

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Graduação em Engenharia de Computação

William Azevedo de Paula

**MODELAGEM MATEMÁTICA PARA PROBLEMA DE SCHEDULING DE
TAREFAS EM AMBIENTE JOB SHOP E VALIDAÇÃO COM ALGORITMO
GENÉTICO**

Belo Horizonte

2013

William Azevedo de Paula

**MODELAGEM MATEMÁTICA PARA PROBLEMA DE SCHEDULING DE
TAREFAS EM AMBIENTE JOB SHOP E VALIDAÇÃO COM ALGORITMO
GENÉTICO**

Monografia apresentada ao Curso de Engenharia da
Computação da Pontifícia Universidade Católica de
Minas Gerais, como requisito parcial para obtenção
do título de Bacharel em Engenharia da Computação

Orientador: Luiz Carlos Figueiredo

Belo Horizonte

2013

À minha mãe, que tanto me incentivou e que fez de tudo para que eu vencesse esta jornada,
mas que infelizmente não está aqui hoje, para ver o meu sucesso.

Ao meu Pai, que tanto lutou para me dar o conhecimento, a educação e a sabedoria de que
precisei para chegar até aqui, os quais jamais terei como retribuir.

À Jéssica, a quem tanto amo, que tanto carinho, compreensão e paciência teve comigo neste
tempo, sem nunca deixar de me motivar e de me ajudar.

A todos os parentes e amigos, que entenderam as minhas ausências e que me deram força para
seguir em frente.

Aos Professores Paulo Amaral, Júlio Conway, Rosely Campos, Alexandre Teixeira, dentre
outros que tanto fizeram para melhorar a qualidade do curso e por despertar nos alunos a
admiração pela profissão de Engenheiro.

AGRADECIMENTOS

Agradeço a todos que participaram da realização deste trabalho, especialmente:

Ao professor Luiz Carlos Figueiredo, pela orientação e atenção ao meu trabalho, fornecendo o conhecimento e apoio de que precisei.

Ao professor Sandro Jerônimo, que muito me ajudou com a concepção deste trabalho, e em cuja aula a ideia inicial surgiu. A ele que, mesmo indiretamente, cumpriu o papel de meu co-orientador.

Ao professor Zenilton, pelos toques e dicas que me forneceu, e que foram cruciais para o desenvolvimento deste trabalho.

A todos que de alguma maneira contribuíram com este.

RESUMO

Este trabalho propõe uma modelagem matemática, baseada em problemas de programação matemática não linear, a ser aplicada na resolução de problemas de *scheduling* de tarefas em ambientes *job shop*, onde várias tarefas são executadas simultaneamente e demandam recursos que existem em quantidade limitada. A resolução apresentada é baseada no conceito de *pipeline*, dividindo as tarefas em etapas (ou estágios), e escalonando-as de maneira a realizar todo o processo produtivo em menor tempo, aproveitando o paralelismo na utilização dos recursos disponíveis. Para validar o modelo, foi desenvolvida uma ferramenta de *software* que utiliza algoritmo genético para encontrar soluções válidas para o problema de escalonamento, utilizando o modelo proposto como função de avaliação do *fitness* dos indivíduos. Foram realizados testes exaustivos, variando o tamanho da população em três cenários de testes, todos representando um ambiente de fabricação de doces e salgados. O desempenho do algoritmo foi validado e comparado com a solução do problema através de uma heurística gulosa, que acabou por ser incorporada ao *software*, sendo utilizada para inicializar a população com a maior quantidade possível de indivíduos com bom *fitness*, aumentando a velocidade de convergência do algoritmo. Os resultados mostram que o modelo funciona muito bem e consegue representar corretamente o processo de execução das tarefas e suas restrições. Porém, a solução do modelo é complexa e a utilização de *algoritmo genético* se mostrou uma boa forma de aplicar o modelo a problemas reais, alcançando boas soluções, embora nem sempre a solução ótima seja encontrada. Em comparação com a heurística gulosa, a combinação de algoritmo genético e modelo matemático proposta se mostrou mais eficiente em problemas de grandes dimensões, onde a heurística não consegue atuar. Para problemas pequenos, a heurística é capaz de encontrar a solução ideal muito rapidamente, fato pelo qual a heurística foi incorporada na ferramenta, melhorando a pontuação de indivíduos na solução inicial e melhorando a eficiência do algoritmo. Um problema encontrado é o excessivo consumo de memória por parte do *software* desenvolvido, aumentando a dificuldade de utilização do mesmo à medida que cresce a complexidade do ambiente que se deseja aplicá-lo.

Palavras Chave: Escalonamento, algoritmo genético, modelagem matemática, Programação Não Linear.

ABSTRACT

This paper proposes a mathematical modeling, based in no-linear mathematical programming, to be applied in solving task scheduling problems in job shop production environments, where multiple tasks are performed simultaneously and require limited resources. The resolution presented is based on the concept of pipeline, dividing tasks into steps (or stages) , and scaling them so as to reduce the makespan. To validate the model, was developed a software tool that use genetic algorithm to find valid resolutions for the scheduling problem, using the proposed model to evaluate the fitness of the solutions. Extensive tests were performed, varying the population size in three testing scenarios, each of them representing a food manufacturing environment. The performance of the algorithm was validated and compared with the solution of the problem through a greedy heuristic, which was eventually incorporated into the software, and is used to initialize the population with as many as possible individuals with good fitness, increasing the convergence speed of the algorithm. The results show that the model works very well and can correctly represent the process of execution of the tasks and their constraints. However, the solution of the model is complex and the use of genetic algorithm proved to be a good way to apply the model in real problems , achieving good solutions, although the optimal solutions are not always found. Compared with the greedy heuristic, the proposed combination of genetic algorithm with the mathematical model is more efficient in large problems, where the heuristic cannot operate. For small problems, the heuristic is able to find the optimal solution very quickly. Because of this, the heuristic was incorporated in the tool, improving the scores of the individuals in the initial population and improving the efficiency of the algorithm. A problem in the developed software is the excessive system memory consumption, increasing the difficulty of using it when grows the complexity of the environment that it is applied to.

Keywords: Scheduling, genetic algorithm, mathematical modeling, Non-Linear programming

LISTA DE ILUSTRAÇÕES

| | |
|--|----|
| Figura 1 - Componentes de um ERP..... | 12 |
| Figura 2 - Fluxo do PCP..... | 17 |
| Figura 3 - Funções e relacionamentos do PCP..... | 18 |
| Quadro 1 - Processo Contínuo x Processo por batelada (lotes) | 21 |
| Quadro 2 - Ambiente Flow Shop x Ambiente Job Shop..... | 22 |
| Figura 4 - Overlap no ambiente Flow Shop x Job Shop..... | 23 |
| Figura 5 - Layouts Produtivos..... | 24 |
| Figura 6 - Gráfico de Gantt de um Schedule, orientado a recursos (Máquinas, ou Machines) | 25 |
| Figura 7 - Gráfico de Gantt de um Schedule, orientado a tarefas (Jobs)..... | 25 |
| Figura 8 - Exemplo de Solução Gráfica..... | 28 |
| Figura 9 - Fluxograma dos passos do AG..... | 32 |
| Figura 10 - Cruzamento de cromossomos..... | 34 |
| Figura 11 - Diagrama de gantt mostrando o fluxo da produção com solução sequencial..... | 38 |
| Figura 12 - Diagrama de gantt mostrando a produção otimizada..... | 39 |
| Figura 13 - Funcionamento da ferramenta de software..... | 41 |
| Figura 14 - Diagrama de casos de uso..... | 41 |
| Figura 15 - Diagrama de Atividades..... | 43 |
| Figura 16 - Solução representável pelo modelo (com folga)..... | 48 |
| Figura 17 - Roleta para os Indivíduos da Tabela 3..... | 56 |
| Figura 18 - Crossover por escalonamento vertical..... | 57 |
| Figura 19 - Crossover por escalonamento horizontal..... | 57 |
| Figura 20 - Crossover de três pontos por linha..... | 58 |
| Figura 21 - Mutação swap..... | 59 |
| Figura 22 - Mutação flip..... | 59 |
| Figura 23 - Visualização do arquivo “input.txt” | 60 |
| Figura 24 - Início da execução do schedullng.exe..... | 65 |
| Figura 25 - Solução inicial do algoritmo..... | 66 |
| Figura 26 - Soluções parciais do algoritmo..... | 67 |
| Figura 27 - Solução final do algoritmo..... | 67 |
| Figura 28 - Curva de evolução de uma população com 5 indivíduos..... | 69 |
| Figura 29 - Curva média de evolução de uma população de 5 indivíduos..... | 69 |
| Figura 30 - Curva de evolução de uma população com 5 indivíduos..... | 70 |

| | |
|---|----|
| Figura 31 - Curva média de evolução de uma população com 5 indivíduos..... | 70 |
| Figura 32 - Curva média de evolução para uma população de tamanho 50..... | 71 |
| Figura 33 - Curva média de evolução ampliada, para uma população de tamanho 50..... | 72 |
| Figura 34 - Curva média de evolução para uma população de tamanho 100..... | 73 |
| Figura 35 - Curva média de evolução para uma população de tamanho 200..... | 74 |
| Figura 36 - Curva média de quantidade de soluções parciais em função do tamanho da população..... | 74 |
| Figura 37 - Pontuação média na população inicial em função do tamanho da população..... | 75 |
| Figura 38 - Pontuação média na população inicial em função do tamanho da população..... | 75 |
| Figura 39 - Representação da melhor solução encontrada para o cenário de testes 2..... | 77 |
| Figura 40 - Curvas individual e média de evolução para uma população de tamanho 5..... | 78 |
| Figura 41 - Curvas individual e média de evolução para uma população de tamanho 10..... | 79 |
| Figura 42 - Curvas individual e média de evolução para uma população de tamanho 50..... | 80 |
| Figura 43 - Curvas individual, média e média ampliada de evolução para uma população de tamanho 100..... | 81 |
| Figura 44 - Curvas individual, média e média ampliada de evolução para uma população de tamanho 200..... | 82 |
| Figura 45 - Gráficos de Soluções Inválidas e Pontuação Média, ambos em função do tamanho da população..... | 83 |
| Figura 46 - Pontuação média na inicialização da população, em função do tamanho da população (normal e empliado)..... | 84 |
| Figura 47 - Quantidade média de soluções parciais por tamanho da população..... | 84 |
| Figura 48 - Representação da melhor solução encontrada para o cenário de testes 3..... | 87 |
| Figura 49 - Curvas individual e média de evolução para uma população de tamanho 5 | 88 |
| Fonte: Autor..... | 88 |
| Figura 50 - Curvas individual e média de evolução para uma população de tamanho 10..... | 89 |
| Figura 51 - Curvas individual e média de evolução para uma população de tamanho 50..... | 90 |
| Figura 52 - Curvas individual e média de evolução para uma população de tamanho 100..... | 91 |
| Figura 53 - Curvas individual e média de evolução para uma população de tamanho 200..... | 92 |
| Figura 54 - Curvas de Soluções Inválidas e Pontuação Média, ambas em função do tamanho da população..... | 93 |
| Figura 55 - Pontuação média na inicializaçã oda população, em função do tamanho da população..... | 93 |
| Figura 56 - Quantidade média de soluções parciais por tamanho da população..... | 94 |

| | |
|---|----|
| Figura 57 - Execução do algoritmo identificada em um gráfico de consumo de memória do Windows 7..... | 95 |
| Figura 58 - Gráficos de Consumo de memória para o cenário de testes 1..... | 96 |
| Figura 59 - Gerenciador de Tarefas do Windows registrando o momento de consumo excessivo da aplicação (Falha de segmentação)..... | 97 |
| Figura 60 - Gráficos de Consumo de memória para o cenário de testes 2..... | 98 |
| Figura 61 - Gráficos de Consumo de memória para o cenário de testes 3..... | 98 |

LISTA DE TABELAS

| | |
|---|----|
| Tabela 1 - Exemplos de recursos e sua disponibilidade..... | 37 |
| Tabela 2 - Tempo de utilização dos recursos pelas tarefas..... | 38 |
| Tabela 3 - Indivíduos da População e o cálculo pontuação..... | 56 |
| Tabela 4 - Sequência de resultados para uma população de tamanho 5..... | 69 |
| Tabela 5 - Sequência de resultados para uma população de tamanho 10..... | 70 |
| Tabela 6 - Sequência de resultados para uma população de tamanho 50..... | 71 |
| Tabela 7 - Sequência de resultados para uma população de tamanho 100..... | 72 |
| Tabela 8 - Sequência de resultados para uma população de tamanho 200..... | 73 |
| Tabela 9 - Recursos disponíveis para o cenário de testes 2..... | 76 |
| Tabela 10 - Tempo de Utilização do Recurso por Processo para o Cenário de testes 2..... | 76 |
| Tabela 11 - Sequência de resultados para uma população de tamanho 5..... | 78 |
| Tabela 12 - Sequência de resultados para uma população de tamanho 10..... | 79 |
| Tabela 13 - Sequência de resultados para uma população de tamanho 50..... | 80 |
| Tabela 14 - Sequência de resultados para uma população de tamanho 100..... | 81 |
| Tabela 15 - Sequência de resultados para uma população de tamanho 100..... | 82 |
| Tabela 16 - Recursos disponíveis para o cenário de testes 3..... | 85 |
| Tabela 17 - Tempo de Utilização do Recurso por Processo para o Cenário de testes 3..... | 86 |
| Tabela 18 - Sequência de resultados para uma população de tamanho 5..... | 88 |
| Tabela 19 - Sequência de resultados para uma população de tamanho 10..... | 89 |
| Tabela 20 - Sequência de resultados para uma população de tamanho 50..... | 89 |
| Tabela 21 - Sequência de resultados para uma população de tamanho 100..... | 90 |
| Tabela 22 - Sequência de resultados para uma população de tamanho 200..... | 91 |

SUMÁRIO

| | |
|---|-----------|
| 1 INTRODUÇÃO..... | 12 |
| 1.1 Motivação..... | 13 |
| 1.2 Objetivos..... | 14 |
| 1.2.1 <i>Objetivos Específicos</i> | 14 |
| 1.3 Justificativa..... | 15 |
| 1.4 Organização do Trabalho..... | 15 |
| 2 REVISÃO DA LITERATURA..... | 16 |
| 2.1 Programação do Controle de Produção (PCP)..... | 16 |
| 2.1.1 Tipos de Processo Produtivo..... | 18 |
| 2.1.1.1 Produção por Fluxo..... | 19 |
| 2.1.1.1.1 Produção por fluxo contínuo..... | 19 |
| 2.1.1.1.2 Produção por fluxo em lotes..... | 20 |
| 2.1.1.2 Produção por Encomenda..... | 23 |
| 2.1.1.2.1. Processos de projeto..... | 23 |
| 2.1.1.2.2. Processos por tarefa..... | 23 |
| 2.1.1.3 <i>Outros tipos de produção</i> | 24 |
| 2.2 O Problema de Scheduling..... | 25 |
| 2.3. Otimização Matemática..... | 26 |
| 2.3.1 <i>Programação Linear</i> | 26 |
| 2.3.2 <i>Programação Não Linear</i> | 28 |
| 2.3.2.1 Programação não linear Irrestrita..... | 29 |
| 2.3.2.2 Programação não linear Restrita..... | 30 |
| 2.4 Algoritmos Genéticos..... | 31 |
| 2.4.1 <i>Inicialização</i> | 32 |
| 2.4.2 <i>Avaliação</i> | 33 |
| 2.4.3 <i>Seleção</i> | 33 |
| 2.4.4 <i>Cruzamento</i> | 33 |
| 2.4.5 <i>Mutação</i> | 34 |
| 2.4.6 <i>Atualização e finalização</i> | 34 |
| 2.5 Pipeline | 35 |
| 2.6 Trabalhos Relacionados..... | 35 |
| 3. METODOLOGIA..... | 37 |
| 3.1 O Ambiente modelado..... | 37 |
| 3.2 Técnicas de Modelagem e solução..... | 40 |
| 3.3 Modelagem do Processo..... | 40 |
| 3.4 Modelagem Matemática..... | 43 |
| 3.4.1 <i>Parâmetros de Entrada</i> | 44 |
| 3.4.2 <i>Função Objetivo</i> | 46 |
| 3.4.3 <i>Restrições</i> | 48 |
| 3.4.3.1 Disponibilidade de Recursos..... | 49 |
| 3.4.3.2 Simultaneidade de Utilização de Recursos..... | 49 |
| 3.4.3.3 Retenção do Recurso..... | 49 |
| 3.4.3.4 Ordem de Utilização dos recursos..... | 49 |
| 3.4.3.5 Utilização de Todos os Recursos..... | 50 |
| 3.4.3.6 Tempo máximo para execução dos processos..... | 50 |

| | |
|--|-----|
| 3.4.3.7 Outras Restrições..... | 51 |
| 3.4.4 Modelo Matemático..... | 51 |
| 3.5 Modelagem do Algoritmo Genético..... | 52 |
| 3.5.1 Indivíduos..... | 52 |
| 3.5.2 Inicialização..... | 54 |
| 3.5.3 Avaliação..... | 55 |
| 3.5.4 Seleção..... | 56 |
| 3.5.5 Cruzamento..... | 56 |
| 3.5.5.1 Escalonamento Vertical..... | 57 |
| 3.5.5.2 Escalonamento Horizontal..... | 57 |
| 3.5.5.3 Corssover de 3 pontos por linha..... | 58 |
| 3.5.6 Mutação..... | 58 |
| 3.5.6.1 Mutação swap..... | 58 |
| 3.5.6.2 Mutação flip..... | 59 |
| 3.5.7 Atualização e finalização..... | 59 |
| 3.6 Implementação..... | 59 |
| 3.6.1 Parâmetros de Entrada..... | 60 |
| 3.6.1.1 Tamanho da população..... | 61 |
| 3.6.1.2 Taxa de mutação..... | 61 |
| 3.6.1.3 Quantidade de cruzamentos..... | 61 |
| 3.6.1.4 Número de gerações..... | 61 |
| 3.6.1.5 Quantidade de recursos..... | 62 |
| 3.6.1.6 Vetor recursos..... | 62 |
| 3.6.1.7 Quantidade de Processos..... | 63 |
| 3.6.1.8 Matriz de Processos..... | 63 |
| 3.6.1.9 Prazo de Entrega..... | 64 |
| 3.6.1.10 Ordem de utilização..... | 64 |
| 3.6.2 Execução do Software..... | 65 |
| 4 TESTES E RESULTADOS..... | 68 |
| 4.1 Cenários de Testes..... | 68 |
| 4.1.1 Cenário de testes 1..... | 68 |
| 4.1.2 Cenário de testes 2..... | 75 |
| 4.1.3 Cenário de testes 3..... | 85 |
| 4.2 Outros testes e Observações..... | 95 |
| 4.2.1 Consumo de Memória..... | 95 |
| CONCLUSÃO..... | 99 |
| REFERÊNCIAS..... | 102 |

1 INTRODUÇÃO

No meio industrial, uma das necessidades predominantes é a de gerenciar o processo produtivo de maneira a haver ganhos de produtividade e diminuição dos custos da produção, o que pode ser conseguido, na maioria das vezes, através de uma utilização eficiente dos recursos produtivos. Essa necessidade fez com que a programação da produção se tornasse a área da gestão que mais tem obtido progressos nessa utilização eficiente de recursos (REIS, 1996).

Além do processo industrial, as empresas necessitam também gerenciar várias outras áreas internas, como estoque, finanças, processo de qualidade, etc. Por esse motivo, as grandes indústrias optam por realizar o gerenciamento integrado de todas essas áreas, adotando para esse fim ferramentas complexas de *software*, chamadas de ERPs (*Enterprise Resource Planning*). Segundo JUNIOR (2011), essas ferramentas reúnem, em um único sistema de informação, diversos módulos para permitir o controle de cada uma dessas áreas, como pode ser visto na Figura 1.

Figura 1 - Componentes de um ERP.



Fonte: JUNIOR, 2011

O módulo do ERP destinado ao controle do processo produtivo, lidando com requisição de materiais para a produção, é o chamado MRP (Material Requirements Planning), o qual posteriormente evoluiu para um planejamento mais eficiente, denominado MRP II (Manufacturing Resource Planning), que além de englobar o MRP, também possui o componente “Planejamento e Controle da Produção” (PCP), que é uma ferramenta utilizada para gerenciar o processo produtivo como um todo, auxiliando as tomadas de decisão com base na demanda da produção e na

disponibilidade de materiais e recursos, e determinando o que produzir, em qual quantidade e como produzir (MATERIAL REQUEST PLANNING, 2013). O PCP tem como principal componente o “planning” ou “Planejamento da Produção” (DERIZ, 2007. p. 14), responsável pela realização do escalonamento eficiente, no tempo, dos recursos concorrentes demandados no processo de manufatura, de forma a atender aos prazos previstos e maximizar ou minimizar determinados parâmetros, como tempo de produção (ou Lead Time), lucratividade, dentre outros.

O planejamento da produção e o impacto que o mesmo exerce sobre o processo produtivo são os focos principais deste trabalho.

1.1 Motivação

Em um processo industrial, dado a grande quantidade de recursos disponíveis, como equipamentos, mão de obra e materiais, que muitas vezes precisam ser alocados por processos concorrentes entre si, isto é, que necessitam utilizar um mesmo recurso em um dado instante, o desempenho desse sistema está diretamente relacionado à maneira como esses recursos são utilizados, e à ordem em que determinadas etapas são inicializadas, dando ao problema uma característica combinatória, de forma que existem diversas configurações possíveis do processo, onde cada configuração pode possuir um desempenho diferente da outra. Uma utilização inteligente dos meios de produção, normalmente realizada através da realização em paralelo das etapas de produção que demandam recursos não concorrentes, pode reduzir significativamente o tempo total da produção, aumentando a produtividade da empresa e, conseqüentemente a lucratividade. Por sua vez, um planejamento ruim, sem implementar algum tipo de pipeline ou paralelismo no processo, irá gerar um “gargalo” na utilização de recursos compartilhados, aumentando o tempo da produção. Além disso, haverá um desperdício de tempo muito maior com o setup dos recursos, como por exemplo, a limpeza de determinados equipamentos ou o tempo para aquecer um forno, uma vez que em um processo em paralelo vários desses recursos teriam o seu setup realizado simultaneamente, economizando tempo.

Dessa forma, um melhor desempenho será obtido ao encontrar uma configuração para o processo que minimize ou maximize o parâmetro utilizado para medir o desempenho, geralmente o tempo. O problema é que, apesar de parecer simples, a grande quantidade de variáveis a serem consideradas, como necessidade e disponibilidade de equipamentos, mão de obra, reservas, lotes, tempos de produção, dentre outros contribuem para que a escala do problema seja grande o suficiente para não poder ser resolvido por métodos que analisam todas as combinações, devendo ser utilizadas heurísticas para encontrar soluções aproximadas.

Uma metodologia geralmente utilizada no desenvolvimento de PCP é a modelagem e resolução de problemas de *scheduling* ou planning, que visam escalonar os recursos do processo produtivo no tempo (REIS,1996) e que, apesar de serem problemas difíceis de resolver, devido à grande quantidade de restrições que podem assumir, geram grandes benefícios quando uma boa solução é encontrada. Apesar de apresentarem muita semelhança entre si, esses problemas são modelados de maneira a atender a casos específicos de planejamento, voltados para áreas específicas da produção, e tem levado a um grande número de pesquisas nas áreas de pesquisa operacional e inteligência artificial, na tentativa de encontrar soluções para esses casos (ALMEIDA; HAMACHER; PACHECO, 2010).

Essa complexidade de resolução e os benefícios que uma boa metodologia para resolução de problemas de *scheduling* pode trazer ao meio industrial serviram como principais motivações para a realização deste trabalho.

1.2 Objetivos

O objetivo geral deste trabalho é apresentar uma modelagem em forma de programação matemática não-linear para o problema de *scheduling*, voltada para um planejamento de produção a ser aplicado em indústrias alimentícias, mas que seja facilmente adaptável a outras áreas de planejamento, abordando diversas variáveis do processo de manufatura; e o desenvolvimento de uma metodologia, baseada em algoritmos genéticos, para resolvê-la em um tempo hábil.

1.2.1 Objetivos Específicos

Os objetivos específicos são:

- a) realização de um estudo da eficiência da aplicação de técnicas de inteligência artificial para resolução de programação matemática não linear em tempo hábil;
- b) implementação de uma ferramenta de *software* capaz de resolver o problema do planejamento de processo produtivos de maneira a minimizar o tempo de produção (*Lead Time*) em tempo hábil, sendo baseada em um modelo matemático proposto, resolvido a partir de algoritmos genéticos;
- c) comparar os resultados da aplicação da técnica proposta com a utilização de uma heurística gulosa, analisando características como tempo de execução e qualidade da solução.

1.3 Justificativa

O desenvolvimento da aplicação proposta pode trazer benefícios para o setor produtivo, permitindo adaptar problemas complexos, com grande quantidade de restrições e de recursos escalonáveis, ao modelo proposto, abstraindo dados da produção em equações matemáticas, as quais devem ser resolvidas de forma rápida, permitindo que um planejamento complexo e eficiente seja abstraído do modelo proposto em tempo hábil. Embora existam vários tipos de ferramentas de *software* voltadas para este tipo de planejamento, utilizando heurísticas cada vez mais eficientes, a solução proposta apresenta a vantagem de convergir para o melhor escalonamento possível para a produção, resultando em um processo produtivo com economia de tempo, devido ao paralelismo na utilização dos recursos não concorrentes.

A elaboração de um modelo matemático que atenda às restrições existentes em um ambiente de produção também é uma contribuição deste trabalho, uma vez que estudos posteriores podem aperfeiçoar ou reutilizar o modelo, além de evoluir as metodologias aplicadas à resolução do mesmo, podendo trazer benefícios cada vez maiores ao meio industrial.

1.4 Organização do Trabalho

Este trabalho é organizado da seguinte maneira: O Capítulo 2 apresenta uma revisão bibliográfica a respeito do problema de *scheduling*, abordando sua história e desenvolvimento, e apresentando algumas importantes modelagens e metodologias para a resolução do mesmo, em várias áreas industriais, baseadas em modelagem matemática e inteligência artificial. Ainda nesse capítulo, é realizado um estudo comparativo a respeito de possíveis técnicas de inteligência artificial a serem aplicadas à resolução de modelos matemáticos de programação não-linear.

No Capítulo 3 é apresentado, de maneira detalhada, o modelo matemático proposto, e realizada uma proposta de implementação para o mesmo, baseada em algoritmos genéticos, revisados no Capítulo 2. Também são apresentados diagramas UML de casos de uso e atividades para o *software* implementado e o cronograma para a realização do projeto.

No Capítulo 4 são apresentados os testes da implementação realizada e os resultados são analisados.

No Capítulo 5 são apresentadas as conclusões e propostas de trabalhos futuros.

2 REVISÃO DA LITERATURA

A seguir são apresentados uma revisão bibliográfica a respeito do problema de *scheduling* e o levantamento do estado da arte a respeito de implementações para o mesmo. Por fim, é apresentado um estudo comparativo a respeito de várias técnicas de inteligência artificial que podem ser aplicadas à resolução de modelos de programação matemática não linear.

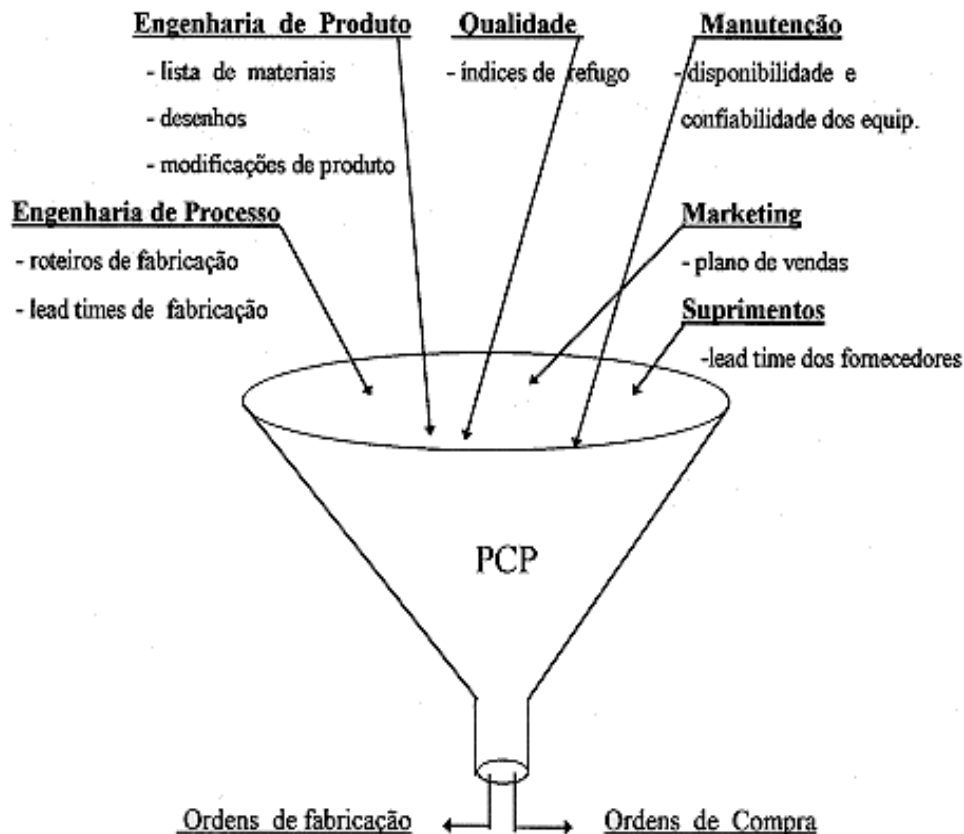
2.1 Programação do Controle de Produção (PCP)

As vantagens da divisão do trabalho no meio industrial provêm da coordenação entre as atividades produtivas. Nesse contexto, destacamos o papel do PCP: integrar os setores de uma empresa envolvidos no processo produtivo, envolvendo planejamento, demanda, controle de estoque, processo de fabricação e a programação do processo produtivo, realizando o *scheduling* da produção (JUNIOR, 2006; p. 4).

O PCP realiza a tomada de decisão quanto à melhor maneira de empregar os recursos do processo produtivo, de forma que a demanda seja atendida no tempo e quantidade previstos e utilizando os recursos corretos, baseando-se nos dados de diversas áreas da empresa (PLANEJAMENTO E CONTROLE DA PRODUÇÃO, 2013).

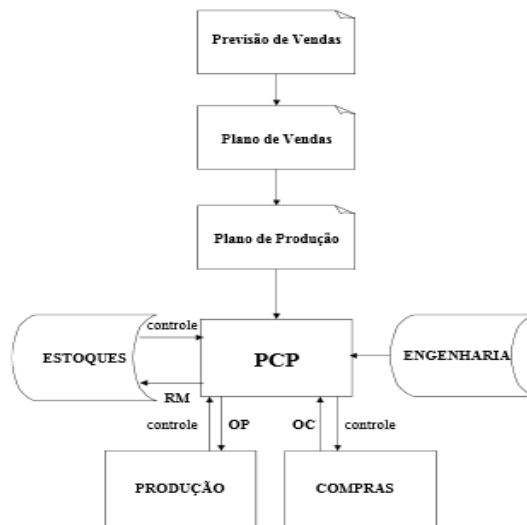
Dentre as várias áreas das quais o PCP utiliza informações, podemos destacar a Engenharia do Produto, onde o PCP busca informações contidas nas listas de materiais; a Engenharia do Processo, onde são encontradas informações a respeito do roteiro de fabricação e do *lead time* da produção; Na área de Marketing, busca as informações a respeito de vendas e pedidos firmes; a Manutenção fornece informações acerca de planos de manutenção; Compras/Suprimentos informa a respeito do fluxo de estoque, dentre muitos outros relacionamentos (MOLINA; RESENDE, 2006, p.1). A Figura 2 representa o fluxo de atividades do PCP.

Figura 2 - Fluxo do PCP



Fonte: O PCP NA INDÚSTRIA AUTOMOTIVA, 2012

Segundo ALMEIDA (2010; p. 8), o PCP oferece suporte ao setor produtivo através da emissão de ordens de produção e ordens de compra. Por meio da avaliação dos níveis de estoque, o mesmo precisa se assegurar se há material suficiente para a realização da produção. Caso não haja quantidade suficiente, são emitidas ordens de compra, as quais são processadas pelo setor de compras, para que seja realizada a aquisição dos materiais, que passam pelo controle de qualidade e por fim entram no estoque. Uma vez disponível a quantidade necessária, o PCP gera requisições de materiais (RMs), as quais devem ser sincronizadas com as ordens de produção, para garantir que o processo produtivo nunca pare por falta de materiais. Cada ordem de produção possui informações de quantidade a produzir, material processado e suas quantidades, local de processamento e data de início e final da fabricação. A sincronização entre as operações garantem que as demandas sejam atendidas no prazo previsto. A Figura 3 apresenta as funções exercidas pelo PCP e o relacionamento entre elas.

Figura 3 - Funções e relacionamentos do PCP

Fonte: ALMEIDA, 2010

2.1.1 Tipos de Processo Produtivo

FERREIRA (2012, p.24) afirma que as atividades a serem realizadas pelo PCP são determinadas com base na classificação do processo produtivo em:

- a) Produção por lotes ou encomenda (fluxo intermitente);
- b) Produção de grandes projetos sem repetição.

FERREIRA também afirma que os três tipos citados podem ser subdivididos em “Produção por Tarefa (*Jobbing*) ou projetos, lote ou batelada, produção em massa e processos contínuos”, embora reconheça que a diferença entre os processos contínuos e em massa consiste apenas nos volumes produzidos, sendo semelhantes nos demais fatores, de forma que ele chega a classificar a produção em massa como um tipo de produção contínua.

ALMEIDA (2010, p.12) e LOPES (2013, p. 3) apresentam classificações diferentes, mas que abrangem os mesmos tipos básicos de processo: segundo ALMEIDA, a produção contínua (Fluxo Contínuo) e a produção por lotes (Fluxo Intermitente ou em Lotes) apresentam características em comum, principalmente no que diz respeito à disposição espacial dos recursos produtivos, sendo portanto subdivisões de um processo denominado “Produção por Fluxo”. Além desse processo, ALMEIDA também destaca a “Produção Funcional”, que equivale ao processos de produção Por Tarefa (*Jobbing*) ou Por Projetos. LOPES, por sua vez separa os processos “Por Tarefa (*Jobbing*)” e “Por Projetos”, devido à diferença entre os volumes gerados do produto final.

BORGES e DALCOL (2002, p. 1) focam apenas nos processo de produção de fluxo contínua e por lotes (intermitente).

Enquanto os demais autores citados classificam o processo produtivo principalmente com base no fluxo e volume de produtos gerados, ALMEIDA apresenta classificações baseadas principalmente na disposição espacial dos recursos produtivos. A seguir, são apresentadas as características dos principais processos produtivos identificados.

2.1.1.1 Produção por Fluxo

Em um processo produtivo por fluxo, os recursos produtivos são fisicamente dispostos de acordo com as etapas a serem realizadas durante a produção, aumentando o fluxo de materiais nos postos de trabalho e mantendo-o constante, o que garante uma alta produtividade, com baixo custo de mão de obra (mão de obra não específica) e qualidade uniforme. Além disso, este processo possui um planejamento de controle da produção simplificado, tendo como única desvantagem a baixa flexibilidade, uma vez que tal arranjo físico dos recursos torna o processo muito específico, muitas vezes dedicado ao processamento de um único tipo de produto (ALMEIDA, 2010, p.12).

Quanto às subdivisões do processo por fluxo, que são a produção contínua e por lotes, BORGES e DALCOL (2002, p.2) afirmam que não há critérios de separação bem definidos, sendo que alguns autores se baseiam em critérios relacionados à escala da produção, enquanto outros baseiam-se na padronização dos recursos ou das tarefas produtivas. Muitos autores apresentam, por sua vez, critérios baseados na combinação desses aspectos.

2.1.1.1.1 Produção por fluxo contínuo

A produção Contínua, ou de fluxo contínuo, é uma subdivisão do processo por fluxo, a qual é utilizada em empresas que produzem grandes volumes de produto, através de métodos altamente padronizados e automatizados, compostos por sequências ininterruptas de operações repetitivas. Nessas empresas, a produção é um processo linear e acelerado, baseada em planejamentos a longo prazo, podendo abranger até o período de um ano inteiro, explorando a disponibilidade de recursos e visando otimizar o processo através da elaboração de planos de produção bem completos e padronizados. Dessa forma, esta modalidade produtiva permite prever despesas, investimentos necessários em equipamentos e mão de obra, dentre outros fatores, visando uma maior economia no processo (FERREIRA, 2012, p.26).

ALMEIDA (2010, p.13) afirma que este modelo produtivo é mais comum em indústrias de

processo, como indústrias químicas e alimentícias, refinarias de petróleo e produção de aço; e em processos de manufatura baseados em linha de montagem. Ele afirma que os recursos são posicionados de maneira a processar apenas um tipo de produto ou subproduto, para posterior montagem do produto final, e funcionam continuamente através de um processo repetitivo e monótono, tendo como benefícios a alta produtividade e, embora um alto volume de produção seja requerido para equilibrar os custos do processo (que são muito altos), uma vez atingido esse ponto (*Break Event Point*) o custo de produção por unidade torna-se baixo. Entre as desvantagens deste processo estão o fato de o desempenho do mesmo depender diretamente da disponibilidade de materiais, e necessidade de um sistema de movimentação e manipulação de materiais projetado especialmente para atender a um produto específico, o que torna baixa a flexibilidade, tornando extremamente altos os custos para modificação dos processos.

Segundo LOPES (2013, p. 3), neste tipo de organização ocorre: “Produção por longos períodos” e os produtos são “produzidos em um fluxo ininterrupto”, com alto volume de produção e baixa variedade. Embora ele diferencie o processo contínuo do processo em massa, devido à característica de o processo contínuo ser ininterrupto, muitos autores não diferenciam os mesmos, enquanto outros, como FERREIRA (2012, p.26), optam por considerar o processo em massa como subdivisão do processo contínuo.

Neste modelo produtivo, o tempo de processamento de uma unidade do produto é geralmente pequeno, embora a velocidade de produção seja alta e o tempo gasto com *set up* dos recursos são altos, o que leva à necessidade de uma produção em grandes quantidades, isto é, o processamento de grandes ordens de produção de uma vez, o que aliado a pequena variedade e complexidade dos produtos e à baixa quantidade de etapas de produção, justificam os investimentos altos em equipamentos destinados a um único produto, para operarem constantemente (BORGES e DALCOL, 2002, p.2).

2.1.1.1.2 Produção por fluxo em lotes

O processo por fluxo em lotes é mais flexível que o processo por fluxo contínuo, pois permite a produção de lotes de tamanhos diferentes de um mesmo produto ou de modelos diferentes de um mesmo produto, mas que conserva características semelhantes, necessitando de poucas alterações na linha produtiva, sendo característico da indústria manufatureira (ALMEIDA, 2010, p.13).

FERREIRA (2012, p.26) afirma que cada lote da produção é dimensionado para atender a uma demanda prevista para um determinado período de tempo, de forma que um novo lote é

produzido após o término do anterior, o que dá também a este processo o nome de “Produção Intermitente”.

Segundo BORGES e DALCOL (2002, p.3), um processo por lotes diferencia-se do processo contínuo principalmente por uma maior quantidade de etapas no processo produtivo e por este ser destinado a produtos mais complexos. A variedade de produtos produzidos é maior, e esses produtos, muitas vezes, necessitam de recursos em comum, o que torna o planejamento da produção muito mais complexo. Por esse mesmo motivo, frequentemente são feitas alterações na linha produtiva para permitir novas configurações de processo.

O Quadro 1 apresenta uma comparação entre os processos contínuo e por lotes.

Quadro 1 - Processo Contínuo x Processo por batelada (lotes)

| Processo Contínuo | Processo por batelada |
|--|---|
| Alta velocidade de produção, pouco trabalho humano | Tempo de lead time grande, muito trabalho humano no processo |
| Clara determinação de capacidade, uma rotina para todos os produtos, baixa flexibilidade | Capacidade não facilmente determinada (diferentes configurações, rotinas complexas) |
| Baixa complexidade do produto | Produtos mais complexos |
| Baixo valor agregado | Alto valor agregado |
| Tempos de parada causam grande impacto | Tempos de parada causam menor impacto |
| Pequeno número de etapas de produção | Grande número de etapas de produção |
| Numero limitado de produtos | Grande número de etapas de produção |

Fonte: BORGES e DALCOL, 2002

ALMEIDA, (2012, p.27), BORGES e DALCOL (2002, p.6) classificam os ambientes de fabricação (Quadro 2) dos sistemas de produção por lotes em:

- a) **ambientes Flow Shop:** Ambiente em que todos os produtos a serem produzidos possuem a mesma sequência de operações nos recursos disponíveis na linha de montagem (FERREIRA, 2012, p.27), e em cujo ambiente os materiais e peças se deslocam pela fábrica em fluxo constante, sendo uma tendência das indústrias de processo, e o arranjo físico (*layout*) dos recursos depende do produto a ser produzido. Utiliza equipamentos especializados, com alto grau de automação e capazes de produzir uma pequena variedade de produtos, porém em grandes quantidades e baixo custo (BORGES e DALCOL, 2002, p.6);
- b) **ambientes Job Shop:** Ambiente em que nem todos os produtos a serem produzidos possuem a mesma sequência de operações (FERREIRA, 2012, P.27), e em cujo ambiente os materiais e peças se deslocam em rotas que variam de acordo com a tarefa a ser executado, e o arranjo físico dos recursos depende do processo de fabricação, e os equipamentos são mais flexíveis, podendo ser utilizados na produção

de vários produtos diferentes (BORGES; DALCOL, 2002, p.6).

Quadro 2 - Ambiente Flow Shop x Ambiente Job Shop

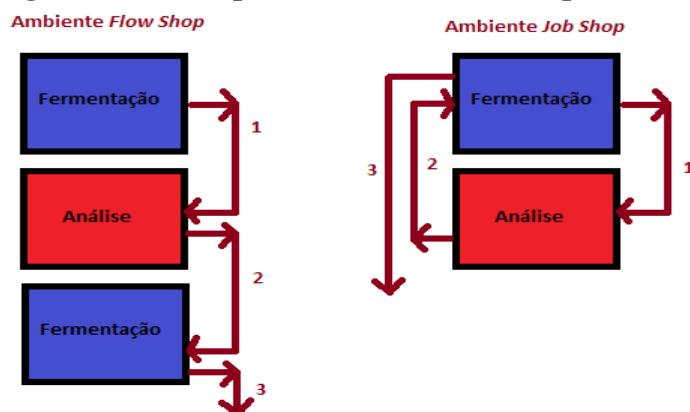
| <i>Flow shop</i> | <i>Job shop</i> |
|--|--|
| Rotas fixas | Rotas variáveis |
| <i>Layout</i> por produto | <i>Layout</i> por processo |
| Equipamento especializado | Equipamento flexível |
| Produção para estocar em grandes volumes | Produção por encomendas em pequenos volumes |
| Maiores <i>lead time</i> para aumentar a capacidade | Menores <i>lead time</i> para aumentar a capacidade |
| Capacidade bem definida | Capacidade difícil de se definir |
| Intensiva em capital | Intensiva em operários |
| Baixos estoques intermediários | Estoques intermediários significativos |
| Menor coordenação de materiais | Maior coordenação de materiais |
| Operadores altamente especializados e treinados p/ monitorar e controlar os equip. de processo | Operários possuem certa habilidade em algum tipo de equip. e/ou máquina que fabricam os produtos |
| <i>Overlapping</i> nas operações | Inexistência de <i>overlapping</i> nas operações |
| Falha nos equipamentos pode parar a planta | Falha nos equipamentos pode parar a produção de alguns itens |
| Atraso no recebimento de materiais e peças pode parar a planta | Atraso no recebimento de materiais e peças atrasam a produção de itens |
| Maior consumo de energia | Menor consumo de energia |

Fonte: BORGES e DALCOL, 2002

BORGES e DALCOL também destacam a diferença entre esses dois ambientes no que diz respeito ao “*overlapping*” dos equipamentos (Figura 4):

“Se for necessário para a elaboração de um produto a sequência de operações: fermentação, análise e fermentação em seu processo produtivo, em um *flow shop* ele irá passar pela seção de fermentação, depois pela seção de análise e em seguida irá para uma outra seção de fermentação (operação conhecida como *overlap*). Se essa mesma sequência for executada em um *job shop*, ele irá passar pela fermentação, em seguida pela análise e logo depois irá retornar à seção de fermentação anteriormente passada.” (BORGES; DALCOL, 2002, p.86).

Figura 4 - Overlap no ambiente Flow Shop x Job Shop



Fonte: Autor. Adaptado de BORGES; DALCOL, 2002

2.1.1.2 Produção por Encomenda

Este tipo de processo produtivo é utilizado por empresas que produzem apenas sob encomenda ou pedido. Normalmente, os produtos são muito complexos e únicos ou até mesmo personalizados, conforme solicitado pelo cliente que efetuou a encomenda, exigindo muito tempo para a “construção” do produto. Há uma demanda por grande variedade de máquinas e mão de obra especializada; a produção é feita para uma data de entrega pré-estabelecida e é difícil fazer as previsões da produção, pois cada produto é único e complexo, possuindo seu próprio plano de produção (FERREIRA, 2012, p.29).

2.1.1.2.1. Processos de projeto

Esta subdivisão dos processos por encomenda engloba as indústrias “artesaniais”, com baixo volume de produção e alta variedade (LOPES, 2013, p.3). Os produtos desenvolvidos são altamente customizados e com longo período de produção, utilizando recursos dedicados, isto é, recursos específicos para a construção de cada tipo de produto (FERREIRA, 2012, p.29).

2.1.1.2.2. Processos por tarefa

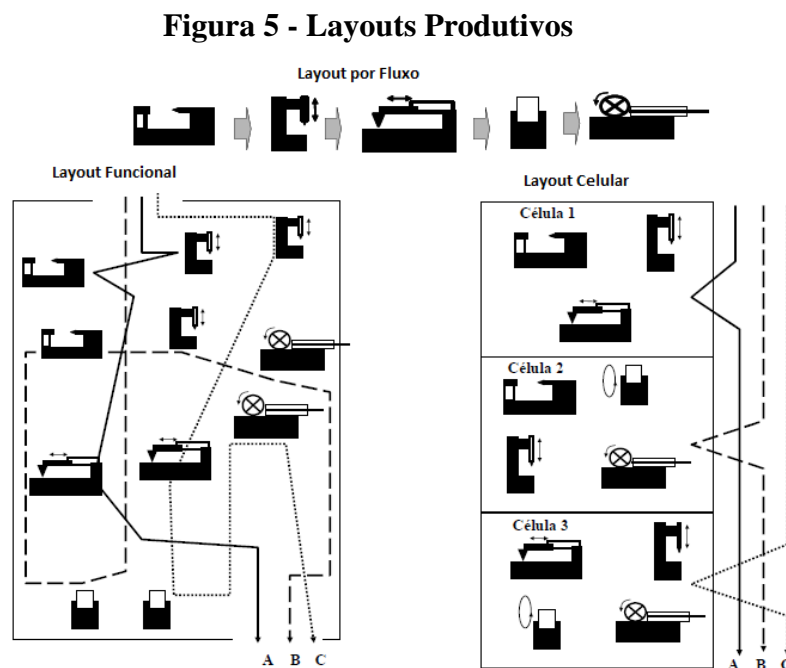
Embora semelhantes, este tipo de processo apresenta volumes de produção superiores aos dos processos de projeto (LOPES, 2013, p.3) e utiliza recursos compartilhados com o processamento de outros produtos (FERREIRA, 2012, p.29).

2.1.1.3 Outros tipos de produção

Além dos tipos de processo produtivo mais importantes, revisados acima, ALMEIDA (2012, p.16) apresenta também outros, os quais diferenciam dos até então apresentados apenas pelo layout da produção, isto é, a distribuição física dos recursos. São eles:

- a) produção funcional, onde os recursos produtivos são organizados no espaço, de acordo com as funções que executam, sendo “posicionados de acordo com a similaridade de seus processos”;
- b) produção celular, que é um processo intermediário entre a produção por fluxo e a produção funcional. Os equipamentos são agrupados em “células”, onde os produtos são executados em fluxo. Em cada célula ocorre um fluxo de produto, de forma que vários fluxos podem ocorrer simultaneamente;
- c) produção posicional, que é um tipo de processo no qual os recursos produtivos são deslocados até o local até onde o produto será produzido, devido à impossibilidade de o produto se “deslocar” na linha de produção. Ex.: Estaleiros, Construção Civil, montagem de caminhões, etc.

A Figura 5 representa alguns dos layouts produtivos.



Fonte: ALMEIDA, 2012

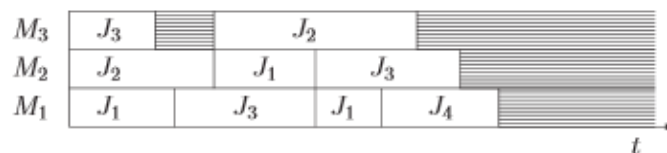
2.2 O Problema de Scheduling

Segundo REIS (1996), os problemas de *scheduling* tratam, de modo geral, de processos ou tarefas compostos por sequências de operações que devem ser executadas em uma ordem pré-determinada, onde cada operação envolve a alocação de uma ou mais máquinas (recursos) durante um intervalo de tempo, as quais podem ser utilizadas em uma única operação por vez. Cada solução possível possui o seu custo, atribuído através de uma função, e o objetivo do problema é encontrar a solução com menor custo.

STEBEL (2006, p. 7) afirma que a palavra inglesa *scheduling* não é traduzida para o português, para evitar confusão com a palavra “programação”, que é utilizada em várias áreas do conhecimento. Ele também afirma que há diferença entre planejamento e o *scheduling*, sendo que o primeiro se refere a metas de produção em períodos na ordem de meses ou anos, baseando-se em previsões de mercado e disponibilidade de recursos. O segundo, por sua vez, diz respeito a períodos menores, na ordem de semanas, dias ou horas, sendo necessário devido à concorrência na utilização de recursos.

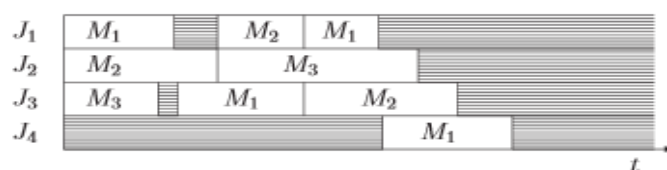
Dado um conjunto de processos a serem realizados e um conjunto de máquinas (recursos) disponíveis requeridos pelos processos, um *schedule* é, para cada processo, alocação de um ou mais intervalos de tempo a uma ou mais recursos, podendo ser representados por gráficos de *Gantt*, os quais podem ser orientados a recursos ou a processos, conforme as Figuras 6 e 7. Uma solução para o problema de *scheduling* consiste em encontrar um *schedule* que atenda a determinadas restrições (BRUCKER, 2006, p.12).

Figura 6 - Gráfico de *Gantt* de um *Schedule*, orientado a recursos (Máquinas, ou *Machines*)



Fonte: BRUCKER, 2006 p.13

Figura 7 - Gráfico de *Gantt* de um *Schedule*, orientado a tarefas (Jobs)



Fonte: BRUCKER, 2006, p.13

O campo de estudo do *Scheduling* é composto por um conjunto de técnicas para a resolução de diversos problemas, onde a complexidade do problema e a natureza do modelo, dentre outros critérios, são a base para se determinar qual ou quais das técnicas devem ser utilizadas, fazendo do *Scheduling* um estudo não apenas de metodologias, mas também de modelos (FERREIRA; 2012; p. 15).

2.3. Otimização Matemática

Um problema de otimização é um problema de programação matemática modelado de maneira a maximizar ou minimizar uma função matemática, chamada de função objetivo, a qual representa o que deve ser otimizado, sujeito a funções de restrição, que representam as restrições que o modelo deve assumir para atender à função objetivo (STEBEL, 2006, p. 45). Os objetivos a serem atingidos, expressos pelo modelo de otimização, geralmente envolvem maximizar ou minimizar a função objetivo (PIRES, 2006, p.4).

A palavra “programação”, no contexto da programação matemática, se refere a “planejamento”, pois consiste em elaborar um modelo matemático de planejamento de tarefas para atender a determinados objetivos, representados por funções matemáticas (HILLIER; LIEBERMAN, 2006, p.25).

Segundo PIRES (2006, p. 5), um modelo de otimização pode ser determinístico (todos os parâmetros do problema são conhecidos) ou estocásticos (valores dos parâmetros são atribuídos conforme funções de distribuição de probabilidade), contínuos (Variáveis podem assumir valores reais) ou discretos (variáveis assumem apenas valores inteiros). Um modelo de otimização determinístico e contínuo pode ser representado, de maneira geral, como na Equação 1.

$$\begin{array}{ll} \text{Minimizar} & f(x) \\ \text{Sujeito a} & h_i(x)=0 \quad i=1,\dots,m_h \\ & g_j(x)\leq 0 \quad j=1,\dots,m_g \\ & x\in\mathbb{R}^n \end{array}$$

$$\text{Em que } f, h_i \text{ e } g_j: \mathbb{R}^n \rightarrow \mathbb{R} \text{ são funções contínuas} \quad (1)$$

2.3.1 Programação Linear

Programas lineares são modelos de otimização matemática onde todas as funções envolvidas (objetivo e restrições) são lineares, podendo ser resolvidos, dentre outros métodos, de maneira

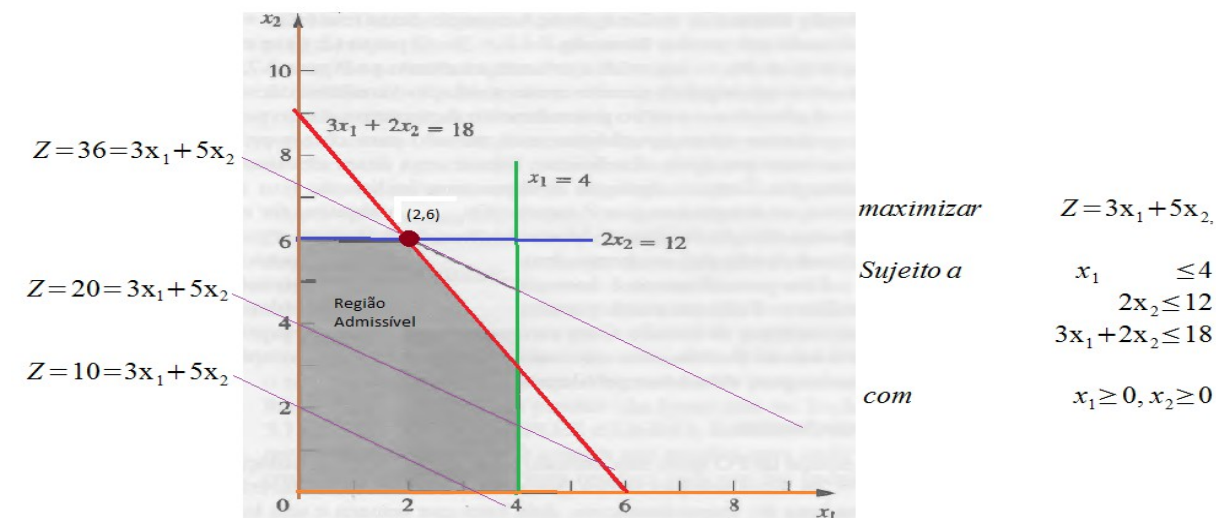
eficiente, mesmo que em grandes dimensões, por um algoritmo chamado *Simplex* (HILLIER; LIEBERMAN, 2006, p.26; BARRICO, 2013, p. 2).

A eficiência do modelo de programação linear é medida pela função objetivo, onde a melhor solução é a que maximiza ou minimiza, conforme o modelo, o valor desta função. As funções de restrição, por sua vez, representam os limites e restrições a que o modelo deve atender na busca pela solução ótima, sendo geralmente representados por inequações lineares (FROSSARD, 2009, p.27). Um problema de programação linear pode ser, tipicamente, representado pelo modelo da Equação 2 (PIRES, 2006, p.5).

$$\begin{array}{ll}
 \text{Minimizar} & c_1 x_1 + c_2 x_2 + c_3 x_3 + \dots + c_n x_n \\
 \text{Sujeito a} & a_{11} x_1 + a_{12} x_2 + \dots + a_{1n} x_n \geq b_1 \\
 & a_{21} x_1 + a_{22} x_2 + \dots + a_{2n} x_n \geq b_2 \\
 & \dots \quad \dots \quad \dots \quad \dots \\
 & a_{m1} x_1 + a_{m2} x_2 + \dots + a_{mn} x_n \geq b_m \\
 & x_1, x_2, \dots, x_n \geq 0
 \end{array} \tag{2}$$

Um problema de programação linear pode ser resolvido através da análise gráfica (resolução gráfica) do problema, obtida traçando-se em um plano cartesiano, onde cada eixo corresponde a um x_i (Portanto, apenas problemas com 2 ou 3 variáveis de decisão podem ser resolvidos pelo método gráfico), as retas referentes às restrições, obtendo assim a **região admissível** ou **região de soluções viáveis**, que é a região delimitada pelas restrições e compreende a todas as respostas possíveis para o modelo, e em seguida selecionar o ponto dentro dessa região que maximiza ou minimiza a função objetivo, escolhendo aleatoriamente um valor para a função que mantenha as variáveis de decisão dentro da região admissível (função Z), e traçar várias retas Z paralelas, até que uma delas atinja um ponto de interseção das restrições. Esse ponto será a solução ótima para o problema (HILLIER; LIEBERMAN, 2006, p. 29; BARRICO, 2013, p.6). A Figura 8 representa a solução gráfica de um problema de programação linear.

Figura 8 - Exemplo de Solução Gráfica



Fonte: adaptado de HILLIER; LIEBERMAN, 2006

Quando o problema exigir que as variáveis sejam inteiras, denominamos o modelo de **Programação Linear Inteira**, ou apenas **Programação Inteira**, que consiste basicamente em um modelo de programação linear com uma restrição adicional de que as variáveis devam ser inteiras. Quando apenas algumas das variáveis devem ser inteiras, denominamos o modelo de **Programação Inteira Mista (PIM)** (HILLIER; LIEBERMAN, 2006, p.462) ou **Programação Linear Inteira Mista (PLIM)**, sendo amplamente utilizado um algoritmo chamado *branch and bound* (BB) para sua resolução, o qual baseia-se em sucessivas divisões da região admissível em subdivisões mais fáceis de serem manipuladas, até encontrar a solução ótima (STEBEL, 2006, p.45).

2.3.2 Programação Não Linear

HILLIER e LIEBERMAN (2006, p.530) afirmam que, embora o modelo de programação linear se aplique a vários problemas do mundo real, em muitos casos o mesmo não é válido, pois em muitos problemas reais há algum grau de não linearidade, como em problemas de planejamento econômico, sendo necessário trabalhar diretamente com modelos de programação não linear.

A programação não linear é destinada à otimização de modelos em que a função objetivo é não linear, e as restrições podem ser lineares ou não lineares (FROSSARD, 2009, p.27), podendo também ser classificados como restritos ou irrestritos (FROSSARD, 2009, p.27; LUENBERGER; YE, 2008, p.2; OLIVEIRA, 1989, p.3).

Não existem algoritmos gerais para a resolução de problemas de programação não linear, de forma que a resolução dos mesmos depende da utilização de métodos e algoritmos voltados para

subclasses específicas desses problemas (PIRES, 2006, p.62), como os métodos de Gradiente, de Lagrange, de Newton, dentre outros (FROSSARD, 2009, p.27).

PIRES (2006, p.62), HILLIER e LIEBERMAN (2006, p. 530) também afirmam que um problema de programação não linear pode ser representado genericamente pela Equação 3.

$$\begin{aligned}
 &\text{Encontrar } x = (x_1, x_2, \dots, x_n) \text{ de modo a} \\
 &\begin{array}{ll}
 \text{Maximizar} & f(x) \\
 \text{Sujeito a} & g_i(x) \leq b_i, \quad i = 1, \dots, m \\
 & x \geq 0
 \end{array} \\
 &\text{onde } f \text{ e } g_i \text{ são funções definidas em } \mathbb{R}^n
 \end{aligned} \tag{3}$$

Quando um problema de Programação Inteira Mista (PIM) envolve restrições não lineares, chamamos o modelo de Programação Não Linear Inteira Mista (PNLIM) (STEBEL, 2006, p.45).

2.3.2.1 Programação não linear Irrestrita

Segundo LUENBERGER e YE (2008, p.183), OLIVEIRA (1989, p.9) e PIRES (2006, p.62), Problemas de otimização irrestritos são descritos pela Equação 4.

$$\begin{aligned}
 &\begin{array}{ll}
 \text{minimizar} & f(x) \\
 \text{sujeito a} & x \in \mathbb{R}^n,
 \end{array} \\
 &\text{onde } f \text{ é uma função real}
 \end{aligned} \tag{4}$$

Como o próprio nome sugere, os problemas de programação não linear irrestrita não possuem restrições, de forma que o objetivo é simplesmente minimizar ou maximizar $f(x)$ ao longo de todos os valores de $x = (x_1, x_2, x_3, \dots, x_n)$ (HILLIER; LIEBERMAN, 2006, p.540). Segundo PIRES (2006, p.62), HILLIER e LIEBERMAN, a condição necessária para a existência de um ótimo em x^* é apresentada nas equações 5 e 6. Caso a função $f(x)$ seja côncava, a condição citada também é suficiente:

$$\nabla f(x^*) = 0 \tag{5}$$

ou

$$\frac{\partial f}{\partial x_j} = 0 \quad \text{em } x = x^*, \quad \text{para } j = 1, 2, \dots, n. \quad (6)$$

Se a condição for suficiente (se $f(x)$ for côncava), encontrar uma solução para x^* consiste basicamente em resolver um sistema de n equações obtidas pela Equação 6 (PIRES, 2006, p. 62; HILLIER; LIEBERMAN, 2006, p.540). Porém, na maioria dos casos em que $f(x)$ é uma função não linear, as n equações também são não-lineares, dificultando a solução analítica dos sistemas, tendo que optar por procedimentos de busca algorítmica (HILLIER; LIEBERMAN, 2006, p.540). Muitas vezes, de maneira analítica, consegue-se apenas “encontrar uma sucessão eventualmente convergente para x^* .” (PIRES, 2006, p. 62).

Quando $f(x)$ não possui uma restrição quanto ao sinal da função, por exemplo $x > 0$, as Equações 5 e 6 podem ser simplificados **pela** Equação 7 (PIRES, 2006, p. 62; HILLIER; LIEBERMAN, 2006, p.540).

$$\begin{cases} \frac{\partial f}{\partial x_j} \leq 0, & \text{em } x = x^*, \text{ se } x_j^* = 0 \\ \frac{\partial f}{\partial x_j} = 0, & \text{em } x = x^*, \text{ se } x_j^* > 0 \end{cases} \quad (7)$$

2.3.2.2 Programação não linear Restrita

Na prática, muitos problemas não lineares apresentam restrições, pois em instâncias de problemas complexos, como por exemplo, um planejamento de produção de uma grande empresa, é necessário subdividir o problema em problemas menores, com restrições aplicadas para limitar o escopo de cada subproblema, como por exemplo, as restrições de orçamento aplicada em um problema de planejamento (LUENBERGER; YE, 2008, p.4).

Segundo OLIVEIRA (1989, p. 60), uma representação genérica, capaz de representar a maioria dos casos reais de programação linear restrita, é apresentado na Equação 8.

$$\begin{array}{ll} \text{Minimizar} & f(x) \\ \text{Sujeito a} & \\ & f_i(x) \leq 0; \quad i = 1, \dots, m \\ & h_i(x) = 0; \quad i = 1, \dots, q \end{array} \quad (8)$$

Quando apenas a função objetivo for não-linear, chamamos o problema de **programação**

linearmente restrita, sendo problemas bem simples com diversos algoritmos, baseados no método *simplex*, disponíveis para sua resolução. Uma especificação deste caso é o problema de **programação quadrática**, que é um problema linearmente restrito com uma função objetivo quadrática e que envolve o quadrado de uma variável básica ou o produto de duas variáveis (HILLIER; LIEBERMAN, 2006, p.541). Segundo PIRES (2006, p.63), em modelos de programação quadrática, a função objetivo é da forma expressa na Equação 9, onde Q é uma matriz de n por n elementos, e b é um vetor, nas restrições, com n componentes, sendo que muitos algoritmos já foram desenvolvidos para este caso, apresentando um desempenho muito bom quando a função objetivo é côncava, e geralmente baseados também no método *simplex*.

$$f(x) = x^T Q x + c^T x \quad (9)$$

Existem muitos outros tipos de programação não linear restrita, como por exemplo a **programação convexa**, onde a função objetivo é côncava e as restrições são convexas; a **programação separável** ou **programação com variáveis separáveis**, que é um tipo especial de programação convexa onde todas as funções envolvidas são de variáveis separáveis; a **programação não-convexa**, que engloba todos os casos que não são atendidos pela programação convexa; a **programação fracionária**, que ocorre quando a função objetivo corresponde a uma razão entre duas outras funções; a **programação geométrica**, que resulta da aplicação de problemas de programação não-linear a modelos de desenvolvimento de engenharia ou certos problemas econômicos e de estatística, onde a função objetivo e as restrições assumem a forma da Equação 10; dentre outros tipos (PIRES, 2006, p. 63; HILLIER, LIEBERMAN, 2006, p.541).

$$g(x) = \sum_{i=1}^N c_i P_i(x)$$

em que:

$$P_i(x) = x_1^{a_{i1}} x_2^{a_{i2}} \dots x_n^{a_{in}} \quad \text{para } i = 1, 2, \dots, N \quad (10)$$

2.4 Algoritmos Genéticos

Os algoritmos genéticos, propostos por Holland em 1970, são modelos baseados na teoria da seleção natural de Darwin, para abordar uma série ampla de problemas, principalmente os de otimização. O processo consiste basicamente em inicializar o modelo com uma população de indivíduos, que modelam possíveis soluções para o problema a ser tratado, e em seguida submeter

essa população ao processo evolucionário, sendo composto pelas etapas de avaliação, seleção, cruzamento, mutação, atualização e finalização (LUCAS, 2002, p.5).

Segundo SOARES (1997, p.9), os AGs são sistemas artificiais onde o problema a ser resolvido é representado através de uma função matemática, onde os indivíduos “mais fortes” geram um melhor valor para a função. Dessa forma, cada indivíduo representa uma possível solução, sendo na verdade uma concatenação de variáveis ou cadeias de caracteres, chamados de cromossomos, onde cada caractere (gene) encontra-se numa posição (locus) e possui um determinado valor (alelo). Trabalhando-se simultaneamente com grupo de indivíduos, os melhores são selecionados para o cruzamento, gerando novos indivíduos. Além disso, alguns indivíduos podem, eventualmente, sofrer mutação (alteração aleatória de um gene).

REDUSINO (2013, p. 1) afirma que algoritmos genéticos são técnicas de busca iterativa que simula um processo evolutivo em uma população de possíveis soluções que, embora seja um método aleatório, é guiado por um mecanismo de adaptação que tende a gerar populações cada vez mais aptas, convergindo para uma solução ótima para o problema.

Figura 9 - Fluxograma dos passos do AG



Fonte: Adaptado de FILITTO, 2008

2.4.1 Inicialização

A inicialização de um algoritmo genético consiste em determinar uma população inicial de indivíduos, a qual passará pelos processos evolutivos em busca da solução ótima. Geralmente, uma população aleatória é gerada, garantindo maior “biodiversidade”, mas outros tipos de funções e aproximações para geração da população inicial podem ser utilizados (LUCAS, 2002, p.10).

2.4.2 Avaliação

Nesta etapa, cada indivíduo da população é avaliado para que seja calculado o seu valor de aptidão, ou *fitness* (LUCAS, 2002, p.11). A função de avaliação (função objetivo) é “o elo de ligação” entre o algoritmo e o modelo proposto (SOUSA, 2013, p.6), recebendo como entrada um cromossomo e retornando um número ou sequência de números que representam o desempenho, aptidão ou *fitness* do cromossomo (FILITTO, 2008, p. 139).

2.4.3 Seleção

Esta é a etapa responsável por fazer com que as melhores características da população perpetuem, através da seleção de dois indivíduos para posterior cruzamento (LUCAS, 2002, p.12). A escolha dos indivíduos deve ser feita de forma que os indivíduos mais adaptados (com maior valor de aptidão ou *fitness*) possuam maior probabilidade de serem selecionados (FILITTO, 2008, p. 139).

O algoritmo de seleção mais utilizado é o algoritmo de Monte Carlo, ou de seleção por roleta, onde cada indivíduo possui uma probabilidade de seleção, de acordo com sua aptidão. Para calcular a probabilidade, as aptidões de todos os indivíduos são somadas para em seguida calcular o percentual para cada indivíduo. Cada indivíduo possuirá o percentual calculado da roleta, de forma que indivíduos com maior aptidão ocupam maior porção da roleta. De acordo com o tamanho da população, a roleta é “girada” várias vezes, selecionando os indivíduos que participarão do processo de geração da nova população (SOARES, 1997, p.13; FILITTO, 2008, p.139).

ZUBEN (p. 18) apresenta outros algoritmos de seleção muito utilizados, como a seleção por Torneio, onde indivíduos são selecionados em pares aleatórios para realizar “competições”, onde vence a competição o indivíduo com melhor *fitness*, e a seleção é feita com base no número de vitórias dos indivíduos em uma quantidade q de competições; O algoritmo de seleção por *rank* (ou classificação), que se baseia na posição dos indivíduos quando ordenados de acordo com seu *fitness*; dentre outros. ZUBEN apresenta também a estratégia conhecida como Elitismo, que consiste em manter os $P\%$ melhores indivíduos da população atual na população seguinte.

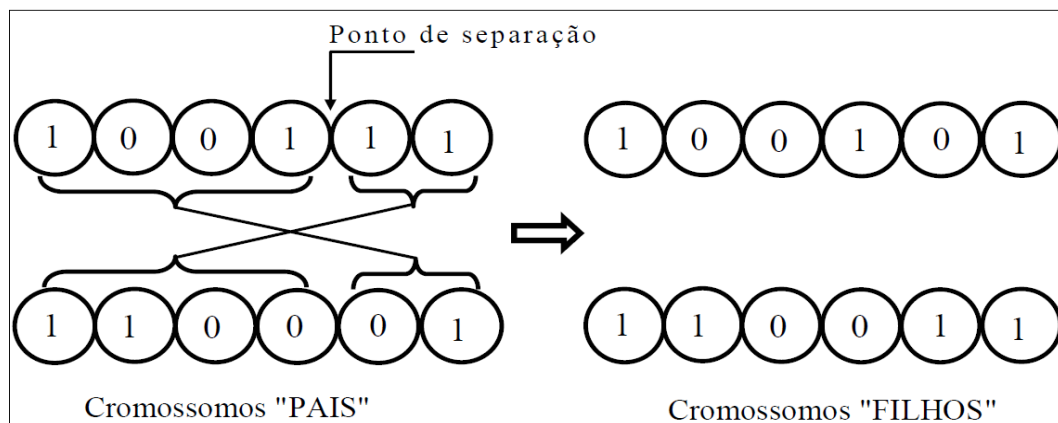
2.4.4 Cruzamento

Após selecionar os indivíduos, os mesmos podem passar (de acordo com uma probabilidade pré-determinada) pelo processo de cruzamento, reprodução ou *crossover* (LUCAS, 2002, p.14),

onde os cromossomos de dois indivíduos selecionados são recombinados, a partir dos “pontos de cruzamento” gerando dois novos indivíduos filhos (FILITTO, 2008, p. 140).

Segundo SOARES (1997, p.14), o cruzamento é simplesmente um processo que permite a troca de material genético entre os indivíduos, sendo um mecanismo que permite facilmente a recombinação de soluções.

Figura 10 - Cruzamento de cromossomos.



Fonte: SOUSA, 2013

2.4.5 Mutação

O processo de mutação consiste na alteração, de acordo com uma probabilidade pré-estabelecida, de um gene aleatório em indivíduos gerados após o processo de cruzamento (LUCAS, 2002, p.16; FILITTO, 2008, p.141).

A mutação tem o objetivo de inserir novas características na população e também de recuperar características perdidas com os processos de cruzamento e até mesmo na própria mutação (SOARES, 1997, p.15).

A mutação geralmente ocorre com baixa probabilidade (CASTRO; ZUBEN, 2013, p.7), e pode ser realizada de várias formas, como por exemplo, a **mutação flip**, onde genes são alterados por um valor válido sorteado aleatoriamente; a **mutação por troca** ou *swap*, onde são sorteados pares de genes que tem o seu material genético trocado entre si; ou **mutação creep**, onde um valor aleatório é somado ou subtraído de um gene (LUCAS, 2002, p.16).

2.4.6 Atualização e finalização

O processo de atualização consiste em introduzir os novos indivíduos na população. Essa

introdução pode ocorrer de várias formas, como por exemplo, substituindo os pais, ou mantendo os n melhores indivíduos e eliminando os demais, dentre outros métodos. Por fim, na etapa de finalização, a solução atual é testada, para verificar se os critérios de parada foram atendidos, como o número de gerações ou algum critério de convergência. Caso o critério de parada tenha sido atendido, o algoritmo é finalizado e a população resultante corresponde à solução para o problema (LUCAS, 2002, p.17).

2.5 Pipeline

Pipeline é uma arquitetura de processadores que permite a execução de instruções em paralelo, aumentando o *throughput* do trabalho, mas sem diminuir a latência de execução das instruções. Esse resultado é obtido dividindo-se o ciclo de instruções do processador em estágios, onde cada estágio utiliza diferentes recursos da CPU, de forma que instruções diferentes podem ser executadas ao mesmo tempo, sendo que cada instrução ocupa um estágio a cada ciclo de *clock* do processador (SILVA, 2013, p.2).

Segundo CASILLO e SILVA (2013, p.6), e SILVA (2013, p. 16) o tempo total de execução em um pipeline é dado pela Equação 11, onde j é o número de estágios, n é a quantidade de instruções (ou tarefas) a serem executadas, e T é o tempo de execução de um estágio.

$$T_{total} = (j + (n - 1)) T \quad (11)$$

2.6 Trabalhos Relacionados

STEBEL (2006) utiliza técnicas de otimização matemática, baseadas em programação linear inteira mista (PLIM) em tempo contínuo, para o problema de *scheduling* de transferência de estocagem em refinarias de petróleo, onde os estados de operação possíveis para os tanques são utilizados para modelar o problema, o qual também considera informações adicionais, como o custo diferenciado de utilização de energia elétrica pelos recursos. Inicialmente, STEBEL apresenta o modelo baseado em PLIM e, para melhoria de desempenho, apresenta uma versão híbrida de PLIM com PLR (programação lógica por restrições).

ALMEIDA et. al (2010) propõem uma solução, utilizando algoritmos genéticos, para o problema de *scheduling* da produção de óleo combustível e asfalto em refinarias de petróleo, previamente analisado e resolvido através de programação matemática por JOLY (1999).

DERIZ (2007) propõe uma solução, utilizando algoritmos genéticos, para resolver de

maneira eficiente o problema de programação da produção em indústrias de manufatura com recursos compartilhados, visando um menor tempo de processamento e baixo tempo de resposta do processo de busca.

FERREIRA (2012) apresenta uma modelagem, utilizando programação linear, para escalonamento a curto prazo da produção em ambientes *job shop*, onde são realizadas várias tarefas utilizando várias máquinas, visando diminuir ao máximo as perdas geradas pelo tempo improdutivo e o tempo de *setup* dos recursos, o qual é dependente da sequência de tarefas a serem executadas.

ÖZDAMAR (1999) propõe uma abordagem, utilizando algoritmos genéticos, para o problema de *scheduling* de projetos com restrição de recursos, onde as atividades podem ser executadas em vários modos de operação, e cada um desses modos demanda uma quantidade diferente de tempo e recursos, os quais podem ser renováveis ou não renováveis, visando minimizar o tempo total de produção.

3. METODOLOGIA

Através deste trabalho, é apresentada uma solução para o problema de *scheduling* que minimiza o *lead time*, ou *makespan*, de um processo produtivo em lotes e ambiente *job shop* com recursos compartilhados e concorrentes, levando em conta parâmetros como o tempo de utilização dos recursos por cada tarefa produtiva e a ordem de precedência entre as tarefas. Também foi desenvolvida uma ferramenta de *software* que implementa a solução proposta.

3.1 O Ambiente modelado

O ambiente modelado é do tipo *job shop*, para uma produção em lotes ou batelada, onde existem t tarefas produtivas a serem executadas, denotadas por T_i , onde $i = 1, 2, \dots, t$. Por exemplo, tem-se as tarefas “Fazer Bolo” (T_1) e “Fazer Torta” (T_2).

Essas tarefas necessitam alocar recursos, que podem ser equipamentos, ferramentas, mão de obra, dentre outros, sendo os r recursos existentes denotados por R_j e com quantidade disponível Q_j , onde $j = 1, 2, \dots, r$. Por exemplo, na Tabela 1 os recursos “batedeira”, “forno”, “freezer”, “Confeiteiro”, “Inspetor” e “embalador” (R_1, R_2, R_3, R_4, R_5 e R_6 , respectivamente) estão associados às suas respectivas quantidades disponíveis.

Tabela 1 - Exemplos de recursos e sua disponibilidade

| Recurso | Quantidade |
|-------------|------------|
| Batedeira | 2 |
| Forno | 1 |
| freezer | 1 |
| Confeiteiro | 1 |
| Inspetor | 1 |
| Embalador | 1 |

Fonte: Autor

Os recursos são compartilhados entre as tarefas, onde tarefas diferentes podem utilizar recursos em comum, porém, um mesmo recurso não pode ser utilizado por mais de uma tarefa simultaneamente. Além disso, cada tarefa pode ou não possuir uma ordem de utilização dos recursos, isto é, alguns recursos podem ser utilizados apenas se outro, de ordem menor, já tiver sido utilizado em um momento anterior, estabelecendo uma relação de dependência entre eles. Para os mesmos dados exemplificados, pode-se considerar as seguintes sequências de utilização (precedência):

- a) “Batedeira → Forno → Confeiteiro → Embalador”, para o Bolo;
- b) “Batedeira → Freezer → Confeiteiro → Embalador”, para a torta;
- c) O recurso “inspetor” não é utilizado por nenhuma das tarefas em questão.

Cada recurso é associado a um tempo médio de utilização por cada tarefa, conforme exemplificado pela Tabela 2.

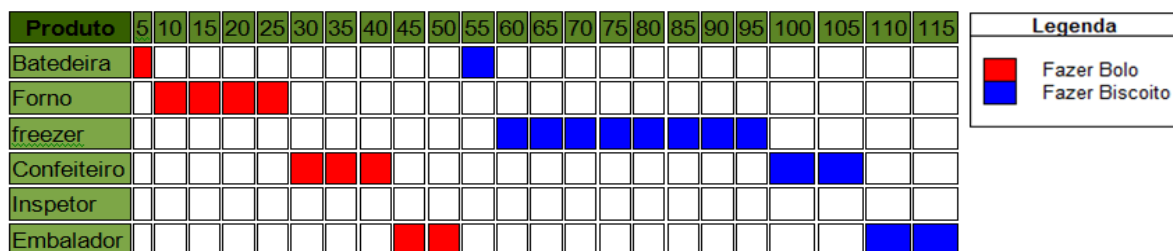
Tabela 2 - Tempo de utilização dos recursos pelas tarefas

| Tarefa | Recurso | | | | |
|-------------|-----------|-------|---------|-------------|-----------|
| | Batedeira | Forno | freezer | Confeiteiro | Embalador |
| Fazer Bolo | 5 | 20 | 0 | 15 | 10 |
| Fazer Torta | 5 | 0 | 40 | 10 | 10 |

Fonte: Autor

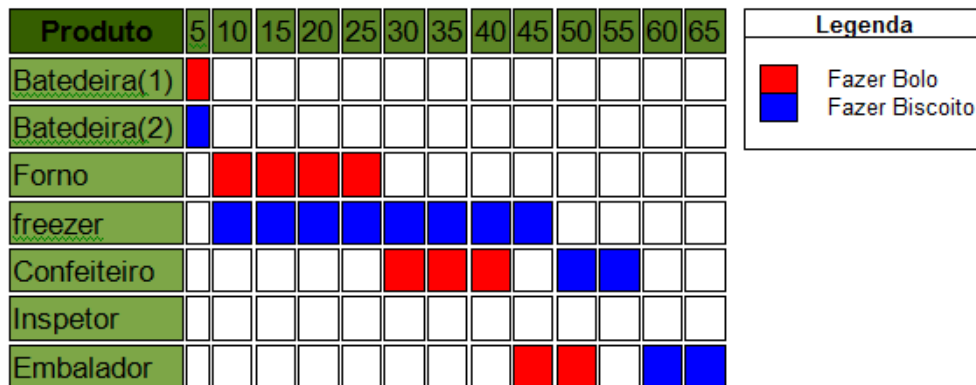
Uma solução possível para o problema seria a mostrada no diagrama da Figura 11, obtida através da execução sequencial das tarefas, de forma que uma tarefa apenas é iniciada após a anterior ser concluída. Embora a implementação deste tipo de programação seja simples, a mesma é ineficiente, pois se pode notar que a mesma possui 115 minutos de produção, sendo que há a possibilidade de escalonar a utilização dos recursos de forma que a produção seja realizada em apenas 65 minutos, conforme o diagrama da Figura12.

Figura 11 - Diagrama de *gantt* mostrando o fluxo da produção com solução sequencial



Fonte: Autor

Figura 12 - Diagrama de *gantt* mostrando a produção otimizada



Fonte: Autor

Como é possível observar pelas figuras acima, a programação sequencial da produção, além de demorar um tempo maior para a realização dos dois processos, ocasiona em um desperdício de recursos, pois nota-se que apenas uma das duas batedeiras disponíveis é utilizada. Uma melhor administração dos recursos levaria a um tempo de produção menor, permitindo a utilização simultânea de recursos concorrentes que existam em maior quantidade.

A metodologia proposta por este trabalho permite que, em ambientes produtivos semelhantes ao descrito, o planejamento seja otimizado apresentando uma solução como a da Figura 12, ou próxima a ela.

3.2 Técnicas de Modelagem e solução

A solução do problema de *scheduling* descrito se dá em duas etapas:

- PNL: Elaboração de uma modelagem para o problema utilizando programação matemática não linear (PNL), representando uma função objetivo para minimizar o *makespan* da produção e restrições quanto à sequência de operações dos recursos pelas tarefas, atendimento aos prazos de entrega, concorrência entre os recursos (não permitir que um mesmo recurso seja alocado para dois processos simultâneos, a menos que haja multiplicidade de recursos), permitindo operação simultânea dos recursos não concorrentes em intervalos de tempo, dentre outras;
- Algoritmos Genéticos: Como o problema modelado é extremamente complexo, devido ao alto número de restrições necessárias para manter a integridade do *schedule*, e ao fato de se tratar evidentemente de um problema combinacional, onde

as variáveis de decisão e o espaço de busca crescem exponencialmente de acordo com o número de tarefas e recursos a serem utilizados, uma solução matemática convencional pode não ser eficaz, além de difícil implementação. Portanto, é proposta uma abordagem de algoritmo genético modificado para atender ao problema, onde o modelo de PNL desenvolvido é utilizado para avaliar a aptidão dos indivíduos e também para inicializar a população apenas com indivíduos que representam soluções possíveis, que atendam ao modelo matemático. Neste modelo, cada indivíduo representa um *schedule* válido, e os operadores genéticos são utilizados para obter indivíduos com maior *fitness*, isto é, que representem um menor tempo de produção (*makespan*).

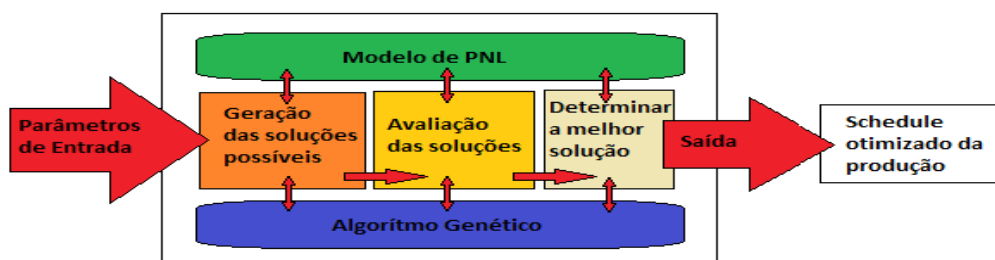
3.3 Modelagem do Processo

A ferramenta desenvolvida é responsável, macroscopicamente, pelas seguintes etapas:

- a) geração das soluções possíveis;
- b) avaliação das soluções;
- c) determinar a melhor solução.

Conforme informado na seção 3.2, esse processo é implementado através de uma combinação de AG com PNL. A Figura 13 ilustra o funcionamento dessa ferramenta.

Figura 13 - Funcionamento da ferramenta de software

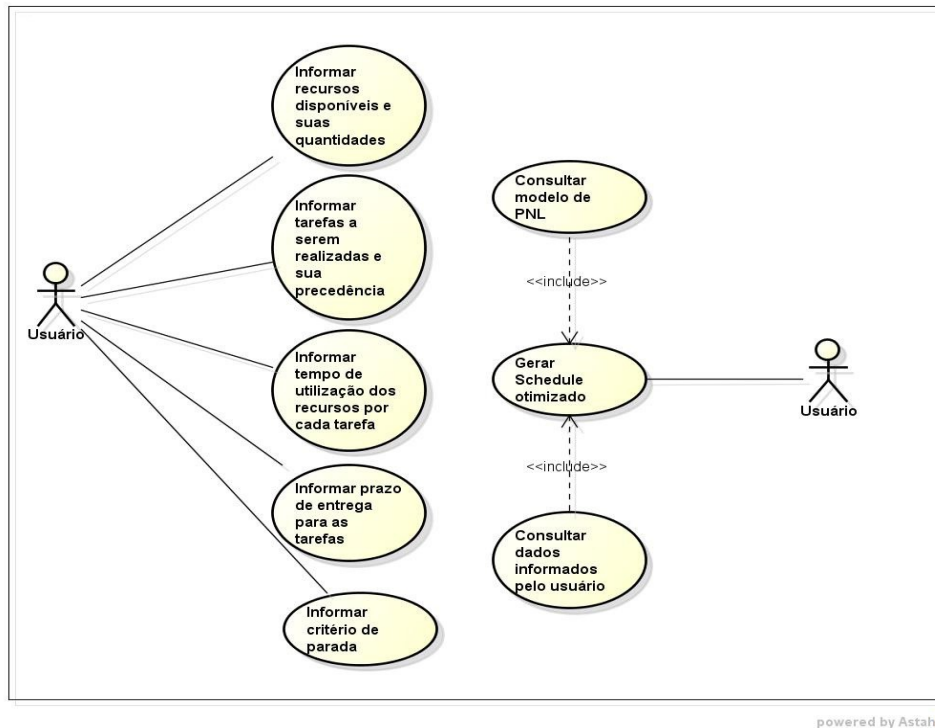


Fonte: Autor

As figuras 14 e 15 representam dois diagramas UML para modelagem do processo, sendo a primeira um diagrama de casos de uso e a segunda um diagrama de atividades, onde esta última representa o fluxo de operação do mesmo. O ator “usuário” utilizado não representa necessariamente um usuário humano, mas sim qualquer meio que possa fornecer as entradas

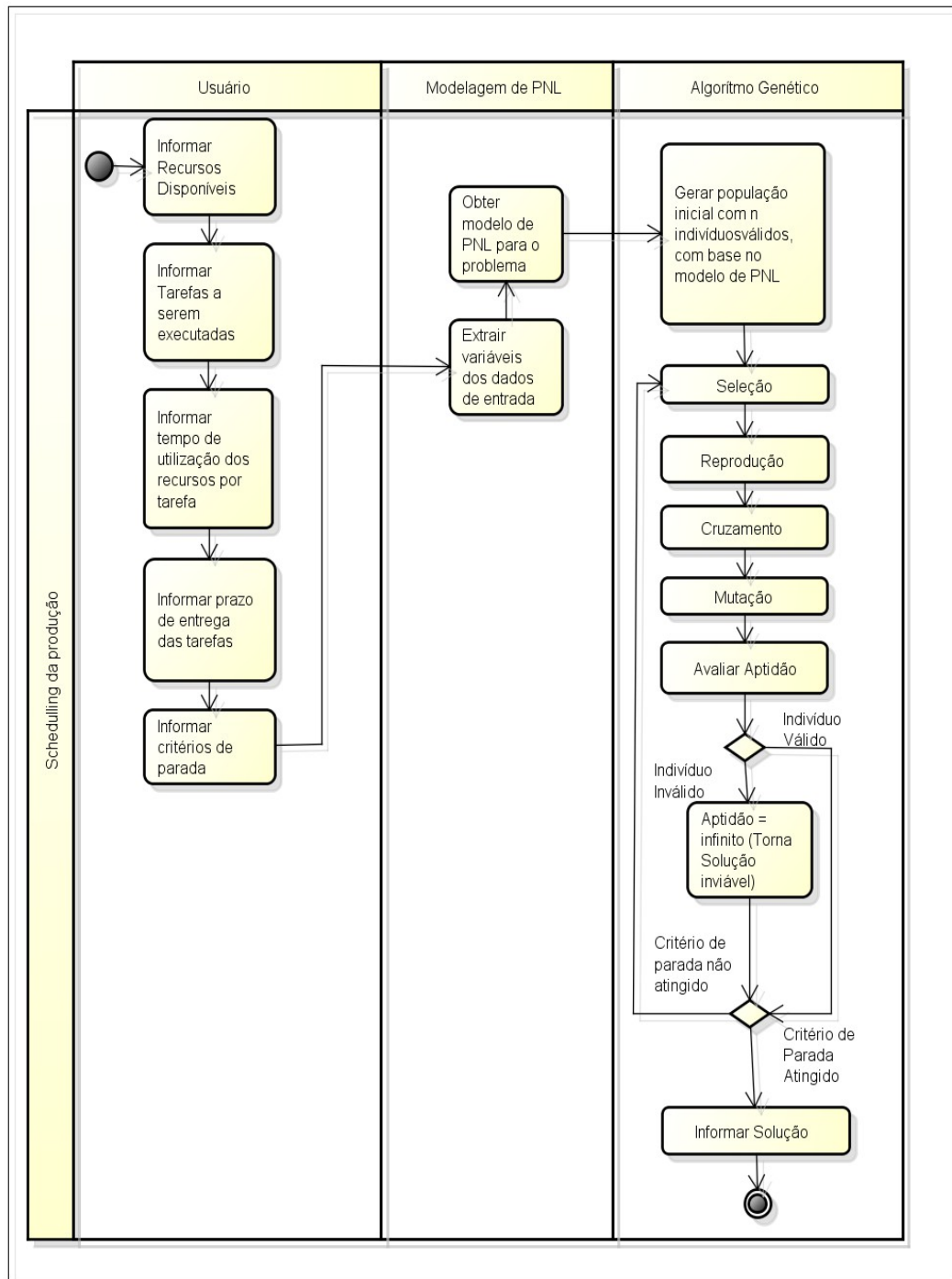
necessárias para a execução dos algoritmos, como por exemplo, um sistema de MRP II.

Figura 14 - Diagrama de casos de uso



Fonte: Autor

Figura 15 - Diagrama de Atividades



powered by Astah

Fonte: Autor

3.4 Modelagem Matemática

Para modelagem do problema, considera-se que os tempos de utilização dos recursos pelos processos já tenham sido calculados previamente, com base em quantidades de matéria prima,

quantidade a produzir e capacidade dos recursos. Portanto, não cabe ao escopo deste modelo preocupar-se com a estimativa do valor de tais variáveis, uma vez que as mesmas já terão sido utilizadas para fornecer os tempos de uso dos recursos, que são utilizados como entrada para o modelo proposto.

Durante o processo de modelagem, é utilizado como exemplo o ambiente apresentado na seção 3.1.

3.4.1 Parâmetros de Entrada

Tem-se então, os seguintes dados iniciais, como entrada para o problema:

- a) p : Quantidade de Processos a ser programados. No exemplo, $p = 2$;
- b) r : Quantidade total de tipos de recursos existentes (recursos repetidos são considerados como um único recurso). No exemplo, $r = 6$;
- c) P : Conjunto de Processos existentes, denotado por: $P = \{P_1, P_2, P_3, \dots, P_p\}$. No exemplo, $P = \{\text{Fazer Bolo}, \text{Fazer Biscoito}\}$;
- d) R : Conjunto de Recursos disponíveis, denotado por: $R = \{R_1, R_2, R_3, \dots, R_r\}$. No exemplo, $R = \{\text{Batedeira}, \text{Forno}, \text{Freezer}, \text{Confeiteiro}, \text{Inspetor}, \text{Embalador}\}$;
- e) matriz de Tempo de Utilização (T), onde cada elemento t_{ij} ($i = 1, 2, \dots, p$ e $j = 1, 2, \dots, r$) corresponde ao tempo de utilização do recurso R_j pelo processo P_i . Para os recursos não utilizados por um processo, utiliza-se tempo zero. A matriz T para o exemplo é a apresentada na Equação 13.

$$T = \begin{bmatrix} t_{1,1} & t_{1,2} & t_{1,3} & \dots & t_{1,r} \\ t_{2,1} & t_{2,2} & t_{2,3} & \dots & t_{2,r} \\ \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \cdot & \dots & \cdot \\ t_{p,1} & t_{p,2} & t_{p,3} & \dots & t_{p,r} \end{bmatrix} \quad (12)$$

$$T = \begin{bmatrix} 5 & 20 & 0 & 15 & 10 \\ 5 & 0 & 40 & 10 & 10 \end{bmatrix} \quad (13)$$

- f) Matriz de Ordenação (O), que contém a ordem que cada recurso deve ser utilizado pelo processo, sendo a ordem representada por números naturais de 1 a r (ou zero,

para os recursos que não são utilizados por um processo). Para um mesmo processo, não pode haver ordem repetida (exceto para o zero), e se uma ordem x é atribuída a um recurso para um processo, todas as ordens de 1 a $x-1$ devem, obrigatoriamente, ser atribuídas a outros recursos desse mesmo processo. Para o exemplo apresentado, tem-se a matriz apresentada na Equação 15.

$$O = \begin{bmatrix} O_{1,1} & O_{1,2} & O_{1,3} & \dots & O_{1,r} \\ O_{2,1} & O_{2,2} & O_{2,3} & \dots & O_{2,r} \\ \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \cdot & \dots & \cdot \\ O_{p,1} & O_{p,2} & O_{p,3} & \dots & O_{p,r} \end{bmatrix} \quad (14)$$

$$O = \begin{bmatrix} 1 & 2 & 0 & 3 & 4 \\ 1 & 0 & 2 & 3 & 4 \end{bmatrix} \quad (15)$$

- g) Vetor de Quantidades (Q), onde cada elemento q_j (com $j = 1, 2, \dots, r$) contém a quantidade disponível do recurso R_j . A matriz Q para o exemplo proposto é apresentada na Equação 17.

$$Q = [q_1 \quad q_2 \quad q_3 \quad \dots \quad q_r] \quad (16)$$

$$Q = [2 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1] \quad (17)$$

- h) Vetor de tempo limite (L), onde cada elemento l_i contém o tempo máximo que o processo P_i pode demorar. Para o exemplo apresentado, supõem-se ambos os processos com mesmo limite de tempo: 500 minutos. O vetor L referente a esse exemplo é apresentado na Equação 19.

$$L = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ \vdots \\ \vdots \\ l_p \end{bmatrix} \quad (18)$$

$$L = \begin{bmatrix} 500 \\ 500 \end{bmatrix} \quad (19)$$

3.4.2 Função Objetivo

O planejamento pode ser dividido em conjuntos de variáveis de decisão que indicam, em um intervalo de tempo discreto, todos os recursos que estão sendo utilizados nesse intervalo. A sequência desses intervalos representa todo o processo produtivo, indicando a ordem em que os recursos devem ser utilizados e também aqueles que podem ser utilizados em paralelo.

No pior caso, o modelo proposto deve gerar um *schedule* que corresponde a um *pipeline* não otimizado (com os processos executadas na pior ordem), mas que ainda assim é uma solução aceitável para o problema, uma vez que um *pipeline* qualquer é mais eficiente do que o processo executado sequencialmente, onde um processo espera o outro finalizar para poder iniciar.

Um *pipeline* com k estágios e n tarefas a serem executadas leva $k+n-1$ unidades de tempo para executar todas as tarefas. De maneira análoga, onde os p processos a serem executados podem ser entendidos como tarefas de um *pipeline*, e os intervalos (ou janelas) de tempo discreto adotados podem ser entendidos como os estágios. Portanto, serão executadas $r + p - 1$ etapas.

Chamando de D_k a matriz formada pelas variáveis binárias de decisão $d_{k,i,j}$, com $k = 1, 2, \dots, r+p-1$; $i = 1, 2, \dots, p$ e $j = 1, 2, \dots, r$; as quais indicam se o recurso j é ou não utilizado pelo processo i durante o intervalo k , e chamando de D o vetor composto por todos os D_k , tem-se, portanto, a representação de um *pipeline* na produção, onde cada D_k representa um dos seus estágios. Alternativamente, D pode ser entendido como um arranjo tridimensional que representa a sequência de etapas do planejamento.

$$D_k = \begin{bmatrix} d_{k,1,1} & d_{k,1,2} & d_{k,1,3} & \dots & d_{k,1,r} \\ d_{k,2,1} & d_{k,2,2} & d_{k,2,3} & \dots & d_{k,2,r} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots \\ d_{k,p,1} & d_{k,p,2} & d_{k,p,3} & \dots & d_{k,p,r} \end{bmatrix} \quad (20)$$

Um exemplo de planejamento é mostrado em (21), onde está representada a melhor solução representável pelo modelo proposto para o exemplo apresentado.

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (21)$$

A matriz D_k possui, portanto, posições com valor 0, indicando os recursos que não estão sendo utilizados nessa etapa. Os valores iguais a 1 indicam os recursos utilizados pelo processo no intervalo k . O tempo da etapa é dado pelo maior tempo de utilização presente na matriz.

$$T_{Dk} = \text{MAX}(d_{k,1,1}t_{1,1}, d_{k,1,2}t_{1,2}, d_{k,1,3}t_{1,3}, \dots, d_{k,1,r}t_{1,r}, d_{k,2,1}t_{2,1}, \dots, d_{k,p,r-1}t_{p,r-1}, d_{k,p,r}t_{p,r}) \quad (22)$$

A resolução do problema consiste em atribuir valores (0 ou 1) às variáveis de decisão, de forma a minimizar o tempo total do planejamento, o qual é dado pela soma do tempo das etapas. Esses valores atribuídos indicam a ordem em que cada etapa deve ser realizada, e quais recursos devem ser utilizados em paralelo (quando possível) para que o tempo da produção seja minimizado. Pode-se então formular a função objetivo para o modelo, a qual é representada pela Equação 23.

$$\text{minimizar} \quad T = \sum_{k=1}^{p+r-1} T_{E_k} \quad (23)$$

A solução ótima para o problema, conforme indicada na Figura 12, a princípio, não é possível de ser representada por este modelo, sendo a melhor resposta possível a indicada pela sequência da Eq. 19, cujos tempos calculados são apresentados, respectivamente, nas Equações 23 a 27.

$$T_{D1} = MAX(5,5) = 5 \quad (24)$$

$$T_{D2} = MAX(20,40) = 40 \quad (25)$$

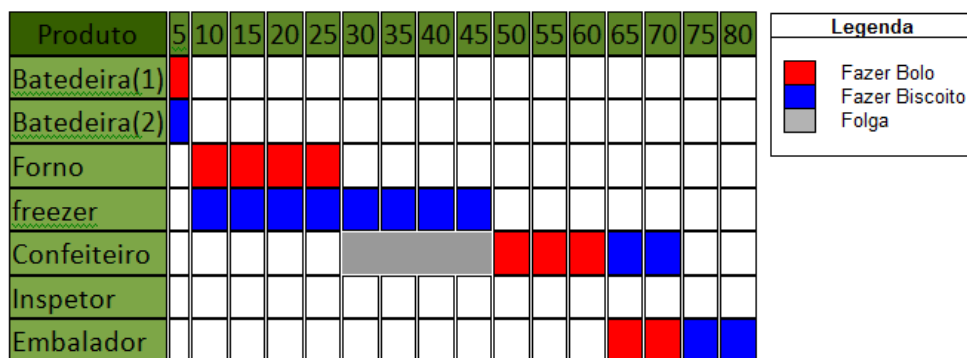
$$T_{D3} = 15 \quad (26)$$

$$T_{D4} = 10 \quad (27)$$

$$T_{D5} = MAX(10,10) = 10 \quad (28)$$

Como o tempo de d_2 é dado pelo tempo de utilização do maior recurso, que no caso é o “Freezer” (ou r_3), e o tempo do mesmo é muito maior que o do “Forno” (ou r_2), este fica com uma folga muito grande (20 minutos) dentro de D_2 , tempo este em que o recurso “Confeiteiro” (ou r_3) poderia ser utilizado, sendo este o motivo de a modelagem não permitir a representação da solução ótima. Porém, a resolução do modelo fornecerá a melhor sequência de utilização possível, de forma que um *software* pode ser desenvolvido para reorganizar os tempos, mantendo a sequência, de forma a diminuir as “folgas”, o que permitiria chegar à solução ótima, conforme mostrado na Figura 12.

Figura 16 - Solução representável pelo modelo (com folga)



Fonte: Autor

3.4.3 Restrições

A aplicação do modelo sobre função objetivo deve obedecer a restrições, as quais são abordadas nesta seção.

3.4.3.1 Disponibilidade de Recursos

Em uma instante (ou etapa) k , um mesmo recurso j não pode ser alocado para mais de um processo, a menos que haja quantidade suficiente do recurso. Portanto, a quantidade de processos para os quais a variável de decisão referente ao recurso j esteja igual a 1 deve ser menor ou igual à quantidade Q_j . A partir dessa situação, levanta-se a restrição representada pela Equação 29.

$$\sum_{i=1}^p d_{k,i,j} \leq Q_j, \quad \forall j, k \text{ tal que } j=1,2,\dots,r; \text{ e } k=1,2,\dots,p+r-1 \quad (29)$$

3.4.3.2 Simultaneidade de Utilização de Recursos

Em um mesmo instante k , o mesmo processo i não pode alocar mais de um recurso simultaneamente. Portanto, o somatório de todas as variáveis de decisão referentes a um processo i dentro de uma mesma janela de tempo não pode ser maior que 1.

$$\sum_{j=1}^r d_{k,i,j} \leq 1, \quad \forall i, k \text{ tal que } i=1,2,\dots,p; \text{ e } k=1,2,\dots,p+r-1 \quad (30)$$

3.4.3.3 Retenção do Recurso

Um mesmo processo não pode ficar com o mesmo recurso alocado em mais de uma etapa. Cada instante k representa um estágio de *pipeline* e, portanto, corresponde à execução de uma tarefa completa. O tempo em que o recurso j precisa ser utilizado pelo processo p deve ser alocado inteiramente dentro desse instante. Matematicamente, a soma de todas as variáveis de decisão referentes à alocação do recurso j pelo processo i não pode ser maior que 1.

$$\sum_{k=1}^{p+r-1} d_{k,i,j} \leq 1, \quad \forall i, j \text{ tal que } i=1,2,\dots,p; \text{ e } j=1,2,\dots,r \quad (31)$$

3.4.3.4 Ordem de Utilização dos recursos

Um recurso não pode ser alocado para um processo se todos os recursos com precedência sobre este não tiverem sido alocados ainda para esse processo. Portanto, na primeira etapa somente

podem ser alocados recursos com ordem de utilização 1.

$$d_{1,i,j}(O_{i,j}-1) = 0, \forall i, j \text{ tal que } i=1,2,\dots,p; j=1,2,\dots,r \quad (32)$$

Nas etapas seguintes à primeira, um recurso somente pode ser alocado se todos os recursos de ordem inferior tiverem sido alocados nas etapas anteriores.

$$\left(\sum_{j=1}^{j=r} d_{k,i,j}\right) \sum_{n=1}^{k-1} \left(\sum_{j=1}^r d_{n,i,j} O_{i,j}\right) = \sum_{m=0}^{(\sum_{j=1}^{j=r} d_{k,i,j} O_{i,j})-1} m, \forall i, j, k \text{ tal que } i=1,2,\dots,p; k=2,3,\dots,p+r-1 \quad (33)$$

3.4.3.5 Utilização de Todos os Recursos

A soma de todas as variáveis de decisão referentes a um processo i , multiplicadas pela respectiva ordem de utilização O_{ij} , deve ser igual à soma de todas as ordens de utilização. Esta restrição garante que todos os recursos necessários sejam utilizados por um processo durante o planejamento.

$$\sum_{k=1}^{p+r-1} \left(\sum_{j=1}^r d_{k,i,j} O_{i,j}\right) = \sum_{j=i}^r O_{i,j}, \forall i \text{ tal que } i=1,2,\dots,p \quad (34)$$

3.4.3.6 Tempo máximo para execução dos processos

Para garantir que todos os processos sejam finalizados dentro de seu prazo individual, adotou-se a seguinte solução, matematicamente: uma variável auxiliar $S_{k,i}$ indica, quando zero, que nenhum recurso mais é alocado na etapa k para o processo i , e nem nas etapas seguintes (quando $S_{n,i} = 0$, indica que todo o processo i foi finalizado na etapa $n-1$). Esta restrição informa que o valor de $S_{k,i}$ quando $k = p+r-1$ (última etapa) deve ser a soma de todos os $d_{k,i,j}$ (como apenas um recurso é alocado para um processo a cada etapa, $S_{k,n}$ será igual a 0 ou 1).

$$S_{(p+r-1),i} = \sum_{j=1}^r d_{(p+r-1),i,j}, \forall i \text{ tal que } i=1,2,\dots,p \quad (35)$$

Para todas as etapas anteriores à última, $S_{k,i}$ deve ser o maior valor entre a soma dos $d_{k,i,j}$ e o $S_{(k+1),i}$ ($S_{k,i}$ da etapa seguinte). Dessa forma, se uma etapa possui $S_{k,i} = 1$, garante-se que todas as anteriores (mesmo que não haja nenhum recurso alocado para o processo i na etapa) também será 1.

$$S_{k,i} = \text{MAX} \left(\sum_{j=1}^r d_{k,i,j}, S_{(k+1),i} \right), \quad \forall k, i \text{ tal que } k=1,2,\dots,(p+r-2); \quad i=1,2,\dots,p \quad (36)$$

Como a variável $S_{k,i}$ possui valor 0 apenas depois da finalização do processo i , pode-se restringir o tempo máximo de cada processo através da Eq. 35, pois $S_{k,i}$ multiplicada pelo tempo total de cada etapa será igual a zero em todas as etapas posteriores à finalização. A soma de todos os tempos de etapa multiplicados pelo $S_{k,i}$ da etapa fornecerá o tempo total de produção do processo i , o qual deve ser menor ou igual ao tempo limite L_i .

$$\sum_{k=1}^{p+r-1} S_{k,i} T_{E_k} \leq L_i; \quad \forall i \text{ tal que } i=1,2,\dots,p \quad (37)$$

3.4.3.7 Outras Restrições

Deve-se garantir que as variáveis de decisão sejam valores binários, portanto devem pertencer ao conjunto $\{0,1\}$, o que é feito através da Equação 38.

$$d_{k,i,j} \in \{0,1\}, \quad \forall k, i, j \text{ tal que } i=0,1,\dots,p+r-1; \quad i=0,1,\dots,p; \quad j=0,1,\dots,r \quad (38)$$

A ordem de utilização dos recursos devem ser números naturais, maiores ou iguais a zero.

$$0 \leq O_{i,j} \leq r, \quad \forall i, j \text{ tal que } i=1,2,\dots,p; \quad j=1,2,\dots,r; \quad O_{i,j} \in \mathbb{N} \quad (39)$$

3.4.4 Modelo Matemático

Utilizando a função objetivo e as restrições levantadas, obtém-se então o modelo de PNL para o *scheduling* da produção, apresentado na Equação 40.

$$\text{minimizar} \quad T = \sum_{k=1}^{p+r-1} T_{E_k}$$

sujeito a:

$$\begin{aligned} & \sum_{i=1}^p d_{k,i,j} \leq Q_j, \quad \forall j, k \text{ tal que } j=1,2,\dots,r; \text{ e } k=1,2,\dots,p+r-1 \\ & \sum_{j=1}^r d_{k,i,j} \leq 1, \quad \forall i, k \text{ tal que } i=1,2,\dots,p; \text{ e } k=1,2,\dots,p+r-1 \\ & \sum_{k=1}^{p+r-1} d_{k,i,j} \leq 1, \quad \forall i, j \text{ tal que } i=1,2,\dots,p; \text{ e } j=1,2,\dots,r \\ & d_{1,i,j}(O_{i,j}-1) = 0, \quad \forall i, j \text{ tal que } i=1,2,\dots,p; \text{ e } j=1,2,\dots,r \\ & \left(\sum_{j=1}^r d_{k,i,j} \right) \sum_{n=1}^{k-1} \left(\sum_{j=1}^r d_{n,i,j} O_{i,j} \right) = \sum_{m=0}^{(\sum_{j=1}^r d_{k,i,j} O_{i,j})-1} m, \quad \forall i, j, k \text{ tal que } i=1,2,\dots,p; k=2,3,\dots,p+r-1 \\ & \sum_{k=1}^{p+r-1} \left(\sum_{j=1}^r d_{k,i,j} O_{i,j} \right) = \sum_{j=i}^r O_{i,j}, \quad \forall i \text{ tal que } i=1,2,\dots,p \\ & S_{(p+r-1),i} = \sum_{j=1}^r d_{(p+r-1),i,j}, \quad \forall i \text{ tal que } i=1,2,\dots,p \\ & S_{k,i} = \text{MAX} \left(\sum_{j=1}^r d_{k,i,j} S_{(k+1),i} \right), \quad \forall k, i \text{ tal que } k=1,2,\dots,(p+r-2); i=1,2,\dots,p \\ & \sum_{k=1}^{p+r-1} S_{k,i} T_{E_k} \leq L_i; \quad \forall i \text{ tal que } i=1,2,\dots,p \\ & d_{k,i,j} \in \{0,1\}, \quad \forall k, i, j \text{ tal que } i=0,1,\dots,p+r-1; i=0,1,\dots,p; j=0,1,\dots,r \\ & 0 \leq O_{i,j} \leq r, \quad \forall i, j \text{ tal que } i=1,2,\dots,p; j=1,2,\dots,r; O_{i,j} \in \mathbb{N} \end{aligned} \tag{40}$$

3.5 Modelagem do Algoritmo Genético

Com o objetivo de validar o modelo proposto e também devido à complexidade envolvida na resolução matemática de problemas não lineares, foi desenvolvido um método heurístico, baseado em algoritmos genéticos, para a resolução do problema, com o objetivo de encontrar soluções que, mesmo não sendo a solução ótima, sejam boas soluções, isto é, atendam aos critérios exigidos. Nas seções a seguir, será apresentada a estrutura do algoritmo proposto, o qual implementa todas as etapas de um algoritmo genético apresentadas na seção 2.4.

3.5.1 Indivíduos

O algoritmo proposto consiste em um ambiente onde uma população de tamanho t , composta por indivíduos modelados de maneira a representar uma solução possível para o problema, passam

pelos processos evolutivos de *crossover* e mutação, com o objetivo de convergir para uma representação otimizada para o problema.

Os indivíduos foram modelados de maneira semelhante à apresentada no modelo de PNL, isto é, como uma sequência de etapas (ou estágios), de maneira analógica a um *pipeline*, porém com a diferença de cada etapa não ser mais representada como uma matriz binária, mas sim como um vetor de números decimais, o que reduz bastante o tamanho dos indivíduos e aumenta o desempenho do algoritmo. Portanto, o indivíduo deixa de ser representado como um vetor tridimensional de números binários, para assumir a forma de uma matriz de números decimais, chamada de I , a qual possui n linhas e k colunas, onde as linhas representam os processos a serem executados e as colunas representam as etapas.

Os recursos alocados em cada etapa são representados por números decimais (o número do recurso na matriz de ordenação). Portanto, para indicar que o recurso 2 está alocado para o processo 1 na terceira etapa da produção, coloca-se o número 2 na célula delimitada pela linha 1 e coluna 3.

$$I_i = \begin{bmatrix} I_{i,1,1} & I_{i,1,2} & I_{i,1,3} & \dots & I_{i,1,k} \\ I_{i,2,1} & I_{i,2,2} & I_{i,2,3} & \dots & I_{i,2,k} \\ \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \cdot & \dots & \cdot \\ I_{i,p,1} & I_{i,p,2} & I_{i,p,3} & \dots & I_{i,p,k} \end{bmatrix}, \text{ onde } k = p + r - 1 \quad (41)$$

Por exemplo, o planejamento apresentado na Equação 21 passa a ser representado conforme a Equação 42.

$$I_x = \begin{bmatrix} 1 & 2 & 4 & 6 & 0 & 0 & 0 \\ 1 & 3 & 0 & 4 & 6 & 0 & 0 \end{bmatrix} \quad (42)$$

A população do problema é, portanto, um conjunto de i indivíduos, sendo representada como um vetor de i posições, onde cada elemento corresponde a um indivíduo I_i ou um vetor tridimensional.

$$I = \left[\begin{array}{c} \left[\begin{array}{cccc} I_{1,1,1} & I_{1,1,2} & \dots & I_{1,1,k} \\ I_{1,2,1} & I_{1,2,2} & \dots & I_{1,2,k} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ I_{1,p,1} & I_{1,p,2} & \dots & I_{1,p,k} \end{array} \right] \left[\begin{array}{cccc} I_{2,1,1} & I_{2,1,2} & \dots & I_{2,1,k} \\ I_{2,2,1} & I_{2,2,2} & \dots & I_{2,2,k} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ I_{2,p,1} & I_{2,p,2} & \dots & I_{2,p,k} \end{array} \right] \dots \left[\begin{array}{cccc} I_{i,1,1} & I_{i,1,2} & \dots & I_{i,1,k} \\ I_{i,2,1} & I_{i,2,2} & \dots & I_{i,2,k} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ I_{i,p,1} & I_{i,p,2} & \dots & I_{i,p,k} \end{array} \right] \end{array} \right],$$

onde $k = p + r - 1$

(43)

A Equação 44 representa uma possível população para o ambiente apresentado na seção 3.1, onde o indivíduo I_2 é a única solução válida para o problema.

$$I = \left[\begin{array}{c} \left[\begin{array}{cccccc} 0 & 2 & 2 & 6 & 0 & 0 & 0 \\ 1 & 2 & 0 & 4 & 0 & 0 & 6 \\ 1 & 2 & 2 & 6 & 0 & 0 & 0 \\ 1 & 2 & 0 & 4 & 0 & 0 & 0 \end{array} \right] \left[\begin{array}{cccccc} 1 & 2 & 4 & 6 & 0 & 0 & 0 \\ 1 & 3 & 0 & 4 & 6 & 0 & 0 \end{array} \right] \left[\begin{array}{cccccc} 1 & 0 & 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 3 & 4 & 0 & 6 \\ 1 & 2 & 2 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right] \end{array} \right]$$

(44)

3.5.2 Inicialização

Para inicializar a população, uma heurística gulosa foi adotada para gerar uma quantidade alta de “boas” soluções já na inicialização do problema, com dois objetivos: O primeiro é o de facilitar a convergência, uma vez que há um grande número de “bons genes” para induzir o processo evolutivo, através dos operadores de *crossover*, a uma solução que combine as melhores característica; o segundo é o de analisar a convergência do algoritmo, verificando se as soluções já são geradas na inicialização (o que tornaria desnecessário o algoritmo genético, uma vez que uma heurística gulosa seria suficiente para resolvê-lo) ou se são geradas através das operações do algoritmo genético.

A heurística utilizada tenta inicializar os indivíduos com uma configuração baseada em *pipeline*, da seguinte maneira:

para cada linha p , faça:

$0 \rightarrow \text{pos};$

para cada coluna k , faça:

$O[\text{pos}] \rightarrow \text{recursoAtual};$

$\text{random}(0,1) \rightarrow \text{rand};$

se $\text{rand} = 0$ faça:

$0 \rightarrow I[p][k]$
 senão, faça:
 $\text{recursoAtual} \rightarrow I[p][k];$
 $\text{pos} + 1 \rightarrow \text{pos};$

Para cada processo, o algoritmo percorre as etapas sorteando se a etapa será utilizada ou não (se não, coloca valor zero na etapa) e, se sim, coloca o próximo valor do vetor de ordenação do processo em questão. Dessa forma, garante-se que nenhum indivíduo da população inicial utilize recursos fora de ordem, que um recurso seja alocado sem os anteriores já terem sido alocados e nem que um recurso seja alocado pelo mesmo processo por mais de uma etapa. Desta forma, grande parte da população inicial será de indivíduos válidos. Os indivíduos inválidos possíveis de serem gerados são aqueles que não utilizem todos os recursos, que aloquem recursos com quantidade insuficiente ou que não atendam ao tempo limite determinado para algum processo.

3.5.3 Avaliação

A avaliação dos indivíduos se dá, primeiramente, através do cálculo do tempo gasto para a execução do *schedule* que ele representa, somando-se os tempos de cada etapa, os quais são calculados conforme a Equação 22.

Em seguida, a validade do indivíduo é testada, convertendo-o para a representação binária e submetendo-o a cada equação matemática que compõem as restrições do modelo de PNL da Equação 40. Os indivíduos inválidos não são descartados, a fim de manter a diversidade da população, mas sofrem uma penalidade que consiste em somar a ele um valor predefinido x , multiplicado pela quantidade de restrições violadas.

Após a verificação da validade da solução, verifica-se, para cada indivíduo, se há outros indivíduos idênticos a ele na população. Caso haja, esses indivíduos sofrem uma penalidade, que consiste em dobrar a sua pontuação atual. Esta medida é tomada com o objetivo de diminuir a probabilidade de cruzamento entre indivíduos idênticos, uma vez que isso faria com que a quantidade de “cópias” do indivíduo aumentasse, podendo levar o algoritmo à estagnação.

Por último, verifica-se a existência de indivíduos diferentes, mas com pontuações iguais. Caso existam, um deles recebe um acréscimo de 1 em sua pontuação, com a finalidade de torná-los diferentes, facilitando os processos de seleção.

Ao fim desse processo, a pontuação acumulada por cada indivíduo se torna o seu *fitness*.

3.5.4 Seleção

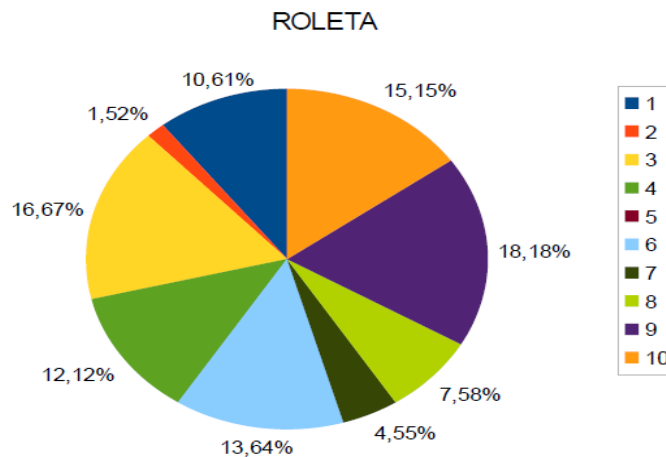
O método de seleção utilizado é o método de seleção por roleta, descrito na seção 2.4.3. Como se trata de um problema de minimização, ao calcular as “fatias” da roleta, cada indivíduo recebe um percentual proporcional à sua pontuação subtraída da pontuação do pior indivíduo, o que elimina todas as probabilidades de o pior indivíduo ser selecionado.

Tabela 3 - Indivíduos da População e o cálculo pontuação

| Indivíduo | Pontuação | Parcela da Roleta (Pior-Pontuação) | Percentual |
|-----------|-----------|------------------------------------|------------|
| 1 | 100 | 35 | 10,61% |
| 2 | 130 | 5 | 1,52% |
| 3 | 80 | 55 | 16,67% |
| 4 | 95 | 40 | 12,12% |
| 5 | 135 | 0 | 0,00% |
| 6 | 90 | 45 | 13,64% |
| 7 | 120 | 15 | 4,55% |
| 8 | 110 | 25 | 7,58% |
| 9 | 75 | 60 | 18,18% |
| 10 | 85 | 50 | 15,15% |
| Pior: | 135 | | |
| Total: | | 330 | 100,00% |

Fonte: Autor

Figura 17 - Roleta para os Indivíduos da Tabela 3



Fonte: Autor

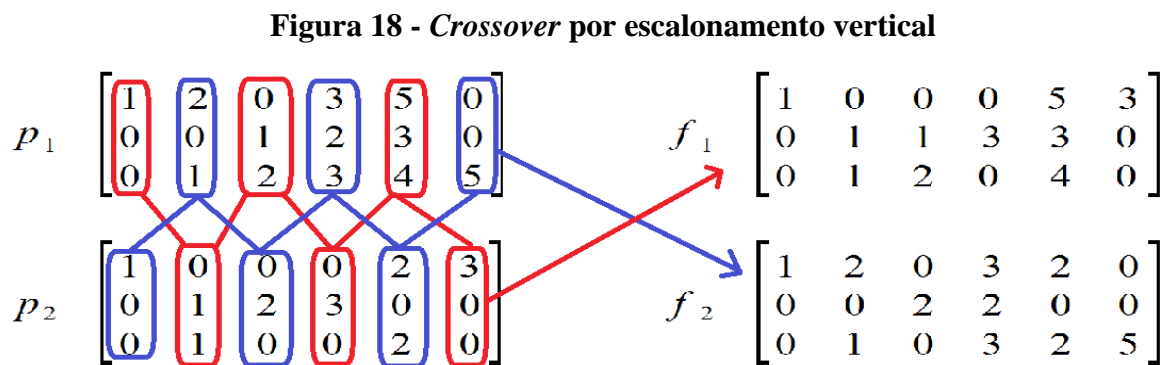
3.5.5 Cruzamento

Para o algoritmo, três diferentes operadores de cruzamento foram definidos, sendo que, em cada geração, um desses processos é sorteado para ser aplicado. Cada um dos três tipos de *crossover* possui igual probabilidade de ser selecionado. O objetivo de realizar diferentes tipos de

crossover é permitir diferentes formas de recombinar os cromossomos do indivíduo, aumentando as possibilidades de solução.

3.5.5.1 Escalonamento Vertical

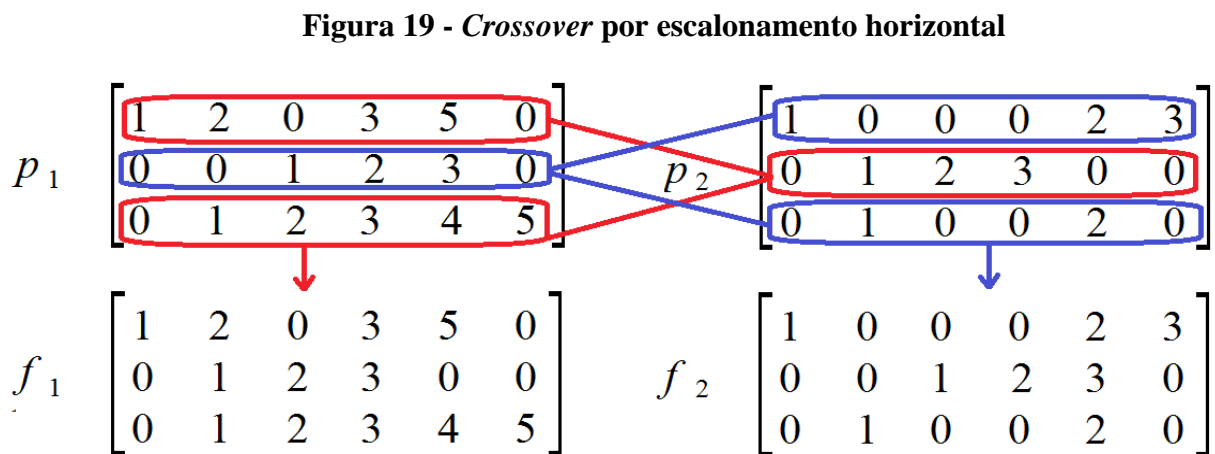
Esta modalidade de *crossover* realiza um cruzamento multiponto, escalonando as colunas dos pais para formar os filhos. A Figura 18 mostra o cruzamento de dois indivíduos, p_1 e p_2 , para formar os filhos f_1 e f_2 , utilizando esta modalidade de *crossover*.



Fonte: Autor

3.5.5.2 Escalonamento Horizontal

Esta modalidade de *crossover* realiza um cruzamento multiponto, escalonando as linhas dos pais para formar os filhos. A Figura 19 mostra o cruzamento de dois indivíduos, p_1 e p_2 , para formar os filhos f_1 e f_2 , utilizando esta modalidade de *crossover*.

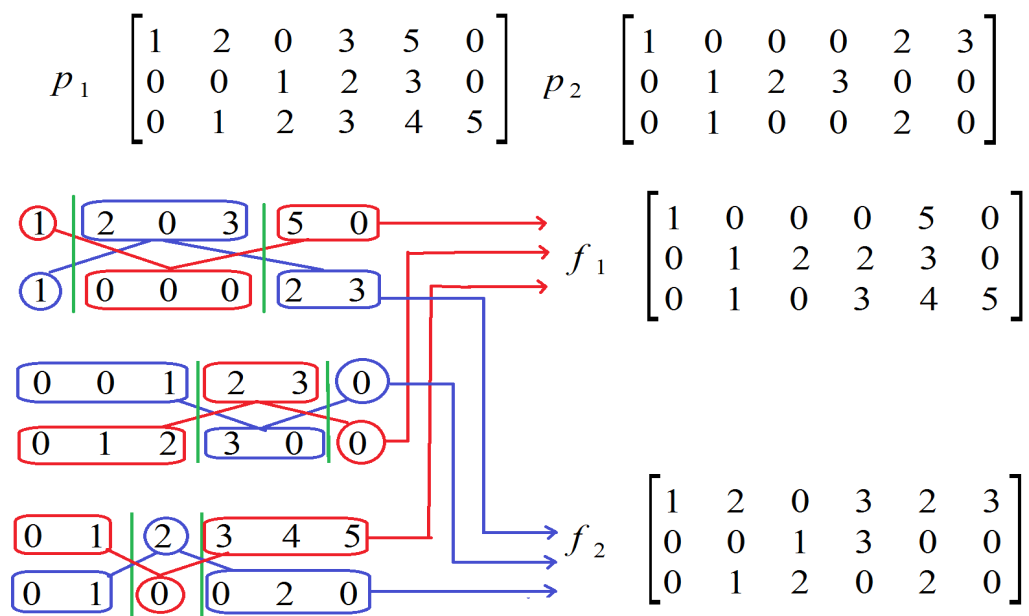


Fonte: Autor

3.5.5.3 Crossover de 3 pontos por linha

Nesta modalidade, a cada linha dos pais, é aplicado um *crossover* de três pontos, como se este fosse um vetor unidimensional, formando uma nova linha para os filhos. Os três pontos do cruzamento são sorteados a cada linha. Para aumentar as possibilidades de indivíduos formados, também é sorteado, a cada linha, de qual indivíduo pai virá o primeiro gene do filho. A Figura 20 apresenta um exemplo deste tipo de cruzamento.

Figura 20 - Crossover de três pontos por linha



Fonte: Autor

3.5.6 Mutação

Para o algoritmo, dois diferentes operadores de cruzamento foram definidos: *swap* e *flip*, sendo que, em cada geração, com uma probabilidade predefinida, um desses processos é sorteado para ser aplicado. Cada um dos dois tipos de mutação possui igual probabilidade de ser selecionado.

3.5.6.1 Mutação swap

Nesta modalidade de mutação, duas posições do indivíduo são sorteadas e o valor das duas é trocado.

Figura 21 - Mutação *swap*

$$\begin{bmatrix} 1 & 2 & 0 & 3 & 5 & 0 \\ 0 & 0 & 1 & 2 & 3 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 2 & 0 & 3 & 0 & 0 \\ 0 & 5 & 1 & 2 & 3 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

Fonte: Autor

3.5.6.2 Mutação *flip*

Nesta modalidade de mutação, uma posição do indivíduo é sorteada e tem o seu valor trocado por outro valor aleatório, dentre os valores de recurso existentes.

Figura 22 - Mutação *flip*

$$\begin{bmatrix} 1 & 2 & 0 & 3 & 0 & 0 \\ 0 & 5 & 1 & 2 & 3 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 2 & 0 & 3 & 0 & 0 \\ 0 & 5 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

Fonte: Autor

3.5.7 Atualização e finalização

Após as operações de *crossover* e mutação, os novos indivíduos são introduzidos na população, substituindo os indivíduos com pior fitness (maior pontuação). A cada geração, o indivíduo com melhor *fitness*, é selecionado como a solução parcial para o problema. Após uma quantidade predefinida de gerações, a solução parcial da última iteração será a solução do problema.

3.6 Implementação

Com a finalidade de validar o modelo de PNL proposto na seção 3.4, o algoritmo genético apresentado na seção 3.5. foi utilizado para implementar um *software*, utilizando linguagem C++ e a ferramenta Dev C++, versão 4.9.9.2.

Para geração de números aleatórios, foi utilizado o método *GARandomInt* da biblioteca GALib, versão 2.4.7, que é uma biblioteca C++, desenvolvida pelo MIT (*Massachusetts Institute of*

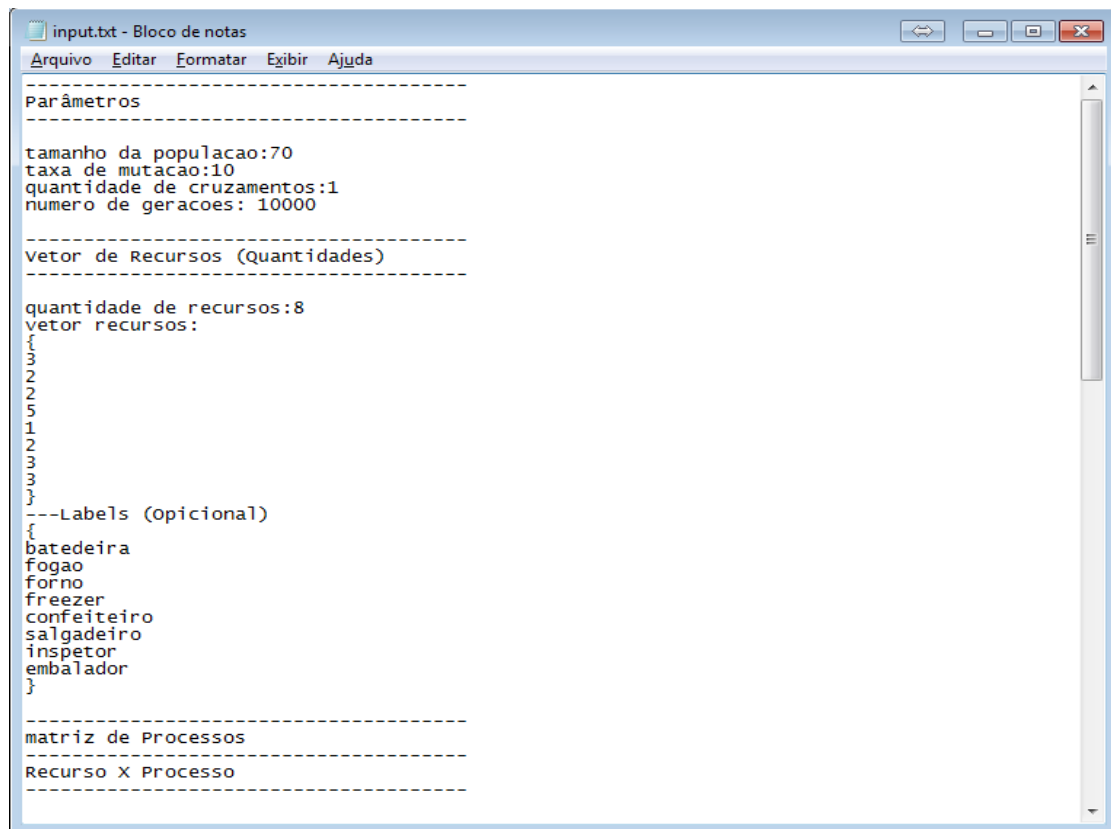
Technology), voltada ao desenvolvimento de algoritmos evolucionários.

Nesta seção é apresentado o funcionamento do *software* desenvolvido, assim como os parâmetros de entrada fornecidos para o mesmo.

3.6.1 Parâmetros de Entrada

O *software* implementado encontra-se disponível através do arquivo executável “*scheduling.exe*”, o qual verifica, ao ser executado, o conteúdo do arquivo “*input.txt*” para inicialização das variáveis. É nesse arquivo que são fornecidos os parâmetros necessários à execução do algoritmo.

Figura 23 - Visualização do arquivo “*input.txt*”



```

input.txt - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda
-----
Parâmetros
-----
tamanho da populacao:70
taxa de mutacao:10
quantidade de cruzamentos:1
numero de geracoes: 10000
-----
Vetor de Recursos (Quantidades)
-----
quantidade de recursos:8
vetor recursos:
{
3
2
2
5
1
2
3
3
}
---Labels (opcional)
{
batedeira
fogao
forno
freezer
confeiteiro
salgadeiro
inspetor
embalador
}
-----
matriz de Processos
-----
Recurso X Processo
-----

```

Fonte: Autor

Os parâmetros devem ser informados utilizando a sintaxe exatamente igual à apresentada nas seções a seguir, sempre em caixa baixa, e na mesma ordem. Qualquer conteúdo diferente do esperado é descartado.

A sintaxe genérica é o nome do parâmetro, em caixa baixa, seguido de dois pontos e o conteúdo associado ao parâmetro.

3.6.1.1 Tamanho da população

Neste parâmetro é informada a quantidade de indivíduos existentes na população. A sintaxe do parâmetro no arquivo é “tamanho da populacao:” e o conteúdo deve ser um número inteiro, representando o número de indivíduos desejados. Por exemplo, para uma população de 70 indivíduos, deve-se utilizar a sintaxe: “tamanho da populacao:70”. Se este parâmetro não for informado, o algoritmo assume o valor padrão de 50 indivíduos.

3.6.1.2 Taxa de mutação

Neste parâmetro é informado a taxa (probabilidade) de mutação desejada para a execução do algoritmo. A sintaxe do parâmetro no arquivo é “taxa de mutacao:” e o conteúdo deve ser um número inteiro, representando o percentual desejado para a taxa de mutação. Por exemplo, para uma taxa de 10%, deve-se utilizar a sintaxe: “taxa de mutacao:10”. Se este parâmetro não for informado, o algoritmo assume a taxa de mutação padrão de 3%.

3.6.1.3 Quantidade de cruzamentos

Neste parâmetro é informado a quantidade de cruzamentos a ser realizada a cada geração do algoritmo. A sintaxe do parâmetro no arquivo é “quantidade de cruzamentos:” e o conteúdo deve ser um número inteiro, representando a quantidade de cruzamentos desejada. Este valor não pode superar metade do tamanho da população. Por exemplo, para 2 cruzamentos por geração, deve-se utilizar a sintaxe: “quantidade de cruzamentos:2”. Se este parâmetro não for informado, será utilizada a quantidade padrão de um cruzamento por geração.

3.6.1.4 Número de gerações

Neste parâmetro é informado a quantidade de gerações do algoritmo. A sintaxe do parâmetro no arquivo é “quantidade de geracoes:” e o conteúdo deve ser um número inteiro, representando a quantidade de gerações desejada. Por exemplo, para 2000 gerações, deve-se utilizar a sintaxe: “numero de geracoes:2000”. Se este parâmetro não for informado, o algoritmo assume a quantidade padrão de gerações, que é 10000.

3.6.1.5 Quantidade de recursos

Neste parâmetro é informado a quantidade de recursos existentes a ser considerada pelo algoritmo. A sintaxe do parâmetro no arquivo é “quantidade de recursos:” e o conteúdo deve ser um número inteiro, representando a quantidade de recursos existentes. Por exemplo, para 8 recursos, deve-se utilizar a sintaxe: “quantidade de recursos:8”. Se este parâmetro não for informado, o algoritmo assume a quantidade padrão de 10 recursos.

3.6.1.6 Vetor recursos

Neste parâmetro é informado o vetor Q , contendo as quantidades disponíveis de cada recurso existente e os nomes (ou rótulos) dos recursos. A sintaxe deste parâmetro é “vetor recursos:” seguido de um vetor, entre chaves, onde cada elemento é um número inteiro representando a quantidade disponível do recurso. A quantidade de elementos no vetor deve ser igual à informada no parâmetro “quantidade de recursos”, e os valores devem ser separados por quebra de linha (enter). A informação do vetor de recursos é obrigatória.

Após o vetor de recursos, devem ser informados os rótulos dos mesmos, isto é, nomes para os recursos, devendo estes ser informados também entre chaves, separados por quebra de linha, e em quantidade igual à quantidade de elementos do vetor de recursos. A informação dos rótulos é opcional, mas mesmo se não forem informados, as chaves deve ser abertas e fechadas. Se os rótulos não forem informados, o algoritmo assumirá o rótulo padrão de “recurso i ”, onde i é o índice do recurso no vetor. Para o exemplo da seção 3.1, a declaração do vetor de recursos é:

vetor recursos:

```
{
2
1
1
1
1
1
1
}
```

```
{
```

Batedeira

```

Forno
Freezer
Confeiteiro
Inspetor
Embalador
}

```

3.6.1.7 Quantidade de Processos

Neste parâmetro é informado a quantidade de processos a serem escalonados pelo algoritmo. A sintaxe do parâmetro no arquivo é “quantidade de processos:” e o conteúdo deve ser um número inteiro, representando a quantidade de processos existentes. Por exemplo, para 4 processos, deve-se utilizar a sintaxe: “quantidade de processos:4”. Se este parâmetro não for informado, o algoritmo assume a quantidade padrão de 10 processos.

3.6.1.8 Matriz de Processos

Neste parâmetro é informada a matriz tempo de utilização (T) do recurso pelo processo, onde cada célula é um número inteiro representando o tempo em que o recurso (coluna) é alocado para o processo (linha). A quantidade de colunas da matriz deve ser igual à quantidade informada no parâmetro “quantidade de recursos:” e a quantidade de linhas deve ser igual à informada no parâmetro “quantidade de processos:”. A informação da matriz de processos é obrigatória, e sua sintaxe é: “matriz processos:” seguido da matriz representada entre chaves. As colunas da matriz devem ser separadas por tabulação ou espaço, e as linhas devem ser separadas por quebra de linha (enter).

Após a matriz, devem ser informados os rótulos dos processos, devendo estes ser informados também entre chaves, separados por quebra de linha, e em quantidade igual à quantidade de linhas da matriz de processos. A informação dos rótulos é opcional, mas mesmo se não forem informados, as chaves deve ser abertas e fechadas. Se os rótulos não forem informados, o algoritmo assumirá o rótulo padrão de “processo i ”, onde i é o índice do processo no vetor. Para o exemplo da seção 3.1, a declaração da matriz de processos é

```

matriz processos:
{

```

```

5      20      0      15      0      10
5      0       40      10      0      10
}
{
Fazer bolo
Fazer torta
}

```

3.6.1.9 Prazo de Entrega

Neste parâmetro é informado o vetor L , contendo os limites de tempo de cada processo. Cada elemento é um número inteiro representando o tempo máximo que pode ser gasto para a realização de todas as etapas dos processos em questão. A quantidade de elementos do vetor deve ser igual à quantidade informada no parâmetro “quantidade de processos:”. A informação deste parâmetro é obrigatória, e segue a seguinte sintaxe: “prazo de entrega:”, seguido do vetor L informado entre chaves, com uma quebra de linha entre cada elemento. Para o exemplo da seção 3.1, a declaração do vetor de prazo de entrega é

```

prazo de entrega:
{
500
500
}

```

3.6.1.10 Ordem de utilização

Neste parâmetro é informada a matriz O , contendo a ordem de utilização de cada recurso por processo. Cada elemento é um número inteiro, e os recursos não utilizados pelo processo devem possuir valor zero. A informação deste parâmetro é obrigatória, devendo a quantidade de colunas ser igual à quantidade de recursos, e a quantidade de linhas igual à quantidade de processo. A sintaxe deste parâmetro é “ordem de utilizacao:”, seguida da matriz de ordenação informada entre chaves, com tabulações ou espaços separando as colunas da matriz, e quebras de linha separando as linhas. Para o exemplo da seção 3.1, tem-se:

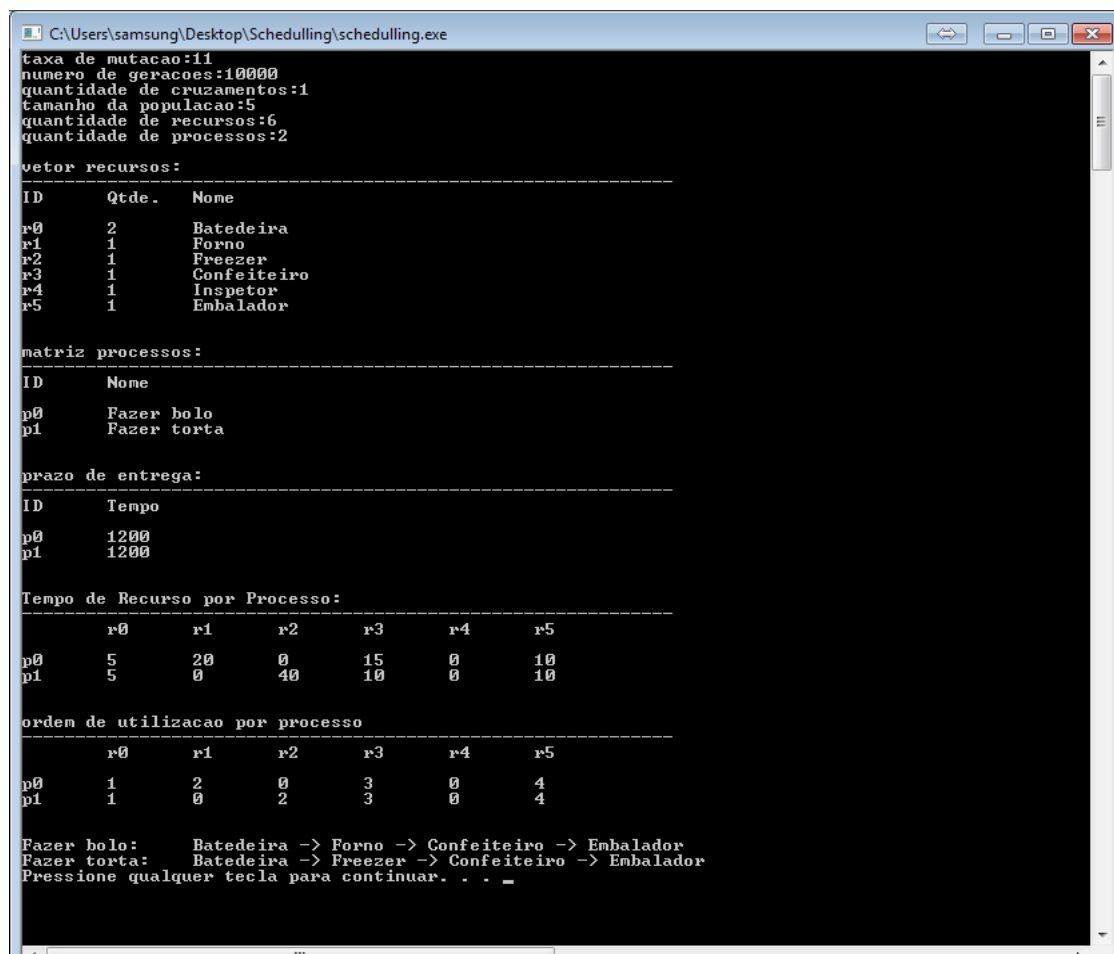
ordem de utilizacao:

```
{
1      2      0      3      0      4
1      0      2      3      0      4
}
```

3.6.2 Execução do Software

Ao executar o *software*, sua primeira ação é acessar o arquivo *input.txt* para obter os parâmetros do algoritmo genético e os dados dos processos e recursos a serem escalonados. Em seguida, esses dados são exibidos, conforme a Figura 24.

Figura 24 - Início da execução do *schedulling.exe*



```

C:\Users\samsung\Desktop\Schedulling\schedulling.exe
taxa de mutacao:11
numero de geracoes:10000
quantidade de cruzamentos:1
tamanho da populacao:5
quantidade de recursos:6
quantidade de processos:2

vetor recursos:
-----
ID      Qtde.  Nome
r0      2      Batedeira
r1      1      Forno
r2      1      Freezer
r3      1      Confeiteiro
r4      1      Inspetor
r5      1      Embalador

matriz processos:
-----
ID      Nome
p0      Fazer bolo
p1      Fazer torta

prazo de entrega:
-----
ID      Tempo
p0      1200
p1      1200

Tempo de Recurso por Processo:
-----
          r0      r1      r2      r3      r4      r5
p0      5        20      0       15      0       10
p1      5         0       40      10      0       10

ordem de utilizacao por processo
-----
          r0      r1      r2      r3      r4      r5
p0      1         2       0       3       0       4
p1      1         0       2       3       0       4

Fazer bolo:      Batedeira -> Forno -> Confeiteiro -> Embalador
Fazer torta:     Batedeira -> Freezer -> Confeiteiro -> Embalador
Pressione qualquer tecla para continuar. . . _
  
```

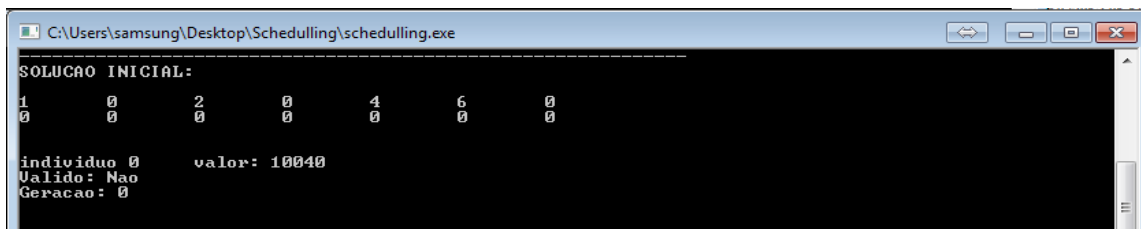
Fonte: Autor

Após pressionar qualquer tecla, o algoritmo genético irá iniciar sua execução, passando por todos os passos do processo evolutivo.

Ao gerar a população inicial, utilizando a heurística apresentada na seção 3.5.2, será exibido o melhor indivíduo (com menor pontuação) dessa população inicial (solução inicial), e ele passa a ser considerado a *solução parcial*. O indivíduo é exibido como uma matriz decimal de p linhas por $(p+r-1)$ colunas, onde p é o número de processos a serem escalonados e r é o número de recursos existentes. Além do indivíduo, são exibidas as seguintes informações sobre o mesmo:

- a) **número do indivíduo:** Utilizado para identificar o indivíduo da população, que são numerados de 0 a $t-1$, onde t é o tamanho da população, definido no arquivo de inicialização;
- b) **validade do indivíduo:** Indica, quando igual a “Sim”, que o indivíduo representa uma solução válida para o problema; ou o contrário, quando igual a “Não”;
- c) **geração:** Número da geração em que o indivíduo foi encontrado. Caso o indivíduo seja encontrado na população inicial, a geração considerada será zero;
- d) **pontuação:** valor de *fitness* do indivíduo.

Figura 25 - Solução inicial do algoritmo



Fonte: Autor

Em seguida, o algoritmo irá executar n iterações, onde n é o número de gerações definidas no arquivo de inicialização. A cada geração, após realizar os processos de *crossover* e *mutação*, os indivíduos da população atual são avaliados e, caso o melhor indivíduo tenha uma pontuação menor do que a solução parcial, ele passa a ser a nova solução parcial e é exibido na tela de execução do algoritmo.

Figura 26 - Soluções parciais do algoritmo

```

C:\Users\samsung\Desktop\Scheduling\scheduling.exe

SOLUCAO INICIAL:
1      0      2      0      4      6      0
0      0      0      0      0      0      0

indivíduo 0      valor: 10040
Valido: Nao
Geracao: 0

-----

SOLUCAO PARCIAL:
0      0      0      0      4      6      0
1      3      4      6      0      2      0

indivíduo 0      valor: 6084
Valido: Nao
Geracao: 4

-----

SOLUCAO PARCIAL:
0      1      2      0      4      6      0
1      3      4      6      0      0      0

indivíduo 0      valor: 100
Valido: Sim
Geracao: 5
  
```

Fonte: Autor

Na iteração número n , o algoritmo irá apresentar a solução final. Se o melhor indivíduo não for encontrado nessa iteração, a solução parcial anterior será apresentada como solução final.

Figura 27 - Solução final do algoritmo

```

C:\Users\samsung\Desktop\Scheduling\scheduling.exe

SOLUCAO PARCIAL:
0      1      2      0      4      6      0
1      3      4      6      0      0      0

indivíduo 0      valor: 100
Valido: Sim
Geracao: 5

-----

SOLUCAO PARCIAL:
0      1      2      0      4      6      0
1      3      4      0      6      0      0

indivíduo 4      valor: 90
Valido: Sim
Geracao: 86

-----

SOLUCAO FINAL:
0      1      2      0      4      6      0
1      3      4      0      6      0      0

indivíduo 4      valor: 90
Valido: Sim
Geracao: 10000

Pressione qualquer tecla para continuar. . .
  
```

Fonte: Autor

4 TESTES E RESULTADOS

Nesta seção, são apresentados e analisados os testes realizados e seus resultados. Os objetivos desses testes são de validar o modelo matemático e do algoritmo propostos, e comparar a solução por esses métodos com a solução por heurística gulosa, verificando a eficácia da heurística e do algoritmo, e o comportamento do mesmo, tanto no resultado quanto em consumo de recursos computacionais, em função de alteração das variáveis.

4.1 Cenários de Testes

Três cenários de testes foram elaborados para podermos avaliar o desempenho do algoritmo, em função das variáveis de entrada. Cada cenário de testes representa um problema de escalonamento de um tamanho diferente. Para cada cenário, o algoritmo é executado dez vezes para cada variação dos parâmetros de entrada, e os resultados são tabelados. A partir desses dados, são elaborados gráficos que permitam analisar fatores como a convergência do algoritmo, e eficácia da heurística de inicialização da população.

Os testes serão realizados em função da variação do tamanho da população: para cada cenário de testes, o algoritmo é executado 10 vezes utilizando cada um dos seguintes tamanhos de população: 5, 10, 50, 100 e 200 indivíduos. Para a realização dos testes quanto, as seguintes configurações foram fixadas:

- a) Taxa de Mutação: 11%;
- b) Número de gerações: 10000;
- c) Quantidade de *Crossover*: O valor mais próximo de 25% do tamanho da população.

4.1.1 *Cenário de testes 1*

O primeiro cenário de testes a ser analisado é o problema de escalonamento apresentado como exemplo na seção 3.1. Trata-se de um problema de tamanho pequeno, com dois processos por 6 recursos. Os dados dos processos e recursos para este problema são apresentados, respectivamente, nos Quadros 3 e 4.

No Quadro 6 são apresentados os resultados dos testes para uma população de 5 indivíduos. Para cada teste, é apresentada a sequência de soluções parciais retornada pelo algoritmo, desde a solução inicial até a solução final. A quantidade de crossover por cruzamento para este teste é 1. Os

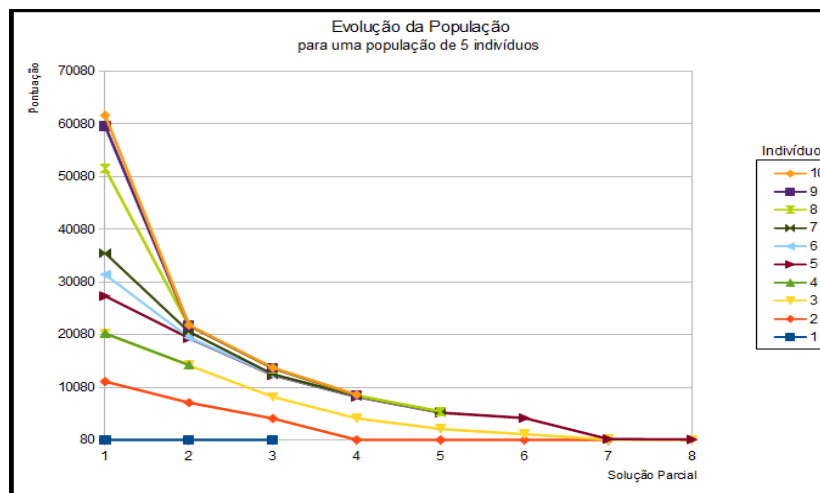
resultados podem ser observados graficamente pela Figura 28, para que possa ser observada a convergência do algoritmo para uma população de 5 indivíduos. A Figura 29 apresenta a curva média, obtida através da média dos resultados das colunas da Tabela 4.

Tabela 4 - Sequência de resultados para uma população de tamanho 5

| Teste | Evolução | | | | | | | Qtde Evolucoes | Resultado |
|-------|----------|------|------|------|------|------|----|----------------|-----------|
| 1 | 95 | 85 | 80 | | | | | 3 | 80 |
| 2 | 11069 | 7073 | 4081 | 95 | 90 | 85 | 80 | 7 | 80 |
| 3 | 9056 | 7068 | 4096 | 4081 | 2068 | 1079 | 90 | 80 | 80 |
| 4 | 95 | 80 | | | | | | 2 | 80 |
| 5 | 7078 | 5100 | 4091 | 4086 | 3092 | 3082 | 85 | 80 | 80 |
| 6 | 4076 | 110 | 95 | 85 | 80 | | | 5 | 80 |
| 7 | 4071 | 1099 | 100 | 90 | 80 | | | 5 | 80 |
| 8 | 16014 | 1089 | 1079 | 85 | 80 | | | 5 | 80 |
| 9 | 8082 | 100 | 85 | 80 | | | | 4 | 80 |
| 10 | 2068 | 90 | 85 | 80 | | | | 4 | 80 |

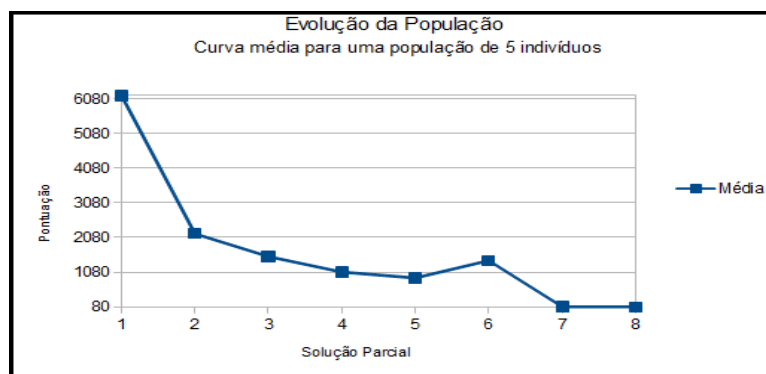
Fonte: Autor

Figura 28 - Curva de evolução de uma população com 5 indivíduos



Fonte: Autor

Figura 29 - Curva média de evolução de uma população de 5 indivíduos



Fonte: Autor

Conforme a seção 3.1, sabe-se que a solução ótima para o problema apresenta pontuação 80.

Analisando os resultados obtidos, percebe-se que para o problema em questão, o algoritmo apresenta um bom desempenho, uma vez que a solução ótima foi encontrada em todos os testes.

Aumentando o tamanho da população para 10 indivíduos, obtêm-se o resultado apresentado na Tabela 5. A quantidade de crossover por cruzamento para este teste é 3.

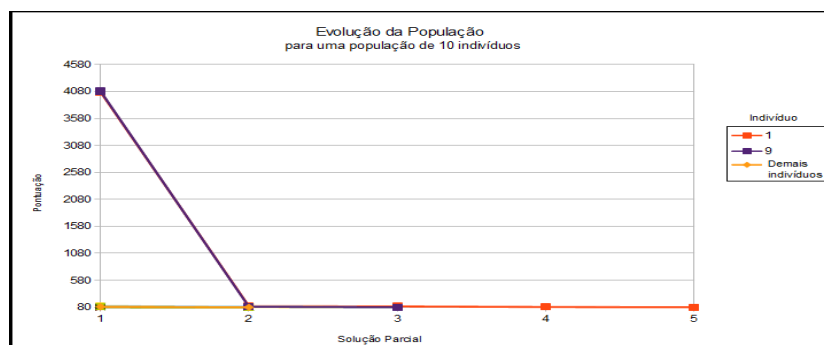
As Figuras 30 e 31 apresentam, respectivamente, a curva com os resultados da Tabela 5 e a curva média, obtida através da média dos resultados.

Tabela 5 - Sequência de resultados para uma população de tamanho 10

| Teste | Evolução | | | | | Qtde Evolucoe | Resultado |
|-------|----------|-----|----|----|----|---------------|-----------|
| 1 | 4071 | 100 | 95 | 85 | 80 | 5 | 80 |
| 2 | 85 | 80 | | | | 2 | 80 |
| 3 | 100 | 85 | 80 | | | 3 | 80 |
| 4 | 85 | 80 | | | | 2 | 80 |
| 5 | 80 | | | | | 1 | 80 |
| 6 | 100 | 90 | 80 | | | 3 | 80 |
| 7 | 85 | 80 | | | | 2 | 80 |
| 8 | 85 | 80 | | | | 2 | 80 |
| 9 | 4086 | 90 | 80 | | | 3 | 80 |
| 10 | 90 | 80 | | | | 2 | 80 |

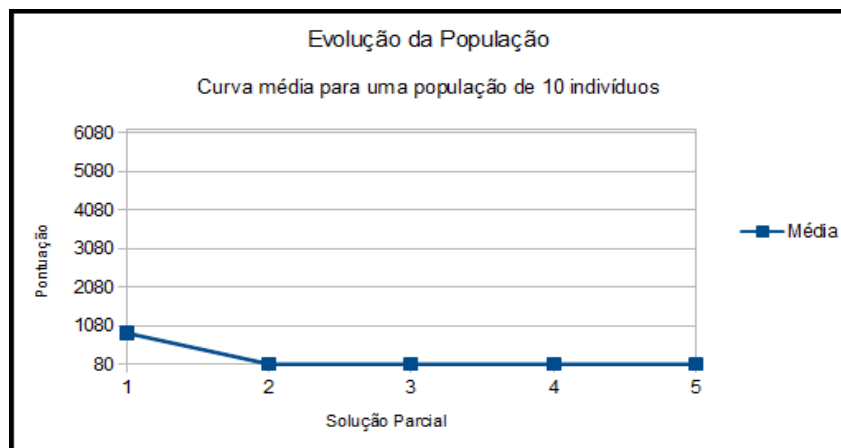
Fonte: Autor

Figura 30 - Curva de evolução de uma população com 5 indivíduos



Fonte: Autor

Figura 31 - Curva média de evolução de uma população com 5 indivíduos



Fonte: Autor

Analizando os gráficos gerados, pode-se perceber que, ao aumentar o tamanho da população, a curva média de evolução teve o tempo de “decaimento” reduzido. Isto se dá pelo fato de que, a aplicação da heurística a uma população maior permitiu a geração de um maior número de “bons” indivíduos logo na inicialização, diminuindo o ponto inicial da curva. Como pode ser analisado através da Tabela 5, houve uma grande quantidade de indivíduos na população inicial com baixa pontuação (entre 80, que é a solução ótima, e 100). Dessa forma, o aumento da população aumentou a velocidade de convergência do algoritmo.

Na Tabela 6 são apresentados os resultados dos testes para uma população de 50 indivíduos e uma taxa de 13 cruzamentos por geração.

Tabela 6 - Sequência de resultados para uma população de tamanho 50

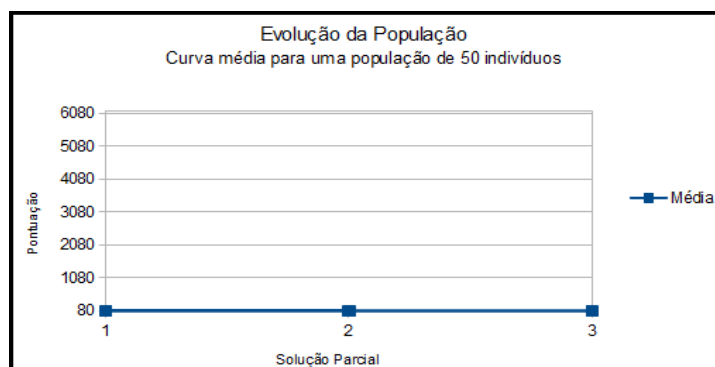
| Teste | Evolução | | | Qtde Evoluções | Resultado |
|-------|----------|----|----|----------------|-----------|
| 1 | 80 | | | 1 | 80 |
| 2 | 85 | 80 | | 2 | 80 |
| 3 | 80 | | | 1 | 80 |
| 4 | 80 | | | 1 | 80 |
| 5 | 90 | 85 | 80 | 3 | 80 |
| 6 | 80 | | | 1 | 80 |
| 7 | 85 | 80 | | 2 | 80 |
| 8 | 80 | | | 1 | 80 |
| 9 | 80 | | | 1 | 80 |
| 10 | 80 | | | 1 | 80 |

Fonte: Autor

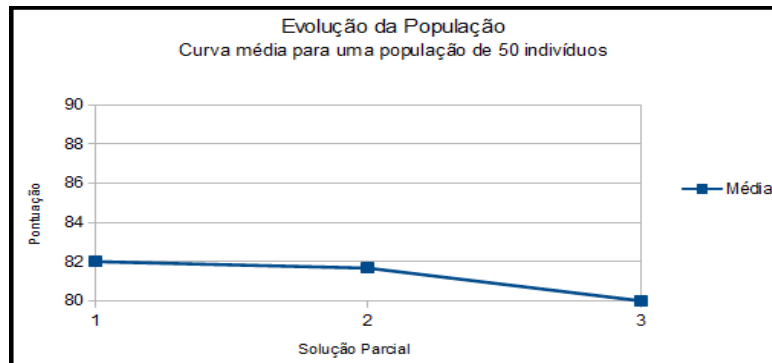
Como pode ser observado pela Tabela 6, em grande parte dos testes foi encontrada a solução ótima do problema logo na inicialização da população, isto é, com a aplicação do método heurístico. Nas demais execuções, soluções muito próximas da solução ideal foram encontradas.

Como todos os testes para esta configuração do problema resultaram em valores muito parecidos, estes não serão exibidos em um gráfico com os resultados individuais de cada teste, pois ocorreria sobreposição das curvas. Nas Figuras 32 e 33 é apresentada a curva média de evolução.

Figura 32 - Curva média de evolução para uma população de tamanho 50



Fonte: Autor

Figura 33 - Curva média de evolução ampliada, para uma população de tamanho 50**Fonte: Autor**

Analisando o gráfico da Figura 32, pode-se verificar um aumento ainda maior no decaimento da curva média de evolução, em relação ao teste com 10 indivíduos. Na Figura 33, foi apresentada uma versão ampliada do gráfico, onde podemos perceber que a curva não é uma reta, como parece na Figura 32.

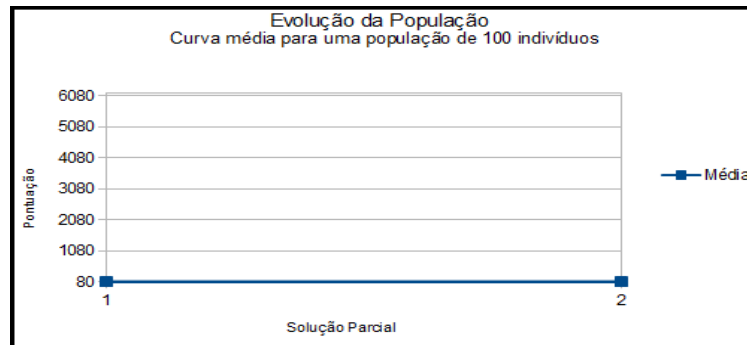
Os resultados dos testes para uma população de 100 indivíduos e taxa de crossover por cruzamento igual a 25, são apresentados a seguir, na Tabela 7 e na Figura 34.

Tabela 7 - Sequência de resultados para uma população de tamanho 100

| Teste | Evolução | Qtde Evolucoes | Resultado |
|-------|----------|----------------|-----------|
| 1 | 80 | 1 | 80 |
| 2 | 80 | 1 | 80 |
| 3 | 80 | 1 | 80 |
| 4 | 80 | 1 | 80 |
| 5 | 80 | 1 | 80 |
| 6 | 80 | 1 | 80 |
| 7 | 80 | 1 | 80 |
| 8 | 80 | 1 | 80 |
| 9 | 80 | 1 | 80 |
| 10 | 85 | 2 | 80 |

Fonte: Autor

Comparando a Tabela 7 com a Tabela 6, percebe-se que ao aumentar a população de 50 para 100 indivíduos fez com que quase todos os testes gerassem a solução ótima na inicialização do problema, através do método heurístico e não do processo evolutivo. A única exceção foi no teste de número 10, onde a solução inicial foi muito próxima da ótima, sendo refinada pelo processo evolutivo.

Figura 34 - Curva média de evolução para uma população de tamanho 100**Fonte: Autor**

Analisando o gráfico da Figura 32, pode-se perceber que a curva de evolução da população se tornou um valor constante, fixo em 80. Isso se dá pelo fato de haver um único teste em que houve duas soluções parciais. Para a primeira solução, como o valor atingido era bem próximo de 80, não foi o suficiente para alterar a média. Para a segunda solução parcial, que ocorre apenas nesse último teste, a solução é a própria média, que também é igual a 80.

Por fim, a Tabela 8 apresenta os resultados dos testes para uma população de 200 indivíduos. A quantidade de crossover por cruzamento para este teste é 50.

Tabela 8 - Sequência de resultados para uma população de tamanho 200

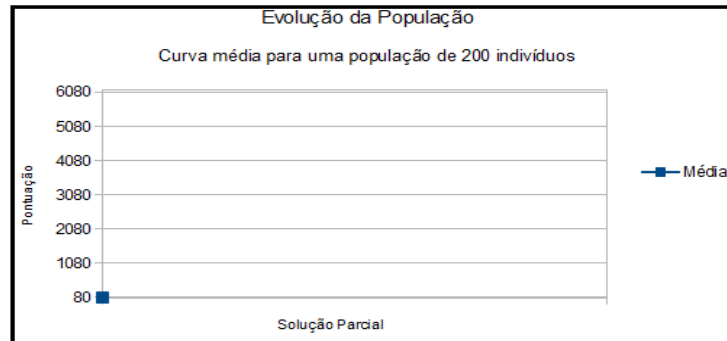
| Teste | Evolução | Qtde Evolucoes | Resultado |
|-------|----------|----------------|-----------|
| 1 | 80 | 1 | 80 |
| 2 | 80 | 1 | 80 |
| 3 | 80 | 1 | 80 |
| 4 | 80 | 1 | 80 |
| 5 | 80 | 1 | 80 |
| 6 | 80 | 1 | 80 |
| 7 | 80 | 1 | 80 |
| 8 | 80 | 1 | 80 |
| 9 | 80 | 1 | 80 |
| 10 | 80 | 1 | 80 |

Fonte: Autor

Para uma população com 200 indivíduos, a solução inicial foi encontrada durante a inicialização da população, para todos os testes realizados. Ou seja, para o cenário de testes em questão, a solução por heurística mostrou-se viável quando utilizada uma população grande. Com 200 indivíduos foi suficiente para obter 100% de testes com a solução ideal, sem a necessidade de recorrer ao processo evolutivo.

A Figura 35 apresenta o gráfico de resultados. Pode-se notar que o gráfico tornou-se, simplesmente, um ponto no plano.

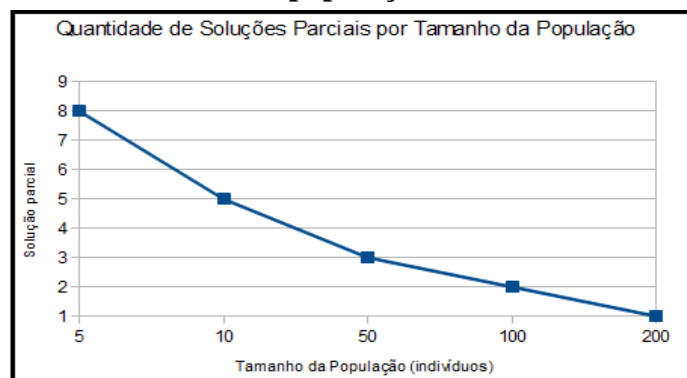
Figura 35 - Curva média de evolução para uma população de tamanho 200



Fonte: Autor

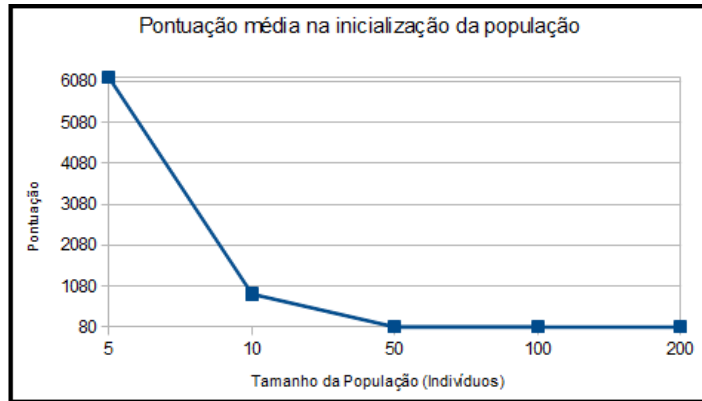
A fato mais notável nos testes realizados anteriormente é o aumento do decaimento da curva de evolução dos indivíduos à medida que o tamanho da população aumenta. O gráfico da Figura 36 permite ver esse decaimento, através da análise da quantidade média de soluções parciais do indivíduo na execução do algoritmo, em função do tamanho da população.

Figura 36 - Curva média de quantidade de soluções parciais em função do tamanho da população

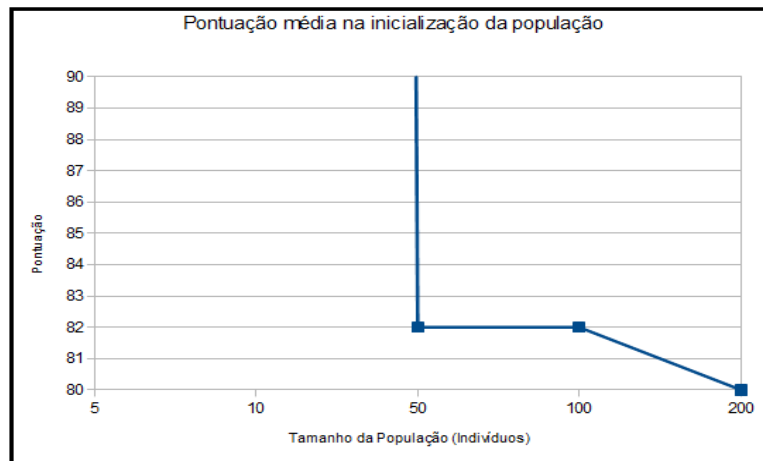


Fonte: Autor

Este fato decorre do aumento da quantidade de testes em que a solução ótima, ou uma solução muito próxima dela, é encontrada na inicialização da população (com a aplicação da heurística apresentada na seção 3.5.2), à medida em que aumenta-se a quantidade de indivíduos, conforme os gráficos das Figuras 37 e 38, pois com indivíduos mais próximos do ideal sendo gerados na inicialização, uma quantidade menor de gerações é necessária para chegar à solução ótima, havendo também uma menor quantidade de soluções parciais.

Figura 37 - Pontuação média na população inicial em função do tamanho da população

Fonte: Autor

Figura 38 - Pontuação média na população inicial em função do tamanho da população

Fonte: Autor

Para o primeiro cenário de testes, o algoritmo genético proposto apresentou um bom desempenho, conseguindo atingir a solução ótima em 100% dos casos, embora para este problema a solução por heurística tenha se mostrado também muito eficiente quando aplicada em grandes populações. Ou seja, para problemas de dimensões pequenas, a heurística é capaz de encontrar excelentes soluções, sem a necessidade de submeter o problema ao processo evolutivo.

4.1.2 Cenário de testes 2

O segundo cenário de testes a ser analisado consiste em um escalonamento de 3 processos por 6 recursos. Trata-se do cenário de testes 1, acrescido de um novo processo.

Os recursos para este cenário de testes são os mesmos do cenário de testes 1, porém com uma diferente quantidade disponível para cada recurso, como pode ser observado na Tabela 9.

Tabela 9 - Recursos disponíveis para o cenário de testes 2

| Recurso | Quantidade |
|-------------|------------|
| Batedeira | 3 |
| Forno | 1 |
| Freezer | 2 |
| Confeiteiro | 1 |
| Inspetor | 1 |
| Embalador | 1 |

Fonte: Autor

Para este cenário, consideremos a existência de três processos a serem escalonados: “Fazer Bolo”, “Fazer Torta” e “Fazer Pudim”, os quais utilizam os recursos na seguinte ordem:

- a) “Batedeira → Forno → Confeiteiro → Embalador”, para o bolo;
- b) “Batedeira → Freezer → Confeiteiro → Embalador”, para a torta;
- c) “Batedeira → Freezer → Forno → Confeiteiro → Inspetor → Embalador”.

A Tabela 10 apresenta o tempo de utilização de cada recurso por cada processo. Tempo igual a zero indica que o recurso não é utilizado pelo processo. O tempo limite considerado para os três processos é 120.

Tabela 10 - Tempo de Utilização do Recurso por Processo para o Cenário de testes 2

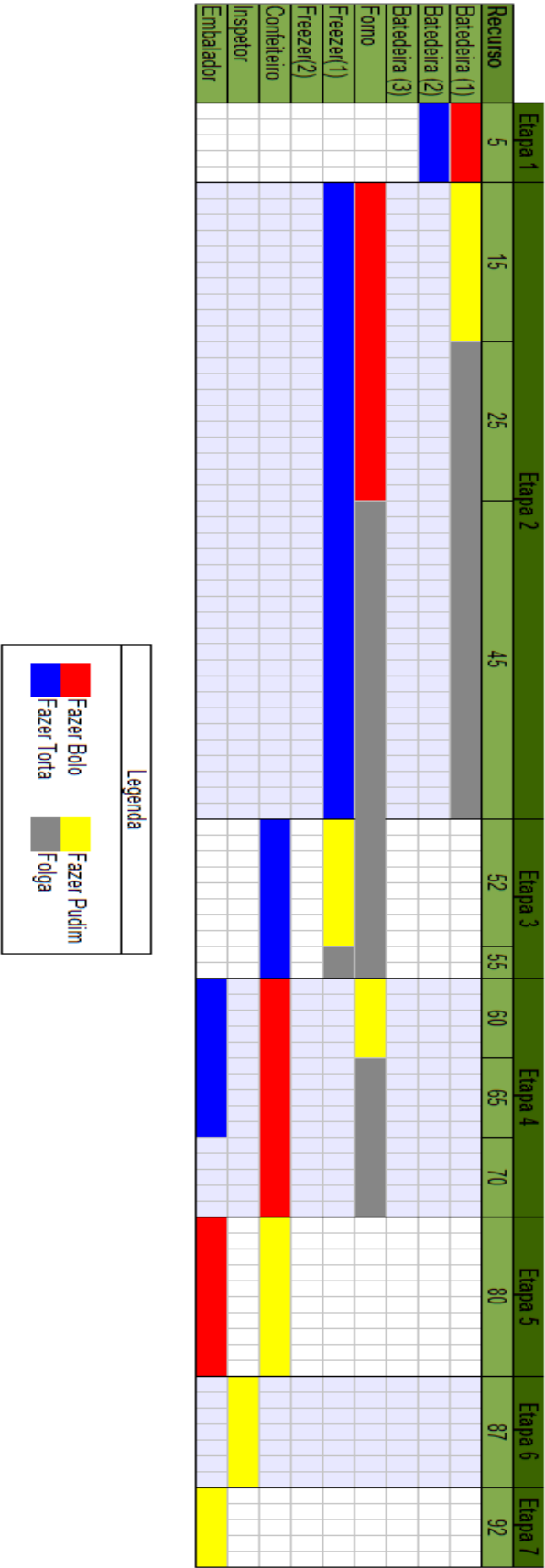
| Tarefa | Recurso | | | | | |
|-------------|-----------|-------|---------|-------------|----------|-----------|
| | Batedeira | Forno | Freezer | Confeiteiro | Inspetor | Embalador |
| Fazer Bolo | 5 | 20 | 0 | 15 | 0 | 10 |
| Fazer Torta | 5 | 0 | 40 | 10 | 0 | 10 |
| Fazer Pudim | 10 | 5 | 8 | 10 | 7 | 5 |

Fonte: Autor

A melhor solução encontrada para o problema é a matriz apresentada em (45). A representação gráfica desse escalonamento é exibida na Figura 39.

$$\begin{bmatrix} 1 & 0 & 2 & 0 & 4 & 6 & 0 & 0 \\ 1 & 0 & 3 & 4 & 6 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 2 & 4 & 5 & 6 \end{bmatrix} \quad (45)$$

Figura 39 - Representação da melhor solução encontrada para o cenário de testes 2



Fonte: Autor

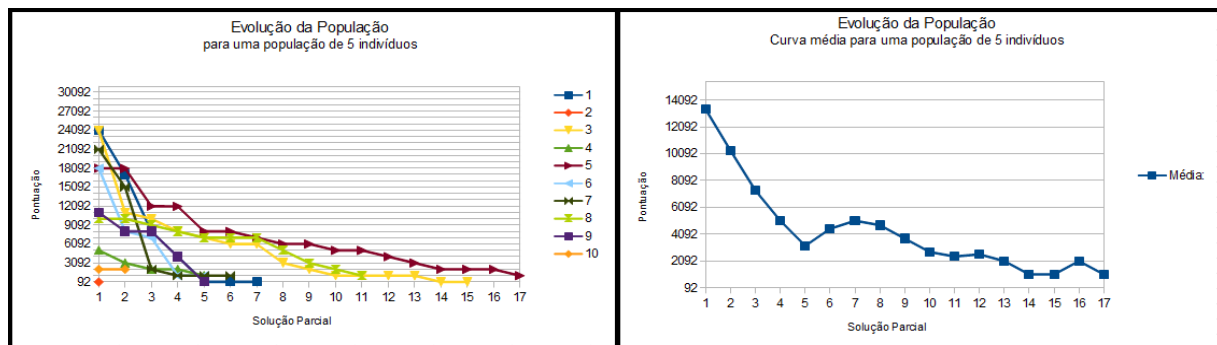
A Tabela 11 apresenta os resultados do segundo cenário de testes para uma população de 5 indivíduos, onde os indicados por “(inválido)” representam soluções finais que não podem ser aplicadas ao problema (soluções inválidas), pois não cumprem uma ou mais restrições do modelo matemático apresentado na seção 3.4.3. A Figura 40 apresenta esses resultados graficamente, através de uma curva individual e uma curva média de evolução dos indivíduos da população.

Tabela 11 - Sequência de resultados para uma população de tamanho 5

| Teste | Evolução | | | | | | | | | | | | | | | | | Qtde Evolucoes | Resultado | |
|-------|----------|-------|-------|-------|------|------|------|------|------|------|------|------|------|------|------|------|------------|----------------|-----------|------|
| 1 | 24081 | 17088 | 8102 | 4106 | 120 | 117 | 115 | | | | | | | | | | | 7 | 115 | |
| 2 | 100 | | | | | | | | | | | | | | | | | 1 | 100 | |
| 3 | 24074 | 11099 | 10110 | 8102 | 7113 | 6107 | 6097 | 3105 | 2096 | 1109 | 1099 | 1094 | 1089 | 100 | 97 | | (invalido) | 13 | 1089 | |
| 4 | 5108 | 3110 | 2111 | 2101 | 1112 | | | | | | | | | | | | (invalido) | 5 | 1112 | |
| 5 | 18092 | 18049 | 12085 | 12055 | 8094 | 8092 | 7100 | 6106 | 6104 | 5100 | 5095 | 4096 | 3100 | 2101 | 2096 | 2091 | 1097 | (invalido) | 13 | 3100 |
| 6 | 18077 | 8112 | 7103 | 1109 | 1089 | | | | | | | | | | | | (invalido) | 5 | 1089 | |
| 7 | 21082 | 15090 | 2101 | 1102 | 1094 | 1089 | | | | | | | | | | | (invalido) | 6 | 1089 | |
| 8 | 10093 | 10090 | 9091 | 8092 | 7103 | 7098 | 7093 | 5095 | 3099 | 2103 | 1104 | | | | | | (invalido) | 11 | 1104 | |
| 9 | 11094 | 8105 | 8085 | 4094 | 108 | | | | | | | | | | | | | 5 | 108 | |
| 10 | 2106 | 2098 | | | | | | | | | | | | | | | (invalido) | 2 | 2098 | |

Fonte: Autor

Figura 40 - Curvas individual e média de evolução para uma população de tamanho 5



Fonte: Autor

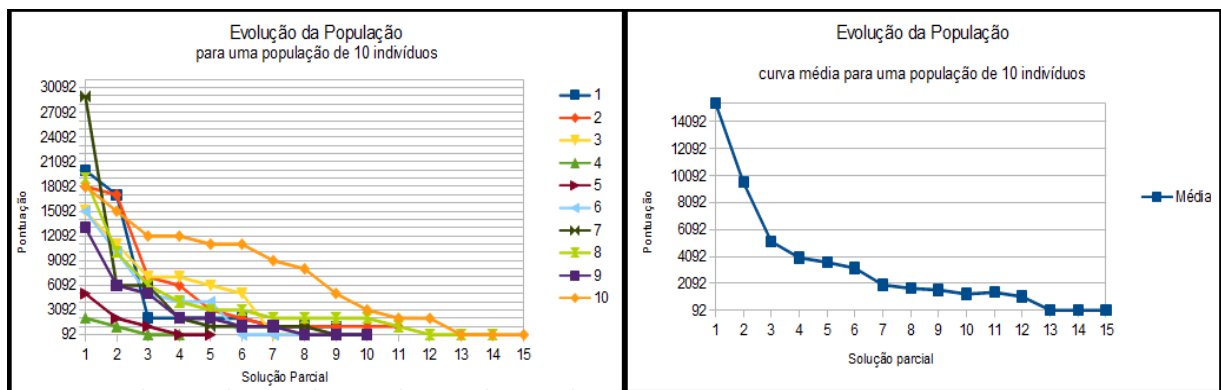
Em comparação com a Tabela 4, que apresenta os resultados para uma população de mesmo tamanho, pode-se perceber que a inclusão de um único processo a mais para ser escalonado tornou o problema mais complexo, de forma que, para uma população de 5 indivíduos, a maioria dos testes resultou em soluções inválidas, ou seja, 10.000 gerações não foram suficientes para que o problema convergisse para uma solução ótima.

Aumentando a população para 10 indivíduos, obtém-se os resultados apresentados na Tabela 12 e na Figura 41.

Tabela 12 - Sequência de resultados para uma população de tamanho 10

| Teste | Evolução | | | | | | | | | | | | | | | Qtde Evolucoes | Resultado | |
|-------|----------|-------|-------|-------|-------|-------|------|------|------|------|------|------|-----|-----|----|----------------|-----------|------|
| 1 | 20063 | 17086 | 2115 | 2113 | 2105 | 2095 | 1109 | 1099 | 1094 | | | | | | | (invalido) | 9 | 1094 |
| 2 | 18090 | 17093 | 7108 | 6087 | 3115 | 2096 | 1117 | 1114 | 1109 | 1104 | 1094 | | | | | (invalido) | 11 | 1094 |
| 3 | 15078 | 11084 | 7111 | 7101 | 6102 | 5107 | 110 | 100 | | | | | | | | | 8 | 100 |
| 4 | 2111 | 1099 | 100 | 95 | | | | | | | | | | | | | 4 | 95 |
| 5 | 5098 | 2101 | 1112 | 118 | 103 | | | | | | | | | | | | 5 | 103 |
| 6 | 15088 | 10098 | 5102 | 4114 | 4098 | 102 | 97 | 92 | | | | | | | | | 8 | 92 |
| 7 | 29051 | 6109 | 6082 | 2106 | 1117 | 1107 | 1104 | 1099 | 110 | 105 | | | | | | | 10 | 105 |
| 8 | 19086 | 10103 | 6114 | 4111 | 3112 | 3107 | 2115 | 2110 | 2105 | 2100 | 1104 | 110 | 105 | 95 | | | 13 | 105 |
| 9 | 13089 | 6087 | 5110 | 2108 | 2103 | 1114 | 1104 | 110 | 105 | 100 | | | | | | | 10 | 100 |
| 10 | 18090 | 15100 | 12093 | 12083 | 11099 | 11089 | 9096 | 8102 | 5110 | 3107 | 2105 | 2100 | 105 | 100 | 97 | | 13 | 105 |

Fonte: Autor

Figura 41 - Curvas individual e média de evolução para uma população de tamanho 10

Fonte: Autor

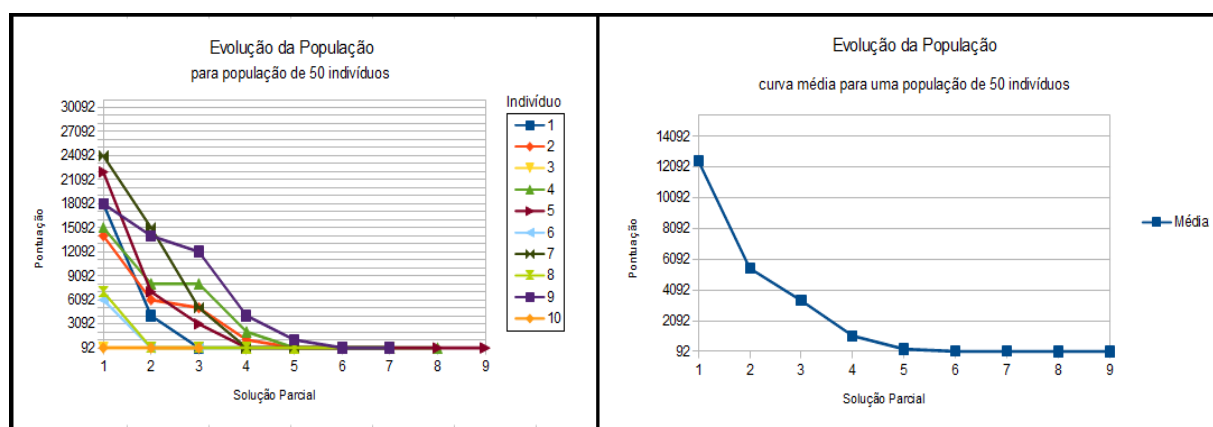
Como pode ser observado, houve uma melhora em relação ao teste anterior (com população de cinco indivíduos), diminuindo para apenas dois resultados inválidos. Em comparação ao cenário de testes 1, percebe-se o aumento da complexidade do problema (pela inclusão do terceiro processo) pelo aumento da quantidade de soluções parciais encontradas antes do fim da execução do algoritmo. A curva média para o cenário de testes 1 (e população de 10 indivíduos) tem o decaimento muito maior do que para este cenário de testes.

Na Tabela 13 estão registrados os resultados deste cenário de testes para uma população de tamanho 50 e 13 cruzamentos por geração. A Figura 42 apresenta os resultados graficamente.

Tabela 13 - Sequência de resultados para uma população de tamanho 50

| Teste | Evolução | | | | | | | | | | Qtde Evolucoes | Resultado |
|-------|----------|-------|-------|------|------|-----|-----|----|----|--|----------------|-----------|
| 1 | 18072 | 4111 | 107 | 103 | 100 | 97 | 92 | | | | 7 | 92 |
| 2 | 14094 | 6091 | 5100 | 1104 | 110 | 100 | | | | | 6 | 100 |
| 3 | 120 | 118 | 103 | | | | | | | | 3 | 103 |
| 4 | 15090 | 8107 | 8099 | 2113 | 110 | 107 | 100 | 97 | | | 8 | 97 |
| 5 | 22061 | 7095 | 3085 | 110 | 105 | 102 | 100 | 95 | 92 | | 9 | 92 |
| 6 | 6089 | 102 | 100 | | | | | | | | 3 | 100 |
| 7 | 24071 | 15085 | 5105 | 115 | 112 | 110 | 105 | | | | 7 | 105 |
| 8 | 7088 | 120 | 118 | 117 | 115 | 113 | 108 | | | | 7 | 108 |
| 9 | 18085 | 14079 | 12095 | 4116 | 1101 | 107 | 97 | | | | 7 | 97 |
| 10 | 105 | 100 | 97 | | | | | | | | 3 | 97 |

Fonte: Autor

Figura 42 - Curvas individual e média de evolução para uma população de tamanho 50

Fonte: Autor

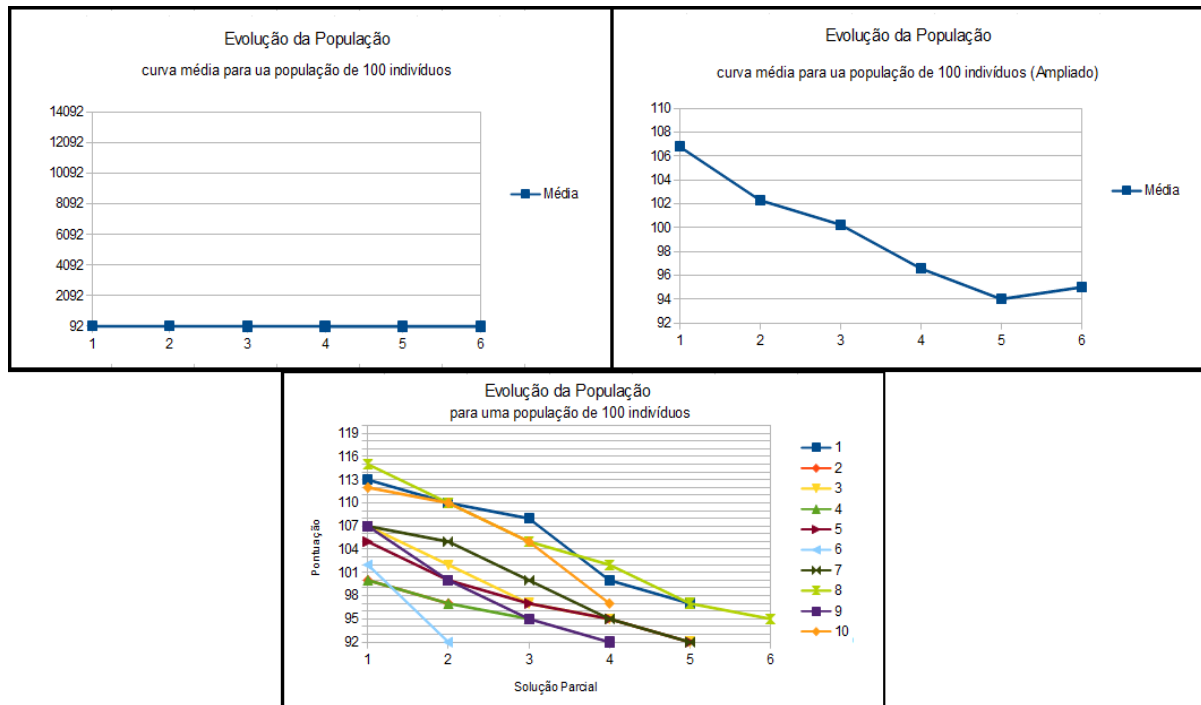
Para este teste, não foram geradas soluções inválidas e, em dois casos, chegou à solução ótima. Nos demais, chegou a soluções bem próximas desta. A curva de evolução da população, por sua vez, teve um grande aumento no decaimento.

Para o teste com população de 100 indivíduos, os resultados encontram-se registrados na Quadro 14, e podem ser observados nas curvas da Figura 43. A quantidade de cruzamentos realizados em cada geração, para esse teste, é 25. A Figura 43 apresenta apenas os gráficos com os resultados de cada teste, o gráfico com a curva média e a curva média ampliada. Como os resultados foram muitos próximos, O gráfico com os resultados individuais dos testes está com uma escala reduzida em comparação com os anteriores, para que as curvas não apareçam sobrepostas.

Tabela 14 - Sequência de resultados para uma população de tamanho 100

| Teste | Evolução | | | | | | Qtde Evolucoes | Resultado |
|-------|----------|-----|-----|-----|----|----|----------------|-----------|
| 1 | 113 | 110 | 108 | 100 | 97 | | 5 | 97 |
| 2 | 100 | 97 | | | | | 2 | 97 |
| 3 | 107 | 102 | 97 | 95 | 92 | | 5 | 92 |
| 4 | 100 | 97 | 95 | | | | 3 | 95 |
| 5 | 105 | 100 | 97 | 95 | 92 | | 5 | 92 |
| 6 | 102 | 92 | | | | | 2 | 92 |
| 7 | 107 | 105 | 100 | 95 | 92 | | 5 | 92 |
| 8 | 115 | 110 | 105 | 102 | 97 | 95 | 6 | 95 |
| 9 | 107 | 100 | 95 | 92 | | | 4 | 92 |
| 10 | 112 | 110 | 105 | 97 | | | 4 | 97 |

Fonte: Autor

Figura 43 - Curvas individual, média e média ampliada de evolução para uma população de tamanho 100

Fonte: Autor

Para este teste, nota-se uma maior aproximação, em todas as execuções, da solução ótima. Uma quantidade maior de indivíduos atingiu essa solução em relação aos testes com população de 50 indivíduos. Devido à aproximação das respostas, o gráfico aproxima-se de uma reta constante, devendo ser observada uma versão ampliada (em menor escala) para perceber que o mesmo ainda não se tornou uma reta.

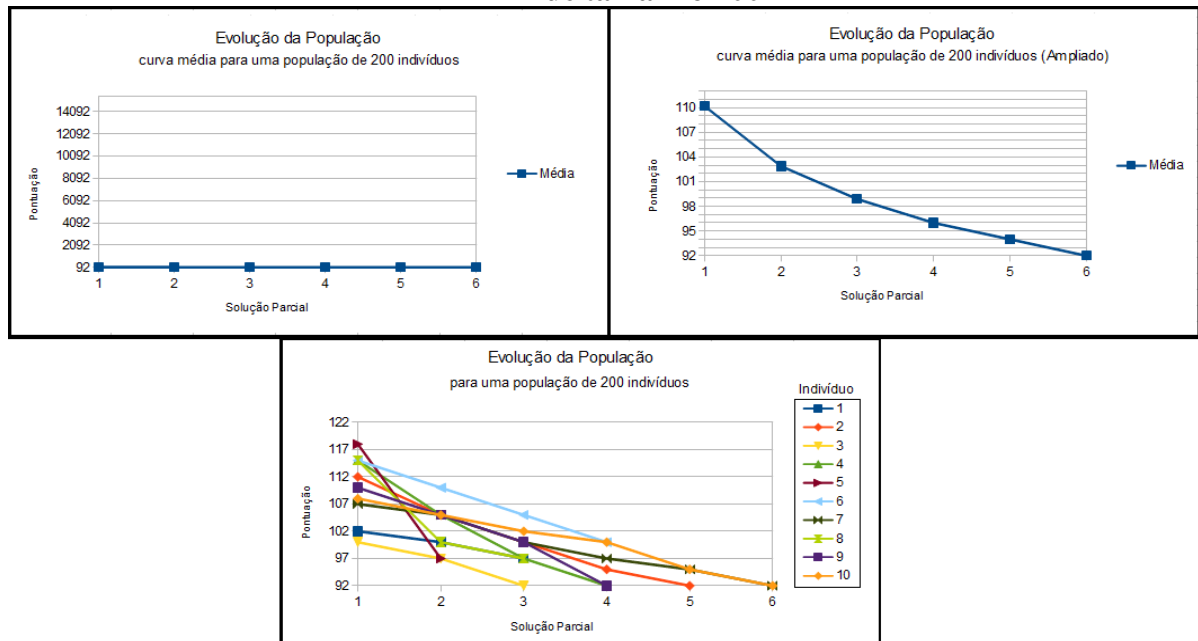
Por fim, estão registrados na Tabela 15 e na Figura 44 os resultados para os testes com uma população de 200 indivíduos e 50 operações de *crossovers* por geração.

Tabela 15 - Sequência de resultados para uma população de tamanho 100

| Teste | Evolução | | | | | | Qtde Evolucoes | Resultado |
|-------|----------|-----|-----|-----|----|----|----------------|-----------|
| 1 | 102 | 100 | 97 | | | | 3 | 97 |
| 2 | 112 | 105 | 100 | 95 | 92 | | 5 | 92 |
| 3 | 100 | 97 | 92 | | | | 3 | 92 |
| 4 | 115 | 105 | 97 | 92 | | | 4 | 92 |
| 5 | 118 | 97 | | | | | 2 | 97 |
| 6 | 115 | 110 | 105 | 100 | | | 4 | 100 |
| 7 | 107 | 105 | 100 | 97 | 95 | 92 | 6 | 92 |
| 8 | 115 | 100 | 97 | | | | 3 | 97 |
| 9 | 110 | 105 | 100 | 92 | | | 4 | 92 |
| 10 | 108 | 105 | 102 | 100 | 95 | 92 | 6 | 92 |

Fonte: Autor

Figura 44 - Curvas individual, média e média ampliada de evolução para uma população de tamanho 200



Fonte: Autor

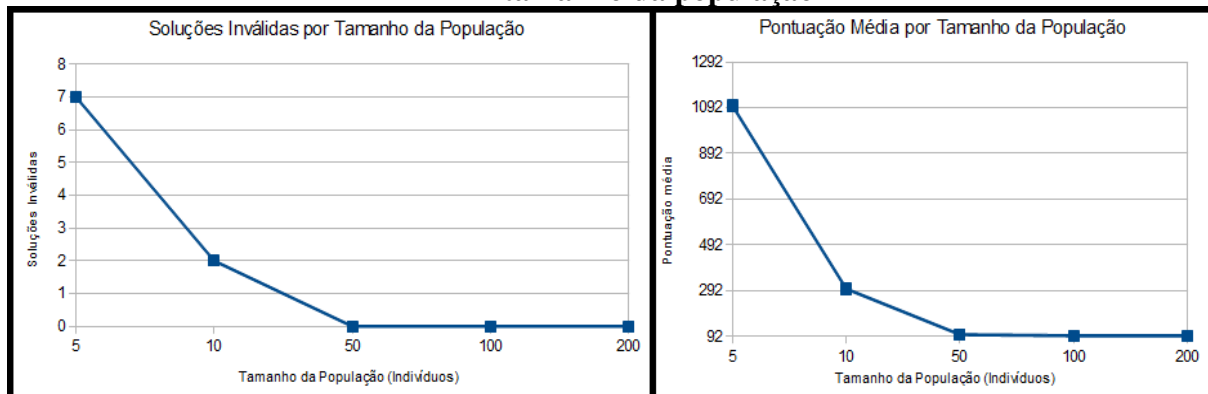
Analisando os resultados obtidos para este cenário de testes, percebe-se que o fato ocorrido no cenário 1 se repete: à medida que se aumenta o tamanho da população, o decaimento da curva de evolução dos indivíduos aumenta. Quanto maior o tamanho da população, maior a probabilidade de encontrar uma boa solução através da *heurística* utilizada na inicialização do algoritmo, havendo uma menor pontuação na população inicial, uma consequente diminuição na quantidade de refinamentos até chegar à solução ótima, e também uma maior quantidade de novos indivíduos introduzidos a cada geração. Essa combinação de fatores contribui para uma melhor convergência.

Para este cenário, nem todos os testes resultaram na mesma solução, como aconteceu no cenário de testes 1. A pontuação dos testes, em média, se aproximou da solução ótima apenas na

medida em que aumentamos o tamanho da população.

A Figura 45 nos permite analisar a convergência do algoritmo, observando a pontuação média atingida nos testes em função do tamanho da população, e a relação entre a diferença de pontuação e a quantidade de indivíduos inválidos.

Figura 45 - Gráficos de Soluções Inválidas e Pontuação Média, ambos em função do tamanho da população

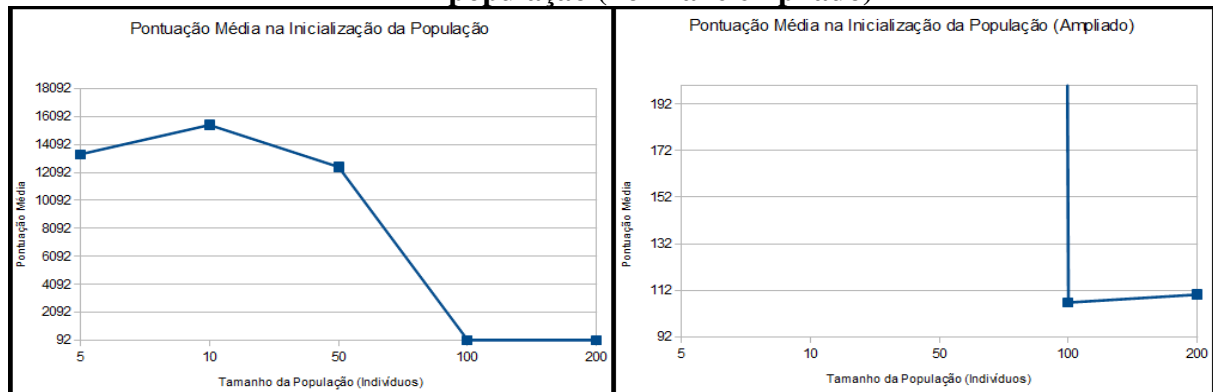


Fonte: Autor

Para uma população de 5 indivíduos, foram geradas muitas soluções inválidas. Por isso, a pontuação média foi alta. Aumentando a população para 10 indivíduos, há uma redução na quantidade de indivíduos inválidos, causando uma pontuação média muito melhor. A partir de 50 indivíduos, não são mais geradas soluções inválidas, de forma que a média cai para próximo de 92, que é a solução ótima.

A Figura 46 apresenta outro gráfico que precisa ser analisado: Pontuação média na inicialização da população. Através desse gráfico, pode-se perceber que, para populações maiores, a heurística de inicialização da população consegue encontrar boas soluções, enquanto para populações pequenas o processo evolutivo precisa ser aplicado. Porém, para este cenário de testes, percebe-se que mesmo para uma população de 200 indivíduos, a solução ótima não é mais gerada na população inicial, embora soluções bem próximas dela sejam encontradas. Podemos perceber que, ao aumentar o tamanho do problema (em relação ao cenário de testes 1), a heurística perde eficácia.

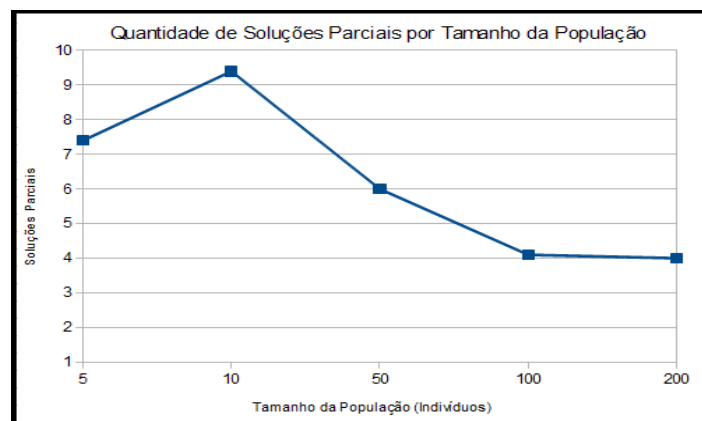
Figura 46 - Pontuação média na inicialização da população, em função do tamanho da população (normal e empliado)



Fonte: Autor

Como consequência do fato observado nos gráficos da Figura 46, A curva de quantidade de soluções parciais tende a acompanhá-lo, como pode ser observado na Figura 47. Dessa forma, para nenhum tamanho de população obteve-se uma média de uma única solução, como ocorreu no cenário de testes 1, evidenciando que, para a heurística de inicialização da população ser capaz de encontrar a solução ótima sem passar pelo processo evolutivo, o tamanho da população deve aumentar à medida que as dimensões do problema aumentam. Pode-se notar que, diferente do cenário de testes 1 (Figura 35), a curva nunca toca o eixo das abscissas (O que representaria uma única solução parcial – a solução inicial da população).

Figura 47 - Quantidade média de soluções parciais por tamanho da população



Fonte: Autor

4.1.3 Cenário de testes 3

Para o terceiro cenário de testes, foi elaborado um ambiente com dimensões maiores. Para este problema, há uma quantidade muito maior de processos a serem escalonados do que nos testes anteriores, e também uma diversidade maior de recursos.

A Tabela 16 apresenta os recursos disponíveis para este problema e suas respectivas quantidades.

Tabela 16 - Recursos disponíveis para o cenário de testes 3

| Recurso | Quantidade |
|-------------|------------|
| Batedeira | 3 |
| Fogao | 2 |
| Forno | 2 |
| Freezer | 5 |
| Confeiteiro | 1 |
| Salgadeiro | 2 |
| Inspetor | 3 |
| Embalador | 3 |

Fonte: Autor

Para este cenário, considerou-se a existência de dez processos a serem escalonados: “Fazer Bolo”, “Fazer Biscoito”, “Fazer Torta”, “Fazer Brigadeiro”, “Fazer Beijinho”, “Fazer Olho de Sogra”, “Fazer Coxinha”, “Fazer Pastel”, “Fazer Kibe” e “Fazer Enrolado”, os quais utilizam os recursos na seguinte ordem:

- a) “Batedeira → Forno → Confeiteiro → Inspetor → Freezer”, para o bolo;
- b) “Batedeira → Fogão → Inspetor → Embalador”, para o biscoito;
- c) “Batedeira → Forno → Confeiteiro → Inspetor”, para a torta;
- d) “Fogão → Freezer → Confeiteiro → Inspetor → Embalador”, para o brigadeiro;
- e) “Batedeira → Freezer → Confeiteiro → Inspetor → Embalador”, para o beijinho;
- f) “Batedeira → Freezer → Confeiteiro → Inspetor → Embalador”, para o olho de sogra;
- g) “Batedeira → Salgadeiro → Fogão → Inspetor → Embalador”, para a coxinha;
- h) “Batedeira → Salgadeiro → Fogão → Inspetor → Embalador”, para o pastel;
- i) “Batedeira → Salgadeiro → Fogão → Inspetor → Embalador”, para o kibe;
- j) “Batedeira → Slagadeiro → Fogão → Inspetor → Embalador”, para o enrolado.

A Tabela 17 apresenta o tempo de utilização de cada recurso por cada processo. Tempo igual

a zero indica que o recurso não é utilizado pelo processo. O tempo limite considerado para os três processos é 180.

Tabela 17 - Tempo de Utilização do Recurso por Processo para o Cenário de testes 3

| Tarefa | Batedeira | Fogão | Forno | Freezer | Confeiteiro | Salgadeiro | Inspetor | Embalador |
|---------------------|-----------|-------|-------|---------|-------------|------------|----------|-----------|
| Fazer Bolo | 10 | 0 | 30 | 30 | 20 | 0 | 5 | 0 |
| Fazer Biscoito | 8 | 10 | 0 | 0 | 0 | 0 | 4 | 5 |
| Fazer Torta | 10 | 0 | 35 | 0 | 15 | 0 | 5 | 0 |
| Fazer Brigadeiro | 0 | 10 | 0 | 5 | 12 | 0 | 6 | 5 |
| Fazer beijinho | 5 | 0 | 0 | 4 | 5 | 0 | 5 | 5 |
| Fazer Olho de Sogra | 5 | 0 | 0 | 4 | 5 | 0 | 5 | 5 |
| Fazer Coxinha | 6 | 10 | 0 | 0 | 0 | 10 | 6 | 6 |
| Fazer pastel | 7 | 8 | 0 | 0 | 0 | 9 | 6 | 6 |
| Fazer kibe | 7 | 15 | 0 | 0 | 0 | 9 | 6 | 6 |
| Fazer Enrolado | 9 | 9 | 0 | 0 | 0 | 10 | 6 | 7 |

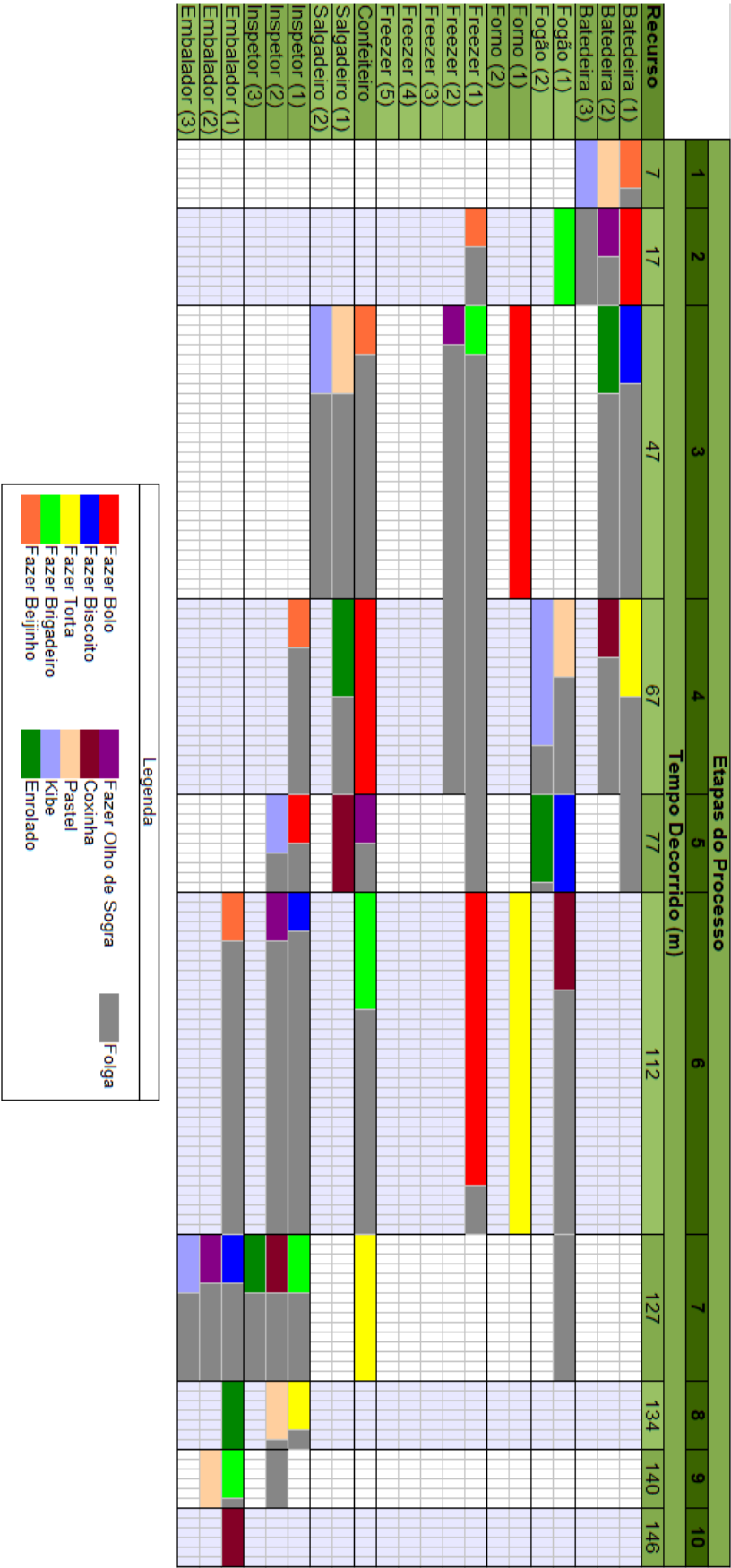
Fonte: Autor

A melhor solução encontrada para o problema é a matriz apresentada em (46). A representação gráfica desse escalonamento é exibida na Figura 48. Não foi possível determinar qual a solução ótima para este ambiente de testes.

$$\begin{bmatrix} 0 & 1 & 3 & 5 & 7 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 2 & 7 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 3 & 5 & 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 4 & 0 & 0 & 5 & 7 & 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 5 & 7 & 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 0 & 5 & 7 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 6 & 2 & 7 & 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 6 & 2 & 0 & 0 & 0 & 7 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 6 & 2 & 7 & 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 6 & 2 & 0 & 7 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(46)

Figura 48 - Representação da melhor solução encontrada para o cenário de testes 3



Fonte: Autor

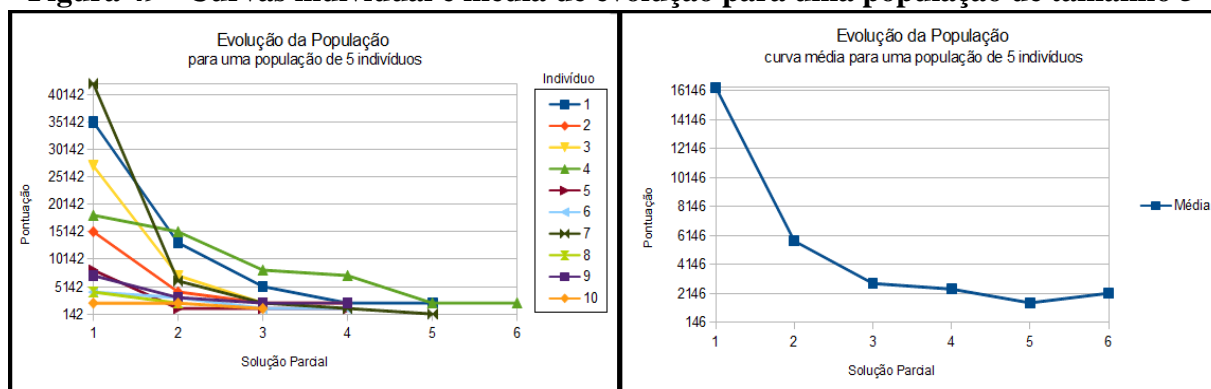
A Tabela 18 apresenta os resultados do terceiro cenário de testes para uma população de 5 indivíduos, onde os indicados por “(inválido)” representam soluções finais que não podem ser aplicadas ao problema (soluções inválidas), pois não cumprem uma ou mais restrições do modelo matemático apresentado na seção 3.4.3. A Figura 49 apresenta esses resultados graficamente, através de uma curva individual e uma curva média de evolução dos indivíduos da população.

Tabela 18 - Sequência de resultados para uma população de tamanho 5

| Teste | Evolução | | | | | | | | Qtde Evolucoes | Resultado |
|-------|----------|-------|------|------|------|------|--|------------|----------------|-----------|
| 1 | 35178 | 13177 | 5178 | 2158 | 2154 | | | (inválido) | 6 | 2154 |
| 2 | 15176 | 4193 | 2160 | 2156 | | | | (inválido) | 5 | 2156 |
| 3 | 27179 | 7165 | 2168 | | | | | (inválido) | 4 | 2168 |
| 4 | 18179 | 15177 | 8169 | 7170 | 2166 | 2161 | | (inválido) | 7 | 2161 |
| 5 | 8171 | 1171 | 1170 | 1165 | | | | (inválido) | 5 | 1165 |
| 6 | 4167 | 3160 | 1178 | 1174 | | | | (inválido) | 5 | 1174 |
| 7 | 42182 | 6190 | 2178 | 1178 | 172 | | | | 5 | 172 |
| 8 | 4179 | 2156 | | | | | | (inválido) | 3 | 2156 |
| 9 | 7179 | 3162 | 2171 | 2161 | | | | (inválido) | 5 | 2161 |
| 10 | 2158 | 2152 | 1159 | | | | | (inválido) | 4 | 1159 |

Fonte: Autor

Figura 49 - Curvas individual e média de evolução para uma população de tamanho 5



Fonte: Autor

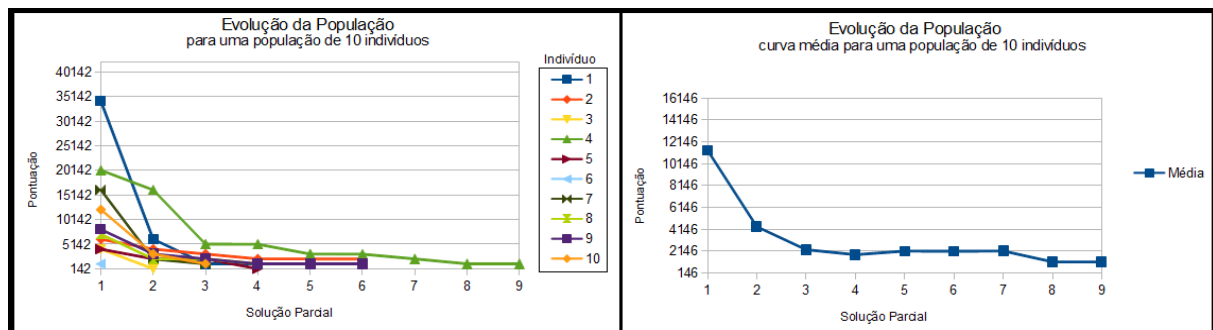
Comparando estes resultados com os apresentados no Quadro 11, percebe-se que ocorre entre os cenários de teste 2 e 3 o mesmo ocorrido entre os cenários 1 e 2: O aumento das dimensões do problema torna o algoritmo menos eficiente para um mesmo tamanho de população. Esse fato é evidenciado pela parcela de 90% de indivíduos inválidos gerados ao final da execução do algoritmo para este cenário de testes, para uma população de 5 indivíduos.

A Tabela 19 contém a sequência de resultados dos testes para uma população de 10 indivíduos. A Figura 50 contém os mesmos dados expressos graficamente.

Tabela 19 - Sequência de resultados para uma população de tamanho 10

| Teste | Evolução | | | | | | | | | | | Qtde Evolucoes | Resultado |
|-------|----------|-------|------|------|------|------|------|------|------|------------|------------|----------------|-----------|
| 1 | 34179 | 6176 | 1169 | 1164 | | | | | | | (invalido) | 5 | 1164 |
| 2 | 6178 | 4162 | 3166 | 2178 | 2158 | 2153 | | | | | (invalido) | 7 | 2153 |
| 3 | 4161 | 178 | | | | | | | | | | 2 | 178 |
| 4 | 20172 | 16174 | 5173 | 5149 | 3172 | 3141 | 2172 | 1178 | 1168 | (invalido) | | 10 | 1168 |
| 5 | 4154 | 2171 | 2155 | 177 | | | | | | | | 4 | 177 |
| 6 | 1155 | | | | | | | | | | (invalido) | 2 | 1155 |
| 7 | 16182 | 2161 | 1167 | | | | | | | | (invalido) | 4 | 1167 |
| 8 | 7175 | 2176 | 2165 | 1166 | | | | | | | (invalido) | 5 | 1166 |
| 9 | 8162 | 3166 | 2168 | 1173 | 1168 | 1162 | | | | | (invalido) | 7 | 1162 |
| 10 | 12175 | 3177 | 1177 | | | | | | | | (invalido) | 4 | 1177 |

Fonte: Autor

Figura 50 - Curvas individual e média de evolução para uma população de tamanho 10

Fonte: Autor

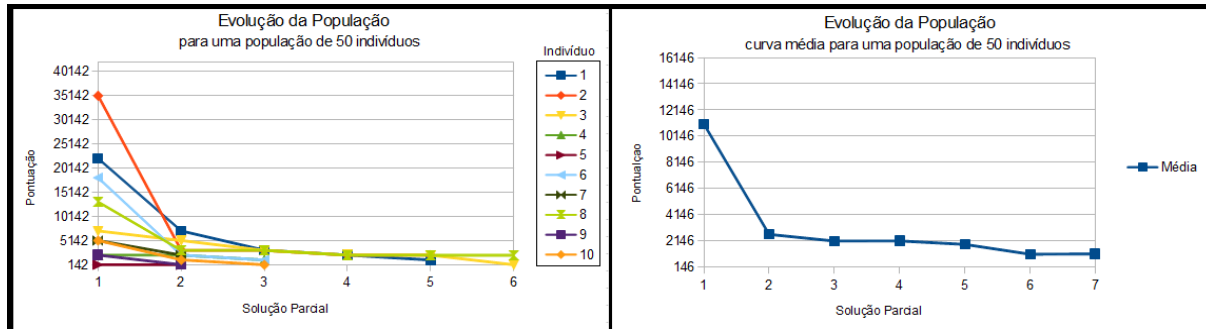
Como nos testes anteriores, o aumento do tamanho da população de 5 para 10 indivíduos causou a diminuição da pontuação média inicial, que consequentemente levou a um maior decaimento da curva de evolução da população e a um resultado final médio com uma pontuação menor. Porém, a melhora na curva média é pequena, e ainda há uma média de 80% de soluções inválidas.

A Tabela 20 e a Figura 41 apresentam os resultados dos testes para uma população de 50 indivíduos.

Tabela 20 - Sequência de resultados para uma população de tamanho 50

| Teste | Evolução | | | | | | | | | Qtde Evolucoes | Resultado |
|-------|----------|------|------|------|------|------|------|------------|--|----------------|-----------|
| 1 | 22178 | 7178 | 3171 | 2176 | 1164 | | | (invalido) | | 6 | 1164 |
| 2 | 35169 | 3142 | 3133 | 2151 | 2147 | | | (invalido) | | 6 | 2147 |
| 3 | 7171 | 5177 | 3146 | 2164 | 2151 | 165 | | | | 6 | 165 |
| 4 | 2172 | 2156 | 1153 | | | | | (invalido) | | 4 | 1153 |
| 5 | 180 | 164 | | | | | | | | 2 | 164 |
| 6 | 18178 | 2168 | 1160 | | | | | (invalido) | | 4 | 1160 |
| 7 | 5177 | 2165 | | | | | | (invalido) | | 3 | 2165 |
| 8 | 13183 | 3158 | 3151 | 2160 | 2146 | 2133 | 1172 | (invalido) | | 8 | 1172 |
| 9 | 2167 | 169 | | | | | | | | 2 | 169 |
| 10 | 5176 | 1176 | 160 | | | | | | | 3 | 160 |

Fonte: Autor

Figura 51 - Curvas individual e média de evolução para uma população de tamanho 50

Fonte: Autor

Para este teste foi constatada uma pequena melhoria nos resultados, em comparação com os testes anteriores, embora essa melhoria seja pequena, uma vez que 60% dos resultados gerados foram inválidos. Além disso, a curva média de evolução teve uma melhora quase desprezível no decaimento.

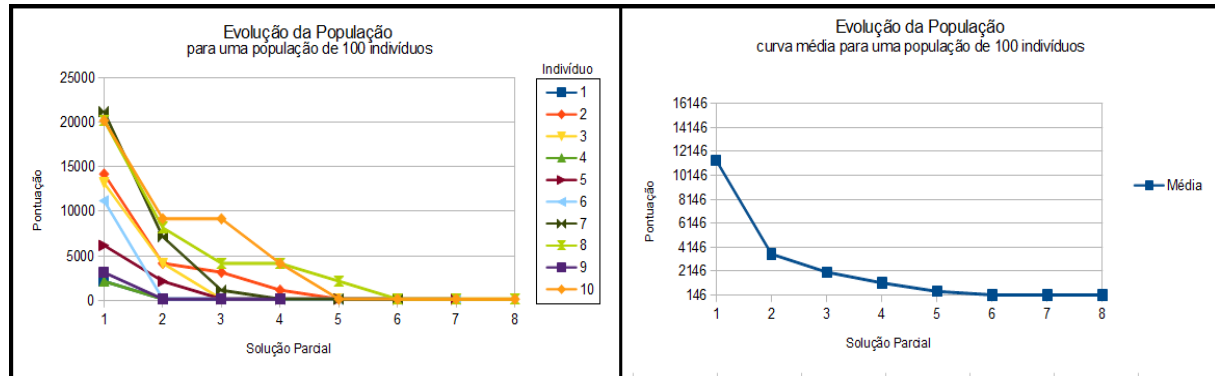
Os testes para uma população com 100 indivíduos estão relatados na Tabela 21 e na Figura 52.

Para um problema com as dimensões deste cenário de testes e para este tamanho de população, ocorre um grande consumo de memória do sistema, ocorrendo “estouro de memória” antes que todas as 10000 gerações sejam executadas. As questões relacionadas a consumo de memória serão analisadas com mais detalhes na seção 4.2.1.

Tabela 21 - Sequência de resultados para uma população de tamanho 100

| Teste | Evolução | | | | | | | | Qtde Evolucoes | Resultado |
|-------|----------|------|------|------|------|-----|-----|-----|----------------|-----------|
| 1 | 2163 | 174 | 167 | 165 | | | | | 4 | 165 |
| 2 | 14183 | 4177 | 3170 | 1174 | 166 | | | | 5 | 166 |
| 3 | 13178 | 4179 | 179 | 164 | 162 | 159 | 157 | | 7 | 157 |
| 4 | 2157 | 165 | | | | | | | 2 | 165 |
| 5 | 6180 | 2168 | 179 | 169 | 166 | 163 | 159 | | 7 | 159 |
| 6 | 11183 | 176 | 171 | 167 | 159 | 154 | 150 | | 7 | 150 |
| 7 | 21178 | 7156 | 1160 | 167 | 164 | 163 | | | 6 | 163 |
| 8 | 20178 | 8177 | 4169 | 4161 | 2171 | 162 | 158 | 155 | 8 | 155 |
| 9 | 3154 | 172 | 159 | 156 | | | | | 4 | 156 |
| 10 | 20179 | 9182 | 9173 | 4167 | 167 | 166 | 162 | 161 | 8 | 161 |

Fonte: Autor

Figura 52 - Curvas individual e média de evolução para uma população de tamanho 100

Fonte: Autor

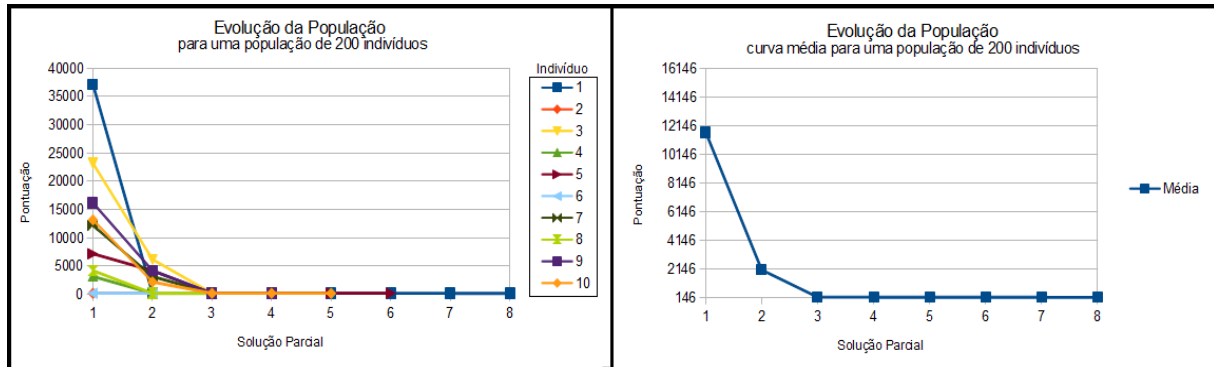
Pode-se notar que, para uma população de 100 indivíduos, as soluções inválidas foram totalmente eliminadas, chegando a 100% de soluções razoáveis (não constituem soluções ótimas, mas representam boas soluções para o problema), e mesmo sem executar todas as 10000 gerações, devido ao problema com memória. Confirma-se então a observação de que, quanto maior as dimensões do problema, maior deve ser o tamanho da população para se obter respostas satisfatórias. Pode-se observar na Figura 52, em comparação com a Figura 51, que a curva média se aproxima bastante do eixo das abscissas, indicando proximidade com a melhor solução encontrada para o problema, que é 146.

Por fim, os testes referentes a uma população com 200 indivíduos são registrados na Tabela 22, e representados nos gráficos da Figura 53. Para este tamanho de população, também ocorrem problemas com memória.

Tabela 22 - Sequência de resultados para uma população de tamanho 200

| Teste | Evolução | | | | | | | | Qtde Evolucoes | Resultado |
|-------|----------|------|-----|-----|-----|-----|-----|-----|----------------|-----------|
| 1 | 37178 | 179 | 171 | 170 | 163 | 158 | 156 | 153 | 8 | 153 |
| 2 | 169 | 161 | 159 | 159 | | | | | 4 | 159 |
| 3 | 23179 | 6169 | 175 | 162 | | | | | 4 | 162 |
| 4 | 3154 | 173 | 159 | 155 | | | | | 4 | 155 |
| 5 | 7167 | 4176 | 165 | 164 | 161 | 159 | | | 6 | 159 |
| 6 | 176 | 172 | 169 | 165 | | | | | 4 | 165 |
| 7 | 12178 | 3153 | 168 | 159 | | | | | 4 | 159 |
| 8 | 4177 | 165 | 146 | | | | | | 3 | 146 |
| 9 | 16179 | 4183 | 166 | 165 | | | | | 4 | 165 |
| 10 | 13171 | 2166 | 178 | 175 | 161 | | | | 5 | 161 |

Fonte: Autor

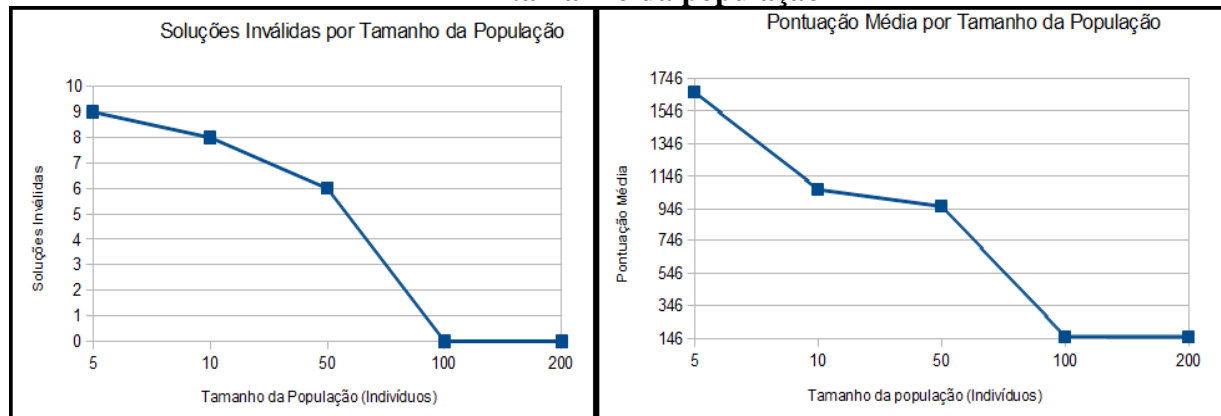
Figura 53 - Curvas individual e média de evolução para uma população de tamanho 200

Fonte: Autor

Comparando a curva média da Figura 53 com a da Figura 52, nota-se que, para este tamanho de população, embora a solução média inicial seja praticamente a mesma (Primeiro ponto na curva média), para uma população de 200 indivíduos a convergência ocorreu muito mais rapidamente. Em ambos os casos obteve-se 8 soluções parciais, mas para uma população de tamanho 200 a curva se aproxima da melhor solução a partir da terceira solução parcial, enquanto para uma população de 100 indivíduos essa aproximação somente ocorre a partir da quinta solução parcial. Nas soluções seguintes, ocorre apenas o refinamento dos resultados, de forma que o algoritmo converge para soluções melhores do que para a população de 100 indivíduos. Graficamente, percebe-se esse fato observando que a curva média se aproxima do eixo das abscissas muito mais rapidamente que para o teste com 100 indivíduos por população, mesmo tendo iniciado a partir de um ponto próximo.

A Figura 54 permite analisar a pontuação média atingida nos testes em função do tamanho da população, e a relação entre a diferença de pontuação e a quantidade de indivíduos inválidos. Pode-se notar que, para este problema, a quantidade de soluções inválidas somente chegou a zero em testes com populações grandes. Para os testes com 50 indivíduos ou menos, a grande maioria das soluções são inválidas. Consequentemente, a pontuação média acompanha esta curva, chegando próximo à melhor solução apenas em populações com pelo menos 100 indivíduos.

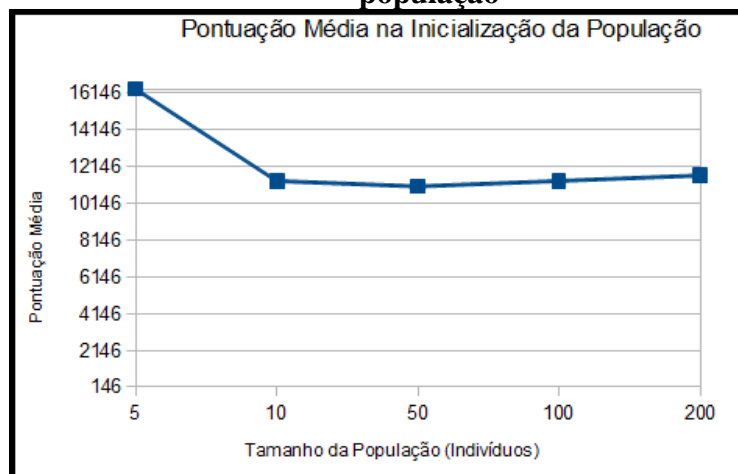
Figura 54 - Curvas de Soluções Inválidas e Pontuação Média, ambas em função do tamanho da população



Fonte: Autor

A curva de pontuação média na inicialização da população, para este teste, teve um comportamento bem diferente dos testes anteriores, como pode ser observado pela Figura 55. Pode-se notar que a curva fica estagnada, não se aproximando da melhor solução. Isto ocorre devido ao fato de a heurística utilizada na inicialização da população ter a eficácia diminuída na média em que aumenta as dimensões do problema. Para que a heurística funcione bem, quando aumenta-se o tamanho do problema, deve-se aumentar também o tamanho da população.

Figura 55 - Pontuação média na inicialização da população, em função do tamanho da população



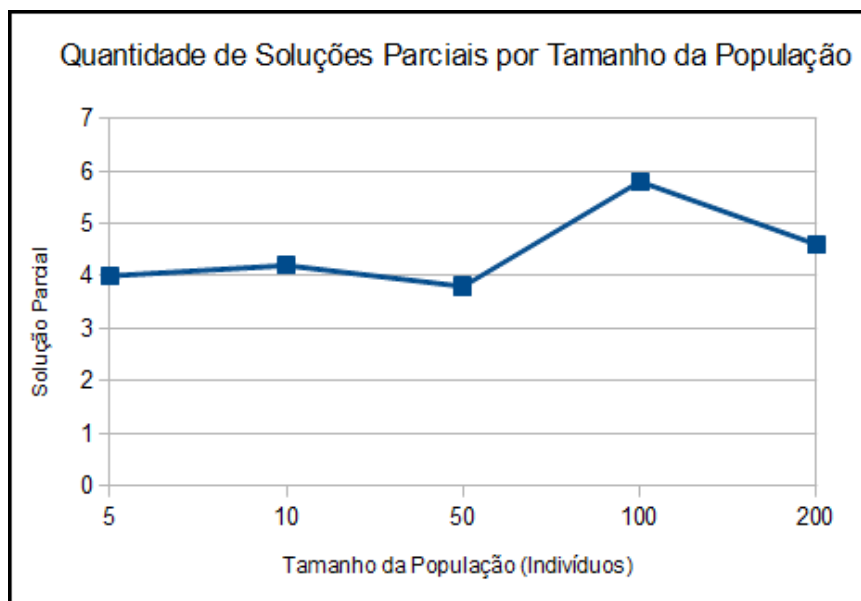
Fonte: Autor

Para este cenário de testes, a heurística apenas gerou soluções iniciais válidas (Ainda assim, não foram as melhores) com tamanhos de população a partir de 500 indivíduos. Esses testes foram realizados apenas com o propósito de confirmar o comportamento da heurística, não tendo sido documentados, uma vez que o escopo de testes definido para análise do modelo foi de 5, 10, 50, 100

e 200 indivíduos.

A Figura 56 apresenta a curva mostrando a quantidade de soluções parciais por tamanho da população. Esta curva também apresentou um comportamento diferentes dos cenários de teste anteriores, tendo aumentado a quantidade de soluções parciais entre os testes com 50 e 100 indivíduos, e voltado a decair no teste seguinte. Uma possível explicação para esse comportamento é a estagnação do algoritmo em soluções inválidas nos testes com 5, 10 e 50 indivíduos, onde a maior parte das soluções não eram viáveis. Para uma população de 100 indivíduos, o algoritmo conseguiu convergir melhor, atingindo as soluções válidas e, conseqüentemente, tendo uma maior quantidade de evoluções. Com 200 indivíduos, a curva voltou a decair pelo mesmo motivo observado na Figura 53: A convergência para este tamanho de população é maior do que para o tamanho 100, havendo novamente uma menor quantidade de soluções parciais.

Figura 56 - Quantidade média de soluções parciais por tamanho da população



Fonte: Autor

4.2 Outros testes e Observações

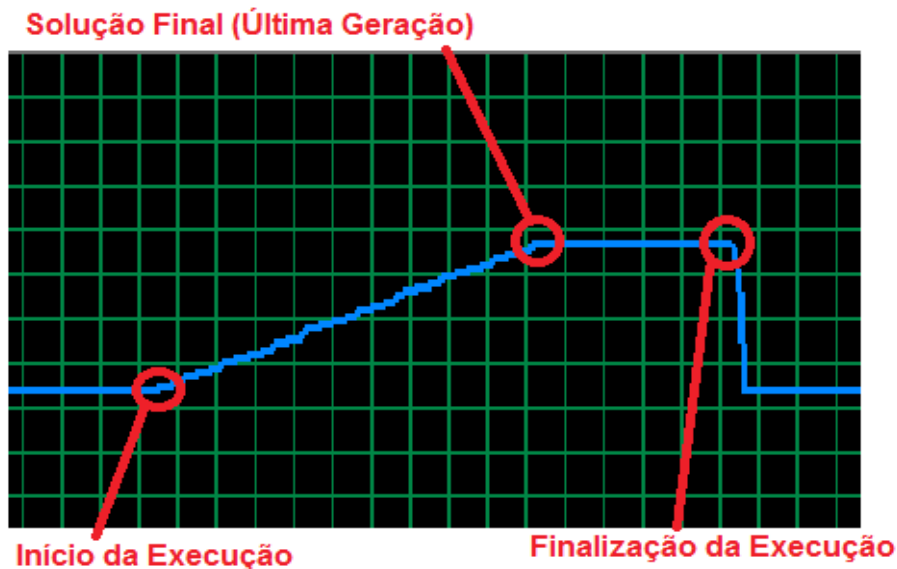
A seguir serão apresentados outros testes e observações realizadas para o algoritmo, envolvendo consumo de memória e processador pela execução do *software*, e a variação de outros parâmetros do algoritmo: A taxa de mutação e a quantidade de *crossover* por geração.

4.2.1 Consumo de Memória

Um dos principais problemas observados durante a execução do *software* desenvolvido foi o consumo excessivo de memória à medida que aumenta o tamanho da população e as dimensões do problema.

Foi observado que, a cada geração, uma quantidade de memória é alocada pelo algoritmo. Por isso, percebe-se que o consumo de memória do sistema aumenta linearmente durante o tempo de execução, e é interrompido quando o mesmo atinge a solução final, apenas liberando toda a memória alocada quando a execução é finalizada (quando o *software* é finalizado). A Figura 57 apresenta essas regiões em um gráfico de consumo de memória do Sistema Operacional Windows 7.

Figura 57 - Execução do algoritmo identificada em um gráfico de consumo de memória do Windows 7



Fonte: Autor

Quanto maior as dimensões do problema, maior será o consumo de memória por geração. Se é aumentado o tamanho da população, o consumo também aumenta, pois tem-se mais indivíduos

consumindo memória a cada geração. E por fim, quanto mais gerações (ou iterações) o *software* executar, mais memória é alocada no decorrer da execução.

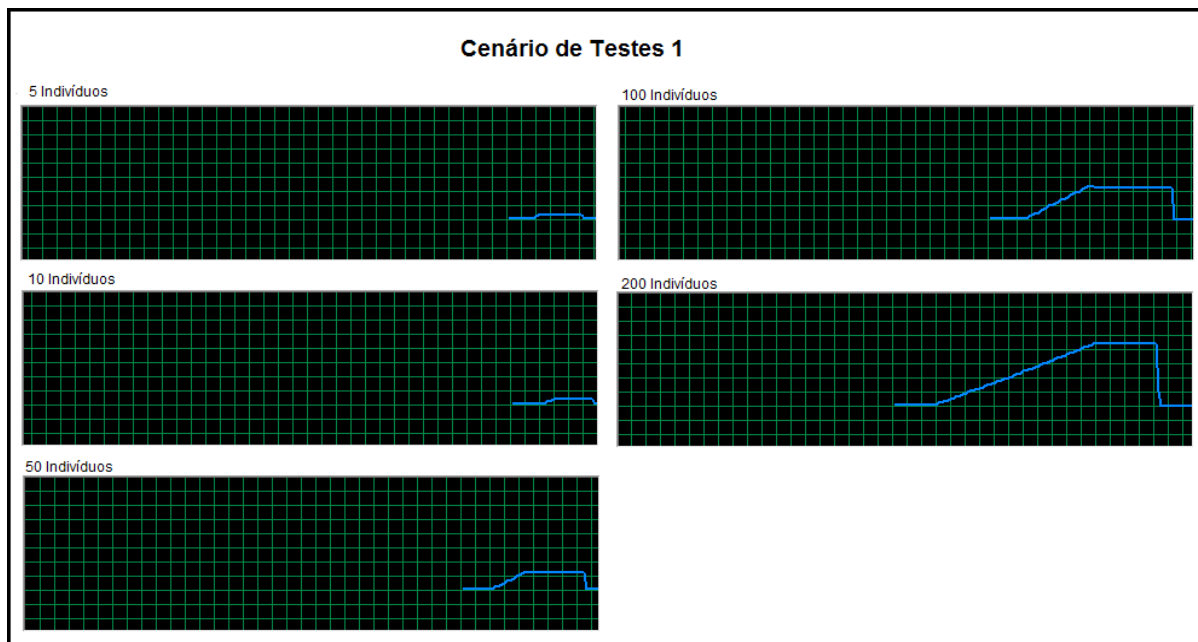
Portanto, pode-se expressar o consumo de memória total do algoritmo (c) em função das dimensões do indivíduo (p linhas por $(p+r-1)$ colunas, onde p é a quantidade de processos e r é a quantidade de recursos), do tamanho da população (t), e do número de gerações (g).

$$c = k \cdot p \cdot (p + r - 1) \cdot t \cdot g; \quad \text{onde } k \text{ é uma constante} \quad (47)$$

A Figura 58 apresenta os gráficos de consumo de memória para a execução do algoritmo, em todos os tamanhos de população utilizados nos testes, em um computador com as seguintes configurações:

- a) AMD Processador Quad-Core E6, 1.5 GHz;
- b) 4 GB de memória RAM (3,48 GB utilizáveis);
- c) Sistema Operacional Windows 7.

Figura 58 - Gráficos de Consumo de memória para o cenário de testes 1



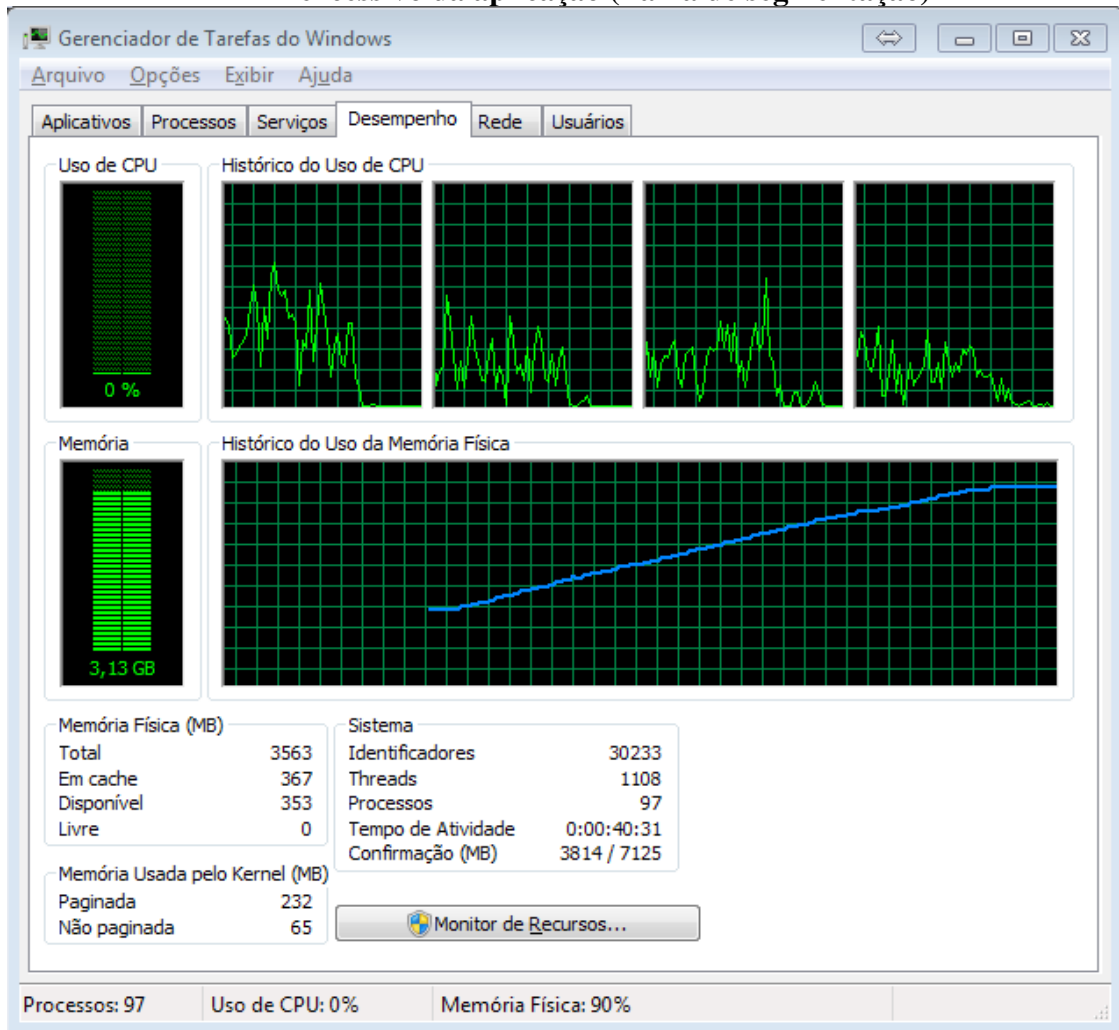
Fonte: Autor

Através da observação dos gráficos, pode-se perceber que, a medida em que se aumenta o tamanho da população, o consumo de memória aumenta proporcionalmente.

Para este problema, embora o consumo para a população de 200 indivíduos seja elevado,

não foi o suficiente para esgotar a memória do sistema. Nos cenários de teste seguintes, o consumo de memória foi mais elevado. Quando o consumo total do sistema se aproxima de 3,48 GB, ocorre falha de segmentação na aplicação, e ela é encerrada, antes de a execução terminar. A Figura 59 mostra um gerenciador de tarefas do *Windows*, no momento em que ocorre o erro na aplicação.

Figura 59 - Gerenciador de Tarefas do Windows registrando o momento de consumo excessivo da aplicação (Falha de segmentação)

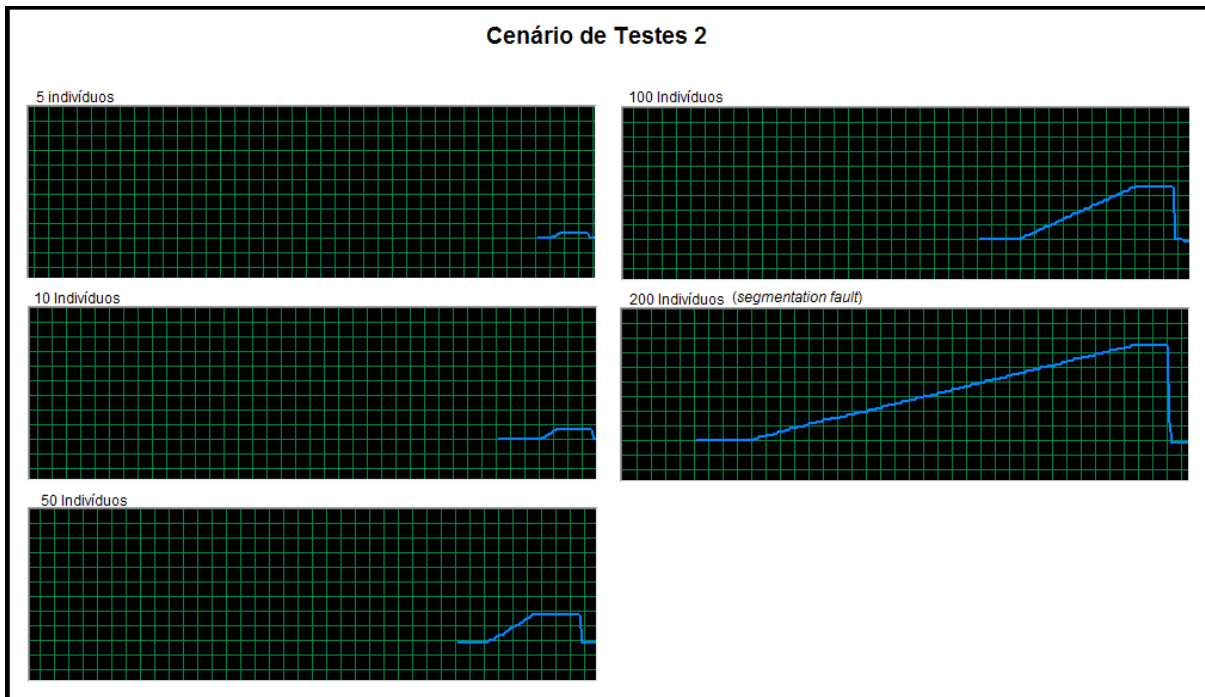


Fonte: Autor

As Figuras 60 e 61 apresentam as sequências de gráficos de consumo de memória para o segundo e terceiro cenários de testes. Os gráficos sinalizados com (falha memória) se referem aos testes onde a execução foi interrompida pela falha de segmentação de memória. No cenário de testes 2, ocorre a falha de segmentação apenas para o teste com 200 indivíduos, e o erro sempre ocorre após o algoritmo já ter chegado à solução ótima.

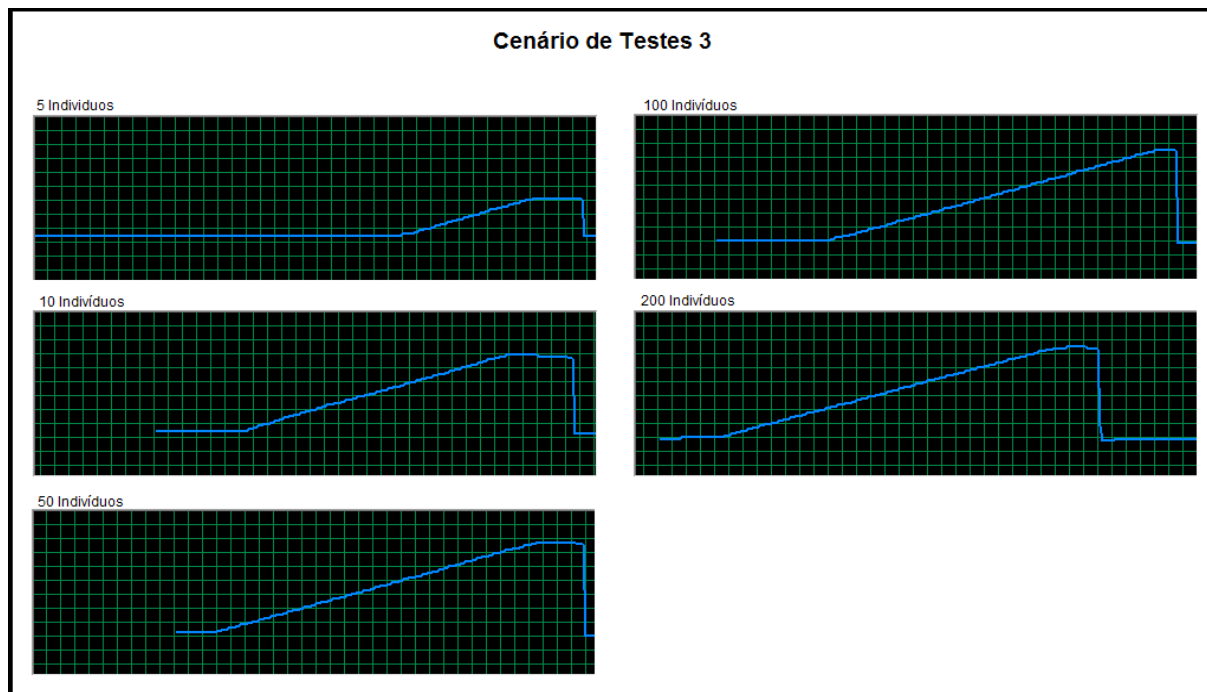
Através da análise desses gráficos, pode-se perceber a relação entre o consumo de memória com o tamanho dos indivíduos e da população.

Figura 60 - Gráficos de Consumo de memória para o cenário de testes 2



Fonte: Autor

Figura 61 - Gráficos de Consumo de memória para o cenário de testes 3



Fonte: Autor

CONCLUSÃO

A grande quantidade de testes realizados permitiu a realização de uma análise detalhada a respeito das características do trabalho desenvolvido.

A primeira constatação é a funcionalidade do modelo matemático proposto. Em todos os testes realizados, a submissão dos indivíduos gerados pelo algoritmo genético ao modelo de PNL gerou uma avaliação correta acerca da validade da solução (isto é, se a solução representada pelo indivíduo atende ou não a todas as restrições do problema). Além disso, a utilização do modelo como função de avaliação forneceu uma base eficiente para determinar as melhores soluções, mas sem excluir as demais, aumentando assim a diversidade da população, e aumentando as probabilidades de convergência.

Outra constatação foi a vantagem obtida pela aplicação de algoritmos genéticos à resolução de problemas combinatoriais, sobre a utilização de uma heurística gulosa. A heurística apresenta excelente desempenho, tanto em qualidade das soluções quanto em tempo de execução, quando aplicada em problemas de pequenas dimensões e com grandes populações. Porém, para problemas mais complexos, o método heurístico se mostrou insuficiente, sendo necessária a utilização de algoritmos genéticos para o refinamento das soluções. Em geral, boas soluções (embora nem sempre ótimas) são encontradas, desde que com o uso de parâmetros adequados ao problema, como tamanho da população e quantidade de cruzamentos por geração.

A utilização de algoritmos genéticos se mostrou, porém, computacionalmente custosa. À medida que as dimensões do problema crescem, o tamanho dos indivíduos e a quantidade de restrições do modelo de PNL crescem também, aumentando o tempo de execução do *software* e havendo um grande consumo de memória do sistema, o que torna a aplicação da ferramenta desenvolvida restrita a ambientes computacionais de alto desempenho, como por exemplo, em grandes servidores.

Outro problema encontrado foi a quantidade de folgas existentes nas soluções geradas, devido ao “truncamento” do tempo de todas as tarefas executadas em um estágio pelo tempo da maior. Esse problema se dá ao fato de o algoritmo não se basear na procura por um caminho crítico, mas sim no processo crítico de cada etapa. Esta abordagem se mostra eficiente quando as tarefas executadas nos estágios possuem tempos de execução aproximados, mas apresenta um “desperdício” de tempo em situações contrárias, onde grande quantidade de folgas são introduzidas, devendo o *schedule* passar por um processo externo de refinamento da solução, caso queira aproveitar ainda mais os benefícios do paralelismo. Portanto, pode-se perceber que o modelo

proposto apresentará melhor desempenho em ambientes nos quais as tarefas são executadas em tempos aproximados. Quanto mais uniforme o processo produtivo, melhor a solução gerada, não havendo necessidade de reprocessamento ou refinamento da solução por processos externos.

Uma vantagem da implementação realizada é que, para que a ferramenta proposta possa ser utilizada em outros tipos de ambientes, que atualmente não são atendidos, precisa-se realizar modificações apenas no modelo matemático, pois é ele quem controla o *fitness* dos indivíduos no algoritmo genético. Dessa forma, ocorre uma separação em “camadas” na ferramenta, onde uma realiza o controle evolutivo, e a outra a avaliação dos indivíduos, devendo esta última ser modificada conforme as necessidades do ambiente a ser modelado.

Portanto, melhorias podem ser feitas no modelo matemático proposto, incluindo ou modificando as restrições, de forma que o mesmo possa ser aplicado a outros tipos de problemas. Dessa forma, grandes benefícios podem ser obtidos em vários tipos de escalonamento de tarefas, como processos produtivos, tarefas computacionais, dentre outros. Através de modificações corretas, o mesmo pode também ser aplicado a outros tipos de ambientes, que não sejam *job shop*.

Dentre as possíveis melhorias no modelo, está a inclusão de restrições para permitir o controle de *setup* de tarefas, isto é, tempo gasto na manutenção dos recursos durante a troca de contexto (trocar o processo que está utilizando o recurso). Também podem ser incluídas restrições para controle de tarefas que precisam ser executadas em sequências ininterruptas, restrições para controle de tempo máximo de espera entre tarefas, ou qualquer outro tipo de restrição que se mostre necessário.

Visando melhorar o desempenho do algoritmo e contornar os problemas encontrados, as propostas para trabalhos futuros incluem:

- a) Alterações nas restrições do modelo matemático proposto, para atender a outros tipos de ambiente e para controle de outros tipos de variáveis do processo produtivo, como tempo de *setup* entre tarefas, tempo de espera, etc.
- b) Desenvolvimento e teste da aplicação de outras técnicas de inteligência artificial, como *swarm* e PSO (*Particle Swarm Optimization*) à execução do modelo matemático proposto;
- c) Comparação da aplicação de técnicas de inteligência artificial à resolução do modelo matemático proposto com a utilização de metodologias matemáticas exatas de resolução;
- d) Revisão e aprimoramento das equações de restrição do modelo de PNL proposto para reduzir a introdução de folgas no processo produtivo, dispensando a utilização de

metodologias externas de refinamento da solução;

- e) Revisar o *software* desenvolvido, com o intuito de reduzir o consumo de memória durante a execução do processo evolutivo.
- f) Avaliar o desempenho do algoritmo em sistema operacional *Linux*.
- g) Avaliar o impacto das alterações dos demais parâmetros, como taxa de mutação e quantidade de *crossover*, sobre o algoritmo. Testes preliminares mostraram que o algoritmo apresenta melhor desempenho com taxas de mutação entre 8 e 11%, e quantidade de *crossover* em torno de 25% do tamanho da população, mas é importante que testes mais detalhados sejam realizados, a fim de descobrir como esses parâmetros afetam o algoritmo, e identificar as melhores taxas com melhor precisão.

REFERÊNCIAS

- ALMEIDA, Dagoberto Alves de. **Planejamento & controle da produção**. 2010. 128f. Universidade Federal de Itajubá, Itajubá. Disponível em: <<http://www.anterior.unifei.edu.br/dagoberto/EPR704PlanejamentoControleProducaoGradEP/PCP/EPR7042010.pdf>>. Acesso em: 22 mai. 2013.
- ALMEIDA, Mayron Rodrigues de; HAMACHER, Sílvio; PACHECO, Marco Aurélio Cavalcanti. Algoritmo genético para programação da produção em refinarias de petróleo. In: **Revista de Inteligência computacional aplicada**, Rio de Janeiro, n.6, Jul. 2010. Disponível em: <http://rica.ele.puc-rio.br/media/Revista_rica_n6_a9.pdf>. Acesso em: 22 ma. 2013.
- BARRICO, Carlos. **Introdução à investigação operacional**. Universidade da Beira Interior, Covilhã, Portugal. Disponível em: <<http://www.di.ubi.pt/~cbarrico/Disciplinas/InvestigacaoOperacional/Downloads/Capitulo1.pdf>>. Acesso em: 22 mai. 2013.
- BORGES, Flávio Hasenclever; DALCOL, Paulo Roberto Tavares. Indústria de processo: comparações e caracterizações. In: ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, 22, 2002, Curitiba. **Anais...** Curitiba: ABEPRO, 2002. 9p.
- BRUCKER, Peter. **Scheduling Algorithms**. 5. ed. Osnabrück, Germany: Springer, 2006. p.11-13.
- CASILLO, Leonardo; SILVA, Ivan Saraiva. **Organização e Arquitetura de Computadores I: pipeline**. Disponível em <<http://www.dimap.ufrn.br/~ivan/orgI/Pipeline.PDF>>. Acesso em 23 nov. 2013.
- CASTRO, Leandro de; ZUBEN, Fernando Von. **Algoritmos genéticos (AG's)**. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia707_02/topico9_02.pdf>. Acesso em: mai, 2013.
- DERIZ, Ana Claudia. **Um método de busca usando algoritmo genético para programação reativa da produção de sistemas de manufatura com recursos compartilhados**. 2007. 119f. Dissertação(Mestrado em Ciência da Computação) – Universidade Federal de São Carlos, Programa de Pós Graduação em Ciência da Computação, São Carlos, SP.
- FERREIRA, Alessandra Henriques. **Proposta de um modelo em programação linear para a solução de problemas de sistemas produtivos *job shop* com setup dependentes da sequência**. 2012. 112f. Tese(Doutorado) – Universidade de São Paulo, departamento de Administração, São Paulo.
- FILITTO, Danilo. Algoritmos genéticos: uma visão explanatória. **Saber acadêmico**, São Paulo, n.6, Dez. 2008. Disponível em: <<http://www.uniesp.edu.br/revista/revista6/pdf/13.pdf>>. Acesso em: 22 mai. 2013.
- FROSSARD, Afonso Celso Pagano. Programação linear: maximização de lucro e minimização de custos. **Revista científica da faculdade Lourenço Filho**, Fortaleza, v.6, n.1, p. 19-48, 2009. Disponível em: <http://www.flf.edu.br/revista-flf.edu/volume06/V6_02.pdf>. Acesso em: 22 mai. 2013.

HILLIER, Frederick S; LIEBERMAN, Gerald J. **Introdução à pesquisa operacional**. 8. ed. São Paulo: Mc Graw Hill, 2006. p.25-31, 530-543.

JUNIOR, Arilson. **O que é ERP – Enterprise Resource Planning?**. Oficina da net. 15 jul. 2011. Disponível em: <http://www.oficinadanet.com.br/artigo/business_intelligence/solucoes-erp>. Acesso em: 02 mar. 2013.

JUNIOR, Muris Lage. Os sistemas de planejamento e controle da produção e o ambiente: uma perspectiva histórica. In: SIMPÓSIO DE ENGENHARIA DE PRODUÇÃO, 13, 2006, Bauru. **Anais...** Bauru: DEP-UNESP, 2006. 7p.

LOPES, Edson Augusto. **Alteração do processo produtivo de uma indústria artesanal para uma produção dedicada**. [s.l]. Disponível em: <<http://www.rgmovei.com.br/14/senai.pdf>>. Acesso em: 22 mai. 2013.

LUCAS, Diogo C. **Algoritmos genéticos: uma introdução**. RS, 2002. Disponível em: <<http://www.inf.ufrgs.br/~alvares/INF01048IA/ApostilaAlgoritmosGeneticos.pdf>>. Acesso em: 22 mai. 2013.

LUENBERGER, David G.; YE, yinyu. **Linear and nonlinear programming**. 3. ed. New York: Springer, 2008. p.2-4, 183-189. (International Series in Operations Research & Management Science).

MATERIAL REQUIREMENT PLANNING. **Wikipédia**: a enciclopédia livre. [s.l]. Disponível em: <http://pt.wikipedia.org/wiki/Material_Requirement_Planning>. Acesso em: 01 mar. 2013.

MOLINA, Caroline Cristina; RESENDE, João batista. Atividades do planejamento e controle da produção (PCP). **Revista científica eletrônica de administração**, Garça, SP, v.6, n.11, 5p, Dez. 2006. Disponível em: <<http://www.revista.inf.br/adm10/pages/artigos/ADM-edic11-anovi-art01.pdf>>. Acesso em: 22 mai. 2013.

OLIVEIRA, Paulo Roberto. **Introdução à programação não linear**, Rio de Janeiro: Encontro de Algoritmos e Otimização, 1989. 85 p.

O PCP NA INDÚSTRIA AUTOMOTIVA. Speed blog. 15 Nov. 2012. Disponível em: <<http://speedb.blogspot.com.br/2012/11/o-pcp-na-industria-automotiva.html>>. Acesso em: 17 mar. 2013.

ÖZDAMAR, Linet. A genetic algorithm approach to a general category project scheduling problem. In: **IEEE transactions on systems, man, and cybernetics-part c: applications and reviews**, NJ, USA, 1999, v.29, n.1, p.49-59, Feb, 1999.

PLANEJAMENTO E CONTROLE DA PRODUÇÃO. **Wikipédia**: a enciclopédia livre. [s.l]. Disponível em: <http://pt.wikipedia.org/wiki/Planejamento_e_controle_da_produção>. mai 2013.

PIRES, Marília. **Programação matemática**. Universidade de Alga, 2005/06. Disponível em: <<http://w3.ualg.pt/~mpires/PMtexto.pdf>>. Acesso em: 22 mai. 2013.

REDUSINO, Augusto Cesar E. **Aplicações de Algoritmos genéticos**. Faculdade Salesiana Maria Auxiliadora, Macaé. Disponível em:

<http://www.fsma.edu.br/si/edicao3/aplicacoes_de_alg_geneticos.pdf>. Acesso em: 22 mai. 2013.

REIS, Joaquim. **Uma introdução ao *scheduling***. Instituto Superior de Ciências do Trabalho e da Empresa, Departamento de Ciências e Tecnologias da Informação, 1996. Disponível em: <<https://repositorio-iul.iscte.pt/bitstream/10071/169/1/SCH-INT.pdf>>. Acesso em: 22 mai. 2013.

SOARES, Gustavo Luís. **Algoritmos genéticos: estudo, novas técnicas e aplicações**. 1997. 145f. Dissertação(Mestrado em Engenharia Elétrica)-Universidade Federal de Minas Gerais, programa de pós graduação em engenharia elétrica, Belo Horizonte.

SILVA, Gabriel P. **Arquitetura de Computadores II: Pipeline**. Disponível em: <<http://equipe.nce.ufrj.br/gabriel/arqcomp2/Pipeline.pdf>>. Acesso em 23 nov. 2013.

SOUSA, Carlos Pimentel. **Algoritmos genéticos**. Fortaleza. Disponível em: <http://www.deti.ufc.br/~pimentel/disciplinas/ica_files/Documentos/Algorimos_Geneticos.pdf>. Acesso em: 22 mai. 2013.

STEBEL, Sérgio Leandro. **Técnicas de otimização aplicadas em problemas de *scheduling* de recursos de estocagem**. 2006. 155f. Tese(Doutorado) – Universidade Tecnológica Federal do Paraná, Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, Curitiba.

ZUBEN, Von. **Algoritmos Genéticos**. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia707_01/topico6_01.pdf>. Acesso em 23 nov. 2013