17
214	1	14	5	1	19
215	13	18	7	10	13
216	18	3	13	8	4
217	10	12	18	12	13
218	13	18	18	10	17
219	18	16	11	4	6
220	4	1	18	3	3
221	5	10	12	17	7
222	10	9	12	5	20
223	5	8	12	11	14
224	13	11	20	4	16
225	15	18	3	18	2
226	14	14	2	9	17
227	11	5	3	17	14
228	20	8	18	1	4
229	9	14	4	13	5
230	8	1	4	17	3
231	7	11	18	19	19
232	13	15	10	3	6
233	8	15	4	18	7
234	1	5	6	9	13
235	12	19	19	4	17
236	6	5	12	14	7
237	16	5	6	7	7
238	12	19	6	2	18
239	12	20	2	20	12
240	8	10	15	8	20
241	15	9	2	8	4
242	8	16	6	14	8
243	5	13	18	11	14

```
20      9       9       7       20
20      14      11      5       4
16      19      6       14      7

FLM5120120
```

```
30
8
8

3
3
6
3
4
3
5
3

6       13      1       5       13      14      16      11
6       18      13      16      6       20      13      13
4       7       2       6       5       3       14      5
16      15      1       5       9       15      2       3
8       11      13      20      12      9       7       19
5       3       14      14      7       8       13      4
20      11      11      2       15      16      16      2
16      5       9       2       12      17      3       1

FLM8308
```

```
30
8
15

2
2
2
1
2
1
2
1
2
1
4
2
2
4
```

```
2

6       1       2       18      7       4       3       11
1       7       18      13      15      4       10      18
8       15      13      13      9       18      13      18
4       7       6       15      6       16      2       10
5       16      3       6       4       5       13      17
11      12      5       12      14      14      17      18
13      12      3       11      2       4       15      2
4       1       12      7       1       4       3       4
18      8       3       9       14      8       1       4
4       20      8       13      9       10      9       5
2       10      8       8       5       11      7       9
12      16      10      6       12      6       10      8
19      4       17      18      2       13      13      8
7       11      13      17      11      2       16      15
13      10      5       16      14      12      5       20

FLM83015
```

```
30
8
30

1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
```

```
1
1
1
1
1
1

7       3       6       16      16      10      9       17
14      4       1       20      12      19      10      6
15      5       5       14      6       1       7       12
16      7       13      17      7       13      10      3
14      15      2       16      10      15      4       12
1       11      5       3       2       16      1       3
20      11      7       11      7       16      14      2
15      1       7       9       6       17      5       12
18      14      2       2       5       12      11      8
16      1       2       15      12      17      17      19
17      6       5       12      6       5       3       8
19      6       13      8       14      12      18      1
7       9       1       7       19      18      14      12
3       10      4       7       10      2       14      14
15      13      19      20      9       4       14      13
10      1       15      4       17      1       18      1
2       8       1       11      3       7       18      9
5       16      15      5       15      5       2       20
15      16      6       17      9       9       12      11
6       6       9       20      19      14      20      6
15      18      16      7       8       19      17      8
5       12      11      18      17      12      4       11
20      20      11      7       13      9       1       19
16      19      14      15      16      14      8       5
20      13      5       2       3       7       18      5
11      2       5       16      3       2       18      15
8       12      20      11      16      4       4       6
9       15      20      18      10      10      19      15
6       13      4       10      20      13      12      14
15      18      18      13      6       15      11      4

FLM83030
```

```
60
8
15

4
4
4
4
```

```
4
4
4
4
4
4
4
4
4
4
4

12      4       11      1       9       3       5       11
15      3       3       4       17      7       14      16
5       2       18      5       11      8       13      10
12      12      14      13      20      4       19      13
5       20      10      4       9       18      14      2
4       20      19      13      20      13      12      3
18      17      20      9       6       17      18      3
10      13      20      1       16      12      4       18
6       4       12      3       14      8       6       13
5       2       12      10      19      11      1       17
13      6       1       3       20      11      15      10
12      6       17      15      19      14      11      12
6       20      20      13      13      2       18      11
18      5       15      3       14      8       10      8
5       10      20      1       17      17      20      13

FLM86015
```

```
60
8
30

2
2
2
1
2
1
2
2
2
2
2
2
1
2
```

```
19 2
20 2
21 4
22 2
23 1
24 2
25 2
26 2
27 2
28 2
29 1
30 2
31 1
32 2
33 2
34 6
35
36 3  5  19 1  9  12 1  4
37 13 16 3  10 13 18 4  15
38 9  12 18 14 10 8  14 1
39 18 5  8  1  12 6  11 20
40 15 7  2  14 8  8  8  3
41 2  4  8  16 14 12 3  8
42 11 20 6  2  17 16 14 15
43 20 12 7  20 5  5  17 16
44 8  17 6  13 19 16 1  19
45 8  16 13 2  10 3  14 1
46 6  10 16 12 8  8  20 13
47 9  12 11 18 4  9  3  16
48 7  13 8  10 18 12 16 15
49 12 4  10 5  5  9  15 16
50 14 1  8  5  2  5  9  6
51 12 18 13 11 6  15 16 4
52 11 1  4  19 16 13 11 11
53 12 7  2  1  15 14 12 17
54 3  19 11 11 3  11 19 18
55 11 1  3  2  20 6  13 13
56 8  16 10 8  19 17 1  18
57 6  12 10 3  17 18 1  3
58 2  13 6  5  19 4  1  7
59 13 6  11 3  15 7  12 10
60 16 5  10 12 3  17 2  16
61 4  4  9  14 7  15 18 9
62 5  8  9  2  16 10 12 4
63 18 11 8  19 15 20 3  14
64 13 1  10 6  4  8  11 11
65 9  11 15 19 17 7  7  2
```

```
66
67 FLM86030
```

```
1  60
2  8
3  60
4
5  1
6  1
7  1
8  1
9  1
10 1
11 1
12 1
13 1
14 1
15 1
16 1
17 1
18 1
19 1
20 1
21 1
22 1
23 1
24 1
25 1
26 1
27 1
28 1
29 1
30 1
31 1
32 1
33 1
34 1
35 1
36 1
37 1
38 1
39 1
40 1
41 1
42 1
43 1
44 1
45 1
```

```
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1

17 16 18 17 1 14 1 18
1 14 8 6 9 2 19 7
4 8 12 6 16 13 16 18
11 5 18 5 18 20 17 1
19 8 10 2 19 20 12 12
18 20 18 16 15 18 13 1
11 12 3 19 12 13 12 12
14 4 20 13 7 15 5 1
7 20 12 14 17 12 1 13
15 3 4 16 1 12 15 5
17 19 19 12 7 8 5 16
20 4 20 1 10 12 7 5
15 14 14 17 20 12 11 11
3 2 15 5 13 5 6 18
7 19 18 19 10 16 10 17
6 15 12 2 10 12 15 4
15 9 1 2 5 11 13 12
5 11 13 12 17 16 7 9
17 15 3 17 7 2 11 14
2 9 11 14 14 17 20 4
3 8 18 4 19 11 15 4
7 20 15 5 12 2 5 11
8 13 16 4 7 10 14 17
3 6 8 5 6 8 5 14
20 3 17 18 4 13 5 8
5 7 17 17 4 8 9 14
10 2 5 1 13 15 5 4
```

```
1 1 8 1 20 1 1 6
9 11 14 8 7 1 16 3
13 3 18 3 14 4 17 5
13 14 17 13 3 20 10 3
2 4 20 12 16 17 5 20
14 20 7 2 5 9 17 8
3 9 2 7 14 20 18 18
4 11 19 16 1 13 14 5
1 17 4 15 7 12 17 13
8 13 19 1 18 9 11 14
13 13 6 4 8 9 6 12
17 15 4 7 13 5 17 4
17 2 13 1 8 3 13 8
10 9 20 8 6 17 7 10
12 19 20 2 8 16 18 17
3 5 10 19 16 2 16 13
2 11 7 13 17 10 12 13
16 20 17 7 20 17 15 5
3 3 13 18 11 5 9 18
9 3 16 14 2 3 17 15
15 9 11 7 9 9 4 2
20 5 19 14 15 6 15 3
16 20 2 19 2 18 12 5
13 4 18 17 18 16 10 19
11 12 3 10 13 12 11 19
1 17 8 15 13 5 12 1
12 17 12 15 18 1 1 12
16 3 4 8 9 7 5 8
13 16 7 11 18 19 20 2
8 8 4 12 2 19 19 6
20 4 3 9 15 11 11 18
6 19 7 6 18 14 3 13
2 10 10 10 19 1 7 5

FLM86060
```

```
120
8
30

4
4
4
4
2
4
2
4
```

```
4
6
4
6
4
2
4
4
4
8
4
2
4
2
4
4
4
4
2
6
4
6

13 10 17 13 12 20 19 17
12 19 10 19 15 9 9 17
4 10 5 7 17 1 11 11
12 6 6 1 13 1 4 1
19 15 17 14 7 11 5 10
10 9 5 7 20 17 9 14
10 8 1 15 11 15 12 19
11 9 15 9 17 12 16 19
19 2 9 16 11 8 9 17
14 15 7 7 15 4 16 6
14 5 11 18 8 10 16 5
20 20 17 15 13 17 3 18
16 17 17 18 18 6 17 11
1 11 17 6 17 6 3 12
20 18 2 10 15 8 13 6
8 7 9 16 19 12 12 4
8 9 3 1 3 20 14 3
6 19 13 19 12 16 10 13
10 6 18 5 15 1 14 17
16 20 18 20 20 5 20 17
17 3 17 13 5 9 17 19
13 19 14 10 1 10 11 15
12 1 10 2 1 16 7 7
12 20 16 5 20 16 9 5
```

```
7       15      9       3       6       10      19      9
14      5       15      11      6       3       6       2
7       13      11      5       7       18      4       19
20      1       15      20      3       8       11      6
16      18      19      4       4       3       7       6
2       17      20      18      2       20      4       19

FLM812030
```

```
120
8
60

2
2
2
2
2
2
2
2
2
1
2
2
2
2
2
2
2
1
2
2
2
2
2
2
2
1
1
2
2
4
2
2
2
1
2
```

```
2
2
2
1
2
2
2
1
2
2
1
2
6
2
2
1
1
2
2
2
2
2
6
2

2 13 1 9 20 19 20 3
1 1 13 4 8 16 1 1
13 17 15 7 16 17 10 8
11 16 18 11 6 18 9 14
13 9 7 7 5 16 17 11
7 13 20 7 19 5 19 4
5 20 10 17 18 9 17 4
18 15 11 9 8 4 14 10
10 2 4 15 3 11 2 1
9 4 20 6 10 14 14 4
12 17 19 15 4 8 19 17
11 7 18 1 7 5 8 17
7 9 19 10 1 20 3 17
18 1 16 3 6 17 7 19
5 12 13 20 9 16 19 15
20 4 12 10 14 14 20 2
20 12 15 13 12 11 10 16
14 12 13 19 10 19 2 1
9 12 12 13 3 2 7 10
12 8 13 17 14 3 10 1
19 16 7 18 4 15 2 9
```

```
12 11 8 1 15 14 14 19
13 13 15 1 14 20 4 9
4 10 12 16 14 19 10 12
14 20 8 14 5 18 20 4
5 8 15 3 15 6 9 12
10 7 9 19 1 3 3 10
8 4 10 15 6 9 16 9
12 18 8 4 13 9 18 13
2 9 3 13 18 17 15 10
16 14 5 4 17 13 12 3
11 10 11 4 3 16 20 10
11 8 19 5 19 10 3 15
18 9 6 8 8 9 17 9
8 1 6 1 8 4 15 18
5 14 13 10 6 17 8 19
17 15 12 17 14 20 12 17
2 7 6 13 20 17 13 16
17 11 8 14 17 16 6 19
7 7 15 18 5 16 6 7
18 17 19 15 18 20 4 17
10 19 16 18 17 19 14 12
6 7 1 6 1 1 9 2
12 20 11 4 6 18 8 20
2 11 16 5 18 15 4 18
15 4 10 15 5 9 8 3
3 17 6 10 2 17 7 16
13 5 16 15 9 16 14 1
16 14 11 16 16 4 9 18
5 19 20 20 4 7 16 4
1 4 1 20 12 15 18 18
7 5 3 6 7 8 17 9
16 1 5 12 13 15 14 11
17 5 16 18 1 3 13 9
2 7 18 17 16 19 8 13
6 6 9 12 9 7 20 1
13 7 4 8 12 14 4 5
3 17 2 13 17 8 9 16
6 1 10 9 2 20 20 6
9 14 10 17 18 13 15 19

FLM812060
```

```
120
8
120

1
1
```

7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	1
21	1
22	1
23	1
24	1
25	1
26	1
27	1
28	1
29	1
30	1
31	1
32	1
33	1
34	1
35	1
36	1
37	1
38	1
39	1
40	1
41	1
42	1
43	1
44	1
45	1
46	1
47	1
48	1
49	1
50	1
51	1
52	1
53	1

54	1
55	1
56	1
57	1
58	1
59	1
60	1
61	1
62	1
63	1
64	1
65	1
66	1
67	1
68	1
69	1
70	1
71	1
72	1
73	1
74	1
75	1
76	1
77	1
78	1
79	1
80	1
81	1
82	1
83	1
84	1
85	1
86	1
87	1
88	1
89	1
90	1
91	1
92	1
93	1
94	1
95	1
96	1
97	1
98	1
99	1
100	1

```
101 1
102 1
103 1
104 1
105 1
106 1
107 1
108 1
109 1
110 1
111 1
112 1
113 1
114 1
115 1
116 1
117 1
118 1
119 1
120 1
121 1
122 1
123 1
124 1
125 1
126
127 8  6  12 9  10 4  11 8
128 8  12 18 5  16 10 5  8
129 1  13 5  1  9  6  12 11
130 20 18 4  2  15 1  1  2
131 15 2  18 10 7  12 18 10
132 6  14 16 5  8  10 16 7
133 17 20 18 15 15 11 8  20
134 4  7  20 17 3  3  2  5
135 15 6  3  20 4  7  20 6
136 18 16 2  15 9  3  20 1
137 14 13 2  19 5  6  8  4
138 17 6  1  8  12 10 2  13
139 8  1  9  15 19 5  5  18
140 1  15 12 1  4  11 8  7
141 17 2  20 8  13 11 16 5
142 2  2  7  13 2  4  19 17
143 18 12 2  8  10 12 5  18
144 3  9  20 13 8  18 15 14
145 9  5  5  7  13 8  9  16
146 6  12 3  16 12 8  18 20
147 7  9  12 1  4  13 3  12
```

148	16	13	6	8	16	1	20	5
149	4	9	9	2	7	17	3	6
150	16	12	1	15	14	14	20	8
151	20	7	5	3	2	2	15	1
152	6	15	12	2	1	18	9	7
153	3	6	18	10	3	13	7	7
154	3	16	7	7	1	13	18	19
155	5	16	14	19	14	9	19	9
156	18	11	11	16	4	7	2	19
157	18	8	12	10	20	6	8	19
158	12	12	4	16	14	12	10	11
159	7	20	10	12	6	13	19	13
160	6	19	6	10	16	20	16	6
161	20	19	3	6	12	1	6	2
162	11	14	2	14	16	13	1	4
163	4	13	16	14	14	8	19	7
164	18	18	8	18	3	8	18	14
165	20	9	9	7	14	8	6	9
166	20	10	4	3	4	12	7	15
167	2	2	17	2	15	2	9	3
168	12	11	19	14	12	19	10	3
169	19	9	11	17	8	17	4	4
170	8	4	7	14	4	17	3	4
171	20	18	6	12	3	3	20	12
172	9	4	15	2	13	17	4	4
173	17	12	15	16	6	8	1	12
174	8	6	5	3	8	12	11	5
175	6	18	5	16	19	4	8	8
176	2	17	1	15	5	15	12	13
177	5	13	17	9	5	14	12	11
178	4	18	20	8	16	15	12	2
179	10	10	7	13	1	9	11	18
180	19	8	7	16	15	4	11	14
181	20	19	12	13	16	14	12	2
182	20	11	5	9	14	11	7	4
183	9	12	13	4	16	8	8	19
184	6	17	17	9	18	10	3	9
185	19	3	6	10	8	11	7	8
186	7	3	17	13	2	3	2	19
187	3	20	15	12	4	12	16	16
188	13	10	18	13	15	13	7	1
189	5	13	14	10	3	12	20	1
190	5	13	14	20	16	2	2	17
191	6	9	7	7	5	12	20	12
192	8	8	19	9	11	11	12	14
193	10	7	18	12	7	4	10	11
194	14	6	20	3	6	16	1	7

195	7	9	19	7	3	1	14	2
196	7	20	14	5	17	8	20	18
197	4	4	14	17	16	19	3	14
198	1	5	10	11	11	1	1	17
199	14	7	1	9	13	4	17	13
200	3	16	18	3	5	14	1	5
201	16	15	15	17	14	7	6	13
202	19	10	2	13	15	8	3	7
203	4	12	7	20	8	17	3	15
204	2	4	5	16	1	5	7	20
205	18	10	15	6	14	4	14	15
206	6	17	12	20	17	8	10	3
207	19	20	19	4	6	15	20	12
208	9	8	11	4	16	7	5	1
209	8	8	14	6	14	2	9	6
210	12	14	12	15	7	6	15	5
211	11	4	17	8	11	1	1	16
212	3	17	1	15	9	2	3	19
213	6	13	13	4	9	19	10	18
214	9	19	10	4	10	9	3	19
215	16	10	1	17	8	3	13	9
216	11	8	11	11	13	20	1	8
217	20	15	17	20	9	4	1	14
218	10	15	11	12	2	10	15	9
219	4	19	14	16	9	19	15	13
220	6	7	9	8	5	5	1	6
221	5	16	9	20	19	6	13	7
222	9	3	11	16	18	20	13	7
223	5	5	14	11	9	19	19	8
224	20	11	5	18	3	10	4	20
225	12	6	6	20	3	10	11	14
226	20	3	8	4	13	14	7	9
227	13	10	8	15	14	18	4	17
228	8	12	16	13	19	13	7	8
229	14	12	16	9	5	8	20	13
230	14	2	17	7	5	13	17	9
231	14	15	17	17	19	2	6	7
232	15	2	3	15	17	4	3	4
233	12	5	5	17	9	7	16	9
234	12	14	16	8	17	10	4	9
235	2	11	14	6	14	8	17	1
236	11	17	8	14	4	16	18	10
237	9	19	5	12	1	20	1	15
238	14	2	10	10	20	11	2	14
239	13	5	8	16	3	9	10	20
240	2	15	15	2	13	14	17	18
241	8	12	17	7	20	11	17	17

```
4       8       17      1       2       19      5       19
14      9       20      5       14      12      9       15
19      6       13      20      2       7       13      13
1       11      4       11      5       3       17      2
8       10      13      7       12      1       12      1

FLM8120120
```