UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

PROGRAMA DE MESTRADO EM LOGÍSTICA E PESQUISA OPERACIONAL

THIAGO COSTA HOLANDA

O RELACIONAMENTO DO PROBLEMA DE SEQUENCIAMENTO CLÁSSICO COM O PROBLEMA DO CAIXEIRO VIAJANTE E SUA RESOLUÇÃO NUMA ABORDAGEM EVOLUTIVA.

FORTALEZA – CE

2015

THIAGO COSTA HOLANDA

O RELACIONAMENTO DO PROBLEMA DE SEQUENCIAMENTO CLÁSSICO COM O PROBLEMA DO CAIXEIRO VIAJANTE E SUA RESOLUÇÃO NUMA ABORDAGEM EVOLUTIVA.

Dissertação apresentada ao Programa de Mestrado em Logística e Pesquisa Operacional da Universidade Federal do Ceará, como requisito parcial para a obtenção do título de Mestre em Ciências (M.Sc.) em Logística e Pesquisa Operacional.

Orientador: Prof. Dr. Jose Lassance de Castro e Silva

FORTALEZA – CE

2015

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca de Pós Graduação em Engenharia

H669r Holanda, Thiago Costa.
O Relacionamento do problema de sequenciamento clássico com o problema do caixeiro viajante e sua resolução numa abordagem evolutiva / Thiago Costa Holanda. – 2015.
70 f. : il. color., enc. ; 30 cm.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós – Graduação em Logística e Pesquisa Operacional, Fortaleza, 2015.
Área de Concentração: Pesquisa Operacional.
Orientação: Prof. Dr. José Lassance de Castro e Silva.

1. Logística. 2. Algoritmo genético. 3. Abordagem evolutiva. 4. Problemas de agendamento. I. Título.

CDD 658.78

O RELACIONAMENTO DO PROBLEMA DE SEQUENCIAMENTO CLÁSSICO COM O PROBLEMA DO CAIXEIRO VIAJANTE E SUA RESOLUÇÃO NUMA ABORDAGEM EVOLUTIVA.

Thiago Costa Holanda

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE MESTRADO EM LOGÍSTICA E PESQUISA OPERACIONAL DA UNIVERSIDADE FEDERAL DO CEARÁ COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM LOGÍSTICA E PESQUISA OPERACIONAL.

Aprovado em 21/09/2015.

BANCA EXAMINADORA

Prof. Dr. José Lassance de Castro Silva (Orientador)
Universidade Federal do Ceará - UFC

Prof. Dr. Maxweel Veras Rodrigues
Universidade Federal do Ceará - UFC

Prof. Dr. Gerardo Valdísio Rodrigues Viana
Universidade Estadual do Ceará - UECE

AGRADECIMENTOS

A Deus, meu sustentador e fonte de todas as forças que há em mim.

Aos meus pais, pelo amor e incentivo que sempre me dedicaram, ensinando-me sempre através do exemplo.

A minha esposa, pela compreensão e devoção aos meus projetos, reconhecendo minhas potencialidades e sendo paciente com minhas falhas

Ao meu irmão, Felipe Holanda, pelo exemplo de filho, aluno, profissional e cristão, que me motivam a ser alguém melhor.

Ao Prof. José Lassance, pela valiosa orientação e ótimos conselhos acadêmicos ao longo do curso.

Aos meus preciosos amigos, em especial ao Jardel Menezes, que percorrem comigo a caminhada da vida, me ajudando e incentivando.

A todos os professores e servidores do GESLOG, pelo trabalho desenvolvido em prol da valorização de educação superior.

DEDICATÓRIA

Dedico este trabalho à minha querida igreja, lugar onde Deus me deu sonhos e transformação.

RESUMO

A resolução de um Problema de Sequenciamento sempre é uma operação que demanda grandes recursos, devido ao grande volume de dados inerentes a formulação do problema. O uso bem sucedido do Algoritmo Genético quando aplicado ao Problema de Sequenciamento Clássico deu-se através dos experimentos computacionais encontrados na literatura. O objetivo geral deste trabalho é relacionar as similaridades do Problema de Sequenciamento Clássico como um Problema do Caixeiro Viajante e resolvê-lo utilizando a metaheurística Algoritmo Genético. Foram realizados experimentos computacionais utilizando as instâncias da OR-Library (Beasley, 1990), conjunto de dados de Taillard (1993). A análise das soluções obtidas por operadores genéticos foram realizadas, com o intuito de mostrar a evolução da busca. O método proposto foi comparado com outros métodos discretos, onde constata-se o bom desempenho do Algoritmo Genético, apresentando melhores resultados em 69 das 90 instâncias testadas.

Palavras-chave: Algoritmo Genético, Problema de Sequenciamento, Problema do Caixeiro Viajante.

ABSTRACT

The resolution of a Flow Shop Problem is always an operation which requires great resources, due to the large volume of data inherent in the problem formulation. The successful use of Genetic Algorithm when applied to the Classic FSP took place through computational experiments found in the literature. The aim of this work is to relate the similarities of the Classic Scheduling Problem as a Traveling Salesman Problem (TSP) and solve it using the Genetic Algorithm metaheuristic. Computational experiments were performed using the OR-Library instances (Beasley, 1990), dataset of Taillard (1993). The analysis of the solutions obtained by genetic operators were carried out in order to show the progress of the search. The proposed method was compared with other discrete methods where there is evidence of the good performance of Genetic Algorithm.

Keywords: Genetic Algorithm, Scheduling Problem, Travelling Salesman Problem.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AG	Algoritmo Genético
FSP	*Flow Shop Problem*
PCV	Problema do Caixeiro Viajante
PS	Problema de Sequenciamento
POCP	Problema de Otimização Combinatória Permutacional

SUMÁRIO

CAPÍTULO I

INTRODUÇÃO

Este capítulo está dividido em cinco seções, sendo a primeira seção uma apresentação da justificativa e relevância do trabalho. Na segunda seção, é discutida a problemática a ser estudada. A terceira seção aborda os objetivos do trabalho desenvolvido. A quarta seção contém a metodologia abordada no trabalho e na última seção é apresentada a estrutura e a organização do trabalho.

1.1 Justificativa e relevância do trabalho

As empresas de manufatura enfrentam a difícil tarefa de determinar a melhor sequência de processamento de seus produtos em suas máquinas que atenda aos objetivos competitivos do negócio. A Pesquisa Operacional denomina este problema como *Scheduling Problem* (SP), na literatura, e o define como: dado um conjunto de tarefas e um conjunto de máquinas, determinar uma sequência específica de tarefas que otimize uma função objetivo.

Existem vários tipos de SP, por exemplo, o S*ingle Machine Scheduling Problem*, *Multiple Machine Scheduling Problem* e *Man power Scheduling Problem*. Este trabalho trata do *Multiple Machine Scheduling Problem* chamado de *Flowshop Scheduling Problem* (FSP). O primeiro artigo publicado sobre este problema foi de Johnson (1954) que formulou e resolveu o *Two-machine Flowshop Problem*. Segundo Gupta e Stafford Jr. (2006), de 1954 a 2004 mais de 1.200 artigos foram publicados abordando diferentes aspectos do FSP.

O FSP é definido como um fluxo unidirecional de n tarefas em m máquinas, a ordem de processamento de todas as tarefas nas m máquinas é a mesma. Considerando o caso geral do FSP, o número de sequências possíveis e distintas é igual a $(n!)\times m$, mesmo para problemas com n e m pequenos a enumeração completa de todas as soluções possíveis e distintas torna-se impraticável.

A metaheurística Algoritmo Genético (AG), baseada na evolução das espécies, tem sido aplicada com sucesso no FSP (Chen *et al*. (1995), Reeves (1995), Murata *et al.* (1996) e Ruiz *et al.* (2006)). Ruiz *et al.* (2006) desenvolveram um AG que teve um bom desempenho quando aplicado no FSP. Testar um algoritmo em problemas conhecidos e disponíveis na literatura é uma forma de permitir que o método possa ser comparado com outros métodos.

A primeira justificativa para a escolha do AG é a possibilidade de mostrar que ele pode ser um bom método de resolução quanto ao uso de recursos computacionais. A segunda justificativa é baseada na hipótese de Silva e Soma (2006) que métodos de resolução exata para problemas da classe FSP geralmente só são aplicados em problemas com até n=20 e que mesmo assim o tempo computacional ainda é muito alto, por isso a importância de desenvolver métodos que encontrem boas soluções em tempo computacional aceitável. A terceira justificativa é que se trata de uma técnica generalista, pode ser aplicada em vários problemas necessitando somente de poucas modificações. E finalmente, a quarta justificativa é o fato de que o AG está sendo usado no setor produtivo, onde se constatou que em setembro de 1998 através do site Evole, especializado em notícias relacionadas à computação evolucionária, noticiou que em 1997 uma empresa de manufatura foi comprada por US$ 53 milhões por uma empresa de *software*, o alto valor pago foi justificado pelo interesse em adquirir um programa de computador baseado em AG desenvolvido pela empresa de manufatura para fazer o sequenciamento das ordens de produção da fábrica (EvoWeb, 2007).

1.2 Problemática

A resolução de um Problema de Sequenciamento sempre é uma operação que demanda grandes recursos, devido ao grande volume de dados inerentes à formulação do problema. Wall (1996) afirma que existem, principalmente, três aspectos nesse contexto:

a) O tamanho do problema: tentativas de se “restringir” o espaço de busca podem ser realizadas e muitos métodos têm sido concebidos nesse sentido, contudo para que esses métodos tenham bom aproveitamento, há uma

demanda de certa quantidade de informações prévias acerca do problema que, nem sempre, estão disponíveis.

b) Incertezas e a natureza dinâmica dos problemas reais: os distúrbios são inevitáveis e podem requerer apenas a substituição de um único recurso, ou eles podem exigir a completa reformulação do plano. As técnicas de otimização precisam estar aptas a enfrentar essas mudanças.
c) Inviabilidade: quando o problema não tem solução. Isso se dá, por vezes, quando as tarefas só podem ser executadas em espaços de tempo muito restringidos. Existem algoritmos capazes de determinar se existe essa inviabilidade.

O mesmo autor, após executar uma série de testes com vários problemas, verificou que o AG geralmente não apresenta bom desempenho quando aplicado à classe de problemas do tipo sequenciamento, devido à própria estrutura do problema e o uso dos tempos relativos, ou seja, a modificação de um único valor afeta todas as tarefas sucessivas. A proposta de trabalhar com um AG moderno não é desmistificar esta observação, tendo em vista que os AGs tiveram bom desempenho quando aplicados ao Problema do Caixeiro Viajante (PCV), e a relação dele com o FSP. Outra, os algoritmos genéticos de Reeves e Chen quando aplicados ao FSP mostraram que esta afirmação não é verdadeira.

Portanto, o problema que esse trabalho visa a tratar tem o seguinte enunciado: é possível desenvolver um Algoritmo Genético para resolução do Problema de Sequenciamento que obtenha bons resultados via Problema do Caixeiro Viajante?

1.3 Objetivos: Geral e Específicos

O objetivo geral deste trabalho é relacionar as similaridades do Problema de Sequenciamento Clássico como um Problema do Caixeiro Viajante e resolvê-lo utilizando a metaheurística Algoritmo Genético.

Como objetivos específicos, têm-se:

a) Desenvolver, implementar, testar e validar um algoritmo genético moderno aplicado no FSP via PCV;
b) Descrever as conclusões e descrição do trabalho científico.

1.4 Metodologia

De acordo com Almeida (2011), um estudo só pode ser considerado científico se são adotados métodos para sua realização. Aqui, a pesquisa será caracterizada do seu ponto de vista metodológico.

Quanto a sua natureza, nossa pesquisa se caracteriza como aplicada, pois fazemos uso de conhecimentos que já tem sido sistematizados por diversos autores com o objetivo de solucionar problemas organizacionais da sociedade. Quanto aos objetivo, nossa pesquisa é descritiva, onde procuramos relatar o objeto de estudo, bem como suas características e os problemas relacionados ao mesmo.

Nossa abordagem é quantitativa, pois o estudo caracteriza-se pelo uso de inúmeras ferramentas matemáticas e estatísticas para o tratamento dos dados, visando medir a relação existente entre as variáveis, que por sua vez são previamente estabelecidas, assim como as hipóteses. Em relação aos procedimentos, a pesquisa deste trabalho é experimental, onde relacionamos como as variáveis se relacionam entre si dentro do ambiente de estudo. (GODOY, 1995, p. 62)

Para a realização deste trabalho foi cumprido um conjunto de ações estratégicas, a saber:

a) Realizar uma revisão bibliográfica sobre os assuntos envolvidos;
b) Desenvolver, implementar e testar a técnica de solução para o problema;
c) Pesquisar e implementar a metaheurística AG nas instâncias mais significativas encontradas na literatura.
d) Comparar os resultados.
e) Escrever a versão final da dissertação.

1.5 Estrutura e organização do trabalho

Esta dissertação está composta por sete capítulos organizados da seguinte forma: no capítulo II, contém a fundamentação teórica a respeito das características do Problema de Sequenciamento, suas definições, modelo matemático e estado da arte, bem como uma introdução aos AGs já utilizados na tentativa de solução deste problema; O Capítulo III apresenta as principais características do Problema do Caixeiro Viajante; O Capítulo IV mostra os conceitos mais importantes referentes à metaheurística do AG; enquanto no Capítulo V é apresentado o modelo de AG que foi utilizado neste trabalho; Já no Capítulo VI, os resultados dos testes realizados e o desempenho dos algoritmos proposto na resolução do problema são abordados com destaque para o desempenho das soluções apresentadas e tempo de execução; Concluindo a dissertação, as considerações finais e sugestões para trabalhos futuros são apresentadas no Capítulo VII.

CAPÍTULO II

O PROBLEMA DE SEQUENCIAMENTO

Neste capítulo, a definição e a modelagem matemática do FSP são apresentadas na primeira e segunda seção, respectivamente. Na terceira seção, um método de resolução do FSP é apresentado dentro da otimização combinatória permutacional. O estado da arte do FSP está descrito na quarta seção.

2.1 Definição do FSP

Antes de definir o FSP é importante esclarecer que *flowshop* não é sinônimo de linha de montagem, mesmo que a característica do *flowshop* seja um fluxo que pareça ser constante de trabalhos através de um conjunto de máquinas em série, conforme Gupta e Stafford Jr.(2006). A seguir são apresentadas três diferenças entre estes dois tipos de modelo de sistema de produção.

a) No ambiente *flowshop*, existe uma variedade de produtos e na linha de montagem existe um produto padrão;

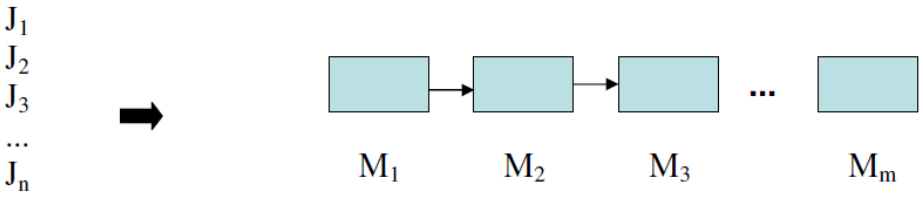
b) No ambiente *flowshop*, as tarefas não são obrigadas a passarem em todas as máquinas dependendo das necessidades tecnológicas e na linha de montagem todas as tarefas têm que passar por todas as estações de trabalho; e

c) No ambiente *flowshop*, cada tarefa tem seu próprio tempo de processamento em cada máquina e na linha de montagem todas as unidades dos produtos têm o mesmo tempo padrão em cada estação de trabalho.

O FSP pode ser definido como sendo: um conjunto de *n* tarefas J_1, J_2,..., J_n, onde cada tarefa deve ser processada em *m* máquinas M_1, M_2, ..., M_m. Cada tarefa demanda *m* operações, com uma operação representando o tempo de processamento da tarefa por máquina. As tarefas seguem o mesmo fluxo de operações nas máquinas, i.e., para qualquer j =1, 2, ..., *n*, a tarefa J_j deve ser processada primeira na máquina M_1, depois na máquina M_2, e assim por diante até a última máquina, no caso a máquina M_m, conforme mostra a Figura 2.1. Caso a tarefa J_j não utilize todas as máquinas, o seu fluxo

continua sendo o mesmo, todavia com o tempo de operação sendo igual a zero. Uma máquina pode processar somente uma operação de cada vez, e iniciada uma operação, ela deve ser processada até a sua conclusão. O problema consiste em realizar todas as *n* tarefas no menor tempo possível (Silva e Soma, 2006).

Figura 2.1 – Ilustração do FSP



Fonte: Silva e Soma (2006)

Conforme Silva e Soma (2006) um *input* do FSP é dado por *n*, *m* e uma matriz P(*n* × *m*) de elementos não negativos, onde P_{ij} denota o tempo de processamento da tarefa J_j na máquina M_i. Seguindo os 4 parâmetros da notação A/B/C/D adotada por Conway *et al.*(1967), o problema é classificado como *n*/m/P/Fmax. Na menos antiga notação paramétrica V/W/Y, proposta por Graham *et al.* (1979), o problema é denotado como sendo F/prmu/Cmax. Estas notações são usadas e solicitadas para serem descritas a fim de não deixar dúvida na definição correta do problema de sequenciamento que está sendo abordado.

O FSP pertence à classe dos problemas NP-completo, no sentido forte, para valor de *m* maior que 3, conforme Garey e Johnson (1979). No caso em que *m* é menor ou igual a 3, o problema pode ser solucionado em tempo polinomial.

Suposições relacionadas às tarefas, conforme Gupta e Stafford Jr.(2006), são:

T1– Cada tarefa é liberada para a fábrica no começo do período de programação.

T2– Cada tarefa pode ter sua própria data de entrega fixa e não sujeita a mudança.

T3– Cada tarefa é independente das demais.

T4– Cada tarefa consiste de operações específicas que são realizadas por somente uma máquina.

T5– Cada tarefa tem uma sequência tecnológica preestabelecida fixa e que é igual para todas as demais tarefas.

T6– Cada operação de uma tarefa requer um tempo de processamento finito e conhecido para ser processada nas várias máquinas. Nesse tempo de processamento estão incluídos tempos de transporte, *setup* e outros. O tempo de processamento é independente dos tempos de processamento das tarefas anteriores e posteriores.

T7– Cada tarefa é processada uma única vez em cada máquina.

T8– Cada tarefa pode esperar entre máquinas consecutivas, ou seja, estoque em processo é permitido.

Suposições em relação às máquinas:

M1– Cada setor é composto de somente uma máquina e a fábrica tem somente uma máquina de cada tipo.

M2– Cada máquina está inicialmente desocupada no início do período de programação.

M3– Cada máquina na fábrica opera independentemente das outras e, por isso, pode operar na taxa de produção máxima.

M4– Cada máquina só pode processar uma tarefa por vez.

M5– Cada máquina está continuamente disponível para processar tarefas durante período de programação e não há interrupções devido a quebras, manutenção ou outras causas.

Suposições relacionadas às políticas de operação

O1– Cada tarefa é processada tão logo seja possível. Por isso, não há intenção de fazer atarefa ficar esperando ou fazer a máquina ficar ociosa.

O2– Cada tarefa é considerada uma entidade individual mesmo que possa ser composta por um conjunto de unidades.

O3– Cada tarefa uma vez iniciada é processada até o fim, ou seja, o cancelamento de tarefas não é permitido.

O4– Cada operação de uma tarefa uma vez iniciada numa máquina é completada antes que outra tarefa possa começar na mesma máquina, ou seja, nenhuma preempção é permitida.

O5– Cada máquina processa as tarefas na mesma ordem.

É importante conhecer estas suposições já que os outros problemas da classe FSP foram criados a partir de alguma modificação nelas. Este é o caso do S*equence Dependent Setup Time Flowshop Scheduling Problem* que foi criado a partir da alteração da suposição T6 que deixa de considerar o tempo de *setup* como fazendo parte do tempo de processamento e passa a considerar os dois tempos separadamente.

2.2 Um Modelo Matemático para o FSP

A seguir é apresentada a notação necessária e um modelo matemático em Programação Linear Inteira e Mista para o FSP, elaborado por Gupta e Stafford Jr.(2006).

Notação:

a) T_{ij} é uma variável do modelo que representa o tempo para iniciar o processamento da tarefa *j* na máquina *i*, com $1 \leq j \leq n$ e $1 \leq i \leq m$;

b) *W* é um número bastante grande;

c) X_{ijk}, é uma variável binária definida por:

X_{ijk}= 1, se a tarefa *j* precede a tarefa *k* na máquina *i*.

0, caso contrário.

com $i = 1, 2, 3, ..., m$, $j = 1, 2, 3,..., (n-1)$ e $k = (j+1), (j+2),..., n$.

Observações:

i) São parâmetros do problema: *n*, *m* e a matriz *P*;

ii) Em (c), *j* e *k* não variam de 1 até *n*, porque iriam comparar as tarefas *j* e *k* na mesma máquina duas vezes; e

iii) As variáveis do tipo X_{ijk} estabelecem a sequência de processamento das tarefas em cada uma das *m* máquinas.

$$\text{Minimizar } Z = \sum_{j=1}^{n} T_{mj} \qquad 2.1$$

sujeito a:

$$T_{rj} \geq T_{r-1,j} + p_{r-1,j}\,, \ \forall \ r = 2,3,\ldots,m \text{ e } j = 1, 2,\ldots, n. \qquad 2.2$$

$$\left.\begin{array}{l} T_{ik} \geq T_{ij} + p_{ij} - W * (1 - X_{ijk}) \\ T_{ij} \geq T_{ik} + p_{ik} - W * X_{ijk} \\ x_{ijk} \in \{0,1\} \end{array}\right\} \quad \forall \ 1 \leq i \leq m, j = 1, 2,\ldots, n\text{-}1 \text{ e } k = j\text{+}1, j\text{+}2,\ldots n \qquad \begin{array}{l} 2.3 \\ 2.4 \\ 2.5 \end{array}$$

$$T_{ij} \geq 0, \ \forall \ 1 \leq i \leq m \text{ e } 1 \leq j \leq n \qquad 2.6$$

Onde:

a) A função objetivo 2.1 minimiza a soma para a conclusão das *n* tarefas nas *m* máquinas através da obtenção do menor tempo para iniciar cada tarefa na última máquina. Para o caso da tarefa *j* não utilizar a máquina *m* substitui-se T_{mj} por T_{rj}, onde *r* é a última máquina a ser utilizada pela tarefa *j* com tempo maior de zero;

b) O grupo de restrições 2.2 representa o fluxo que as tarefas devem seguir para serem concluídas. Cada equação do tipo 2.2 determina que a (*i*+1)-*ésima* operação da tarefa *j* não pode iniciar até que a *i-ésima* operação da tarefa *j* na máquina *i* seja concluída.

c) Os grupos de restrições 2.3 e 2.4 comparam a relação de precedência das *n* tarefas nas *m* máquinas. Estes grupos de restrições também não permitem que uma máquina processe duas tarefas ao mesmo tempo;

d) As restrições do grupo 2.5 representam o atendimento à definição da variável X_{ijk} como sendo binária;

e) As restrições do grupo 2.6 definem a não-negatividade dos tempos de processamento;

f) O valor de W faz com que uma das restrições do grupo 2.3 se torne coerente com a definição da variável X_{ijk} da seguinte forma:

i) Se $X_{ijk} = 1$, então 2.3 fica $T_{ik} \geq T_{ij} + P_{ij}$, coerente com a definição de X_{ijk}, enquanto 2.4 fica $T_{ij} \geq T_{ik} + P_{ik} - W$, ou seja, T_{ij} maior ou igual que um número negativo, logo $T_{ij} \geq T_{ik} + P_{ik} - W$ se torna verdadeiro devido ao grupo de restrição 2.6.

ii) Se $X_{ijk} = 0$, então 2.4 fica $T_{ij} \geq T_{ik} + P_{ik}$, coerente com a definição de X_{ijk}, enquanto 2.3 fica $T_{ik} \geq T_{ij} + P_{ij} - W$, ou seja, T_{ik} maior ou igual que um número negativo, logo $T_{ik} \geq T_{ij} + P_{ij} - W$ se torna verdadeiro devido ao grupo de restrições 2.6.

iii) Podemos adotar W como sendo $1000 \times \text{Max}\ \{P_{ij}\}$, $1 \leq i \leq m$ e $1 \leq j \leq n$.

Complexidade do modelo:

a) Variáveis do tipo X_{ijk}: $mn\ (n - 1) / 2$;

b) Variáveis do tipo T_{ij}: mn;

c) Restrições do grupo 2.2: $n(m - 1)$;

d) Restrições do grupo 2.3 e 2.4: $mn(n - 1)$;

e) Número total de variáveis: $mn(n + 1) / 2$; e

f) Número total de restrições: $n(mn - 1)$.

2.3 Um Modelo Combinatorial Permutacional do FSP

Um POCP pode ser definido por um terno (S, g, n), onde S é o conjunto de todas as soluções do problema, g é sua função ou procedimento que aplica a cada solução viável $s \in S$ um número real e n é uma instância do problema. O número de

soluções existentes para um POCP é representado por |S| (cardinalidade de *S*) e igual a *n*!. O objetivo é encontrar uma solução $s^* \in S$ que otimize a um dado critério de desempenho representado pela função *g*. Representa-se *s* como uma permutação de *n* elementos, ou seja, $s = \langle a_1, a_2, ..., a_n \rangle$, com $a_i \neq a_j$, tal que $1 \leq i, j \leq n$ e $i \neq j$.

Segundo Silva e Soma (2006) o FSP pode ser modelado como um POCP da seguinte forma:

a) Um elemento $s = \langle J_1, J_2, ..., J_n \rangle$ do conjunto de soluções viáveis *S* é representado por uma permutação das *n* tarefas, com a ordem de *s* determinando a sequência na qual as tarefas serão processadas; e

b) O procedimento *g*, dado a seguir, determina o valor do tempo gasto para processar as sequências, mais precisamente tem-se que ele representa o tempo utilizado no processamento da última tarefa de *s* na última máquina M_m.

Figura 2.2 – Procedimento para calcular g(s).

Entrada: *m*, *n*, permutação *s*, Matriz *P*(*m* x *n*).
Saída: *g* (tempo gasto para processar todas as *n* tarefas usando a seqüência *s*)

```
for(i=1; i<=m; i++)
  for(j=1; j<=n; j++)  t[i][j]=0;
for(i=2; i<=m; i++) t[i][ s[1] ]=t[i-1][ s[1] ]+ p[i-1][ s[1] ];
for(j=2; j<=n; j++)
  for(i=1; i<=m; i++)
  {  if (i==1) {  t[1][ s[j] ]=t[1][ s[j-1] ]+p[1][ s[j-1] ];  }
     else {
       y=t[i][ s[j-1] ]+p[i][ s[j-1] ];   /* tempo da tarefa anterior na maquina atual */
       x=t[i-1][ s[j] ]+p[i-1][ s[j] ];   /* tempo da maquina anterior, para tarefa atual */
       if (x>=y) t[i][ s[j] ]=x;  else  t[i][ s[j] ]=y;
     }
  }
g=t[m][ s[n] ] + p[m][ s[n] ];
```

* O elemento t_{ij} representa o tempo para iniciar a tarefa J_j na máquina M_i.

Fonte: Silva e Soma (2006)

Para determinar a sequência *s* com menor valor de *g* seria necessário enumerar e avaliar todas as *n*! sequências distintas de *S*. A Tabela 2.1 abaixo mostra alguns

valores de *n*, a quantidade de soluções distintas de *S* e o tempo computacional, caso o tempo de processamento do procedimento *g* para cada sequência fosse igual 0,001 segundos.

TABELA 2.1 – Tempo de processamento para valores de *n* dada a cardinalidade de *S*.

n	Quant. Soluções	Tempo Computacional
5	1,20 x 10^2	0,12 seg
10	3,63 x 10^6	3.628,80 seg
20	2,43 x 10^{18}	7,71 x 10^7 anos

Fonte: Silva e Soma (2006)

Os resultados da Tabela 2.1 demonstram que para valores de *n* maiores que *20* fica inviabilizada a enumeração completa de todas as soluções. Um exemplo da aplicação do problema é dado a seguir.

Exemplo: Procura-se minimizar o tempo de processamento de 3 tarefas a serem processadas em 2 máquinas, sendo que todas as tarefas iniciam primeiro na máquina 1 e depois na máquina 2. Os tempos de processamento das tarefas nas máquinas, dados em horas, são apresentados na Tabela 2.2 abaixo.

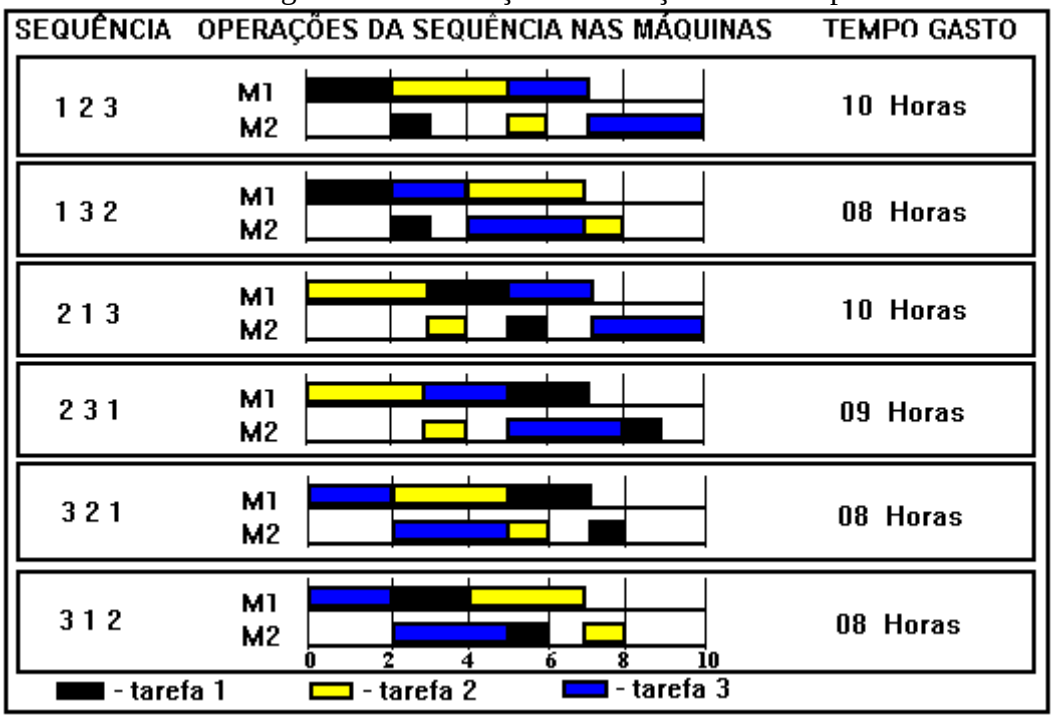
Tabela 2.2 - Tempo de operacionalização das tarefas nas máquinas, dado em horas.

	Tarefa 1	**Tarefa 2**	**Tarefa 3**
Máquina 1	02	03	02
Máquina 2	01	01	03

Fonte: Silva e Soma (2006)

Quer-se então determinar qual a sequência de processamento das 3 tarefas nas 2 máquinas com o menor tempo possível. Com este objetivo, a medida de desempenho usado neste caso é denominado, na literatura, de *make span*.

Figura 2.3 – Ilustração das soluções do exemplo 01.



Fonte: Silva e Soma (2006).

Existem somente 6 tipos distintos de ordenar as tarefas nas máquinas: 123, 132, 213, 231, 312, e 321. A Figura 2.3, dada anteriormente, mostra o tempo gasto por cada uma das sequências (soluções do problema) e as devidas alocações de processamento nas 2 máquinas. Conclui-se que as sequências 132, 312 e 321 são as soluções ótimas do problema, com um tempo gasto de 8 horas para processar as 3 tarefas. Se uma determinada técnica de resolução aproximada do problema tivesse apresentado a sequência 123, com tempo gasto de 10 horas, haveria um desvio de 25% da solução ótima. Isto equivale a duas horas a mais das sequências ótimas.

2.4 Estado da Arte do FSP

A análise do histórico do progresso das técnicas utilizadas na resolução dos problemas da classe FSP serve para situar o método proposto neste trabalho. Gupta e Stafford Jr. (2006) analisaram o desenvolvimento da pesquisa em relação ao FSP desde o trabalho de Johnson (1954) até 2004. Esse período foi dividido em cinco décadas (1955-1964, 1965-1974, 1974-1984, 1985-1994 e 1995-2004) e para cada período as

suposições, as formulações para o problema e as abordagens de solução foram analisadas.

A primeira década tratou o FSP principalmente do ponto de vista teórico. Além da formulação de Johnson (1954) para duas máquinas foi desenvolvido o *m-machine flowshop* para a minimização do *make span*. Foram desenvolvidas poucas técnicas para a solução do FSP. As duas técnicas que mais se destacaram foram a programação matemática (Wagner, 1959; Manne, 1960) e a simulação de Monte Carlo (Sisson, 1959; Muth e Thompson, 1963). O tamanho dos problemas resolvidos era pequeno por três motivos: i) falta de capacidade computacional; ii) falta de eficientes programas de computador; e iii) a maioria das variações do *Two-machine Flowshop Problem* era NP-*hard*.

A segunda década apresentou um maior número de técnicas de solução e outras funções objetivo além do *makespan*. Os primeiros a proporem a abordagem combinatorial foram Dudek e Teuton (1964). A técnica *branch and bound* para o FSP foi desenvolvida por Lomnicki (1965). Nessa época, também começou o desenvolvimento das primeiras heurísticas para encontrar boas soluções para o FSP de grandes dimensões.

Na terceira década,com a publicação da teoria do NP-*Completude* por Garey e Johnson (1979), a pesquisa em relação ao FSP passou a ter duas direções. Uma direção na tentativa de identificar a complexidade de vários FSP (Brucker, 1998; Lawler*et al.* 1993) e a outra no desenvolvimento de novas heurísticas. Nessa década, também ocorreu a proposição de vários novos FSP como o que separa o tempo de *setup* do tempo de processamento, que considera a data de entrega na função objetivo e que considera o tempo de processamento estocástico.

Na quarta década, surgiu o *hybrid flowshop* que consiste em cada centro de trabalho pode ser constituído de múltiplas máquinas em paralelo. Nessa década, iniciou-se o uso das metaheurísticas (Aarts e Lenstra, 1997): *Tabu Search*; *Simulated Annealing;* e Algoritmo Genético. Também foram desenvolvidas técnicas baseadas em inteligência artificial, sistemas de apoio à decisão e sistemas especialistas (sistemas que

utilizam o conhecimento empírico acumulado da resolução de problemas que já ocorreram para ajudar a resolução de novos problemas).

Na quinta década, continuou o crescimento na criação de novos problemas, funções objetivo e abordagens de resolução. A principal novidade foi o aumento das pesquisas considerando funções multiobjetivo (T'Kindt e Billaut, 1993).

Portanto, percebe-se que esse problema tem sido alvo de muita pesquisa na academia, ao longo das últimas décadas e a sua resolução tem sido proposta sob a ótica de diversas heurísticas. No próximo capítulo, iremos estudar um outro problema clássico da literatura, que será base metodológica do presente trabalho, e, depois, far-se-á a relação entre os dois problemas.

CAPÍTULO III

O PROBLEMA DO CAIXEIRO VIAJANTE

Neste capítulo, a definição e a modelagem matemática do Problema do Caixeiro Viajante são apresentadas na primeira seção. A segunda seção mostra a complexidade do Problema do Caixeiro Viajante. Algumas variações do problema são apresentadas na terceira seção, enquanto na quarta seção são descritos alguns métodos de resolução do problema.

3.1 Definição do Problema do Caixeiro Viajante

No PCV, tem-se um conjunto com *n* cidades $c_1, c_2, \dots, c_n$ e todas as distâncias (ou custo) entre elas, onde d_{ij} representa a distância (ou custo) para "*ir*" da cidade c_i até a cidade c_j. O objetivo do problema consiste em determinar uma rota, para o caixeiro, que percorra todas as *n* cidades somente uma vez com a menor distância percorrida. O PCV pode ser modelado como um POCP, *P* = (*S*, *g*, *n*), de acordo com a definição dada no capítulo anterior, da seguinte forma:

a) Um elemento *s*=<$a_1, a_2, \dots, a_n$> do conjunto de soluções viáveis *S* é representado por uma permutação das *n* cidades, com a ordem de *s* determinando a sequência (rota) na qual as cidades serão visitadas;

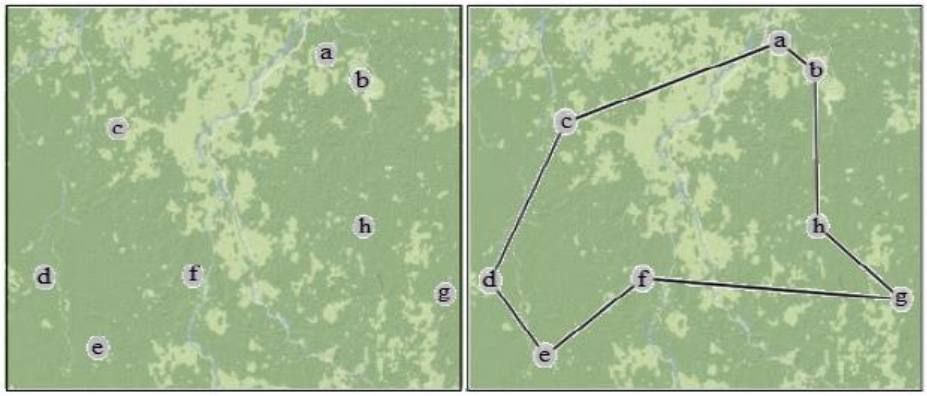
b) A função *g* que calcula a distância total percorrida numa sequência *s* é dada por

$$g(s) = (\sum_{i=1}^{n-1} d_{C_i,C_{i+1}}) + d_{C_n,C_1}$$

Outra formulação do PCV ocorre na da Teoria dos Grafos, onde são definidos dois conjuntos, um composto por elementos chamados de vértices e outro composto por uma coleção de relações binária de pares entre eles, denominados de arestas. No caso,

seja *G = (N, A, D)* um grafo onde *N* é o conjunto de vértices do grafo que representam as localidades a serem visitadas, *A* é o conjunto de arestas que representam as possibilidades de composição de uma rota e *D* é uma matriz completa composta por pesos não negativos associados aos arcos, que podem representar, por exemplo, distância ou tempo gasto para ir de um ponto a outro. O PCV consiste em determinar o circuito com o menor custo, passando por cada vértice uma única vez. A este circuito, denomina-se ciclo ou circuito hamiltoniano. Um exemplo de circuito hamiltoniano em um grafo pode ser visto na Figura 3.1. Se a matriz *D* é simétrica, então o PCV é classificado como Simétrico e há *(n-1)!/2* circuitos hamiltonianos possíveis distintos. Caso contrário, o PCV é dito Assimétrico e o número de circuitos hamiltonianos possíveis passa a ser *(n-1)!*.

Figura 3.1– Exemplo de ciclo hamiltoniano com *n*=7.



Fonte: SILVA (2013)

A essência do Problema do Caixeiro Viajante dentro de muitas aplicações práticas é evidente. Em todos estes casos, o custo ou a distância entre duas localidades quaisquer é conhecido, sendo o objetivo fundamental, determinar uma sequência na qual os locais especificados aparecem ordenados de acordo com o custo total incorrido para percorrê-los, passando por cada localidade uma única vez. Na prática, o PCV pode ser aplicado na definição de planos de rotas (HOFFMAN & PADBERG, 2012), na confecção de placas de circuito impresso (VITTES, 1999), na análise da estrutura de cristais (BLAND & SHALLCROSS, 1987), no suporte à rede de abastecimento de graneis líquidos, no sequenciamento de tarefas, na fabricação de chips (KORTE, 1989), no mapeamento de genoma (AVNER et al., 2001), no sequenciamento de DNA

(GONNET KOROTENSKY & BENNER, 2000), dentre outras. Neste trabalho, O PCV é abordado no contexto de roteirização, atividade que tem por fim buscar os melhores trajetos que um veículo deve fazer através de uma malha viária (BALLOU, 2001).

A visibilidade acadêmica do PCV veio por volta das décadas de 1920 e 1930, quando o problema passou a ser estudado em Viena, mais precisamente por Karl Menger e, posteriormente, em Princeton, por Hassler Whitney e MerrilFlood (MAREDIA, 2010). Foi na década de 1940, que Merril e Hassler, definiram o nome do problema como *Traveling Salesman Problem* (Problema do Caixeiro Viajante) e o divulgou para o *RAND Corporation*, que se interessou pelo problema devido o estimulo financeiro oferecido pela força aérea norte americana em perspectiva da obtenção de soluções ótimas para os problemas de transporte. A RAND, fundada em 1948, é uma organização privada, sem fins lucrativos, que ao longo dos anos dedicou-se a resolver problemas interdisciplinares e quantitativos, traduzindo conceitos teóricos de Ciências Aplicadas e Pesquisa Operacional em aplicações úteis a outras áreas como educação, saúde, sociologia, defesa, justiça e economia aplicada.

Na década de 1950, o PCV ficou popular, dado o prestígio da RAND Corporation e sua relação com outros problemas de otimização estudados naquele período. Foi, também, neste período, que Dantzig, Fulkerson e Johnson propuseram o primeiro método para resolver de forma otimizada uma instância de 49 cidades para o PCV, o que para os padrões da época, era uma instância grande. Aproveitando o sucesso sobre inúmeras aplicações do Algoritmo SIMPLEX (projetado por Dantzig em 1947), os três pesquisadores formularam o problema por meio de Programação Linear Inteira, através da seguinte forma:

$$\min Z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \cdot x_{ij} \qquad (3.1)$$

Sujeito as restrições:

$$x_{ii} = 0 \, , i = 1,2,3, \dots , n. \qquad (3.2)$$

$$x_{ij} \in \{0,1\}, \mathrm{i}, \mathrm{j} = 1,2,3 \dots, \mathrm{n}.. \qquad (3.3)$$

$$\sum_{i=1}^{n} x_{ij} = 1, j = 1, \dots , n.. \qquad (3.4)$$

$$\sum_{j=1}^{n} x_{i\,i} = 1, j = 1, \ldots, n.. \quad (3.5)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, |s \subset V : S \neq \emptyset. \quad (3.6)$$

A equação (3.1) representa a função objetivo do problema e consiste em minimizar o custo para que o caixeiro viajante percorra todos os *n* pontos distintos uma única vez e retornar ao ponto de partida inicial. As restrições do problema são dadas pelas equações (3.2) até (3.5). A equação (3.2) restringe que não é possível ir de uma cidade para ela mesma, enquanto que a equação (3.3) representa uma variável binária x_{ij} de confirmação de deslocamento, tal que se $x_{ij}=1$, o caixeiro vai diretamente da cidade *j* à cidade *i* na rota que está sendo apresentada. Em caso contrário, $x_{ij}=0$. As equações (3.4) e (3.5) determinam que o caixeiro deva passar uma única vez em cada ponto, seja na chegada, ou na partida, devendo seguir para o próximo ponto até que todo o percurso seja realizado e retorne ao ponto inicial. A Equação (3.6), evita que ocorra ciclagem menor que *n* na rota, ou seja, que sejam produzidas rotas somente de ciclos hamiltonianos.

No inicio da década de 1970, Held e Karp (1971) abordaram com sucesso o problema valendo-se de programação dinâmica e relaxação lagrangeana, otimizando uma instância com 64 cidades. Em 1977, Gröetschel (1977), utilizando métodos de programação matemática, encontrou para uma instância de 120 cidades, o percurso ótimo. Em 1987, os pesquisadores Padberg e Rinaldi (1987), atuando como funcionários da *Tektronics Incorporated*, mostraram que o problema também pode ser usado em diversas áreas e não apenas para cidades, quando realizaram a otimização de 2392 pontos de um *layout* com um algoritmo *Branch & Cut*.

3.2 Complexidade do PCV

O PCV, assim como outros problemas de otimização, possuí complexidade computacional de tal forma que todos os esforços programáveis conhecidos para resolver tais problemas crescem exponencialmente com o tamanho do problema. Em 1972, Richard M. Karp mostrou que o problema pertence a classe de problemas *NP Completo*, fornecendo assim uma explicação matemática para a complexidade em resolver de forma otimizada o PCV.

A complexidade de um problema computacional é dada pelo consumo de recursos computacionais (tempo, memória, etc.) de algoritmos para apresentar uma solução aceitável, dado um critério específico do problema. A teoria da complexidade computacional, fundamentada por Cook (1971), estuda a complexidade de problemas, classificando-os em duas classes genéricas conhecidas como *P (Polynomial time)* e *NP (Non-Deterministic Polynomial Time)*. A classe *NP* é composta por problemas onde as instâncias de médio e grande porte são intratáveis pelos algoritmos determinísticos conhecidos em tempo polinomial, devido ao consumo demasiado de recursos computacionais.

Tabela 3.1 – Explosão combinatória.

n	n!	Tempo
5	120	0,00012 segundos
10	3628800	3,62880 segundos
12	479001600	8 minutos
15	1307674368000	15 dias
20	2,43E+018	77.147 anos
50	3.041409320171337804361 2E+0064	∞
100	9.332621544394415268169 9E+0157	∞
500	1.220136825991110068791 2E+1134	∞
1000	4.023872600770937735436 2E+2567	∞

Fonte: SILVA (2013)

Sendo a função de complexidade do PCV exponencial, torna-se inviável a análise de todas as soluções possíveis mesmo para problemas de pequena complexidade, ou seja, para valores de $n>10$, conforme apresenta a Tabela 3.1, que mostra a explosão combinatorial de problemas desta natureza. O tempo estimado está calculado com base numa máquina hipotética da ordem de 1 microssegundo, para a execução das instruções de controle do programa, de acesso aos dados, cálculo das distâncias, comparações, chamadas a subprogramas, etc. (VIANA, 1998).

Garey e Johnson (1979) classificaram os problemas segundo sua complexidade em indecidíveis, tratáveis e intratáveis. Um problema é tido como indecidível se nenhum algoritmo pode ser desenvolvido para resolvê-lo. Já os problemas intratáveis são problemas decidíveis, porém difíceis para os quais possivelmente não existem algoritmos que os resolvam em tempo polinomial. Por fim, os problemas classificados

como tratáveis, são aqueles que possuem algoritmos aptos a resolvê-los em tempo polinomial. Na literatura, problemas tratáveis e intratáveis são classificados ainda como sendo de decisão, que consistem na verificação da veracidade ou não de uma determinada questão para o problema, de localização, que buscam uma estrutura que satisfaça requisitos especificados por uma questão do problema, e de otimização, que visam obter a melhor solução possível, dentre as soluções viáveis para o problema. O PCV pertence a classe de problemas de otimização, porém pode ser formulado como um problema de decisão da seguinte forma: dadas *n* cidades e a distância d_{ij} entre duas cidades *i* e *j*, e um inteiro não negativo *k*, verificar se há um roteiro para o caixeiro cujo custo seja menor que *k*.

Vários trabalhos direcionaram seus estudos na busca por um método polinomial para resolver o PCV, pois Cook (1971) e Karp (1972) mostraram que uma grande quantidade de problemas intratáveis pode ser reduzida, em tempo polinomial, ao problema do caixeiro viajante. Porém, o histórico literário mostra que grandes instâncias deste problema requerem sempre um tempo exponencial. Uma importante noção neste contexto é do subconjunto de *NP* a qual pertence o PCV, denominado *NP-Completo*. Os problemas mais difíceis da classe *NP* são atribuídos a classe *NP-Completo*, isto deve-se ao fato de que nenhum algoritmo de tempo polinomial foi descoberto para qualquer problema desta classe. Se descoberto algum algoritmo de tempo polinomial para qualquer problema *NP–Completo*, então todos os problemas desta classe poderão ser resolvidos em tempo polinomial.

3.3 Variações e Problemas Semelhantes

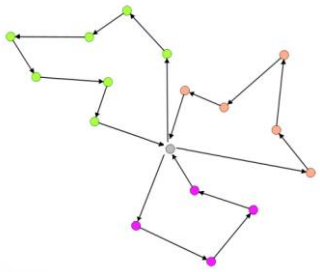
As variações clássicas do PCV encontradas na literatura foram propostas para contemplar problemas cuja formulação clássica não poderia representar de maneira satisfatória. A seguir, são apresentadas algumas destas variações:

PCV Estocástico: Esta variação foi formulada para contemplar problemas em que as localidades que se deseja visitar juntamente com os custos para visitá-las podem variar conforme os resultados obtidos durante o processo de resolução. Toriello (2012) abstraiu a aplicação prática do PCV Estocástico no roteamento dinâmico do veículos, no

qual o custo de uma decisão é conhecida apenas probabilisticamente de antemão, mas é revelado dinamicamente antes de a decisão ser executada.

PCV Múltiplo: Como o próprio nome sugere, nesta variação, múltiplos caixeiros são designados para que todos os pontos do problema sejam visitados. Logo, roteiros com o custo mínimo são configurados para que cada caixeiro viajante atenda ao menos a um ponto da rede. A Figura 3.3 ilustra um esquema básico de resolução do PCV Múltiplo. Bektas (2006) realizou uma revisão literária deste problema, enfatizando a aplicação prática desse problema juntamente com técnicas propostas para sua resolução.

Figura 3.3 – Esquema básico de resolução do PCV Múltiplo.



Fonte: Bektas (2006)

No contexto de roteirização, há problemas semelhantes ao PCV que sugiram da necessidade de se refletir, por meio da inclusão de restrições e especificações, a realidade das decisões que precisam ser tomadas diariamente por organizações ou entidades do segmento de transporte. As dificuldades de se modelar um problema de roteirização advém da grande quantidade de parâmetros que podem influenciar esse tipo de problema, logo, uma classificação adequada permite que seja implementada uma estratégia de resolução precisa. Bodin *et al.* (1983) classificaram os problemas de roteamento em *roteirização pura* e *programação de veículos.* Problemas de *roteirização pura* visam construir rotas viáveis com o menor custo possível considerando apenas aspectos espaciais. Já em problemas de *programação de veículos*, constam tanto os aspectos espaciais quanto os temporais, como restrições de horários preestabelecidos para cada atividade a ser executada. Bodin e Golden (1981) relataram que são os problemas de *programação de veículos* que normalmente ocorrem na prática. Ronen

(1988) baseou-se no ambiente operacional e nos objetivos dos problemas para distingui-los em três classes.

A primeira classe é formada por problemas relacionados com o transporte de passageiros, a segunda é composta por problemas referentes a prestação e programação de serviços, como roteirização de coleta de lixo ou entrega postal, e a terceira classe remete á problemas de transporte de cargas. Desrochers, Lenstra e Savelsbergh (1990) desenvolveram um esquema que suporta o desenvolvimento de modelos e sistemas de transporte, servindo também para classificar diversos problemas de roteirização e programação de veículos. A ideia é fornecer diretrizes a uma representação teórica do problema real, com base nos tipos de veículos, endereços, objetivos e características do problema, servindo como base para o desenvolvimento de modelo e sistemas, e facilitando escolha do algoritmo mais apropriado. Abaixo, são descritos alguns problemas presentes em Bodin*et al.* (1983), Ronen (1988), Desrochers, Lenstra e Savelsbergh (1990) e outros clássicos encontrados na literatura:

Problema de Roteirização de Veículos (*Vehicle Routing Problem*– VRP): O objetivo deste problema é traçar caminhos mínimos para que uma frota de veículos, inicialmente em um ou mais depósitos, possa atender uma demanda de consumidores dispersos em um espaço geográfico, em um determinado horário, com diferentes demandas pelos produtos a serem distribuídos. A diferença deste problema para o problema de múltiplos caixeiros está no acréscimo da restrição de capacidade de veículos. Há também outras restrições aplicadas a este problema, como janela de tempo para o atendimento de clientes e tempo máximo de viagem dos veículos. O VRP foi proposto em 1959 por Dantzig e Ramser, através de um estudo realizado sobre o problema de distribuição de gasolina por uma frota de veículos. Um estudo recente sobre este problema pode ser encontrado na publicação de Drexl (2012).

Problema de Dimensionamento e Roteirização de uma Frota de Veículos (*Fleet Size and Vehicles Routing Problem*– FSVRP): Trata-se de uma variação do problema de roteamento de veículos, onde deve-se primeiramente especificar a quantidade de veículos necessários (frota ilimitada) para o roteamento, e depois, determinar o roteiro mínimo de cada veículo, para que no final, as demandas de todos os clientes sejam atendidas com os custos fixos de veículos e os custos variáveis de roteirização sendo

mínimos. Neste problema, se os veículos da frota possuem a mesma capacidade e custos fixos, pode-se dizer que a frota é homogênea. Caso contrário, a frota é denominada heterogênea. Subramanian et. al. (2011) abordaram este problema por meio de Meta-Heurísticas Híbridas.

Problema de Coleta e Entrega (*Pickup and Delivery Problems*– PDP): Este problema é tido como uma variação do VRP, onde há uma restrição de precedência no atendimento da demanda dos clientes, pois as cargas são transportadas pelos veículos do depósito aos clientes e entre os clientes. Estratégias de resolução do PDP podem ser vistas em Berbeglia, Cordeau e Laporte (2010).

3.4 Métodos de Resolução

Na literatura, destacam-se os métodos exatos ou heurísticos na resolução de problemas complexos como o PCV. Métodos exatos visam obter sempre o resultado ótimo, com isso, se faz necessário um maior esforço computacional para que o resultado seja alcançado em um tempo aceitável. Como exemplos de métodos exatos, pode-se citar Programação Linear, Programação Dinâmica, os algoritmos *Branch* & *Bound* (LAND & DOIG, 1960).

A Programação Dinâmica é uma metodologia de construção de algoritmos voltados para a resolução de problemas de otimização, cujo processo de obtenção de soluções particiona o problema em subproblemas, trabalhando do fim para o princípio (SZWARCCFITER, 1984). Proposta por Bellman em 1954, esta metodologia visa reduzir a dificuldade de resolução decompondo um problema numa sequência de problemas inter-relacionados mais simples. Em Programação Linear, um modelo matemático é formulado para o problema através de uma função objetivo sujeita á restrições e solucionado pelo algoritmo SIMPLEX.

O Algoritmo *Branch* & *Bound* opera a partir da enumeração inteligente dos pontos candidatos à solução ótima de um problema, realizando sucessivas partições do espaço de busca e cortando a árvore de pesquisa através da consideração de limites inferiores e superiores calculados ao longo da enumeração. Os limites inferiores são configurados a partir de métodos de relaxação que removem restrições do problema em

questão. Na grande parte das aplicações deste algoritmo, os limites superiores são obtidos por heurísticas que produzem boas soluções em curtos intervalos de tempo. O esquema de enumeração divide o problema em vários subproblemas menos complexos até que o limite inferior seja igual ao superior ou o limite inferior seja maior que a melhor solução corrente. Merz (2000) citou que esta abordagem produz uma ramificação (*branching*) na qual cada nó corresponde a um problema e os nós descendentes, os subproblemas.

O algoritmo *Branch* & *Cut* baseia-se na combinação de um algoritmo *Branch* & *Bound* com técnicas de planos de corte visando melhorar a etapa de relaxamento e a exploração do politopo definido pelas soluções viáveis do problema considerado. Este algoritmo é indicado para problemas que são impossíveis de serem tratados eficientemente apenas por ferramentas de programação linear devido ao seu grande número de restrições, e funciona a partir do seguinte conceito: se uma solução ótima associada à relaxação linear é inviável, um novo problema de separação deve ser "resolvido" buscando a identificação de uma ou mais restrições violadas pela relaxação corrente (*cutting procedure*). O novo problema obtido (com restrições adicionais) é novamente resolvido via programação linear e o processo é repetido até que novas classes de desigualdades violadas não sejam mais encontradas (MERZ, 2000). Exemplos de algoritmos *Branch & Bound* e *Branch & Cut* aplicados ao PCV podem ainda ser encontrados em Crowder e Padberg (1980), Balas e Toth (1985), Jfinger, Reinelt e Thienel (1994), Applegate et al. (1998, 2006 e 2009) e Erdogan, Cordeau e Laporte (2010).

Dumitrescu e Stützle (2003) atentaram para vantagens e desvantagens no uso de abordagens exatas na resolução de problemas de otimização. Dentre as vantagens, destacam que soluções ótimas podem ser obtidas caso o algoritmo tenha sucesso na sua execução. Como desvantagem, citam o consumo demasiado de recursos computacionais durante a resolução de instâncias de grande porte. Hoffman e Padberg (1985) e Wolsey (1998) elaboraram um estudo sobre a aplicação destes métodos em problemas de otimização. Em Laporte (1992), podem ser encontradas outras abordagens exatas aplicadas ao PCV.

Na prática, quando não se necessita achar uma solução ótima para o problema, dado que o tempo necessário para a mesma possa ser demasiadamente longo, utilizam-se então métodos heurísticos, que conseguem encontrar boas soluções ou, em alguns casos, soluções ótimas, com um consumo inferior de recursos computacionais ao das abordagens exatas.

No próximo capítulo, será descrito a questão dos Algoritmos Genéticos, base teórica fundamental no desenvolvimento dessa pesquisa. Neste trabalho, mostra-se que o PCV pode resolver instâncias do *Flow Shop Problem*, utilizando a Matriz Distância *D* de um PCV para o FSP. Maiores detalhes sobre o procedimento são explicitados no Capítulo 5.

CAPÍTULO IV

ALGORITIMOS GENÉTICOS

Neste capítulo, um histórico do surgimento dos Algoritmos Genéticos é reportado na primeira seção. O modo de operação do algoritmo é dado na segunda seção. A representação da estrutura das soluções, critérios de seleção, operadores genéticos e critérios de parada do AG são dados na terceira, quarta, quinta e sexta seção, respectivamente.

4.1 Histórico

O Algoritmo Genético foi criado por Jonh Holland durante as décadas de 1960 e 1970 (Holland, 1975), para simular computacionalmente o comportamento da seleção natural. Foi um aluno de Holland, Goldberg, o primeiro a aplicar o AG num problema de otimização, na área de projeto de gasodutos (Haupt e Haupt, 2004). Depois desta aplicação, o AG passou a ser considerado uma técnica de busca baseada nos princípios da genética e seleção natural. Os indivíduos são avaliados por uma função que atribui um valor chamado aptidão a cada indivíduo da população, segundo sua qualidade em relação à função objetivo do problema. Os indivíduos são escolhidos por um procedimento inspirado na seleção natural para passarem por operações genéticas que resultam em descendentes que comporão a nova população. Os estudos mostram que a nova população tem a tendência de ter indivíduos com aptidões melhores do que a população anterior (Mitchell, 1998; Haupt e Haupt, 2004). Este processo de gerar novas populações é chamado de geração. O melhor indivíduo da última população é a solução a ser apresentada para o problema.

Nos AGs, as soluções potenciais de um problema específico são codificadas em uma estrutura de dados semelhante a um cromossomo, sobre a qual são aplicados operadores de recombinação com o objetivo de preservar informações críticas. AGs têm sido aplicados em uma grande quantidade de problemas (WHITLEY, 1993) e são definidos por Gen (2006) como técnicas estocásticas de busca baseadas nos princípios da seleção natural e da genética.

4.2 Modo de Operação

Um AG baseia-se em conjuntos de soluções chamadas populações, formadas por indivíduos, os quais são soluções codificadas. Esses indivíduos, os cromossomos, são avaliados por uma função objetivo, de forma que os melhores têm mais chance de permanecer na população ou de gerar novos indivíduos a partir de suas próprias características. Com isso, espera-se obter uma evolução dos valores de avaliação dos cromossomos ao longo do tempo e, assim, obter a melhoria das soluções. O critério de parada é a convergência das soluções para praticamente uma solução dominante ou a imposição de limites de tempo ou de iterações. Nesse momento, o individuo com a melhor avaliação representa a melhor solução encontrada para o problema tratado pelo algoritmo (KOZA, 1992; GEN, 2006).

Figura 4.1– Pseudocódigo de AG genérico.

```
Inicio
    gera população inicial
    avalia a população
    Enquanto o critério de parada não for atingido faça
        executa cruzamento
        executa mutação
        avalia os novos indivíduos
        seleciona os indivíduos a substituir e seus substitutos
        atualiza o critério de parada
    Fim
    apresenta a melhor solução
Fim
```

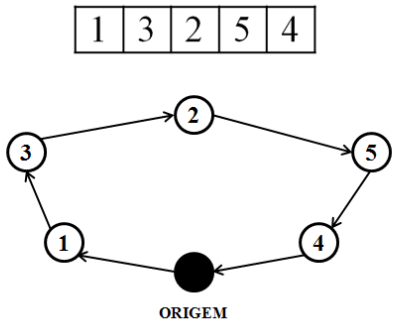
Fonte: BEZERRA (2012)

Segundo Koza (1992) e Gomes (2008), os passos na preparação de um AG são a determinação da representação do cromossomo, da função de aptidão, das regras de geração da população inicial, do método de seleção, dos operadores genéticos, da estratégia geracional, dos parâmetros e variáveis para controlar o algoritmo, do modo de reconhecimento do resultado e do critério de parada. Na Figura 4.1, é apresentado o pseudocódigo de um AG genérico.

4.3 Representação ou codificação

Baker (2003) afirma que o ponto de partida para qualquer AG é a representação das soluções da população e ressalta que, apesar de cromossomos binários terem sido favorecidos por vários estudiosos, boas implementações têm sido feitas com o uso de representações não binárias. Em um cromossomo, cada variável é um gene. Os valores possíveis de cada gene são os alelos, e a posição de cada gene é seu *locus* (REEVES, 1993). O cromossomo representa o mapeamento de cada ponto possível no espaço de busca explorado pelo AG. Essa representação das possíveis soluções pode ser intuitiva, ou não. E, como a sua escolha pode facilitar a própria solução do problema, requer cuidado, inspiração e bom senso. Na Figura 4.2, é apresentado um exemplo de codificação não binária, do tipo permutacional, em que os genes correspondem a pontos de uma rota, e a sequência dos genes no cromossomo correspondem à sequência de visita da rota. Esse é o tipo de representação usada neste trabalho.

Figura 4.2 – Exemplo de cromossomo permutacional e a rota que representa.



Fonte: BEZERRA (2012)

Holland (1975) introduz a noção de *schema* para formalizar o conceito de *building blocks*. O *schema* ou máscara é uma forma de representar configurações em que se fixam valores de determinados genes, deixando os demais em "aberto", representados por asteriscos. A *ordem* de um *schema* é a quantidade de elementos definidos. Por exemplo, o *schema* de ordem quatro (1 2 3 * * 6), considerando o universo de permutações de 1 a 6, representa os elementos (1 2 3 4 5 6) e (1 2 3 5 4 6). Estas são instâncias desse *schema*, de forma que o mesmo define um hiperplano. Assim,

o termo *schema* serve para denotar tanto *a máscara* em si quanto o conjunto de cromossomos que possuem essa máscara (MITCHELL, 1998; VIANA, 1998).

Na avaliação de uma população com *n* indivíduos, o AG estima implicitamente a aptidão média de todos os *schemas* que a compõem, variando suas participações de acordo com seus resultados. Por exemplo, os *schemas* cuja aptidão estimada fica acima da média, recebem mais números de tentativas (membros da população). Esse é o papel da seleção: achar os melhores *schemas* a cada geração. Já a avaliação simultânea e indireta dos *schemas* na população é conhecida como *paralelismo implícito*. E à medida que um AG avança em sua exploração do espaço de soluções, a estimativa da média de aptidão de um *schema* se torna mais assertiva porque o algoritmo avaliou mais instâncias desse mesmo *schema* (MITCHELL, 1998).

Por isso, é clara a importância da medida de aptidão. Seu cálculo deve ocorrer para cada indivíduo da população. Assim como a representação dos indivíduos, a escolha da medida de aptidão e da sua função geradora requer bom senso. Uma medida de desempenho dos indivíduos mal definida pode levar o AG a fazer seleções erradas, distanciando sua exploração das regiões com as soluções realmente desejadas. Além disso, a complexidade da extração do valor de aptidão de um cromossomo requer capacidade computacional, o que pode impactar no tempo de processamento do AG.

4.4 Estratégia geracional e seleção

A estratégia geracional de um AG é o tratamento dado para as diferentes gerações de populações durante o seu processamento. É a política de renovação e de manutenção de indivíduos de acordo com seus desempenhos e os interesses e objetivos do AG implementado. Na forma tradicional apresentada por Holland (1975), os novos elementos criados, por cruzamento ou mutação, são inseridos em uma nova população que substituirá a antiga. Na estratégia *steady-state*, segundo Baker (2003), os filhos entram na população à medida que são criados e ao mesmo tempo em que são retirados os piores indivíduos, de forma que o tamanho da população permanece constante. Há também o *elitismo*, que implica na manutenção de uma determinada quantidade dos

melhores indivíduos da população para a próxima geração. Mitchell (1998) ressalta a melhora significativa da performance de AGs devido ao elitismo.

Nos AGs, o modo de seleção dos indivíduos de acordo com as suas aptidões também desempenha papel fundamental. A seleção ocorre para a escolha dos indivíduos que serão preservados entre gerações ou para definir os pais que serão combinados para gerar filhos. Como exemplos de modelos de seleção, há o método da roleta, a seleção por torneios, a por *ranking* e a dinâmica. Na roleta, a chance de seleção de um indivíduo é função da sua aptidão, podendo ser linear ou não. Já no torneio, indivíduos escolhidos aleatoriamente formam grupos de tamanho fixo, e, dada uma determinada probabilidade, o melhor individuo do grupo é escolhido para cruzamento, caso contrário, qualquer um é escolhido aleatoriamente.

Na seleção por *ranking*, tenta-se prevenir a convergência prematura da população. Nesse tipo de seleção, os indivíduos da população são classificados de acordo com sua aptidão. E a chance de seleção do individuo é função da sua classificação. Dessa forma, indivíduos em posição imediatamente superior ou inferior na classificação têm praticamente as mesmas chances independentemente dos valores absolutos da medida de aptidão. Com isso, mantém-se a pressão alta quando a variabilidade é baixa e a reduz na situação oposta. Este racional também é a base dos processos de seleção dinâmicos, do tipo Boltzmann, segundo os quais, em momentos distintos do processamento, são necessárias pressões distintas de seleção.

4.5 Operadores genéticos e hibridização

O papel dos operadores genéticos em um AG é criar novos indivíduos a partir de cromossomos existentes na população. Existem operadores de cruzamento e de mutação. No primeiro caso, considerado por Mitchell (1998) e Gen (2006) o principal operador genético, são criados cromossomos filhos pela recombinação das características dos cromossomos pais, os quais são escolhidos conforme o método de seleção vigente.

Já as mutações são operadores de retaguarda que produzem mudanças aleatórias e espontâneas em vários cromossomos. Seu papel, normalmente, é pequeno

nos AGs, pois estes tomam como base os efeitos criativos da recombinação cromossômica propiciada pelas operações de cruzamento e a exploração dos efeitos do princípio *darwiniano* da sobrevivência dos mais fortes (KOZA, 1992). Ainda assim, as mutações podem atuar tanto na diversificação quanto na intensificação da busca pelo espaço de soluções independente do mecanismo codificado.

Mitchell (1998) expõe a ideia de Holland (1975) na qual a adaptação é a tensão entre a busca pelo novo e a exploração desse novo encontrado. A autora explica que a tensão surge porque qualquer movimento no sentido de explorar novas áreas do espaço de soluções, experimentando novos *schemas* são testar instâncias cuja aptidão são baixas e desvia a exploração de *schemas* já testados, com bons resultados. Ela afirma também que o sistema tem que continuar testando novas possibilidades, mas também tem que incorporar e usar continuamente a experiência passada como guia para o comportamento futuro, pois um balanceamento ótimo entre explorar o certo e o incerto deve ser encontrado. De certa forma, também por causa dessa "indecisão" e do seu caráter generalista, diz-se que os AGs realizam buscas superficiais e por isso é conveniente hibridizá-los com outras heurísticas.

Com a abordagem híbrida, técnicas de otimização local, são aplicadas a cada novo filho da população. Nessas situações, os AGs são usados para explorar a população enquanto os métodos heurísticos são usados para explorar as soluções. Por causa dessas características complementares dos AGs e das heurísticas convencionais, o método hibrido normalmente se sai melhor que cada qual individualmente (GEN, 2006).

Blum e Roli (2003) reforçam esse conceito ao afirmarem que a maior parte dos casos de sucesso com modelos evolucionários utilizam procedimentos de busca local. Para estes autores, as razões se tornam aparentes ao se analisar as forças de cada método: métodos baseados em populações são melhores em identificar áreas promissórias no espaço de soluções, enquanto heurísticas de busca local são boas em explorar essas áreas. Essas heurísticas partem de uma solução provavelmente razoável ou boa e trabalham de forma orientada ao problema, refinando o cromossomo trabalhado de uma forma que o AG não faria.

4.6 Critérios de Parada e Outros Parâmetros

Em se tratando de um método não exato, no qual se busca ótimo sem necessariamente encontrá-lo, o AG pode seguir avaliando e gerando indivíduos indefinidamente. Por isso, é preciso estabelecer um critério de parada para interromper a busca e apresentar um resultado, normalmente a melhor solução encontrada até a interrupção do processamento. Reeves (1995) cita o número de gerações e o tempo de processamento como os critérios mais comuns de encerramento de execução de um AG.

Juntamente com o critério de parada e seu valor, outros parâmetros também devem ser estabelecidos para a execução de um AG. Koza (1992) coloca o tamanho da população e o número máximo de gerações como os principais parâmetros de controle de um AG, enquanto os secundários são as taxas de reprodução e mutação. E, dependendo do modelo e da implementação, pode haver outros parâmetros.

Apesar de existirem estudos nesse sentido, Mitchell (1998) não acredita na existência de uma formulação geral de parametrização por causa da variedade de problemas, codificações e outras especificidades possíveis em diferentes aplicações. A autora afirma ainda que o tamanho ótimo de uma população, a taxas de cruzamento e de mutação, muda ao longo do processamento.

Através da revisão bibliográfica foram escolhidos oito componentes como sendo os mais importantes num projeto de AG:

1. Escolha da representação do cromossomo (solução de um problema);
2. Definição da função de aptidão (função objetivo do problema);
3. Definição da população inicial;
4. Escolha do método de seleção;
5. Escolha dos operadores genéticos *crossover* e *mutação*;
6. Escolha da estratégia geracional;
7. Escolha do critério de parada; e
8. Escolha dos valores dos parâmetros (calibragem do AG).

Neste trabalho, foi desenvolvido um AG específico para o PCV, sendo que este problema usa as instâncias do FSP através da metodologia usada no trabalho de Ali e Dulaimi (2008). Desta forma, o FSP está sendo solucionado através da aplicação do AG na resolução do PCV. No capítulo seguinte, serão apresentadas esta transformação (FSP → PCV) e o algoritmo genético desenvolvido e implementado.

CAPÍTULO V

ALGORITMO PROPOSTO

Este capítulo é apresentado em duas seções, onde na primeira seção é apresentado o algoritmo genético proposto para resolver o FSP através do PCV, e na segunda seção é apresentada a transformação usada para o FSP se transformar num PCV.

5.1 Algoritmo Genético Aplicado ao PCV

Como já explicitado nesse trabalho, o objetivo do PCV consiste em determinar uma rota, para o caixeiro, que percorra todas as *n* cidades somente uma vez com a menor distância percorrida. O algoritmo genético foi proposto para resolver uma instância do PCV. Desta forma, serão descritas aqui nesta seção as oito componentes usadas na construção do nosso algoritmo genético.

a) O cromossomo representa uma permutação das *n* cidades, definindo uma rota (tour de visitas) ou uma solução viável do problema. Exemplo, para n=5, as permutações <1 3 5 4 2>,<3 5 4 2 1>,<5 4 2 1 3>, <4 2 1 3 5> e <2 1 3 5 4> representam a rota 1->3->5->4->2->1. Trabalhamos com permutação iniciando sempre pelo ponto 1. Caso seja gerada uma permutação fora deste formato é possível transformá-la neste formato, bastando verificar onde está o ponto 1 na referida permutação e iniciar a rota representada por ela a partir deste ponto. Do exemplo dado, exceto a primeira permutação que já inicia com 1, qualquer uma das outras permutações podem ser transformada na primeira.

b) A função custo (*The fitness function*) é dada pela distância total percorrida nesta rota, função g(s) dada no início da seção 3.1.

c) O AG proposto tem uma população inicial com 40 indivíduos (P), selecionados pelas heurísticas gulosas de inserção mais baratas (HIMBM), conforme descrita em Silva (2013). As 40 melhores soluções destas heurísticas serão inseridas na população inicial do AG obedecendo a uma ordem crescente na função *fitness*.Desta forma, os melhores indivíduos da população são P_1, P_2, ..., P_{40}, nesta ordem.

d) A Seleção será feita com todos os indivíduos da população dois a dois.

e) O cruzamento sendo feito entre todos os indivíduos da população, dando 780 cruzamentos (combinação de 40 dois a dois). Pode-se também dividir a população em faixas e cruzar todos os elementos de uma faixa com todos os elementos de outra(s) faixa(s). Por exemplo, podemos dividir a população em 4 faixas (F_1, F_2, F_3 e F_4) com dez elementos cada. Na Faixa 1 ficam os dez primeiros melhores indivíduos, os indivíduos da 11ª a 20ª posição na população atual ficam na Faixa 2, e assim sucessivamente. O cruzamento pode ser feito com todos os indivíduos da F_1 com F_3 e F_2 com F_4, gerando 200 cruzamentos em vez de 780. No estudo feito para determinar a melhor maneira de esquematizar a seleção ficou constatado que o uso da primeira opção produziu melhores resultados que o da segunda opção, tratar o cruzamento por faixa.

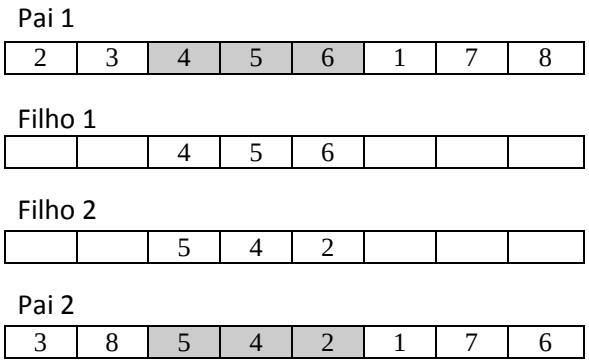
O operador *crossover* é definido da seguinte forma. Primeiro divida a solução em 3 partes (permutação), sendo dois pontos de corte c_1 e c_2, com $1 \leq c_1 \leq n$, $1 \leq c_2 \leq n$ e $c_1 \neq c_2$. O operador *crossover* implementado foi o *Order Crossover* (OX) (Goldberg, 1989). O operador *crossover* OX foi criado baseado na idéia dos bons blocos construídos. Por isso, baseia-se nas posições relativa e absoluta das cidades na rota. Este procedimento é descrito a seguir:

1 – São escolhidos dois pais através do método de seleção (Pai 1 e 2);

2 – Com base nos pontos de cortes determinados, o cromossomo foi dividido em três blocos, conforme mostram as Figuras 5.1 e 5.2. O segundo bloco de cada cromossomo é copiado para cada um dos Filhos (O_1 e O_2). Esta etapa preserva as posições absoluta e relativa das estruturas do cromossomo de cada pai em cada filho, dando a idéia de que haja hereditariedade; e

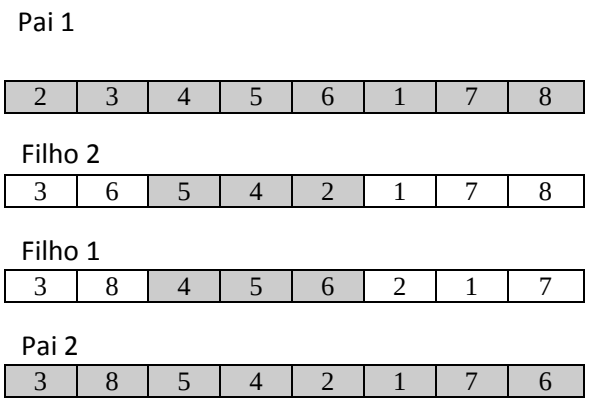
3 – As posições não-preenchidas de cada filho são copiadas das posições do outro pai no sentido da esquerda para a direita (Figura 5.3). Este procedimento faz com que seja preservada a ordem relativa das cidades nas rotas.

Figura 5. 1 – Segunda etapa do *crossover* OX.



Fonte: BEZERRA (2012)

Figura 5. 2 – Terceira etapa do *crossover* OX.



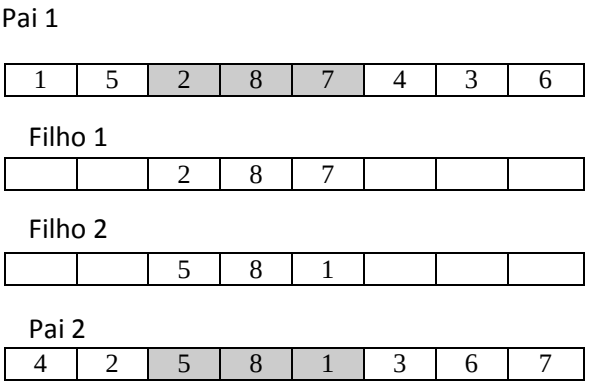
Fonte: BEZERRA (2012)

Neste exemplo, o valor de n=8, c_1=3 e c_2=5, onde: $c_1=\lfloor n/3 \rfloor+1$e $c_2=2*\lfloor n/3 \rfloor + 1$.

Outras combinações de blocos no cruzamento acima foram usadas, mas elas não mostraram evolução na busca. Testou-se o *crossover* de um ponto c_1=n/2, e pôde se observar que os resultados não foram melhores que o crossover de 2 pontos. Um método mais moderno de *crossover* denominado *Partially Matched* (PM) também foi implementado e analisado. Segue uma ilustração deste método na Figura 5.3 e 5.4 abaixo. Ele também usa dois pontos de corte, que podem ser c_1 e c_2 dados anteriormente. O segundo bloco do Pai 1 é copiado para o Filho 1, assim como o

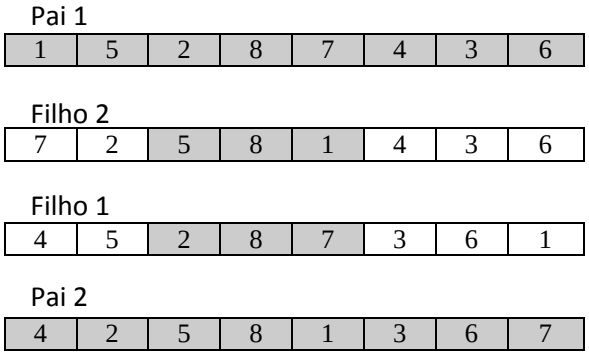
primeiro e terceiro bloco do Pai 2. Depois serão feitos os ajustes para a geração de uma solução viável, representada pelo Filho 1, substituindo no primeiro e terceiro bloco os genes repetidos do Filho 1, pelos genes que se encontram na mesma posição daqueles repetidos que estão no segundo bloco do Pai 2. Estes genes são aqueles que não se encontram no Filho 1 após as cópias dos blocos, no caso específico do exemplo dado através das Figuras 5.3 e 5.4, são as cidades 5 e 1. O procedimento também se aplica da mesma forma para o Pai 2 e Filho 2.

Figura 5. 3 – Segunda etapa do *PM crossover*.



Fonte: BEZERRA (2012)

Figura 5. 4 – Terceira etapa do *PM crossover*.



Fonte: BEZERRA (2012)

A mutação usa a heurística *opt4* (diferente de 4opt), de baixa complexidade descrita em Silva (2013), com bom desempenho, para ser aplicada na melhor solução da iteração atual, ou as *k* melhores soluções (k>1).

f) A estratégia geracional é responsável por controlar a substituição de indivíduos de uma geração para a outra. A estratégia geracional proposta por Holland (1975) cria um conjunto do tamanho da população de indivíduos gerados a partir da população atual, usando os operadores de seleção, *crossover* e mutação. No final este conjunto substitui a população atual. Neste tipo de estratégia existe a possibilidade de que bons indivíduos desapareçam de uma geração para a outra. Por isso, surgiram outras estratégias como a elitista, a *population overlaps* e a *steady-state*. Na estratégia elitista o melhor indivíduo é preservado para a próxima população, enquanto o restante da população é substituída por novos indivíduos. Na estratégia *population overlaps* uma fração da população G (*generation gap*) é substituída por novos indivíduos, enquanto a outra fração é preservada para a próxima população. Na estratégia *steady-state* só o melhor indivíduo gerado é copiado para a próxima população. No nosso AG foi utilizada a estratégia elitista por ter tido melhor desempenho.

g) O critério de parada usa geralmente o limite de valores propostos para três variáveis: número de iterações do algoritmo, tempo máximo de execução e o desvio das soluções. Estes valores foram determinados depois das análises feitas com os testes de execução do algoritmo com vários problemas da literatura. A priori usou-se o valor médio na função objetivo das 40 soluções da população atual e comparou-se com o valor médio da população imediatamente anterior, quando estes valores forem iguais o durante 3 iterações seguidas, o algoritmo pára e apresenta a melhor solução encontrada como solução do problema. Com isto evitaram-se de realizar avaliações de soluções que eram distintas mas que tiveram o mesmo desempenho de soluções similares, muitas vezes estas soluções são repetidas, foi o que se constatou.

h) A última etapa do AG é a calibração dos valores dos seus parâmetros, como tamanho da população, taxa de *crossover*, entre outras. Segundo Mitchell (1998) os parâmetros de um AG interagem entre si de forma não-linear sendo de difícil calibração. Muitos trabalhos têm sido realizados nesta área sem opinião consensual (Ruiz *et al.*, 2006) e uma desvantagem do AG é a dificuldade do processo de calibração dos seus parâmetros (Dréo *et al.*, 2006). Vários testes foram realizados com as pequenas instâncias do PCV sendo oriundas do FSP e foram comentadas nas respectivas componentes.

5.2 Algoritmo proposto

O Problema de Sequenciamento pode ser representado como um Problema do Caixeiro Viajante. Existe uma forma bastante simples de relacionar as instâncias do FSP como a matriz de distâncias D do PCV, usando a matriz de Baker, conforme descrito em Al-Dulaimi e Ali (2008). A Tabela 5.1 ilustra a composição da matriz D.

Tabela 5.1 – Matriz de Distância do PCV para o FSP.

0	D_{12}	D_{13}	...	D_{1n}
D_{21}	0	D_{23}	...	D_{2n}
D_{31}	D_{32}	0	...	D_{3n}
...	...	...	...	...
D_{n1}	D_{n2}	D_{n3}	...	0

Fonte: Ali (2008)

Nessa Matriz de Distâncias, D_{ik} representa o atraso no início da tarefa k (medida a partir do início da tarefa i) e o valor total de qualquer rota representa o *makespan* para a sequência correspondente. D_{ik} é calculado pela equação abaixo:

$$D_{ik} = P_{1i} + \max\left(0, P_{2i} - P_{1k}, P_{2i} + P_{3i} - P_{1k} - P_{2k}, \ldots, \sum_{v=2}^{m} P_{vi} - \sum_{l=1}^{m-1} P_{lk}\right)$$

Onde P_{ij} representa o tempo de processamento da tarefa j (j=1,...,n) na máquina i (i=1,...,m). Subtraindo P_{1i} de cada linha i para os valores da Tabela 5.1 tem-se:

$$D'_{ik} = D_{ik} - P_{1i}$$

Os valores resultantes são mostrados abaixo na Tabela 5.2, denominada Matriz de Custo Reduzido.

Tabela 5.2 – Matriz de Custo Reduzido

0	D'_{12}	D'_{13}	...	D'_{1n}
D'_{21}	0	D'_{23}	...	D'_{2n}
D'_{31}	D'_{32}	0	...	D'_{3n}
...	...	...	...	...
D'_{n1}	D'_{n2}	D'_{n3}	...	0

Fonte: Ali (2008)

Logo, a matriz de custo reduzido acima pode ser utilizada para representar qualquer instância do FSP como sendo uma instância relacionada com o Problema do Caixeiro Viajante.

Nas Tabelas 5.3 a 5.5, abaixo, apresentam se uma instância do PS, com n=4 e m=5, a matriz distância e a matriz custo reduzido, respectivamente.

Tabela 5.3 – Matriz P do Problema de Sequenciamento.

Tarefa	M_1	M_2	M_3	M_4	M_5
J_1	5	10	6	6	2
J_2	6	11	8	6	3
J_3	7	13	8	8	3
J_4	9	15	11	10	4

Fonte: Ali (2008)

Tabela 5.4 – Matriz Distância da Instância do PS.

Cidade	J_1	J_2	J_3	J_4
J_1	0	9	8	6
J_2	12	0	10	8
J_3	15	14	0	11
J_4	22	18	17	0

Fonte: Ali (2008)

Tabela 5.5 – Matriz Custo Reduzido da matriz *D* dada na Tabela 5.4

Cidade	j_1	J_2	J_3	J_4
J_1	0	4	3	1
J_2	6	0	4	2
J_3	8	7	0	4
J_4	13	9	8	0

Fonte: Ali (2008)

A seguir descreve-se como foi obtido o valor de D_{12}, baseado nos dados da Tabela 5.3 para a matriz *P*.

D_{12}= 5 + Max{0, 10-6, 10+6-6-11, 10+6+6-6-11-8, 10+6+6+2-6-11-8-6)

D_{12}= 5 + Max{0, 4, -1, -3, -7) = 5 + 4 = 9.

Portanto, usando esta regra é possível transformar qualquer instância do FSP numa instância do PCV. Aplicar um método de resolução do PCV, produzido a partir de uma instância do FSP, funciona como se este método estivesse sendo aplicado diretamente na resolução do FSP. A seguir, os experimentos computacionais realizados serão apresentados.

CAPÍTULO VI

EXPERIMENTOS COMPUTACIONAIS

Os testes do AG foram executados num micro-computador PC Intel (RAM de 8 GB e 1.8 GHz) e o código implementado em linguagem ANSI C. As tabelas, a seguir, mostram o desempenho do método com instâncias da OR-Library (Beasley, 1990), conjunto de dados de Taillard (1993) separadas em 9 classes. Tabulou-se, por classe: a média dos desvios, mínimo e máximo dos desvios e a quantidade de melhores resultados em comparação com outros métodos de acordo com o operador crossover utilizado *one-point* (1P), *two-point* (2P) e *Partially Matched* (PM). Também estão tabulados os tempos de processamento computacional de cada teste realizado. Colocou-se em negrito os melhores resultados do operador *crossover* que obteve o melhor desempenho entre eles para cada instância testada.

Tabela 6.1 – Resultados do AG para 1ª classe (n= 20, m=5).

	Desvio (%)			Tempo (s)		
Instância	1P	2P	PM	1P	2P	PM
ta001	1,56	**0,63**	1,56	0,79	2,72	0,49
ta002	0,66	**0,52**	0,66	0,83	2,82	0,72
ta003	**7,03**	7,77	7,59	1,10	1,86	0,47
ta004	4,95	**2,94**	6,19	0,80	2,96	0,60
ta005	**1,46**	**1,46**	**1,46**	1,40	3,15	0,43
ta006	7,03	**4,85**	6,95	0,84	2,01	0,37
ta007	3,63	**2,74**	3,63	0,93	2,86	0,57
ta008	2,99	**1,91**	2,99	1,02	3,12	0,61
ta009	**3,82**	**3,82**	4,88	1,13	2,36	0,69
ta010	**4,42**	**4,42**	4,51	0,94	3,87	0,46
Média	3,76	**3,11**	4,04	0,98	2,77	0,54
Mínimo	0,66	**0,52**	0,66	0,79	1,86	0,37
Máximo	**7,03**	7,77	7,59	1,40	3,87	0,72
Melhores	4	9	1			

Fonte: Elaborada pelo autor.

O principal indicador utilizado nas comparações entre os métodos, ao longo desse capítulo, é o percentual de desvio das soluções, dado por ((s* - s') / s*)x100, onde

s* é a melhor solução do problema e s' é a melhor solução encontrada pelo método de resolução aplicado ao problema.

Na Tabela 6.1, o melhor resultado para esta 1ª classe (*n*=20 e *m*=5) foi do operador *two-point* (2P), apresentando 9 melhores resultados, e com média geral de 3,11%. Contudo, por conta de um maior número de operações, foi o que demandou maior tempo computacional (2,77 segundos). O menor mínimo e máximo foram obtidos para o 2P e 1P, respectivamente. No problema ta005, os três operadores apresentaram resultados iguais (1,46%). O problema ta006 apresentou o menor tempo de processamento (0,37s). O operador com pior desempenho foi o PM, com apensar um problema com melhor desempenho, bem como a maior média da classe (4,04%).

Tabela 6.2 – Resultados do AG para a 2ª classe (n= 20, m=10).

	Desvio (%)			Tempo (s)		
Instância	1P	2P	PM	1P	2P	PM
ta011	6,38	**6,07**	6,38	2,63	5,97	1,47
ta012	7,17	7,47	**6,63**	1,55	5,24	0,76
ta013	7,09	**6,22**	8,42	1,69	4,03	1,59
ta014	**7,19**	7,84	10,09	2,21	4,48	0,97
ta015	10,01	**8,74**	9,37	1,78	5,21	0,93
ta016	**9,95**	**9,95**	10,02	1,47	6,38	1,41
ta017	**5,66**	6,40	7,01	1,68	5,38	0,88
ta018	8,52	**7,48**	8,91	2,38	4,93	0,92
ta019	**3,83**	4,39	5,02	2,07	6,11	1,25
ta020	10,69	9,05	**8,67**	1,44	5,93	1,06
Média	7,65	**7,36**	8,05	1,89	5,37	1,12
Mínimo	**3,83**	4,39	5,02	1,44	4,03	0,76
Máximo	10,69	**9,95**	10,09	2,63	6,38	1,59
Melhores	4	5	2			

Fonte: Elaborada pelo autor.

Os dados da Tabela 6.2 mostram um resultado mais equilibrado para a 2ª classe, (*n*=20 e *m*=10) o operador 2P continua tendo o melhor desempenho com 5 melhores resultados e a menor média geral (7,36%), contudo, vêem-se bons resultados também para o operador 1P, com 4 melhores resultados e média geral de 7,65%. O menor mínimo e máximo foram obtidos para o 1P e 2P, respectivamente. O tempo de processamento foi maior para o 2P, com 5,37 e menor para o PM, com 1,12 segundos.

No problema ta016, os operadores 1P e 2P apresentaram os melhores resultados. A pior média foi para o operador PM, com 8,05% e apenas 2 dos melhores resultados para a classe.

Tabela 6.3 – Resultados do AG para a 3ª classe (n= 20, m=20).

	Desvio (%)			Tempo (s)		
Instância	1P	2P	PM	1P	2P	PM
ta021	6,31	**5,05**	6,09	2,34	6,18	2,01
ta022	10,24	**10,15**	10,24	4,50	6,36	1,78
ta023	6,66	5,98	**5,93**	3,13	12,37	2,55
ta024	**5,53**	6,16	6,84	4,53	7,68	2,21
ta025	5,15	**4,67**	**4,67**	4,20	9,49	2,28
ta026	7,28	**4,45**	8,81	2,78	12,25	1,90
ta027	**5,41**	6,03	6,38	4,90	11,20	2,59
ta028	5,77	**4,91**	5,27	4,59	9,04	2,62
ta029	6,44	**5,63**	6,97	4,76	13,19	2,28
ta030	10,01	**6,80**	**6,80**	3,53	8,78	3,65
Média	6,88	**5,98**	6,80	3,93	9,65	2,39
Mínimo	5,15	**4,45**	4,67	2,34	6,18	1,78
Máximo	10,24	**10,15**	10,24	4,90	13,19	3,65
Melhores	2	7	3			

Fonte: Elaborada pelo autor.

A partir de análises da Tabela 6.3 vê-se a tendência de melhora predominante do operador 2P permanece, com 7 dos melhores resultados da classe e média geral de 5,98%. Este operador permanece também com os maiores resultados em tempos computacionais, sendo de 9,65 segundos, em média. Contudo, nessa classe houve uma melhoria para os resultados do operador PM, que foi o segundo melhor desempenho, com 3 melhores resultados e média geral de 6,80%, bem como melhor tempo computacional (2,39 segundos). Nos problemas ta025 e ta030 os operadores 2P e PM apresentaram os melhores resultados. O método com desempenho mais fraco foi o 1P, com média de 6,88%, para esta classe de problemas.

Para a 4ª classe (Tabela 6.4), novamente vê-se um equilíbrio entre os dois melhores operadores (2P e 1P), com médias gerais de 2,58% e 2,80%, respectivamente. O operador 2P obteve 5 dos melhores resultados, enquanto o 1P apresentou 4. Seus tempos computacionais tiveram médias de 27,21 e 11,31 segundos. No problema ta031,

os três operadores apresentaram resultados idênticos. O operador PM foi o operador mais rápido (9,44 segundos), porém com resultados menos eficientes.

Tabela 6.4 – Resultados do AG para a 4ª classe (n= 50, m=5).

	Desvio (%)			Tempo (s)		
Instância	1P	2P	PM	1P	2P	PM
ta031	**0,66**	**0,66**	**0,66**	10,95	23,53	19,77
ta032	**2,72**	3,35	3,35	12,45	19,68	9,52
ta033	**2,37**	2,86	3,43	15,24	42,75	8,51
ta034	3,96	**3,42**	3,96	11,27	42,85	12,49
ta035	1,75	**0,80**	1,75	9,14	19,70	8,18
ta036	**1,84**	1,87	2,51	11,14	29,00	9,73
ta037	4,37	3,45	**3,41**	8,89	27,91	7,19
ta038	4,92	4,32	**4,06**	12,06	19,40	5,93
ta039	3,29	**3,10**	3,72	8,54	17,14	6,92
ta040	2,16	**2,01**	2,55	13,46	30,15	6,13
Média	2,80	**2,58**	2,94	11,31	27,21	9,44
Mínimo	0,66	0,66	0,66	8,54	17,14	5,93
Máximo	4,92	4,32	**4,06**	15,24	42,85	19,77
Melhores	4	5	3			

Fonte: Elaborada pelo autor.

Tabela 6.5 – Resultados do AG para a 5ª classe (n= 50, m=10).

	Desvio (%)			Tempo (s)		
Instância	1P	2P	PM	1P	2P	PM
ta041	11,00	**9,73**	10,36	43,75	72,11	25,32
ta042	10,71	**10,29**	10,95	19,51	36,37	16,30
ta043	13,98	**12,12**	13,49	34,76	119,98	24,86
ta044	10,19	**9,21**	9,83	16,65	46,45	31,01
ta045	**9,98**	10,89	11,53	19,99	39,97	11,95
ta046	10,31	10,05	**9,88**	20,34	47,06	13,73
ta047	**7,73**	**7,73**	8,44	38,89	37,76	29,44
ta048	9,94	**8,96**	9,15	18,19	40,13	17,57
ta049	11,67	**10,63**	11,43	23,09	47,04	10,56
ta050	9,62	**9,00**	10,77	18,06	39,15	18,82
Média	10,51	**9,86**	10,58	25,32	52,60	19,96
Mínimo	**7,73**	**7,73**	8,44	16,65	36,37	10,56
Máximo	13,98	**12,12**	13,49	43,75	119,98	31,01
Melhores	2	8	1			

Fonte: Elaborada pelo autor.

Os dados dos experimentos computacionais da 5ª classe, acima, mostram que a melhor média geral de desvios foi de 9,86%, do operador 2P, com o maior número de melhores resultados igual a 8, dos 10 problemas testados. Contudo, percebe-se que há um considerável crescimento do tempo computacional médio demandado, o operador 2P demandou 52,60 segundos, o dobro do operador 1P (25,32). Nessa classe, nenhum dos operadores apresentou resultados idênticos para um problema. O operador PM apresentou somente 1 melhor resultado.

Analisando os dados da Tabela 6.6, dada a seguir, tem-se que na 6ª classe encontra-se uma melhoria do método PM. Foi o operador com segundo melhor desempenho, com média geral de 14,84% e 4 melhores resultados. Novamente, o operador 2P foi o melhor, com 6 resultados eficiente e média geral de 14,69%. O tempo de processamento do operador 2P foi de 131,88 segundos. No problema ta058 os três operadores obtiveram resultados idênticos. O pior resultado foi do operador 1P, com média de 15,10% e apenas 2 melhores resultados. O operador 2P foi o que consumiu mais tempo de processamento, enquanto PM foi que consumiu menos tempo.

Tabela 6.6 – Resultados do AG para a 6ª classe (n= 50, m=20).

	Desvio (%)			Tempo (s)		
Instância	1P	2P	PM	1P	2P	PM
ta051	15,65	**14,90**	16,49	57,66	129,09	46,91
ta052	15,29	14,91	**14,72**	68,77	81,25	47,25
ta053	14,09	**13,95**	14,48	64,71	94,24	43,73
ta054	15,98	15,65	**15,27**	42,30	142,82	34,17
ta055	17,17	**16,04**	16,16	38,53	261,45	32,56
ta056	15,03	13,74	**12,35**	50,34	131,75	42,43
ta057	**13,18**	13,64	13,34	58,17	115,67	39,38
ta058	**14,64**	**14,64**	**14,64**	30,13	102,23	23,70
ta059	15,06	**14,81**	16,13	58,22	107,66	49,92
ta060	14,88	**14,64**	14,83	68,91	152,66	64,10
Média	15,10	**14,69**	14,84	53,77	131,88	42,42
Mínimo	13,18	13,64	**12,35**	30,13	81,25	23,70
Máximo	17,17	**16,04**	16,49	68,91	261,45	64,10
Melhores	2	6	4			

Fonte: Elaborada pelo autor.

Tabela 6.7 – Resultados do AG para a 7ª classe (n= 100, m=5).

	Desvio (%)			Tempo (s)		
Instância	1P	2P	PM	1P	2P	PM
ta061	1,53	**0,75**	1,53	80,4	153,88	80,4
ta062	**1,75**	1,92	**1,75**	69,02	194,99	69,02
ta063	1,95	**1,80**	1,95	72,15	119,06	72,15
ta064	1,58	**0,90**	1,34	161,75	167,92	185,12
ta065	**2,06**	2,25	2,25	103,2	180,82	103,03
ta066	0,33	**0,19**	0,33	59,74	1479	101,43
ta067	0,30	2,29	**0,00**	109,68	194,23	72,78
ta068	2,94	**2,82**	3,32	120,98	351,49	85,94
ta069	1,69	**1,36**	1,80	89,21	191,8	124,23
ta070	2,05	**1,54**	2,05	105,29	27,82	105,29
Média	1,62	**1,58**	1,63	97,14	306,10	99,94
Mínimo	0,30	0,19	**0,00**	59,74	27,82	69,02
Máximo	2,94	**2,82**	3,32	161,75	351,49	185,12
Melhores	2	7	2			

Fonte: Elaborada pelo autor.

Na Tabela 6.7, dada anteriormente, tem-se os resultados para os problemas da 7ª classe de testes. A tendência é a mesma dos resultados vistos até então, o operador 2P tem tido melhores resultados, nesse caso foram 7 melhores resultados do total, e melhor desvio médio geral (1,58%). Contudo, deve-se observar que os tempos tem se elevado bastante, o tempo máximo nessa classe foi de 351,49 segundos para o problema ta068, enquanto o PM só demandou, no máximo 185,12 segundos, isto deve-se ao valor de *n*=100 ser bastante grande. Em nenhum dos problemas da classe os operadores apresentaram resultados idênticos. No problema ta067, o resultado do operador PM foi idêntico ao ótimo do problema. O tempo do operador 2p foi quase 3 vezes os dos demais operadores. O operador PM encontrou a solução ótima em um problema teste (ta067).

Tabela 6.8 – Resultados do AG para a 8ª classe (n= 100, m=10).

	Desvio (%)			Tempo (s)		
Instância	1P	2P	PM	1P	2P	PM
ta071	8,09	**7,05**	8,35	132,77	302,21	149,86
ta072	8,73	**8,17**	8,71	192,21	186,62	200,23
ta073	**6,71**	**6,71**	**6,71**	219,24	238,03	138,41
ta074	10,38	**10,17**	10,59	208,17	360,69	175,20
ta075	9,55	**9,05**	9,88	223,47	293,94	232,40
ta076	8,07	**7,79**	**7,79**	118,31	242,92	242,92
ta077	**5,40**	5,99	5,61	94,06	429,92	223,36
ta078	7,80	**7,41**	7,80	149,66	291,86	199,86
ta079	5,57	**5,01**	6,05	203,71	194,36	168,32
ta080	5,59	**5,17**	6,02	149,65	261,17	254,10
Média	7,59	**7,25**	7,75	169,13	280,17	198,47
Mínimo	5,40	**5,01**	5,61	94,06	186,62	138,41
Máximo	10,38	**10,17**	10,59	223,47	429,92	254,10
Melhores	2	8	2			

Fonte: Elaborada pelo autor.

A análise dos dados da Tabela 6.8, acima, referente a 8ª classe de problemas testes, observa-se que o operador 2P apresentou melhor desempenho com média de 7,25% e 8 melhores resultados do total. No problema ta073 os três operadores apresentaram resultados idênticos, e no problema ta076, os operadores 2P e PM também apresentaram resultados idênticos. O operador menos eficiente foi o PM, pois apresentou maior desvio médio geral (7,75%) e segundo pior tempo computacional demandado (198,47 segundos).

Tabela 6.9 – Resultados do AG para a 9ª classe (n= 100, m=20).

	Desvio (%)			Tempo (s)		
Instância	1P	2P	PM	1P	2P	PM
ta081	**15,44**	15,95	16,79	296,35	642,34	649,05
ta082	14,17	**14,07**	**14,07**	457,57	1050,23	1050,23
ta083	13,71	13,90	**13,50**	528,97	1284,88	500,75
ta084	14,26	**13,19**	14,45	387,84	489,56	469,51
ta085	15,20	**14,28**	15,00	524,62	486,56	357,87
ta086	15,12	**13,84**	14,55	496,07	897,52	527,11
ta087	16,66	17,08	**16,40**	494,94	1297,54	343,62
ta088	17,85	**16,56**	17,32	460,45	878,35	375,41
ta089	14,80	**14,68**	14,80	375,33	5323,00	466,01
ta090	**12,84**	12,88	13,93	632,88	581,34	710,62
Média	15,00	**14,64**	15,08	465,50	1293,13	545,02
Mínimo	**12,84**	12,88	13,50	296,35	486,56	343,62
Máximo	17,85	**17,08**	17,32	632,88	5323,00	1050,23
Melhores	1	6	3			

Fonte: Elaborada pelo autor.

A última classe de teste realizada foi a 9ª classe, onde a Tabela 6.9 mostra os seus resultados. Nela é possível ver a tendência do bom desempenho do operador 2P quando comparado com os demais operadores 1P e PM. O operador 2P foi o mais eficaz, com 6 melhores resultados, e desvio médio geral de 14,64%. Contudo, o tempo computacional máximo do mesmo operador foi consideravelmente elevado (5323,00 segundos) no problema ta089, enquanto sua média de tempo de execução ficou em 1293,13 segundos, quase 22 minutos. O operador PM apresentou 3 dos melhores resultados, com média geral de 15,08%. No problema ta082, os operadores 2P e PM tiveram resultados idênticos. O operador 1P obteve o menor mínimo de desvio na classe, alcançado no problema ta090.

Com base nos experimentos realizados e tabulados nas 9 tabelas anteriores, elaborou-se a Tabela 6.10 que apresenta uma síntese dos resultados, por classe. Tem-se: a média dos desvios, mínimo e máximo dos desvios e a quantidade de melhores resultados em comparação com outros operadores, bem como o tempo médio de processamento por classe.

Tabela 6.10 – Síntese dos resultados obtidos pelo AG por tipo de operador e classe.

AG	Classe	1	2	3	4	5	6	7	8	9	Geral
1P	**Média (%)**	3,76	7,65	6,88	2,80	10,51	15,10	1,62	7,59	15,00	7,88
	Mínimo (%)	0,66	**3,83**	5,15	**0,66**	**7,73**	13,18	0,30	5,40	**12,84**	0,30
	Máximo (%)	**7,03**	10,69	10,24	4,92	13,98	17,17	2,94	10,38	17,85	17,85
	Melhores	4	4	2	3	2	2	2	2	2	22
	Tempo Médio (s)	0,98	1,89	3,93	11,31	25,32	53,77	**97,14**	**169,13**	**465,50**	
2P	**Média (%)**	**3,11**	**7,36**	**5,98**	**2,58**	**9,86**	**14,69**	**1,58**	**7,25**	**14,64**	**7,45**
	Mínimo (%)	**0,52**	4,39	**4,45**	**0,66**	**7,73**	13,64	0,19	**5,01**	12,88	0,19
	Máximo (%)	7,77	**9,95**	**10,15**	4,32	**12,12**	**16,04**	**2,82**	**10,17**	**17,08**	**17,08**
	Melhores	**9**	**5**	**7**	**4**	**8**	**6**	**7**	**8**	**6**	**60**
	Tempo Médio (s)	2,77	5,37	9,65	27,21	52,60	131,88	306,10	280,17	1293,13	
PM	**Média (%)**	4,04	8,05	6,80	2,94	10,58	14,84	1,63	7,75	15,08	7,97
	Mínimo (%)	0,66	5,02	4,67	**0,66**	8,44	**12,35**	**0,00**	5,61	13,50	**0,00**
	Máximo (%)	7,59	10,09	10,24	**4,06**	13,49	16,49	3,32	10,59	17,32	17,32
	Melhores	1	2	3	3	1	4	2	2	3	22
	Tempo Médio (s)	**0,54**	**1,12**	**2,39**	**9,44**	**19,96**	**42,42**	99,94	198,47	545,02	

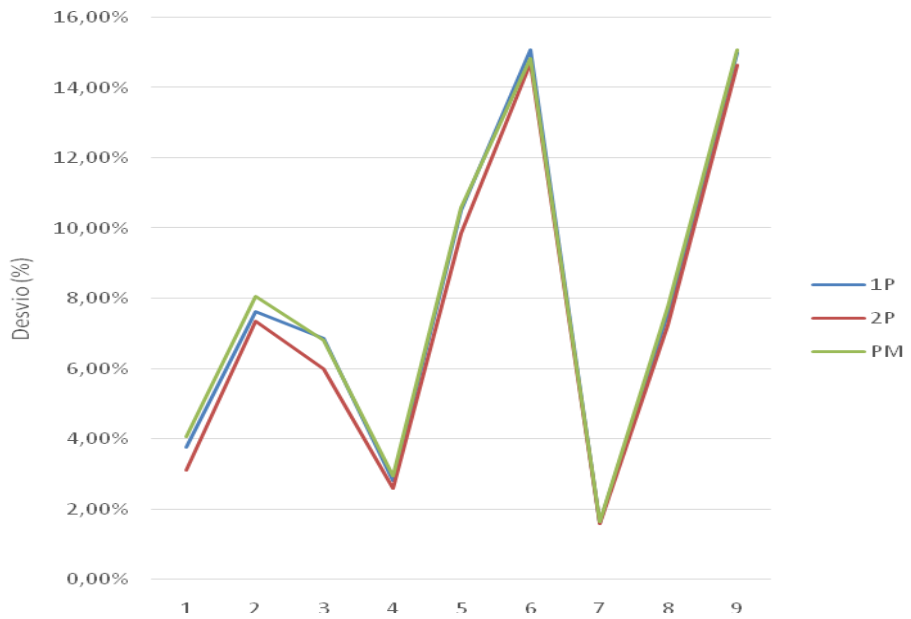
Fonte: Elaborada pelo autor.

A partir de análises da Tabela 6.10 pode-se concluir algumas observações:

a) O operador 2P apresentou melhor desempenho médio e melhor resultado em todas as nove classes testadas. Este mesmo operador obteve cinco melhores mínimos e sete menores máximos. Contudo, apresentou também tempo computacional mais elevado.
b) O operador PM obteve os melhores tempos computacionais nas 6 primeiras classes. O melhor desempenho de um problema teste se deu pelo operador PM, problema ta067, onde chegou-se ao resultado ótimo. Este mesmo operador obteve três melhores mínimos e apenas um melhor máximo.
c) Até a 6ª Classe o operador PM apresentava-se como o mais rápido, contudo, da 7ª em diante gastou mais tempo computacional que o 1P.

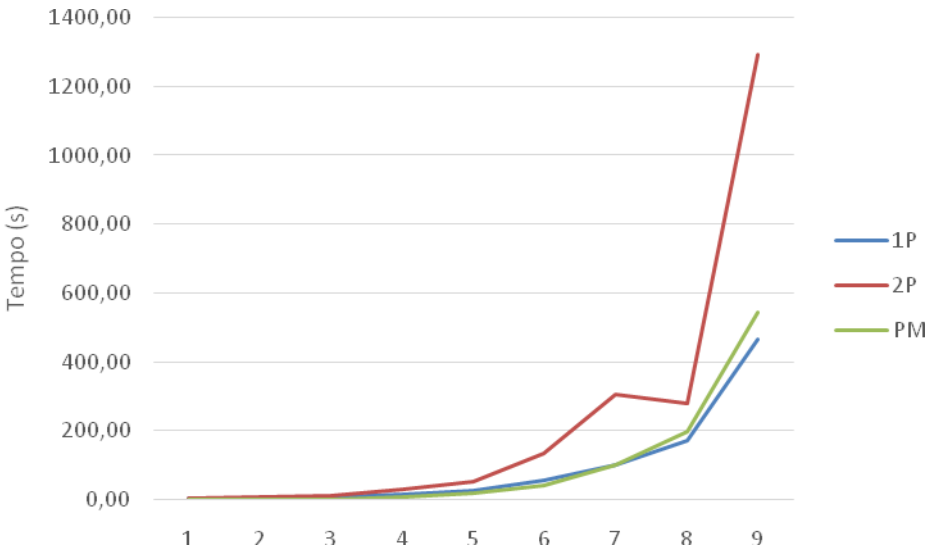
As Figuras 6.1 e 6.2 ilustram mais detalhadamente os resultados tabulados da Tabela 6.10, mostrando mais precisamente o comportamento de cada método, através dos valores dos desvios obtidos com os experimentos computacionais, bem como o tempo computacional demandado ao longo dos experimentos, respectivamente.

Figura 6.1 – Desvio médio dos operadores nas classes.



Fonte: Elaborada pelo autor.

Figura 6.2 – Tempo computacional por tipo de operador.



Fonte: Elaborada pelo autor.

A seguir, uma comparação dos resultados do AG proposto será feito com outros métodos discretos de resolução aplicados ao FSP com o *makespan* sendo o critério de desempenho. As Tabelas 6.11 a 6.19 dadas a seguir apresentam as instâncias avaliadas,

os desvios médios dos resultados do AG aqui desenvolvido e das outras heurísticas da literatura (PALM, CDS HP), o valor ótimo do problema (*z**) e o tempo computacional do AG. No final de cada classe, também estão inseridas: média, mínimo, máximo e melhor (número de instâncias em que o método apresentou melhor resultado). Os tempos computacionais aqui apresentados são, em suma, maiores do que os vistos nas tabelas de classes anteriores, pois o Algoritmo Genético executa os três operadores de cruzamento 1P, 2P e PM, nesta ordem. Os melhores resultados destes três operadores vão gerar os indivíduos da população seguinte. O tempo computacional dos outros métodos não foram levados em consideração por terem sido executados em máquinas distintas.

Tabela 6.11 – Comparação dos resultados do AG com outros métodos na classe 1 (n= 20, m=5).

Instância	*z**	PAL	CDS	HP	AG	AG (z)	AG (seg)
ta001	1278	8,29	8,76	4,93	**0,63**	1286	5,04
ta002	1359	5,89	4,78	3,24	**0,52**	1366	3,38
ta003	1081	**7,49**	15,54	11,38	7,59	1163	2,80
ta004	1293	15,24	9,67	11,83	**2,32**	1323	4,62
ta005	1235	10,12	7,13	9,47	**0,73**	1244	4,00
ta006	1195	12,47	9,79	13,47	**5,52**	1261	6,10
ta007	1239	12,99	12,43	7,18	**2,26**	1267	3,91
ta008	1206	8,87	11,19	10,12	**2,65**	1238	5,31
ta009	1230	15,93	10,57	8,13	**3,82**	1277	3,44
ta010	1108	10,92	5,05	10,92	**1,53**	1125	6,07
Média		10,82	9,49	9,07	**2,76**		4,47
Mínimo		5,89	4,78	3,24	**0,52**		2,80
Máximo		15,93	15,54	13,47	**7,59**		6,10
Melhores		1	0	0	**9**		

Fonte: Elaborada pelo autor.

A Tabela 6.11 mostra que, na 1ª classe, há uma grande superioridade do AG em relação a outras heurísticas, o mesmo apresentou 9 melhores resultados para os 10 problemas da classe. O desvio médio geral foi de 2,76% e o tempo médio de processamento foi 4,47 segundos. O único problema onde o AG não obteve melhor

resultado foi o ta003, onde a heurística PALM obteve resultado ligeiramente melhor, contudo esta heurística foi a que apresentou pior desvio médio geral (10,82%).

Tabela 6.12 – Comparação dos resultados do AG com outros métodos na classe 2 (n= 20, m=10).

Instância	*z**	PAL	CDS	HP	AG	AG (z)	AG (seg)
ta011	1582	13,15	11,06	12,83	**6,07**	1678	10,73
ta012	1659	17,42	11,75	8,62	**6,63**	1769	8,11
ta013	1496	15,57	9,96	12,83	**5,28**	1575	6,11
ta014	1377	15,11	12,35	13,44	**6,68**	1469	9,01
ta015	1419	16,14	9,80	15,57	**8,74**	1543	5,99
ta016	1397	9,31	13,89	13,89	**8,38**	1514	5,43
ta017	1484	16,91	9,84	14,69	**5,66**	1568	6,20
ta018	1538	14,63	16,25	16,84	**7,48**	1653	6,97
ta019	1593	15,25	7,97	9,92	**3,83**	1654	6,75
ta020	1591	19,3	18,42	13,89	**9,43**	1741	7,45
Média		15,28	12,13	13,25	**6,82**		7,28
Mínimo		9,31	7,97	8,62	**3,83**		5,43
Máximo		19,30	18,42	16,84	**9,43**		10,73
Melhores		0	0	0	**10**		

Fonte: Elaborada pelo autor.

Na 2ª classe, conforme Tabela 6.12, dada acima, o resultado do AG foi ainda melhor, quando comparado com os demais métodos, pois apresentou melhores resultados em todas as classes do problema. O tempo de processamento médio para o AG foi de 7,28 segundos.

A Tabela 6.13 mostra os resultados da 3ª classe, onde se percebe que continua uma predominância da heurística AG, com desvio médio geral de 5,74% e 9 melhores resultados para a classe. A segunda melhor heurística, CDS, apresentou apenas 1 melhor resultado e desvio médio de 9,64%, desempenho consideravelmente inferior em relação ao AG. O problema ta022 foi o único da classe onde o AG não foi mais eficiente. O tempo de processamento médio e máximo para a classe foram, respectivamente, 15,20 e 22,28 segundos.

Tabela 6.13 – Comparação dos resultados do AG com outros métodos na classe 3 (n= 20, m=20).

Instância	*z**	PAL	CDS	HP	AG	AG (z)	AG (seg)
ta021	2297	22,68	11,41	7,71	**5,05**	2413	6,66
ta022	2099	11,05	**8,86**	11,77	10,15	2312	11,39
ta023	2326	15,13	10,28	7,39	**5,98**	2465	16,03
ta024	2223	18,26	9,49	11,88	**3,73**	2306	18,16
ta025	2291	18,03	9,38	11,79	**4,23**	2388	22,28
ta026	2226	15,54	8,81	10,2	**6,56**	2372	15,66
ta027	2273	8,05	9,5	11,88	**6,03**	2410	18,22
ta028	2200	10,68	7,36	11,5	**3,59**	2279	19,74
ta029	2237	23,11	7,91	11,31	**5,63**	2363	15,23
ta030	2178	20,89	13,36	15,52	**6,47**	2319	8,63
Média		16,34	9,64	11,10	**5,74**		15,20
Mínimo		8,05	7,36	7,39	**3,59**		6,66
Máximo		23,11	13,36	15,52	**10,15**		22,28
Melhores		0	1	0	**9**		

Fonte: Elaborada pelo autor.

Tabela 6.14 – Comparação dos resultados do AG com outros métodos na classe 4 (n= 50, m=5).

Instância	*z**	PAL	CDS	HP	AG	AG (z)	AG (seg)
ta031	2724	1,84	3,38	4,88	**0,66**	2742	53,92
ta032	2834	7,3	6,99	4,94	**2,54**	2906	39,17
ta033	2621	5,95	3,13	5,61	**2,37**	2683	52,62
ta034	2751	3,96	4,83	6,94	**3,42**	2845	61,70
ta035	2863	3,49	6,11	5,38	**0,80**	2886	47,26
ta036	2829	9,23	7,14	6,61	**1,45**	2870	44,27
ta037	2725	4,4	8,95	6,97	**3,05**	2808	61,98
ta038	2683	5,33	5,67	10,59	**0,00**	2683	34,88
ta039	2552	7,09	9,09	6,11	**3,10**	2631	35,33
ta040	2782	4,78	5,75	7,44	**1,58**	2826	42,32
Média		5,34	6,10	6,55	**1,90**		47,35
Mínimo		1,84	3,13	4,88	**0,00**		34,88
Máximo		9,23	9,09	10,59	**3,42**		61,98
Melhores		0	0	0	**10**		

Fonte: Elaborada pelo autor.

Os experimentos realizados na 4ª classe, descrito na Tabela 6.14, dada acima, constata-se a predominância total do AG sobre as demais heurísticas da classe,

inclusive, obtendo um resultado ótimo no problema ta038. O desvio médio do AG nessa classe foi de 1,90%, frente a 5,34% da heurística PAL, segundo melhor método nesta classe. O tempo computacional médio e máximo foi de 47,35 e 61,98 segundos, respectivamente.

Tabela 6.15 – Comparação dos resultados do AG com outros métodos na classe 5 (n= 50, m=10).

Instância	z*	PAL	CDS	HP	AG	AG (z)	AG (seg)
ta041	2991	16,28	14,38	17,42	**9,73**	3282	100,30
ta042	2867	15,56	13,22	15,7	**10,29**	3162	55,11
ta043	2839	16,98	15,53	19,51	**12,12**	3183	64,64
ta044	3063	14,63	10,77	12,41	**7,90**	3305	60,76
ta045	2976	15,15	13,41	18,48	**10,15**	3278	109,87
ta046	3006	10,55	13,11	16,63	**9,81**	3301	68,75
ta047	3093	11,77	13,81	15,55	**7,86**	3336	79,04
ta048	3037	10,5	11,52	15,9	**8,96**	3309	64,11
ta049	2897	17,85	12,22	18,23	**9,80**	3181	80,91
ta050	3065	11,06	11,88	18,66	**9,00**	3341	60,04
Média		14,03	12,99	16,85	**9,56**		74,35
Mínimo		10,50	10,77	12,41	**7,86**		55,11
Máximo		17,85	15,53	19,51	**12,12**		109,87
Melhores		0	0	0	**10**		

Fonte: Elaborada pelo autor.

Os dados da Tabela 6.15 mostram os experimentos realizados na 5ª classe. O AG apresentou todos os melhores resultados, com desvio médio de 9,56%, desvio mínimo de 7,86% e desvio máximo de 12,12%. O tempo computacional médio demandado foi de 74,35 segundos. A heurística CDS foi a que apresentou segundo menor desvio médio geral, com 12,99%.

Os comparativos da 6ª classe de problemas estão na Tabela 6.16, dada a seguir, onde pode ser visto uma perda parcial de eficiência do AG. Até as classes anteriores, o AG apresentava quase total superioridade em eficiência. Aqui, vê-se que ele apresentou 6 dos melhores resultados e desvio médio de 14,44%, enquanto o CDS apresentou 3 melhores resultados e desvio médio de 15,77%, além de ter apresentado, no problema ta057, o menor desvio da classe para todas as heurísticas, com 12,58%.

Tabela 6.16 – Comparação dos resultados do AG com outros métodos na classe 6 (n= 50, m=20).

Instância	*z**	PAL	CDS	HP	AG	AG (z)	AG (seg)
ta051	3771	13,29	**14,77**	21,06	14,90	4333	185,32
ta052	3668	17,31	14,94	18,87	**13,47**	4162	129,03
ta053	3591	17,24	16,65	20,11	**13,95**	4092	126,32
ta054	3635	16,45	17,74	25,25	**15,02**	4181	127,06
ta055	3553	23,16	**16,01**	19,34	16,10	4125	136,75
ta056	3667	17,59	16,36	18,3	**13,64**	4167	187,37
ta057	3672	17,27	**12,58**	22,17	13,18	4156	199,19
ta058	3627	18,83	17,51	23,24	**14,64**	4158	155,76
ta059	3645	24,75	15,56	22,41	**14,81**	4185	139,45
ta060	3696	**13,56**	15,53	24,95	14,64	4237	148,34
Média		17,95	15,77	21,57	**14,44**		153,46
Mínimo		13,29	**12,58**	18,30	13,18		126,32
Máximo		24,75	17,74	25,25	**16,10**		199,19
Melhores		1	3	0	**6**		

Fonte: Elaborada pelo autor.

Tabela 6.17 – Comparação dos resultados do AG com outros métodos na classe 7 (n= 100, m=5).

Instância	*z**	PAL	CDS	HP	AG	AG (z)	AG (seg)
ta061	5493	4,66	1,8	3,6	**0,62**	5527	146,55
ta062	5268	**0,91**	5,6	6,57	1,75	5360	236,27
ta063	5175	2,9	6,14	7,75	**1,80**	5268	308,26
ta064	5014	**0,7**	5,17	7,08	1,32	5080	391,75
ta065	5250	**1,28**	4,02	9,64	2,25	5368	189,28
ta066	5135	2,71	2,41	5,63	**0,19**	5145	184,57
ta067	5246	2,48	5,93	7,89	**2,21**	5362	228,79
ta068	5034	4,55	7,01	8,82	**2,82**	5176	472,38
ta069	5448	2,9	5,69	7,95	**1,36**	5522	171,24
ta070	5322	1,97	7,53	6,86	**1,54**	5404	269,57
Média		2,51	5,13	7,18	**1,59**		259,87
Mínimo		0,70	1,80	3,60	**0,19**		146,55
Máximo		4,66	7,53	9,64	**2,82**		472,38
Melhores		3	0	0	**7**		

Fonte: Elaborada pelo autor.

A Tabela 6.17 mostra os resultados comparativos da 7ª classe do problema. O AG continua sendo o método mais eficaz, com 7 melhores resultados e desvio médio geral de 1,59%. Essa classe apresenta pequenos desvios médios em relação à classe anterior, aproximando-se mais do resultado ótimo, isso para todas as heurísticas. Contudo, o tempo computacional cresceu bastante, apresentando tempo computacional médio e máximo demandado de, respectivamente, 259,87 e 472,38 segundos.

Tabela 6.18 – Comparação dos resultados do AG com outros métodos na classe 8 (n= 100, m=10).

Instância	*z**	PAL	CDS	HP	AG	AG (z)	AG (seg)
ta071	5770	**6,78**	7,61	14,49	7,05	6177	79,05
ta072	5349	10,10	9,80	16,06	**7,93**	5773	434,32
ta073	5676	7,95	**6,13**	12,4	6,71	6057	668,16
ta074	5781	**9,20**	10,31	16,62	10,17	6369	516,15
ta075	5467	11,03	10,08	14,87	**8,87**	5952	410,64
ta076	5303	10,69	8,32	15,39	**8,03**	5729	62,59
ta077	5595	15,14	10,83	12,89	**5,70**	5914	422,66
ta078	5617	9,81	10,98	12,89	**7,41**	6033	499,53
ta079	5871	**3,58**	8,14	11,77	5,01	6165	309,05
ta080	5845	7,08	9,27	11,91	**5,17**	6147	321,03
Média		9,14	9,15	13,93	**7,21**		372,32
Mínimo		**3,58**	6,13	11,77	5,01		62,59
Máximo		15,14	10,98	16,62	**10,17**		668,16
Melhores		3	1	0	**6**		

Fonte: Elaborada pelo autor.

Os resultados dos experimentos computacionais realizados para a penúltima classe estão apresentados na Tabela 6.18, dada acima, onde se vê, mais uma vez, o bom desempenho do AG, porém com uma pequena redução de sua eficiência em comparação com os desempenhos anteriores, obtendo 6 melhores resultados e desvio médio geral de 7,21%, seguido do PAL, que apresentou 3 melhores resultados e desvio médio de 9,14%. Nesta classe o tempo de execução do AG foi considerado alto com média de 372,32 segundos, tendo tempo máximo de 668,16 segundos (quase 12 minutos).

Tabela 6.19 – Comparação dos resultados do AG com outros métodos na classe 9. (n= 100, m=20).

Instância	*z**	PAL	CDS	HP	AG	AG (z)	AG (seg)
ta081	6106	15,87	**13,33**	22,57	15,44	7049	733,58
ta082	6183	14,15	**12,84**	18,76	14,07	7053	943,95
ta083	6252	15,5	15,63	20,73	**13,71**	7109	1195,73
ta084	6254	**12,55**	12,92	19,28	13,19	7079	992,26
ta085	6262	15,92	**13,59**	19,45	14,28	7156	1270,73
ta086	6302	**12,81**	15,57	19,22	13,84	7174	1588,40
ta087	6184	17,71	**15,57**	24,18	16,14	7182	1170,41
ta088	6315	19,73	**14,57**	25,1	16,56	7361	1484,18
ta089	6204	17,2	15,99	23,16	**14,73**	7118	973,29
ta090	6404	14,07	**11,87**	20,61	12,84	7226	920,05
Média		15,55	**14,19**	21,31	14,48		1127,26
Mínimo		12,55	**11,87**	18,76	12,84		733,58
Máximo		19,73	**15,99**	25,10	16,56		1588,40
Melhores		2	**6**	0	2		

Fonte: Elaborada pelo autor.

A 9ª classe de instâncias mostra que o AG já não é mais a melhor heurística de solução do problema, apresentando somente 2 melhores resultados e desvio médio geral de 14,48%. A melhor heurística para essa classe foi a CDS, com 6 melhores resultados e desvio médio de 14,19%, mostrando que o AG perde um pouco da sua eficiência. A heurística PAL teve desempenho semelhante ao AG, com 2 melhores resultados e desvio médio geral de 15,55%. CDS também apresentou o menor desvio da classe para todas as heurísticas. Os tempos computacionais médio e máximo do AG foram de 733,58 e 1588,40 segundos.

A Tabela 6.20 apresenta uma síntese dos resultados mais relevantes das tabelas nove tabelas dadas anteriormente de comparação dos métodos de resolução do FSP. Nela, são dados os valores, por classe, das médias, mínimos e máximos e melhores (número de instâncias em que o método obteve melhor resultado), separados por Heurística. Em negrito, encontram-se os melhores resultados que se destacaram naquela classe.

Tabela 6.20 – Resumo dos experimentos computacionais.

Método	Classe	1	2	3	4	5	6	7	8	9	Geral
PALM	Média	10,82	15,28	16,34	5,34	14,03	17,95	2,51	9,14	15,55	11,88
	Mínimo	5,89	9,31	8,05	1,84	10,50	13,29	0,70	**3,58**	12,55	0,70
	Máximo	15,93	19,30	23,11	9,23	17,85	24,75	4,66	15,14	19,73	24,75
	Melhores	1	0	0	0	0	1	3	3	2	10
CDS	Média	9,49	12,13	9,64	6,10	12,99	15,77	5,13	9,15	**14,19**	10,51
	Mínimo	4,78	7,97	7,36	3,13	10,77	**12,58**	1,80	6,13	**11,87**	1,80
	Máximo	15,54	18,42	13,36	9,09	15,53	17,74	7,53	10,98	**15,99**	18,42
	Melhores	0	0	1	0	0	3	0	1	**6**	11
HP	Média	9,07	13,25	11,10	6,55	16,85	21,57	7,18	13,93	21,31	13,42
	Mínimo	3,24	8,62	7,39	4,88	12,41	18,30	3,60	11,77	18,76	3,24
	Máximo	13,47	16,84	15,52	10,59	19,51	25,25	9,64	16,62	25,10	25,25
	Melhores	0	0	0	0	0	0	0	0	0	0
AG	Média	**2,76**	**6,82**	**5,74**	**1,90**	**9,56**	**14,44**	**1,59**	**7,21**	14,48	**7,16**
	Mínimo	**0,52**	**3,83**	**3,59**	**0,00**	**7,86**	13,18	**0,19**	5,01	12,84	**0,00**
	Máximo	**7,59**	**9,43**	**10,15**	**3,42**	**12,12**	**16,10**	**2,82**	**10,17**	16,56	**16,56**
	Melhores	**9**	**10**	**9**	**10**	**10**	**6**	**7**	**6**	**2**	**69**
	Tempo Médio	4,47	7,28	15,20	47,35	74,35	153,46	259,87	372,32	1127,26	229,06

Fonte: Elaborada pelo autor.

A seguir, far-se-ão algumas conclusões a partir da análise da Tabela 6.20:

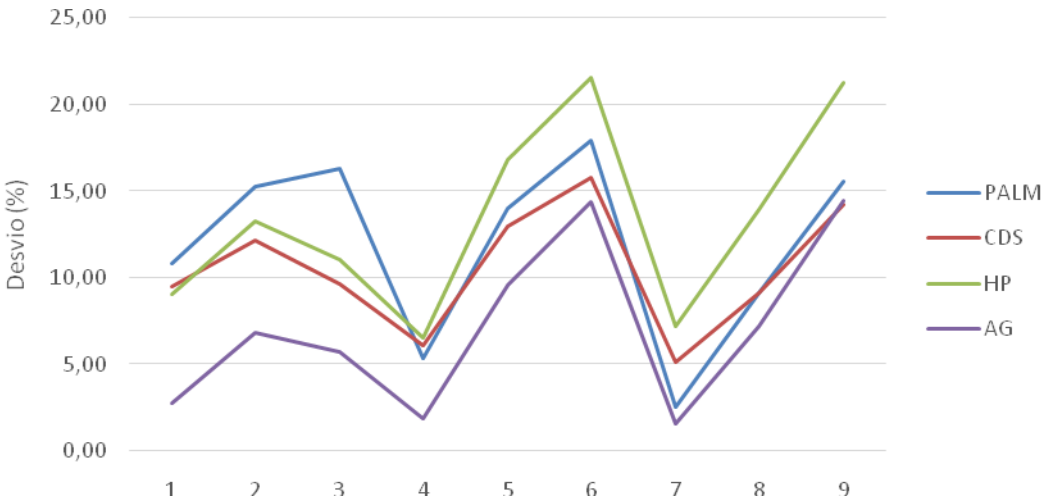
a) A heurística que obteve melhor desempenho geral foi o AG, com média de desvios de 7,16%, apresentando também desvio mínimo de 0,00% e menor desvio máximo de 16,56 %.
b) AG apresentou seus melhores resultados em problemas de pequeno, médio e grande porte, onde em várias classes (2, 4 e 5) apresentou os melhores resultados em todas os problemas testados.
c) A partir da 6ª classe, o método começa a reduzir sua eficiência quando comparado com os outros métodos de resolução do problema. Na última classe os melhores resultados foram apresentados pelo método CDS.
d) O melhor resultado do AG foi na instância ta038, na classe 4, onde se chegou ao resultado ótimo do problema.
e) O método HP não apresentou melhor resultado em nenhuma das classes.
f) O tempo computacional de AG, para problemas de maior porte, é considerável aceitável, pois o tempo máximo foi de 1588,4 segundos

(26,47 minutos) para um problema com n=100 e m=20. Isso significa que o método se adapta muito bem a problemas práticos.

g) A melhor solução mais rápida do problema foi obtida no problema ta002, com 3,38 segundos.

A Figura 6.3 mostra, graficamente, o desempenho dos desvios para os quatro métodos, onde é bem significativo a melhora dos resultados obtidos pelo o método aqui desenvolvido.

Figura 6.3 – Desvio médio dos métodos de resolução do problema.



Fonte: Elaborada pelo autor.

Portanto, neste capítulo descrevemos os resultados computacionais obtidos através dos vários testes realizados. Ao longos das tabelas, desenvolveu-se conclusões parciais das classes e, por fim, foi feita uma análise global com as "tabelas-resumo". No próximo capítulo será desenvolvida a análise global dos resultados com a conclusão do trabalho.

CAPÍTULO VII

CONCLUSÕES

Os experimentos computacionais, baseados em problemas da literatura, indicam que o AG obteve sucesso na resolução do problema proposto. A diversificação na busca por boas soluções tende a gerar bons resultados mesmo que a busca se torna exaustiva. Pode-se afirmar que esse é um procedimento adequado e barato computacionalmente, que exige poucos recursos computacionais, para serem aplicados na resolução do problema FSP.

A análise das soluções do AG mostrou que ele tem a capacidade de melhorar bastante a qualidade das soluções, mesmo depois que a solução já ter atingido uma boa qualidade através de seus operadores genéticos.

O objetivo geral deste trabalho é relacionar as similaridades do Problema de Sequenciamento Clássico como um Problema do Caixeiro Viajante e resolvê-lo utilizando a metaheurística Algoritmo Genético moderno aplicado no FSP via PCV, bem como comparar os resultados obtidos pelo algoritmo proposto com os resultados encontrados na literatura, nos capítulos 2, 3 e 4 nos dedicamos a explanação do estado da arte desses três temas, e no capítulo 4 nós fizemos a correlação entre os mesmos.

No capítulo 5, foi descrito a natureza do algoritmo que o trabalho estava propondo, com a respectiva transformação de instâncias para posterior aplicação do AG. No capítulo 6, descreveu-se os experimentos computacionais. O AG desenvolvido para uso no PCV tem como finalidade obter um sequenciamento ideal de máquinas, o alvo é maximização da eficiência total da sequência através, somente, da Matriz de Tempo de Processamento. Em 69 dos 90 problemas testados nas instâncias de Taillard, o AG apresentou melhores soluções que os outros métodos discretos de resolução do problema. A perda de eficiência do método se deus em poucas instâncias. Contudo, o AG mostrou ser um método muito eficiente e rápido nos problemas de pequeno e médio porte (n<=50). Também no capítulo 6, foi mostrado a comparação dos resultados obtidos pelo nosso método, com os de outras heurísticas da literatura.

É reconhecido que ainda há muito para explorar nesse campo de pesquisa, o presente trabalho pretende ter dado sua contribuição. Dentre os muitos contextos ainda

não explorados na literatura tem-se como sugestões de trabalhos futuros: aplicar o AG em problemas reais para verificar o seu desempenho; aplicar o AG em outros problemas da classe POCP e comparar o seu resultado com os melhores métodos desses problemas; testar outros valores para os parâmetros do AG, inclusive outros operadores genéticos; paralelizar o AG quando for tratar os problemas de grande porte, através do processamento distribuído com populações iniciais diferentes para cada processador envolvido na paralelização do método.

REFERÊNCIAS BIBLIOGRÁFICAS

AARTS, E.; LENSTRA, J.K. **Local search in combinatorial optimization**. Wiley Interscience, Chichester, England. 1997.

ALI, H.; AL-DULAIMI, B. **A Novel Genetic Algorithm Approach for Solving Flow Shop Problem**. International Journal of Computer and Network Security, Vol 8. N. 9, 229-235, 2008.

ALMEIDA. M.S. **Elaboração de Projeto, TCC, dissertação e tese:** uma abordagem simples, prática e objetiva. Atlas, SP. 2011.

APPLEGATE, D.; Bixby, R.; Chvátal, V.; Cook, W.. **Finding cuts in the TSP (A preliminary report)**. DIMACS Technical Report, 1995.

_____. **The Traveling Salesman Problem**: **A Computational Study**. Princeton University Press. 2006.

APPLEGATE, D. L.; BIXBY, R.E.; CHVÁTAL, V.; COOK, W. J.; ESPINOZA, D. G.; GOYCOOLEA, M.; HELSGAUN, K.. **Certification of an optimal TSP tour through 85900 cities**. In: Operations Research Letters., Vol. 37, No. 1, pp. 11–15, 2009.

AVNER P.; BRULS T.; PORAS I.; ELEY L.; GAS S.; RUIZ P.; WILES M. V.; SOUSA-NUNES R.; KETTLEBOROUGH R.; RANA A.; MORISSETTE J.; BENTLEY L.; GOLDSWORTHY M.; HAYNES A.; HERBERT E.; SOUTHAM L.; LEHRACH H.; WEISSENBACH J.; MANENTI G.; RODRIGUEZ-TOME P.; BEDDINGTON R.; DUNWOODIE S.; COX R. D. . **A radiation hybrid transcript map of the mouse genome**. Nature Genetics 29: 194–200, 2001.

BAKER, B. M. **A genetic algorithm for the vehicle routing problem**. Computers & Operations Research, v. 30, n. 5, p. 787-800, 2003.

BEZERRA, F.P.; **Um algoritmo genético aplicado no Problema de Roteirização Periódica de Veículos com caso prático.** Dissertação (Mestrado). Universidade Federal do Ceará, Fortaleza, 2012.

BLICKLE, T. Tournament Selection. Separata de: BACK, T.; FOGEL, D.; ICHALEWICZ, Z. (Ed.). **Handbook of Evolutionary Computation**. Oxford: Oxford University Press, 1997.

BLUM, C.; ROLI, A. **Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison**. Computing Surveys, v. 35, n. 3, p. 268-308, 2003.

BALAS, E.; TOTH, P.. **Branch and bound methods. In The Traveling Salesman Problem**. Edited by Lawler, E. L.; Lenstra, J. K.; Rinnooy Kan, A. H. G. and Shmoys, D. B. John Wiley & Sons Ltd, 361-401, 1985.

BALLOU, H.. **Gerenciamento da Cadeia de Suprimentos: Planejamento, Organização e Logística Empresarial**. Porto Alegre: Bookman, 2001.

BEKTAS, T. **The multiple traveling salesman problem: an overview of formulations and solution procedures**. Omega.34, pp. 209–219, 2006.

BERBEGLIA, G.; CORDEAU, J.; LAPORTE, G.. **Dynamic pickup and delivery problems**. In: European Journal of Operational Research, 2009.

BLAND, R. E.; SHALLCROSS, D. F.. **Large Traveling Salesman Problem Arising from Experiments in X-ray Crystallography: a Preliminary Report on Computation**, Relatório Técnico No. 730, School of OR/IE, Cornell University, Ithaca, New York, 1987.

BODIN, L. D.; GOLDEN, B.; ASSAD, A.; BALL, M.. **Routing and scheduling of vehicle and crews: The state of the art**. Computers & Operations Research, v.10, n.2, p.63-211, 1983.

BRUCKER, P. **Scheduling Algorithms**. Springer-Verlag, Berlin, 1998.

COOK, S. A. **The complexity of theorem-proving procedures**. In: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, ACM, New York, NY, USA, pp. 151-158, 1971.

CHEN, C.-L.; VEMPATI, V.S.; ALJABER, N.. **An application of genetical gorithms for flow shop problems**. European Journal of Operational Research, v.80, p.389-396, 1995

CAMPBELL, H.G.; DUDEK, R.A.; SMITH, M.L.. **An heuristic algorithm for n job m machine sequencing problem**. Management Science, v.16, p.630-637, 1970

CONWAY, R.W.; MAXWELL, W.L.; MILLER, L.W. **Theory of scheduling**. Reading, MA: Addison-Wesley, 1967.

DANNENBRING, D.G.. **An evaluation of flow shop sequencing heuristics**. Management Science, v.23, p.1174-1182,1977

DAVIS, L. D. **Adapting operator probabilities in genetic algorithms**. Separata de: SCHAFFER, J. D. (Ed.). Proceedings of the Third International Conference on Genetic Algorithms, 1989.

DESROCHERS, M.; LENSTRA, J.K.; SAVELSBERGH, M.W.P.. **A classification scheme for the vehicle routing and scheduling problems**. European Journal of Operational Research, v.46, n.3, p.322-332, 1990.

DETOFENO, T. C.; STEINER, M. T. A.. **Optimizing Routes for the Collection of Urban Solid Waste: a Case Study for the city of Joinville**. In: IJIE (Iberoamerican Journal of Industrial Engineering), v. 2, pp. 124-136, 2010.

DREXL, M. **Rich vehicle routing in theory and practice**. In: Logistics Research, Volume 5, Issue 1-2 , pp. 47-63 , 2012.

DUMITRESCU, I.; STÜTZLE, T.**A survey of methods that combine local search and exact algorithms**. Technical Report AIDA-03-07, FG Intellektik, FB Informatik, TU Darmstadt, Germany, 2003.

DUDEK, R.A.; TEUTON Jr., O.F.. **Development of m-stage decision rule for scheduling n jobs through m machines**. Operations Research, v.12, p.471-497,1964

ERDOGAN, G.; CORDEAU, J.-F.; LAPORTE, G. .**A Branch-and-Cut Algorithm for the Non- Preemptive Capacitated Swapping Problem**. In: 24th European Conference on Operational Research, Lisbon, Portugal, 2010.

GAREY, M.R.; JOHNSON, D.S. **Computers and Intractability: a Guide to the Theory of NP-completeness**. W.H. Freeman and Company, San Francisco, CA, 338 pp., 1979.

GEN, M. **Genetic Algorithms and Their Applications**. Springer Handbook of Engineering Statistics**.** Springer London.p. 749-773, 2006.

GODOY, A. S. **Introdução a pesquisa qualitativa e suas possibilidades.** Revista de Administração de Empresas. São Paulo, v.35, n.2, p. 57-63. 1995.

GOLDBERG, D.E. **Genetic Algorithms in Search, Optimization, and Machine Learning**. Addison Wesley, Reading, MA, 372 pp., 1989.

GOMES, F. R. A. **Algoritmo genético aplicado aos problemas de sequenciamento Flowshop sem e com restrição de espera**. 141 f. Dissertação (Mestrado) – Universidade Federal do Ceará, Fortaleza, 2008.

GONNET, G.; KOROTENSKY, C.; BENNER, S.. **Evaluation measures of multiple sequence alignments**. In: Journal of Computational Biology 7(1-2), 261-276, 2000.

GRÖTSCHEL,M.. **Polyedrische Charakteri sierungen kombinatoris cher Optimierungs probleme**. Anton Hain Verlag, Meisenheim/Glan, 1977.

GRAHAM, R.L.; LAWLER, E.L.; LENSTRA, J.K.; KAN, A.H.G.R.. **Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics**, 5:287–326, 1979

GUPTA, J.N.D.; STAFFORD Jr, E.F.. **Flowshop scheduling research after five decades**. European Journal of Operational Research, v.169, p.699–711, 2006.

HELD, M.; KARP, R.M..**The traveling-salesman problem and minimum spanning trees: part II**. Mathematical Programming, Vol. 1, pp. 6-25, 1971.

HOFFMAN, K. ; PADBERG, M. .**Traveling Salesman Problem**. Karla Hoffman home page, Encyclopedia Articles. Disponível na Internet em http://iris.gmu.edu/%7Ekhoffman/papers/trav_salesman.html, acessado em 21 maio de 2014.

HOLLAND, J.H. **Adaptation in natural artificial systems**. University of Michigan Press, Michigan, 1975.

JOHNSON, S.M.. **Optimal two- and three-stage production schedules with setup times included**. Naval Research logistics Quarterly1, p.61–68, 1954.

KORTE, B. H. **Applications of combinatorial optimization**. In: Mathematical Programming: Recent Developments and Applications, M. Iri and K. Tanabe, Eds., Kluwer, Dordrecht, p.1-55, 1989.

KOZA, J. R. **Genetic Programming:** On the Programming of Computers by Means of Natural Selection. Cambridge: MIT Press, 1992.

LAND, A.H.; DOIG, A.G.. **An automatic method for solving discrete programming problems**. Econometrica 28, 497–520, 1960.

LAWLER, E.L.; LENSTRAS, J.K.; RINNOOY KAN, A.H.G.; SHMOYS, D.B., **Sequencing and scheduling: Algorithms and complexity**. In: Graves, S.C. (Ed.),Handbooks in Operations Research and Management Science, Vol. 4. Elsevier Science Publishers, Amsterdam, pp. 445–552,1993.

LOMNICKI, Z.A. **A branch and bound algorithm for the exact solution of the three machine scheduling problem**. Operational Research Quarterly, v.16, p.89-100, 1965.

MANNE, A. **On the job-shop scheduling problem**. Operations Research, v.8,p.219–223,1960.

MAREDIA,A.. **History, Analysis, and Implementation of Traveling Salesman Problem (TSP) and Related Problems**. Department of Computer and Mathematical Sciences. University of Houston-Downtown, 2010.

MERZ, *P.* **Memetic algorithms for combinatorial optimization problems: Fitness landscapes and effective search strategies**. Parallel Systems Research Group. Department of Electrical Engineering and Computer Science. University of Siegen.Ph.D. Thesis, 2000.

MITCHELL, M. **An introduction to genetic algorithms.** Cambridge: MIT Press, 1998.

MUTH, J.; THOMPSON, G.L. **Industrial Scheduling**. Prentice-Hall, Englewood Cliffs,New Jersey, 1963.

MURATA, T.; ISHIBUCHI, H.; TANAKA, H. **Genetic algorithms for flowshop scheduling problems**. Computers& Industrial Engineering, v.30, p.1061-1071, 1996.

NAWAZ, M.; ENSCORE, E.; HAM, I.**A heuristic algorithm for the m-machine,n-job flowshop sequencing problem**. OMEGA. The International Journal of Management Sciences, v.11, p.91-95, 1983.

REEVES, C. **Modern heuristic techniques for combinatorial problems**. Berkshire: McGraw Hill Book Company, 1993.

REEVES, C.R.**A genetic algorithm for flow shop sequencing**. Computers and Operations Research, v.22, p.5-13, 1995.

RONEN, D. **Perspectives on practical aspects of truck routing and scheduling**. European Journal of Operational Research, v.35, n.2, p.137-145, 1988.

RUIZ, R.; MAROTO, C.; ALCARAZ, J..**Two new robust genetic algorithms for the flowshop scheduling problem**. The International Journal the Management Science(Omega), v.34, p.461-476, 2006.

SILVA, J.L.C.; SOMA, N.Y. **Um método heurístico aplicado no problema de programação flow shop permutacional**. In: XXXVIII Simpósio Brasileiro de Pesquisa Operacional,2006, Goiânia. Anais do XXXVIII SBPO, 2006.

SILVA, B.C.H.; **Otimização de rotas utilizando abordagens heurísticas em um ambiente georeferenciado.** Mestrado (Dissertação), Universidade Estadual do Ceará, Fortaleza, 2013.

SISSON, R.L. **Methods of sequencing in job shops - a review**. Operations Research, v.7, p.10–29, 1959.

SUBRAMANIAN, A.; PENNA, P. H. V.; UCHOA, E.; OCHI, L. S..**A Hybrid Algorithm for the Fleet Size and Mix Vehicle Routing Problem**. International Conference on Industrial Engineering and Systems Management, 2011.

SZWARCCFITER, J. L. .**Grafos e Algoritmos Computacionais**. Editora Campus Ltda, 1984.

T'KINDT, V.; BILLAUT, J.-C. **Multi-criteria scheduling**. Springer-Verlag, Berlin, 1993.

TORIELLO, A..**A Dynamic Traveling Salesman Problem with Stochastic Arc Costs**.In: INFORMS Annual Meeting. Phoenix, Arizona, 2012.

VIANA, G. V. R. **Meta-heurísticas e programação paralela em otimização combinatória**. Fortaleza: UFC, 1998.

VITTES, F. J. **Optimizing the perfomace of a chip shooter machine**. Faculty of the Virginia Polytechnic Institute and State University. Dissertação (Master of Science In Industrial and Systems Engineering), 1999.

WAGNER, H.M.. **An integer linear-programming model for machine scheduling**. Naval Research Logistics Quarterly, v.6, p.131–140, 1959.

WALL, M.B.**A Genetic Algorithm for Resource-Constrained Scheduling.** Department of Mechanical Engineering. Thesis, 1996.

WHITLEY, D.**A Genetic Algorithm Tutorial**. Fort Collins: Colorado State University, 1993

WOLSEY, L.A. **Integer Programming**. John Wiley& Sons, 1998.