UNIVERSIDADE FEDERAL DO AMAZONAS - UFAM
INSTITUTO DE COMPUTAÇÃO - ICOMP
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA - PPGI

INCORPORANDO TÉCNICAS DE MINERAÇÃO DE DADOS A META-HEURÍSTICAS POPULACIONAIS

UNIVERSIDADE FEDERAL DO AMAZONAS - UFAM
INSTITUTO DE COMPUTAÇÃO - ICOMP
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA - PPGI

IVANEIDE ALVES PROTÁSIO

INCORPORANDO TÉCNICAS DE MINERAÇÃO DE DADOS A META-HEURÍSTICAS POPULACIONAIS

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas como requisito parcial para a obtenção do grau de Mestre em Informática.

Orientador: Fabíola Guerra Nakamura

Manaus

Fevereiro de 2014

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Protasio, Ivaneide Alves
P967i Incorporando técnicas de mineração de dados a meta-heurísticas populacionais / Ivaneide Alves Protasio. 2014
72 f.: il. color; 31 cm.

Orientadora: Fabíola Guerra Nakamura
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Meta-heurísticas Populacionais. 2. Problema do Caixeiro Viajante. 3. Mineração de Dados. 4. Algoritmo Genético. I. Nakamura, Fabíola Guerra II. Universidade Federal do Amazonas III. Título

Agradecimentos

Tudo que almejamos na vida depende sempre de muito trabalho e persistência, principalmente, nos momentos difíceis... É aí que contamos com a força que vem do amparo amigo, do incentivo e do companheirismo... Que faz toda a diferença... E pelo que muito venho a agradecer...

A Deus, por tudo que me foi permitido alcançar e viver até este momento e me dar forças nos momentos difíceis.

Agradeço à minha orientadora Prof. Dr^{a}. Fabiola Guerra Nakamura por ser uma fonte de inspiração, crítica e por ter me guiado durante a pesquisa. De uma maneira bem especial quero agradecer aos professores Marco Cristo e Eulanda Miranda que muito contribuíram para este trabalho. A todos os colegas do Laboratório Otimização pela grande convivência e solidariedade sempre que precisei.

Agradeço o apoio fantástico a minha mãe que não pôde ao menos terminar a 4a série, mas que me deu a melhor educação que um filho pode receber (e não falo somente de estudo) e fez de tudo para que eu tivesse as melhores possibilidades. Agradeço profundamente a minha mãe Ivanilde Alves Protásio.

Ao meu amado Fábio, que sempre acreditou e sempre esteve ao meu lado me apoiando, encorajando e *debugando* os meu códigos quando eu queria apagar tudo e começar do zero. Agradeço o seu amor e companheirismo que me traz força para enfrentar qualquer obstáculo.

Por último, não menos importante, à secretaria, em especial, à Elienai por sempre me ajudar com os problemas administrativos. À FAPEAM, por conceder a bolsa de estudos permitindo a realização deste trabalho e à UFAM por me proporcionar passagens e diárias para que eu pudesse participar de conferências.

"Ninguém baterá tão forte quanto à vida.
Porém, não se trata de quão forte pode bater,
se trata de quão forte pode ser atingido
e continuar seguindo em frente.
É assim que a vitória é conquistada."
(Rocky Balboa)

Resumo

Vários problemas do mundo real podem ser modelados como problemas de otimização combinatória. Em geral são problemas complexos e de larga escala, não podendo ser resolvidos por métodos exatos, pois os mesmos necessitariam de tempo computacional impraticável. Desse modo, as meta-heurísticas têm sido amplamente empregadas para a resolução de tais problemas. Duas das principais dificuldades destes métodos são escapar das regiões sub-ótimas e evitar a convergência prematura do algoritmo. Para tentar solucionar estes problema, propõe-se o uso de técnicas híbridas buscando desenvolver estratégias que sejam aplicáveis a diversos algoritmos de otimização.

O presente trabalho investiga a eficiência da incorporação de técnicas de Mineração de Dados (MD) as Meta-heurísticas Populacionais Colônia de Formiga e Algoritmo Genético com o intuito de guiá-las a gerar novas e melhores soluções. Para a validação da proposta, serão utilizados o Problema do Caixeiro Viajante e diferentes versões das meta-heurísticas híbridas serão testadas e analisadas.

A técnica escolhida para guiar a obtenção de novas soluções, a partir dos padrões obtidos com a Mineração de Dados, foi o de Agrupamento de soluções similares, na tentativa de reduzir o espaço de busca em problemas de otimização combinatória. O algoritmos de mineração utilizados são o *K-Means* e o *Ward* que utilizam técnicas de particionamento e hierárquico respectivamente.

Experimentos Computacionais foram realizados com o objetivo de avaliar o uso de MD em Meta-heurísticas Populacionais tradicionais, atai como Algoritmo Genético e Colonia de Formiga. Estes experimentos mostraram que a utilização de padrões minerados podem auxiliar na obtenção de boas soluções em relação as técnicas tradicionais
.

Palavras-chaves: Meta-heurísticas Populacionais, Problema do Caixeiro Viajante, Mineração de Dados.

Abstract

Several real-world problems can be modeled as combinatorial optimization problems. This is are usually complex and large scale problems can not be solved by exact methods , since they would require impractical computational time . Thus, meta-heuristics have been widely used for solving such problems. Two of the major difficulties of these methods are to escape from sub-optimal regions and to avoid premature convergence of the algorithm . To try to solving this problem , we use o hybrid techniques in order to develop strategies that are applicable to many optimization algorithms .

This study investigates the efficiency of incorporating of data mining techniques to ant colony and genetic algorithm Population Metaheuristcs in order to guide them to generate new and better solutions. To validate the proposal, we use the Travelling Salesman Problem and the Problem Sets Cover and different versions of the hybrid meta-heuristics are tested and analyzed .

The technique chosen to guide the search of new solutions , from the patterns obtained with the Data Mining , was grouping similar solutions in an attempt to reduce the search space in combinatorial optimization problems . The mining algorithms used are the K -means and *Ward* which use techniques of hierarchical and partitioning respectively.

Computational experiments were performed in order to evaluate the use of MD in Meta-Population traditional heuristics . These experiments showed that the use of mined patterns can assist in obtaining good solutions .

Keywords: Population metaheuristics, Traveling Salesman Problem, Data Mining.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Meta-heurísticas são técnicas utilizadas para obtenção de boas soluções, em tempo razoável, para problemas de otimização combinatória. Estas fazem uso de abordagens não determinísticas que geram soluções que se aproximam do ótimo, mas no entanto, sem a garantia de que se encontre o ótimo global [Osman and Laporte, 1996]. Busca Tabu (TS, do inglês *Tabu Search*), Algoritmos Genéticos (GA, do inglês *Genetic Algorithm*), Recozimento Simulado (SA, do inglês *Simulated Annealing*), Otimização por Colonia de Formigas (ACO, do inglês *Ant Colony Optimization*) e GRASP (do inglês *Greedy Randomized Adaptive Search Procedures*) são exemplos de meta-heurísticas que têm sido aplicadas a problemas da vida real de diversas áreas da ciência nas últimas décadas [Martins, 2012].

Para que uma meta-heurística explore um espaço de busca de forma inteligente, obtenha soluções de ótima qualidade e consiga mover-se para áreas não exploradas quando necessário, etapas de diversificação e intensificação devem ser atingidas. A diversificação consiste na geração aleatória de uma ou mais soluções no espaço de busca, enquanto que a intensificação corresponde á melhoria destas soluções usando um procedimento de busca local. A diversificação é utilizada para permitir a fuga da solução dos chamados mínimos locais, ao passo que a intensificação é utilizada para melhorar a qualidade da solução localmente, buscando o ótimo global. Toda meta-heurística deve possuir componentes de diversificação e intensificação. Esses dois componentes devem ser balanceados e bem utilizados [Blum and Roli, 2003].

Um tópico importante na pesquisa de meta-heurísticas é o desenvolvimento de meta-heurísticas híbridas. Tais métodos resultam da combinação de conceitos e procedimentos de diferentes meta-heurísticas ou da combinação de meta-heurísticas com conceitos e processos de outras áreas de pesquisa responsáveis pela execução de tarefas específicas que podem melhorar a técnica original. Um exemplo deste último,

é a proposta deste trabalho, que incorpora técnicas de Mineração de Dados (MD) a Meta-heurísticas Populacionais.

A MD refere-se à extração automática de conhecimento potencialmente útil a partir de conjuntos de dados na forma de regras ou padrões, dentre as quais se destacam as regras de associação, padrões de sequência, regras de classificação e agrupamentos de dados. O conhecimento extraído representa características importantes do conjunto de dados. Assim, a MD fornece um meio para entender melhor as características implícitas dos dados brutos, que é fundamental em um processo de tomada de decisão [Clarke et al., 2009]. Os principais objetivos do uso de MD são a descrição e a predição. Na descrição, os padrões e regras descrevem características importantes dos dados, enquanto na predição deseja-se prever o valor desconhecido de um determinado atributo, baseado na análise histórica dos dados armazenados.

O desafio deste trabalho é desenvolver métodos híbridos utilizando MD e Meta-heurísticas Populacionais, que mostrem quando a técnica é capaz de atingir a melhor solução, ou uma solução quase ideal com poucas chances de melhorias. Os padrões minerados podem ser usados para orientar a busca das soluções ótimas ou perto de ótimas em menos tempo computacional.

As técnicas de otimização escolhidas para tal estudo foram Colônia de Formigas e Algoritmo Genético, que têm como características importantes, a utilização de população. Para validar a proposta foi utilizado o Problema do Caixeiro Viajante (PCV), que serve como modelo para inúmeros outros problemas de Otimização Combinatória. A ideia central da proposta apresentada nesta dissertação consiste em criar uma versão híbrida das Meta-heurísticas Populacionais já citadas, que incorpora técnicas de MD.

Essa combinação de técnicas ocorre da seguinte forma. Inicialmente gera-se um conjunto de soluções, através das Meta-heurísticas, em seguida, dispara-se um processo de MD que agrupa esse conjunto de soluções em dois grupos. No grupo composto pelas soluções que apresentam melhor qualidade extrai-se uma solução, então, utiliza-se uma técnica de Refinamento na mesma.

1.1 Motivação

A busca pela solução de problemas de elevado nível de complexidade em tempo computacional viável (problemas *NP-completo* e *NP-difícil*), como o PCV, tem sido um dos grandes desafios de pesquisadores da área de otimização. Um dos caminhos mais promissores, tem sido o uso de meta-heurísticas para obter soluções aproximadas de boa qualidade para esta classe de problemas. Elas fazem uso de abordagens não deter-

minísticas que exploram o espaço de busca de maneira eficiente, encontrando soluções que se aproximam do ótimo, no entanto sem a garantia de que se encontre o ótimo global, mas sem ter de explorar o espaço de busca exaustivamente.

Motivado pelas dificuldades em torno da resolução destes problemas e os bons resultados na literatura referentes as meta-heurísticas híbridas, este trabalho foca no desenvolvimento de métodos híbridos, utilizando MD e Meta-heurísticas Populacionais. Com intuito de resolver de forma satisfatória e em um tempo aceitável problemas com dimensões cada vez maiores.

1.2 Objetivo

O presente trabalho tem como meta principal a elaboração de um método de otimização híbrido utilizando MD e Meta-heurísticas Populacionais para tratar o problema do caixeiro viajante, que possui um alto nível de complexidade.

As Meta-heurísticas são heurísticas de busca no espaço de soluções que podem utlizar uma solução ou um conjunto de soluções. Com o intuito de melhorar tais técnicas que utilizam populações para a geração de soluções, utiliza-se MD para minenar padrões nesses conjuntos de soluções.

Tal método é abaseado no algoritmo Colonia de Formigas, inspirado no comportamento coletivo de colônias de formigas na busca de alimento, e Algoritmo Genético, fundamentado na teoria de seleção natural e mecanismos da genética. E dos algoritmos de agrupamento *K-means* e o *Ward*.

1.3 Estrutura do Documento

Este documento está organizado em sete capítulos, sendo este o primeiro e os demais são como se segue:

- No Capítulo 2 são apresentados os resultados das pesquisas sobre os trabalhos relacionados.
- Capítulo 3 será apresentado os principais conceitos ligados ao Problema do Caixeiro Viajante (PCV), sua definição, modelagem e a complexidade do mesmo.
- No Capítulo 4 é apresentada uma visão geral das Meta-heurísticas Populacionais, bem como conceitos necessários para o seu entendimento, além de uma introdução das duas Meta-heurísticas utilizadas o AG e o *Ant System* e os algoritmos implementados para resolver o PCV.

- O Capítulo 5 tem o objetivo principal de descrever as técnicas de M.D. e os algoritmos de Agrupamento(*K-means* e o *Ward*) utilizados na hibridização do método proposto.
- Capítulo 6 apresenta a estratégia híbrida proposta, utilizando os algoritmos citados anteriormente e os resultados obtidos referentes aos experimentos realizados em torno do método proposto.
- Por fim, no Capítulo 7, são apresentas as considerações finais a cerca do desempenho da estratégia proposta e suas principais contribuições, bem como algumas sugestões para trabalhos futuros e publicações realizadas.

Capítulo 2

Trabalhos Relacionados

Nos últimos anos, tornou-se evidente que a concentração em uma única meta-heurística é muito restritiva. A combinação de uma meta-heurística com outras técnicas de otimização ou outras áreas de pesquisa, chamada meta-heurística híbrida, pode proporcionar um comportamento mais eficiente e uma maior flexibilidade quando se trata de problemas do mundo real e em larga escala. Isso se deve ao fato de unir e explorar as vantagens das estratégias puras quando aplicadas individualmente. Muitos trabalhos recentes foram desenvolvidos utilizando meta-heurísticas híbridas para solução de vários problemas, principalmente utilizando a MD.

A maioria dos métodos encontrado na literatura utilizam técnicas de Mineração de Dados não-supervisionados, ou seja, técnicas que não possuem nenhum conhecimento prévio a cerca do problema tratado. Os trabalhos foram divididos em relação aos métodos não supervisionados utilizados. Assim temos duas seções, uma que trata os métodos que utilizam regras de agrupamento e outra que utiliza regras de associação.

2.1 Métodos utilizando Regras de Agrupamento

Em Kim and Cho [2001], foi proposto um Algoritmo Genético eficiente com menor avaliação da aptidão por agrupamento. Ele divide a população inteira em vários grupos ou *clusters* e apenas um indivíduo representante de cada *cluster* é avaliado através da função de aptidão ou*fitness*. Os valores de aptidão dos demais indivíduos do mesmo grupo, são estimados de acordo com sua distância euclidiana aos indivíduos representativos. Este GA híbrido com agrupamento pode reduzir de forma eficiente o número de avaliações sem qualquer perda de desempenho. Os resultados de vários testes mostraram que o algoritmo proposto tem pelo menos a mesma performance que o GA simples, com muito menos avaliações. Este tipo de abordagem é muito útil para problemas que

exigem alto custo para avaliar indivíduos. Yoon and Cho [2011] e Yoo and Cho [2004], propuseram o mesmo procedimento, utilizando a versão *fuzzy* do algoritmo *K-means*, o *C-Fuzzy*, para solucionar o Problema do Caixeiro Viajante e outras noves funções de referências.

No trabalho de Jin and Sendhoff [2004], o método k-vizinho mais próximo é aplicado para agrupar os indivíduos de uma população de cada geração de um AG em um número de *clusters*. Para cada grupo, apenas o indivíduo que é mais próximo ao centro do *cluster*, será avaliado através da função de *fitness* original. Os indivíduos testados (Centros de conglomerados) são usados para criar um conjunto para rede neural, que é utilizada para estimar os valores de aptidão dos indivíduos restantes. Os primeiros trabalhos utilizando essa abordagem foram publicados em Jin et al. [2000] e Jin et al. [2002].

Em de Oliveira and Lorena [2004] e Oliveira and Lorena [2007] criou-se uma abordagem híbrida conhecida como Busca por Agrupamento (CS, do ingles *Clustering Search*) que consiste em dividir o espaço de busca em grupos (clusters) e aplicar, em cada grupo e de acordo com certos critérios, diversas meta-heurísticas como, por exemplo, Simulated Annealing, GRASP, Busca Tabu, VNS e outras. Foram aplicados em diversos problemas, como o problema das p-medianas e [Chaves et al., 2007] e o problema do caixeiro viajante com coleta de prêmios [Chaves and Lorena, 2008] e [Chaves and Nogueira Lorena, 2011].

A metodologia proposta em Martínez-estudillo et al. [2006] baseia-se na combinação de um Algoritmo Evolutivo, um processo de *clustering* e um procedimento de Busca Local aplicado para a evolução da estrutura e os pesos das redes neurais com base no produto de unidade. Existem duas versões desse método. Na primeira, o processo de agrupamento é aplicado em um grande subconjunto dos melhores indivíduos da população final. Depois disso, aplica-se uma Busca Local no melhor indivíduo de cada *cluster*. A segunda, realiza o mesmo processo dito anteriormente em um número fixo, G_0, de gerações. A solução final é o melhor indivíduo entre os ótimos locais encontrados durante o processo evolutivo. Neste método é muito importante determinar a quantidade de melhores indivíduos da população final, assim como o número de *clusters*. Uma das suas principais vantagens é que o custo computacional da aplicação do algoritmo de Busca local para apenas um pequeno número de indivíduos praticamente não afeta o tempo total gasto pelo algoritmo. O uso de um algoritmo de agrupamento permite a seleção de indivíduos que representam diferentes regiões no espaço de busca. Deste modo, os indivíduos otimizados são mais propensos a convergir para um mínimo local diferente. Se aplicarmos o algoritmo de otimização para cada indivíduo na população, vamos obter muitas soluções semelhantes com uma perda de tempo considerável.

A segunda versão não só obteve os melhores resultados na média, mas também nos desvios padrão.

2.2 Métodos utilizando Regras de Associação

Técnicas de M.D. são frequentemente usados para controlar a população. Várias obras lidam com a introdução de novos indivíduos na população. Isto poderia ser realizado para diversificar a população como na estratégia aleatória. A fim de levar a pesquisa para espaços de busca promissores, poderia introduzir regularmente os indivíduos que são construídos com base em informações das soluções encontradas no passado.

No trabalho de Ribeiro et al. [2006] os autores desenvolveram uma versão híbrida da meta-heurística *GRASP* (*Greedy Randomized Adaptative Search Procedure*) que incorpora técnicas de Mineração de Dados para resolver o problema de Empacotamento de Conjuntos. Esta meta-heurística híbrida é dividida em duas partes. Na primeira, após executa um número significativo de iterações do *GRASP*, o processo de mineração de dados extrai padrões de um conjunto de soluções de elite(conjunto das melhores soluções obtidas), que norteiam as seguintes iterações do algoritmo. Na segunda fase, são executadas outras iterações adaptadas, nas quais os padrões extraídos são utilizados para guiar a construção das novas soluções. As soluções são representadas por conjuntos de regras e padrões que são definidos como subconjuntos de itens que ocorrem em um número significativo de soluções. O processo de mineração desses padrões é o problema bem conhecido na mineração de dados chamado de Mineração de Conjuntos Frequentes, um sub-problema de regra de associação de mineração. Foram utilizados os algoritmos *DCI* (*Direct Count & Intersect*) e *FPMax**. Os resultados obtidos mostraram que a estratégia de extração de dados acelera o processo de encontrar uma boa solução através do algoritmo tradicional. Essa proposta híbrida foi utilizada com sucesso em três problemas de otimização combinatória: Problema de *Multicast* Confiável [Fonseca et al., 2008] e Problema das P-Medianas [Plastino et al., 2011]. Martins [2012] também desenvolveu uma meta-heurística hibrida, utilizando o mesmo módulo de MD, mas utilizando a técnica de *Multistar*. Foi publicado um *survey* desse método em Santos et al. [2008].

Em Ochi et al. [2006], os autores propõem hibridizar um algoritmo genético e o algoritmo Apriori (Apriori é um algoritmo clássico para gerar regras de associação [Agrawal and Srikant, 1994]) que visa descobrir as sub-rotas que são comumente encontradas nas melhores soluções da população para o problema de roteamento de veículos com coleta de prêmios. O processo começa com a criação de um conjunto de soluções

elite (CE), que vai manter as S melhores soluções geradas no processo de busca. Inicialmente, as S primeiras soluções geradas são inseridos no CE. Posteriormente, este conjunto é atualizado sempre que uma solução, que é melhor do que a pior solução do CE e diferente dos demais, é gerada. O CE será a base de dados na qual se tentará descobrir padrões relevantes. A fim de fazer isso, desenvolveu-se uma versão ligeiramente modificada do algoritmo *Apriori*, a qual descobre sequências frequentes (ao invés de regra de associação). Uma vez que o conjunto de subsequências frequentes estiver disponível, este é usado para guiar a construção de indivíduos, utilizando um algoritmo construtivo. O algoritmo recebe o suporte mínimo como parâmetro de entrada. Este parâmetro está relacionado com o tamanho do CE e define o número minimo de ocorrências no CE que uma sequência deve ter para ser considerado frequente. Assim, o algoritmo vai descobrir todas as sequências (Sub-rotas no CE) de todos os tamanhos ≥ 1, que satisfazem o suporte mínimo. Por exemplo, um suporte mínimo de 0.5, indica que, pelo menos metade do CE soluções devem incorporar a sequência considerada.

2.3 Considerações Finais

Neste capítulo foram apresentados os trabalhos que encontramos na literatura relacionados ao nosso trabalho. Note que a maioria dos trabalhos envolvem Meta-heurísticas que utilizam uma população e técnicas de MD não supervisionado, ou seja, técnicas que não necessitam de conhecimento prévio dos dados.

A maioria dos métodos tentam encontrar padrões nas soluções dos algoritmos ou buscar por regiões ditas como promissoras. Nosso trabalho tem como objetivo reduzir o conjunto de soluções, reunindo soluções similares dentro do mesmo *cluster*, sem perder as características desse conjunto e tentando encontrar padrões no mesmo. Para evitar um esforço computacional extra, o processo de agrupamento é realizado no conjunto de soluções final da Meta-heurística.

A partir da literatura, podemos observar que a interação entre as técnicas de mineração de dados e algoritmos evolutivos e outras meta-heurísticas tem sido um casamento muito lucrativo durante a última década, sendo hoje em dia um dos temas mais promissores em investigação na informática. Uma revisão da literatura mais detalhada referente a esse tema pode ser encontrada em [Jourdan et al., 2006] e uma classificação geral das estratégias investigadas nessa seção encontra-se na tabela 2.1.

Tabela 2.1. Classificação das estratégias propostas na literatura.

Problemas	Meta-Heurísticas	Técnicas de MD	Referências
Problema do Caixeiro Viajante e Redes Neurais	Algoritmo Genético	Algoritmo *K-means*	Kim and Cho [2001], Jin and Sendhoff [2004], Jin et al. [2002], Jin et al. [2000] e Martínez-estudillo et al. [2006]
Diversão funções	Algoritmo Genético	Algoritmo *C-fuzzy*	Yoon and Cho [2011], Yoo and Cho [2004] e Herrera and Lozano [2001]
Problema de Empacotamento	GRASP	Algoritmo *DCI*	Ribeiro et al. [2006]
Problema de Roteamento de Veículos Coleta de Prêmios	Algoritmo Genético	Algoritmo *Apriori*	Ochi et al. [2006]
Problemas Contínuos	Algoritmo de Otimização de Domínio (DOA) e a arquitetura de Amostragem Inteligente(SS)	K-Vizinhos mais próximos(KNN)	De Melo et al. [2007] e de Melo et al. [2007]
Problema do Caixeiro Viajante	GRASP, Algoritmos Genéticos e algoritmo	algoritmo *Q-learning*	dos Santos et al. [2009] e dos Santos et al. [2010]
Projeto de Circuitos Lógicos Combinacionais	Algoritmo Genético		Louis [2003a] e Louis [2003b]
Funções de Teste	Algoritmo Genético	Regressão e Interpolação	Branke and Schmidt [2005]

.

Capítulo 3

Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (PCV) em inglês Traveling Salesman Problem (TSP), é um problema clássico de Otimização Combinatória, que tem sido utilizado em experimentos de diversos métodos de otimização. Isso se deve ao fato do PCV ter uma fácil descrição e compreensão, grande dificuldade de solução e larga aplicabilidade. Segundo [Campello and Maculan, 1994], o PCV é um dos mais tradicionais e conhecidos problemas de programação matemática.

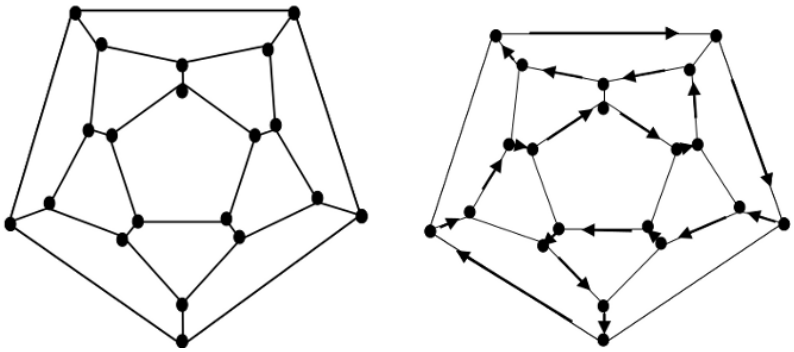
Suponha que um caixeiro viajante deseja visitar n cidades e que, entre alguns pares de cidades existem rotas. Cada rota tem uma distância ou um custo para percorrê-la. O custo final é a soma dos custos das rotas percorridas. O problema consiste em encontrar uma rota para um caixeiro viajante tal que este:

- Parta de uma cidade inicial;
- Passe por todas as demais cidades uma única vez;
- Retorne à cidade origem no final do percurso;
- Percorra a trajetória pelo menor caminho possível.

Essa trajetória, que passa umavez única vez em cada cidade, também é conhecida como ciclo Hamiltoniano, que possui a seguinte definição: Um circuito diz-se Hamiltoniano, se passar uma vez somente por todos os vértices de uma rede [Boaventura, 1996]. A designação provém do irlandês Willian Rowman Hamilton que, em 1857 propôs um jogo denominado *"Around the World"*.

Nesse jogo, os vértices de um dodecaedro de madeira representavam as 20 cidades mais importantes do mundo na época. O objetivo do jogo consistia em encontrar um percurso através dos vértices do dodecaedro, com início e fim no mesmo vértice

(cidade) e que passasse por cada vértice (cidade) apenas uma vez[Goldbarg, 2005]. Este problema na verdade corresponde ao PCV clássico e a figura 3.1 mostra um exemplo de jogo de Hamilton e uma solução para o mesmo. Hamilton não foi o primeiro a propor esse problema, mas o seu jogo ajudou a divulgá-lo. Modernamente, a primeira menção conhecida do problema é devida a Hassler Whitney em 1934 em um trabalho na Princeton University.



(a) Jogo de Hamilton.

(b) Uma solução do jogo de Hamilton.

Figura 3.1. Exemplo do jogo de Hamilton e a solução Goldbarg [2005].

Em relação ao custo, apesar da distância ser a variável mais comum, pode haver variações e outras informações podem ser quantificadas neste caso, tais como pedágios ou condições da rodovia - por exemplo, uma rodovia sem asfalto pode ser penalizada com o acréscimo de uma distância.

O PCV pode ainda ser classificado como simétrico ou assimétrico. É simétrico se, para todos os pares de nós i, j, os custos c_{ij} e c_{ji} forem iguais. Caso contrário, o mesmo é dito assimétrico. Uma revisão histórica e mais completa sobre o PCV pode ser encontrada em Hoffman e Wolfe [Hoffman and Wolfe, 1985] e ainda em Gutin e Punnen [Gutin and Punnen, 2002].

3.1 Modelagem Matemática

Existem diversas formulações matemáticas para esse problema. Entretanto, a abordagem utilizada neste trabalho será a formulação de Dantzig, Fulkerson e Johnson [Dantzig et al., 1954], presente em [Goldbarg, 2005] visto que a mesma é frequentemente utilizada na literatura, pois apresenta modos peculiares para a caracterização do problema e, também, por ser de fácil compreensão.

Seja $G = (N, E)$ um grafo completamente conectado onde $N = \{1, ..., n\}$ é o conjunto de nós(vértices), que representam as cidades, e $E = \{1, ..., m\}$ é o conjunto de arestas de G que ligam todos os nós. O custo c_{ij} associado com cada aresta, ligando os vértices i e j, representa a distância entre as cidades [Szwarcfiter, 1986].

O problema consiste em localizar o menor ciclo Hamiltoniano do grafo G. O tamanho do ciclo é calculado pelo somatório dos custos das arestas que formam o ciclo. Os nós do grafo são, frequentemente, referenciados como " cidades " e, em outras palavras, o objetivo é visitar todas as cidades passando apenas uma vez por cada cidade, retornando ao ponto de origem. Este percurso deve ser feito de forma a minimizar a distância total percorrida.

Minimizar $z = \sum_{j=1}^{n} \sum_{i=1}^{n} c_{ij} x_{ij}$

Sujeito a:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \in \mathbb{N} \tag{3.1a}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in \mathbb{N} \tag{3.1b}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1 \qquad \forall S \subset \mathbb{N} \tag{3.1c}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in \mathbb{N} \tag{3.1d}$$

Onde a variável binária x_{ij} assume valor igual a 1, se o arco $(i, j) \in A$ for escolhido para integrar a solução, e 0 em caso contrário, e S é um subgrafo de G, em que $|S|$ representa o número de vértices desse subgrafo. Nessa formulação, assumimos implicitamente que x_{ij} não existe e que teremos $n(n-1)$ variáveis inteiras $0-1$ e $O(2^n)$ restrições.

3.2 Complexidade Computacional

Um algoritmo obvio para solucionar este problema precisa calcular o peso de $(n-1)!/2$ circuitos hamiltoniano de G. A complexidade de tal algoritmo é claramente de ordem fatorial e, dessa forma, muito ineficiente. Infelizmente nenhum algoritmo eficiente foi encontrado para solucioná-lo. De fato, embora possa ser descrito de uma maneira simples, o PCV pertence à classe dos problemas *NP-Difícil* [Karp, 1972]. Sua versão de Otimização é equivalente, em complexidade, à versão de decisão que é *NP-Completo*. Isso significa que podemos verificar se existe ou não um ciclo Hamiltoniano de custo

total menor ou igual a w, em tempo polinomial e o problema é tão difícil de ser solucionado quanto qualquer outro problema da classe NP [Papadimitriou and Steiglitz, 1982].

Existem vários problemas que podem ser modelados como instâncias do Problema do Caixeiro Viajante. Ele possui aplicações em diversas áreas tais como: indústrias (minimizar custo por meio da melhor rota utilizada por equipamentos no processo de montagem), empresas (transportadoras, entrega e coleta de cargas), automobilística (redução de custo de viagem), transporte de passageiros, roteirização de serviços de reparos ou serviços públicos (como coleta de lixo, entrega postal), sequenciamento de genoma [Applegate et al., 2007].

Em 1991, [Reinelt, 1991] criou uma uma biblioteca contendo várias instâncias do PCV que vinham sendo (e são até hoje) testadas e discutidas na literatura. Essa biblioteca é conhecida como TSPLIB e contém mais de 100 exemplos com tamanhos que variam de 14 até 85.900 cidades. Restam apenas três instâncias dessa biblioteca que ainda não foram resolvidas. Essas instâncias são d18512, pla33810 e pla85900 que têm 18.512, 33.810 e 85.900 cidades, respectivamente. Utilizamos essa biblioteca neste trabalho, mas infelizmente não testamos essas instâncias.

3.3 Considerações Finais

Neste capítulo foram apresentados os principais conceitos ligados ao PCV, e teve como objetivo conceituar e caracterizar o problema, visto que durante toda a dissertação este problema é tomado como exemplo. Foram apresentadas: definição, formulação matemática, complexidade e aplicações do mesmo.

Ele é um dos problemas mais estudados da Área de Otimização Combinatória. Sua enorme importância se dá pelo fato de existir um número muito grande de problemas reais que podem ser modelados como um PCV, bem como o desenvolvimento de novas técnicas para resolução de problemas vem sendo criadas para o mesmo, ou, pelo menos, sendo a ele primeiramente aplicadas para mostrar suas eficácias.

O PCV não é um problema de fácil resolução, um dos problemas principais é o alto tempo computacional necessário para encontrar uma solução. Uma das abordagens para resolver esse problema consiste em encontrar uma solução aproximada, através de Meta-heurísticas, que podem ser encontradas na literatura. Por exemplo, soluções que utilizam algoritmos genéticos, Busca Local e outros. Apesar de existir muitas instâncias ainda não solucionadas, é notável o avanço que se tem alcançado na solução do PCV nos últimos anos.

Capítulo 4

Meta-heurísticas Populacionais e Heurística de Refinamento

Uma meta-heurística pode ser definida como um conjunto de conceitos que pode ser utilizado para definir métodos heurísticos aplicáveis a uma ampla gama de problemas. Em outras palavras, uma meta-heurística pode ser vista como uma estrutura algorítmica geral que pode ser empregada na resolução de diferentes problemas de otimização, com um número relativamente reduzido de modificações que a adaptem para o tratamento de cada problema específico [Dorigo, 2004].

Meta-heurísticas podem ser classificadas como meta-heurísticas de trajetória ou meta-heurísticas populacionais. Dentre as meta-heurísticas de trajetória mais conhecidas, encontram-se a Recozimento Simulada e a Busca Tabu. Estes métodos trabalham apenas com uma solução por iteração. A solução inicial é geralmente gerada aleatoriamente e, ao passar das iterações, descreve um caminho ou trajetória de soluções pelo espaço de busca, obtido pela transição de uma solução para outra de acordo com os movimentos permitidos pela meta-heurística [Oliveira, 2008].

Meta-heurísticas Populacionais, diferentemente das meta-heurísticas de trajetória, geram um conjunto de soluções, armazenam as melhores soluções e as manipulam concorrentemente de formas diversas em cada iteração. Elas usam uma estratégia simples e eficiente que é a exploração do espaço de busca de soluções em pontos bem espalhados na etapa inicial do processo. Esta estratégia possibilita ao algoritmo mapear o espaço de busca e determinar regiões que poderão ser mais exploradas em iterações subsequentes. Nessas iterações, surgem melhores pontos de qualidade aproximando-se do ótimo global [Melo, 2009].

Uma das Meta-heurísticas Populacionais mais conhecidas é o Algoritmo Genético (AG), elaborada por Holland (1975) e posteriormente aprimorada por Goldberg (1989),

que se fundamenta nos conceitos de genética populacional e da teoria da evolução das espécies postulada por Darwin (1859). Outra Meta-heurística Populacional introduzida mais recentemente na literatura foi Colônia de Formigas (Ant Colony Optimization) que baseia-se no comportamento utilizado pelas colônias de formigas para traçar rotas entre o formigueiro e as fontes de alimentação [Dorigo, 1992].

4.1 Algoritmo Genético

Algoritmos genéticos (AG) são um ramo dos algoritmos evolucionários e como tal podem ser definidos como uma técnica de busca inspirada em uma metáfora do processo de evolução dos seres vivos, baseado na seleção natural de Darwin [Darwin, 1872]. Foram inicialmente apresentados por John Henri Holland (considereado o pai de GA) em seu trabalho intitulado de "Adaptation in Natural and Artificial Systems " em 1975 [Holland, 1975], onde ele propôs um modelo heurístico computacional que mantinha os mesmos mecanismos encontrados em sistemas naturais.

Um Algoritmo Genético é um procedimento iterativo, que mantém uma população de estruturas (chamadas indivíduos ou cromossomos), que representam possíveis soluções de um determinado problema ou soluções candidatas [Goldberg, 1989]. O processo da criação da população inicial é realizado de maneira aleatória, mas após cada iteração do algoritmo, denominado nesse contexto como geração, um novo conjunto de estruturas é criado a partir de estruturas bem adaptadas selecionadas da geração(iteração) anterior.

Esse novo conjunto é criado através da aplicação de operadores genéticos como seleção, *crossover*, mutação entre outros. Cada indivíduo recebe uma avaliação, através da função *fitness*, que é uma quantificação da sua qualidade como solução do problema em questão. Baseado nesta avaliação serão aplicados os operadores genéticos de forma a simular a sobrevivência do mais apto [Linden, 2005], pois eles exploram regiões desconhecidas no espaço de busca e com base no valor dos *fitness* os melhores indivíduos são selecionados. A figura 4.1 mostra o esquema de um AG.

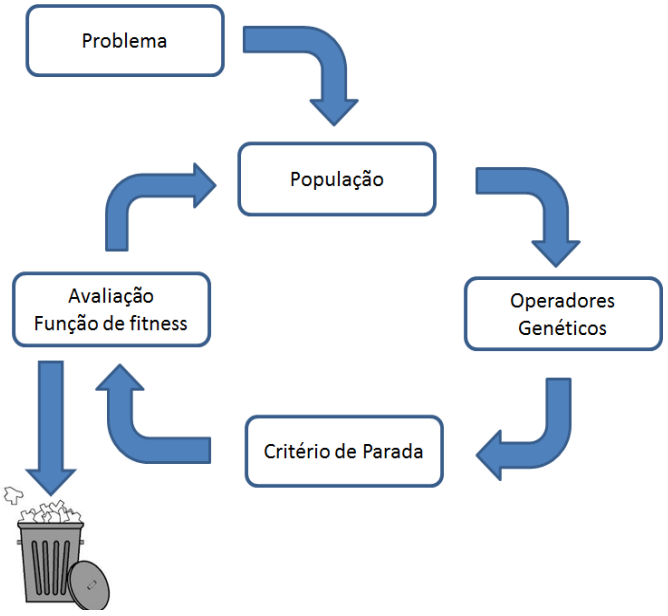


Figura 4.1. Esquema de um Algoritmo Genético.

As principais diferenças entre os AG's e os procedimentos de busca tradicionais são: trabalham com conjunto de variáveis (estrutura) e não com variáveis individuais; são probabilísticos, sendo assim, tem-se soluções diferentes para uma mesma população inicial e os mesmos parâmetros. Além disso, trabalham com grandes populações de pontos, mas não procuram em todos os pontos possíveis, apenas em um subconjunto destes pontos utilizando informações históricas para tentar caminha na direção correta [Linden, 2005].

Entre os principais fatores que têm feito do AG uma técnica bem sucedida destacam-se: simplicidade e facilidade nas operações, eficácia na busca da região onde, provavelmente, encontra-se o máximo global. Além de apresentar um notável equilíbrio entre diversificação e intensificação. No início da pesquisa genética, há uma ampla e diversificada população randômica e os operadores tendem a intensificar a busca na vizinhança de cada uma das soluções.

4.1.1 Algoritmo Genético para o PCV

Para o Problema do Caixeiro Viajante os indivíduos são representados por vetores *n-dimensionais* que correspondem à sequência das cidades que serão visitadas. Cada posição (gene) indica qual cidade está naquela posição. A função *fitness* de avaliação é a função objetivo do PCV.

Michalewicz em [Tao and Michalewicz, 1998] propôs um algoritmo evolutivo baseado em um operador especial chamado *Inver-Over*. Ele tem o propósito de ser um

algoritmo evolucionário "puro", ou seja, não usa nenhum tipo de busca local e, ao mesmo tempo, ser um modelo competitivo com relação aos algoritmos que usam busca local. Este operador incorpora, numa solução, o conhecimento tirado de outros indivíduos na população. Ele trabalha de forma isolada, substituindo tanto o operador de *crossover* (cruzamento) quanto o de mutação, através de uma mistura destes dois operadores. Por um lado, a inversão é aplicada a uma parte de um único indivíduo, a seleção de um segmento a ser invertido depende de outros indivíduos na população. Seu funcionamento é dado pelo pseudo-código 1.

Ele aplica uma forte seleção sobre os indivíduos e usa um ou dois pais a cada iteração. É aplicado a todos os pais, ao invés de uma seleção de pais feita por algum mecanismo aleatório, como a roleta viciada. Isso ocorre pois o operador equivale, de certa forma, a um processo de busca local otimizando as conexões localmente para melhorar o indivíduo corrente ao máximo [Linden, 2005].

Ele possui as seguintes características:

- Cada indivíduo compete apenas com os seus descendentes;
- Existe apenas um operador de variação (*Inver-Over*), mas este é adaptativo;
- O número de vezes que o operador é aplicado a um indivíduo durante uma única geração, é variável. Componentes de adaptação são: o número de inversão aplicado a um único indivíduo, e o segmento a ser invertido (determinado por um outro indivíduo escolhido aleatoriamente).

O parâmetro p representa uma probabilidade (baixa) que é usada como controle do operador *Inver-Over*. Se o número sorteado ficar abaixo dela, faz-se o equivalente a uma inversão. Caso contrário seleciona-se um outro pai aleatoriamente na população, para fazer uma operação que tem uma certa semelhança com o *crossover* tradicional de dois pais. Este operador é ilustrado na figura 4.2. Suponha que S seja o indivíduo atual e a cidade corrente C é 3. Se o número gerado aleatoriamente não exceder p, outra cidade C' é selecionada no mesmo individuo (C' é 8) e o segmento apropriado é invertido gerando o individuo S'.

Algoritmo 1: *Inver-over*

```
1  inicializa a população aleatoriamente;
2  enquanto não chegar o número máximo de gerações ou o individuo não
   mudar por k gerações faça
3      para cada individuo S_i ∈ P faça
4          S' ← S_i;
5          Selecione aleatoriamente uma cidade c em S';
6          repita
7              se rand() ≤ p então
8                  Selecione uma cidade c' do restantes das cidades S'
9              senão
10                 Selecione aleatoriamente outro individuo em P;
11                 c' recebe a próxima cidade depois de c no novo indivíduo;
12             fim se
13             Inverte a sessão da cidade posterior a c até a cidade c' em S';
14             c ← c';
15         até cidade posterior ou cidade anterior a cidade c em S' for c';
16         se eval(S') ≤ eval(S) então
17             S_i ← S‘;
18         fim se
19     fim para
20 fim enqto
```

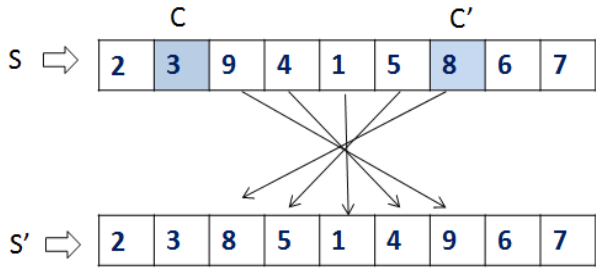


Figura 4.2. Ocorre mutação quando o número aleatório é menor que a probabilidade p.

Se o número sorteado é maior que p, então outro indivíduo é selecionado aleatoriamente da população. Procura-se nesse novo indivíduo a cidade C' próxima de C, nesse caso C' é 5, o segmento a ser invertido começa na cidade 3 e termina na cidade 5 consequentemente a nova saída é S'. Este processo termina quando a cidade C' selecionada esta próxima da cidade C (anterior ou próxima). O individuo é avaliado e se for melhor que o pai S toma o seu lugar. A figura 4.3 ilustra essa situação.

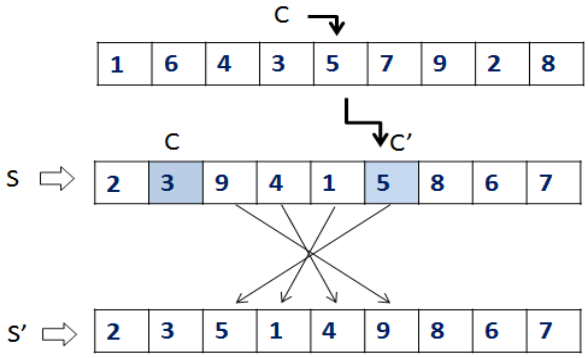


Figura 4.3. Ocorre *crossover* quando o número aleatório é maior que a probabilidade p.

Este algoritmo pode ser entendido como um conjunto de procedimentos paralelos de "subida de montanha" (hill climbing). Cada "alpinista" (hill climber) executa um número variável de trocas de adjacências. Entretanto, o operador *Inver-Over* tem componentes adaptativos: o número de inversões aplicadas a um único indivíduo e o segmento a ser invertido é determinado por outro indivíduo selecionado aleatoriamente. Desta forma, é sempre possível visualizar este método como um algoritmo evolucionário com uma forte pressão seletiva e um operador de variação adaptativo. GAs que usam o operador *Inver-over* costumam ser velozes e precisos, tendo atingido um resultado muito próximo da solução ótima em vários casos de teste, embora não há um estudo teórico formal que prove que esta situação se repetirá em todos os problemas possíveis Wang et al. [2012].

4.2 Otimização por Colônia de Formigas

A Meta-heurística Otimização por Colônia de Formigas (ACO) é baseada em um processo de construção de solução que se inspira no comportamento coletivo de formigas reais para solucionar inúmeros problemas de otimização [Dorigo, Maniezzo, Colorni 1996]. A ideia é imitar a forma como as formigas encontram um caminho mais curto entre seu ninho e uma fonte de alimentos.

O algoritmo de otimização por formigas foi proposto por Marco Dorigo, em 1992, em sua tese de doutorado [Dorigo, 1992]. Inicialmente as formigas estão em seu ninho e com o passar do tempo elas vão percorrendo as trilhas depositando no solo uma certa quantidade de substância, substância essa chamada feromônio, até chegarem ao seu destino: a fonte de alimento.

Dessa forma, as formigas que vierem posteriormente serão atraídas por esta substância tendo alta probabilidade de seguir este mesmo caminho. Esta substância química, no decorrer do tempo, sofre o processo de evaporação. A quantidade de ferormônio acumulada em cada trilha será decisiva para sua escolha, quanto mais ferormônio a trilha contiver maior será a probabilidade de esta trilha ser seguida.

O que se nota na natureza é que, após um determinado tempo, as formigas tenderão a seguir caminhos mais curtos, devido a um maior trânsito de formigas, e consequentemente um maior despejo de feromônio, por unidade de tempo. Isso causa uma maior concentração de feromônios em caminhos mais curtos e faz com que todas as formigas utilizem um único caminho, o mais curto (o sistema convirja para solução ótima, ou próxima dela) [Dorigo, 2004].

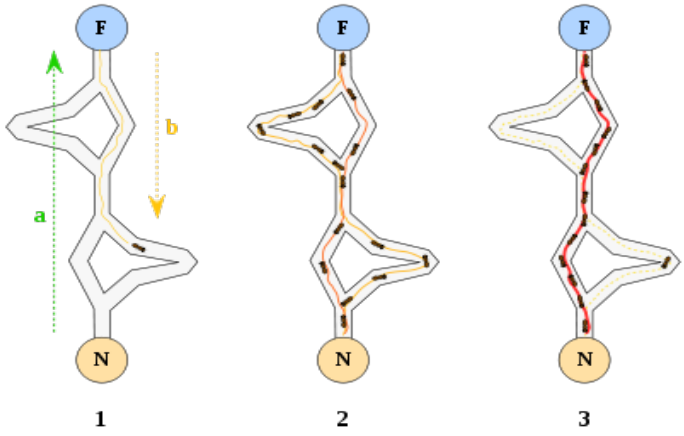


Figura 4.4. Formigas convergem para o caminho mais curto. Fonte: Esquema baseado em Goos and Pasteels [1999].

Na figura 4.4, a formiga inicia a exploração em **a**, ao encontrar alimento em Ⓕ retorna ao ninho Ⓝ depositando feromônio durante o caminho **b**. Depois, as demais formigas são recrutadas para a exploração da área do alimento. Após certo tempo, a maioria das formigas escolhem a menor rota. Eventualmente, algumas formigas são liberadas para explorar ou patrulhar o território.

É um modelo baseado em população, no qual cada indivíduo é uma possível solução do sistema, isso permite a exploração simultânea de diferentes soluções por meio de diferentes formigas. As melhores formigas influenciam a exploração das demais, através das estratégias empregadas para atualizar o feromônio nos caminhos. De acordo com [Bonabeau et al., 1999], os algoritmos baseados em Colônia de Formigas são um dos mais bem sucedidos exemplos de inteligência Coletiva e têm sido aplicados em diversas classes de problemas combinatoriais. Suas principais vantagens são:

- Robustez: mesmo quando um ou mais indivíduos falham, a colônia continua a executar suas tarefas.
- Flexibilidade: a colônia tem a capacidade de se adaptar rapidamente a mudanças externas e internas.
- Auto-organização: colônia requer relativamente pouca supervisão ou controle.

4.2.1 *Ant System* para o PCV

O *Ant System (AS)* foi o primeiro algoritmo de Otimização a utilizar a formulação de Colônia de Formigas, proposto por Dorigo, Maniezzo e Colorni [Dorigo et al., 1996] e que visa resolver o Problema do Caixeiro Viajante. Sua principal importância é servir como base para a criação do paradigma ACO de criação de algoritmos e possui um determinado grau de aleatoriedade na comunicação das formigas que minimiza o tempo de captura do alimento distribuído entre várias fontes.

O algoritmo se inicia com m formigas distribuídas aleatoriamente pelas cidades. Todos as arestas (i, j) são inicializadas com a mesma quantidade $\tau_{ij} \geq 0$ de feromônio. Na construção de sua solução, as formigas selecionam as cidades a serem visitadas de acordo com uma regra de transição que define a probabilidade p_{ij}^k da formiga k ir da cidade i para a cidade j enquanto constrói sua rota. Esta regra de decisão probabilística, que decide qual a próxima cidade a ser visitada, é dada pela fórmula:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}$$

onde α e β são parâmetros para indicar respectivamente a importância do feromônio (τ_{ij}) e da informação heurística (η_{ij}) e N_i^k o conjunto das cidades ainda não visitadas pela formiga k.

Essa probabilidade de escolha de um caminho é uma função da informação heurística, que no caso do PCV é o inverso da distância entre as duas cidades, e da quantidade de feromônio presente na aresta que conecta a essa cidade. Se a formiga já percorreu aquele caminho, a probabilidade de escolha é zero; caso contrário, é positiva. Para não permitir passos inválidos, a formiga tem uma memória que armazena os caminhos já percorridos. Assim, enquanto ela constrói uma solução, ela é forçada a não visitar o mesmo caminho mais de uma vez, até que a solução esteja completa.

Quando a formiga termina um caminho completo, ela deposita uma certa quantidade de feromônio em cada aresta (i, j) que ela visitou[4]. No AS, não há depósito de feromônio durante a fase de construção da solução. Ao dar um passo, a formiga calcula

a probabilidade de todas as cidades que ela ainda não visitou e escolhe uma cidade através do método conhecido como "roleta russa ". O pseudo-código do algoritmo AS aplicado ao PCV é apresentado em 2.

Algoritmo 2: Ant System para o PCV.

```
1  Inicializar parâmetros;
2  Inicializar matriz heurística;
3  Inicializar matriz de feromônio;
4  Coloque cada formiga em uma cidade aleatória;
5  para t = 1 ate numero de iteracoes faça
6      para k = 1 ate m faça
7          enquanto a formiga k nao construir a viagem S_k faça
8              Selecione a proxima cidade pela regra p_ij^k;
9          fim enqto
10         Calcule a distancia L_k da viagem S_k;
11         se L_k ≥ L* então
12             L* ← L_k
13         fim se
14     fim para
15     Atualizar os feromonios;
16 fim para
17 Retornar S*;
```

AS não realiza nenhuma ação global, nem busca local. Assim, a próxima fase, após a construção da solução é a atualização do feromônio, que envolve tanto o incremento do feromônio, quanto a sua evaporação e é realizado por todas as formigas. A regra de atualização da quantidade de feromônio aplicada nas arestas utilizadas na iteração é dada por:

$$\tau_{ij} = \underbrace{(1-\rho)\tau_{ij}}_{evaporacao} + \underbrace{\sum_{k=1}^{m} \Delta\tau_{ij}^{k}}_{deposito}$$

sendo:

$$\tau_{ij}^{k} = \begin{cases} Q/L_k, \text{ se a aresta } (i,j) \text{ pertence a viagem } S_k \\ 0 \qquad , \text{caso contrario} \end{cases}$$

onde $\rho \in (0,1]$ é um parâmetro para regular a taxa de evaporação do feromônio (coeficiente de redução), m o número de formigas e $\Delta\tau_{ij}^{k}$ é a quantidade de feromô-

nio depositada nas arestas percorridas pela formiga k, L_k ´é o comprimento total do caminho percorrido pela formiga k.

4.3 Heurística de Refinamento

Um algoritmo de busca local define, para cada solução, uma vizinhança composta por um conjunto de soluções com características " muito próximas ". Dada uma solução corrente, uma das formas de implementar um algoritmo de busca local é percorrer a vizinhança dessa solução em busca de outra com valor menor (para um problema de minimização). Se tal solução vizinha for encontrada, torna-se a nova solução corrente e o algoritmo continua. Caso contrário, a solução corrente é um ótimo local em relação à vizinhança adotada.

4.3.1 *VNS* para o PCV

A abordagem utilizada no algoritmo de busca local para o PCV foi Busca em Vizinhança Variável (Variable Neighborhood Search, VNS) [Hansen and Mladenović, 1997]. Trata-se de um método que se utiliza de mais de uma vizinhança, podendo combinar mudanças estocásticas e determinísticas de vizinhança. Ele explora o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança tanto na descida para mínimos locais quanto na fuga destes.

Contrariamente à maioria dos outros métodos de busca local, o VNS não segue uma trajetória, mas sim, explora vizinhanças gradativamente mais "distantes" da solução corrente, pelo fato de gerar vizinhos aleatoriamente, movendo-se para a nova solução se e somente se uma melhora for produzida. Além disso, um método de busca local é aplicado repetidamente para transformar a solução vizinha em um ótimo local. Esta rotina de busca local também pode usar diferentes estruturas de vizinhança. O pseudocódigo do método é apresentado no algoritmo 3.

Inicialmente, define-se o conjunto de estruturas de vizinhança utilizadas, através dos movimentos entre os componentes da solução. Ou seja, as diferentes estruturas de vizinhança são obtidas a partir de uma operação chamada movimento. Seja $N_k, (k = 1, \ldots, k_{max})$, um conjunto finito de estruturas de vizinhanças e $N_{(s)}^{(k)}$ o conjunto de soluções na k-ésima vizinhança da solução corrente s. A vizinhança utilizada para o PCV foi definida pela operação de troca entre duas cidades.

Algoritmo 3: Busca em Vizinhança Variável.

```
1  Seja s_0 uma solução inicial e r o número de estruturas diferentes de
   vizinhança;
2  s ← s_0 solucao corrente;
3  enquanto Criterio de parada não satisfeito faça
4      k ← 1 Tipo de estrutura de vizinhança;
5      enquanto k ≤ r faça
6          Gere um vizinho aleatorio s' ∈ N_(k)(s);
7          s'' ← BuscaLocal(s');
8          se f(s'') < f(s) então
9              s ← s'';
10             k ← 1 ;
11         senão
12             k ← k + 1
13         fim se
14     fim enqto
15 fim enqto
16 Retorne s;
```

Depois de escolhida a estrutura de vizinhança, é necessária a geração de uma solução inicial, a qual consiste de uma solução s com a sequência das cidades visitadas pelo caixeiro viajante. Atribuímos a essa solução inicial o valor gerado pela contagem de frequência descrita anteriormente.

Então a cada iteração seleciona-se aleatoriamente um vizinho s' dentro da vizinhança $N_{(s)}^{(k)}$ da solução s corrente. Esse vizinho é então submetido a um procedimento de busca local. Se a solução ótima local, s'', for melhor que a solução s corrente, a busca continua de s'' recomeçando da primeira estrutura de vizinhança $N_{(s)}^{(1)}$. Caso contrário, continua-se a busca a partir da próxima estrutura de vizinhança $N_{(s)}^{(k+1)}$.

Este procedimento é encerrado quando uma condição de parada for atingida, tal como o tempo máximo permitido de CPU, o número máximo de iterações ou número máximo de iterações consecutivas entre dois melhoramentos. O VNS possui alguns princípios básicos, entre eles [Hansen et al., 2008]:

- Um ótimo local com relação a uma vizinhança não necessariamente corresponde a um ótimo local com relação a outra vizinhança;
- Um ótimo global corresponde a um ótimo local para todas as estruturas de vizinhança;

- Para muitos problemas, ótimos locais, com relação a uma vizinhança são relativamente próximos.

Os itens 1 e 2 sugerem, o uso de várias estruturas de vizinhança nas buscas locais para abordar um problema de otimização, ou seja, a ideia é definir um conjunto de estruturas de vizinhanças que possam ser utilizadas de forma determinística, estocástica e ambas.

O último item, de natureza empírica, indica que um ótimo local frequentemente fornece algum tipo de informação sobre o ótimo global. Este é o caso em que os ótimos local e global compartilham muitas variáveis com o mesmo valor, o que sugere uma investigação sistemática da vizinhança de um ótimo local até a obtenção de uma nova solução de melhor valor.

4.4 Considerações finais

Sabe-se que a utilização de Meta-heurísticas aumentou significativamente a habilidade de se encontrar soluções de alta qualidade, em tempo relativamente baixo, para problemas difíceis na Área de Otimização Combinatória. Neste capítulo foram apresentadas duas Meta-heurísticas Populacionais para resolver o Problema do Caixeiro Viajante: Algoritmo Genético e Otimização por Colônia de Formigas. Essas Meta-heurísticas trabalham com um conjunto de soluções concorrentemente em cada iteração.

O GA é uma das Meta-heurísticas mais utilizadas atualmente. É de fácil implementação e tem apresentado resultados relevantes na resolução de problemas. O ACO costuma apresentar resultados em geral de qualidade tão boa quanto o GA, com a vantagem de necessitar de uma população menor e possivelmente menos iterações, portanto, menos avaliações.

Com o desenvolvimento de Meta-heurísticas surge, também, a necessidade de verificar sua eficácia e eficiência junto com outras técnicas. No próximo capítulo será abordado, uma estratégia híbrida que utiliza técnicas de Agrupamento para melhorar as Meta-heurísticas Populacionais.

Capítulo 5

Mineração de Dados

Mineração de Dados (MD) é definida como um processo de descoberta de padrões em grandes quantidade de dados. Este processo pode ser automático (mais usado) ou semi-automático [Witten, 2011]. A noção de encontrar padrões úteis em grandes volumes de dados pode ser conhecida por diversos nomes, tais como mineração de dados, extração de conhecimento, descoberta de informações, arqueologia de dados e processamento de padrões de dados. O termo mineração de dados é o mais usado por profissionais da computação, estatísticos e analistas de dados.

A MD pode ser aplicada de duas formas: como um processo de aprendizagem supervisionada e não supervisionada. Na aprendizagem supervisionada é sugerida uma classe acerca da relação entre os dados e tenta-se predizer em qual classe uma instância não vista pertence, com base nos exemplos utilizados. Na aprendizagem não supervisionada não é feita nenhuma suposição antecipada, ou seja, nenhuma classe é conhecida. Este tipo de aprendizado é utilizado geralmente para análise exploratória dos dados, utilizando técnicas de agrupamento ou regras de associação [Han, 2006].

Conforme dito na introdução desta dissertação, o processo de hibridização desenvolvido neste trabalho utiliza as técnicas de agrupamento, tendo em vista que não se sabe nada em relação aos dados gerados. Desta forma, este capítulo visa apresentar os algoritmos utilizados para tal hibridização.

5.1 Regras de Agrupamento (Clustering)

Análise de agrupamento, ou *clustering*, é o nome dado para o grupo de técnicas computacionais cujo propósito consiste em segmentar populações heterogêneas de dados em grupos ou *clusters*, baseando-se nas características que estes dados possuem. A ideia básica é colocar em um mesmo grupo dados que possuem características simila-

res em seus atributos de acordo com algum critério pré-determinado [Hartigan, 1975]. Esta segmentação é feita de modo a maximizar as diferenças encontradas entre *clusters* diferentes e minimizar as diferenças entre exemplos pertencentes ao mesmo *cluster*.

Um exemplo [Keogh, 2003], seria dividir a população da figura 5.1 em dois grupos de acordo com algum critério.

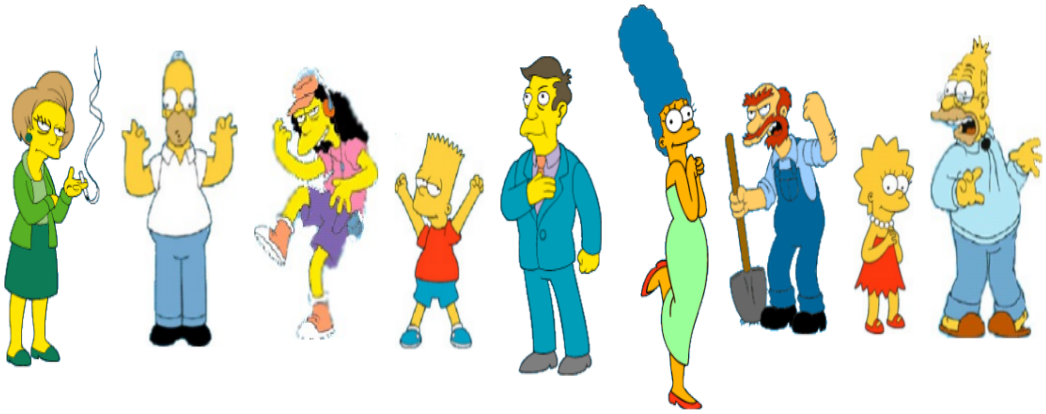
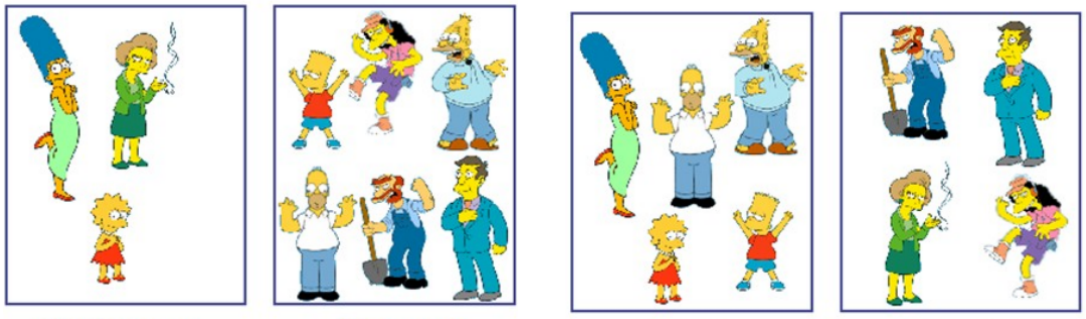


Figura 5.1. Exemplo de grupo de pessoas que podem ser dividida de acordo com alguma similaridade.



(a) Mulheres e Homens. (b) Família e Escola.

Figura 5.2. Possíveis soluções de agrupamentos.

Existem outras soluções, além da figura 5.2: crianças e adultos, gordos e magros, inteligentes e nem tanto, fumantes e não-fumantes.

Para saber quão próximo estão os objetos, devemos primeiramente desenvolver uma escala quantitativa de maneira a medir a similaridade entre estes objetos. Esta medida avalia o quanto dois objetos são parecidos e se devem, ou não, ser colocados em um mesmo grupo. Uma abordagem comum para comparação de dois objetos i e j é a associação de uma função de distância, onde o número $d(i, j)$ representa a

distância entre os objetos. Primeiramente tem-se um conjunto de objetos organizados em uma matriz, onde cada linha da matriz representa um objeto i e cada coluna representa os valores de um atributo assumidos por cada um dos objetos. Este conjunto é transformado em uma matriz de distâncias, através das funções $d : \Gamma \times \Gamma \Rightarrow \mathbb{R}$, onde Γ denota o conjunto de dados que estamos trabalhando [Linden, 2009].

$$\begin{bmatrix} x_{1,1} & x_{1,2} & \dots & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & \dots & x_{2,n} \\ x_{3,1} & x_{3,2} & \dots & \dots & x_{3,n} \\ \vdots & \vdots & \vdots & \vdots & \\ x_{n,1} & x_{n,2} & \dots & x_{n,n-1} & x_{n,n} \end{bmatrix} \xrightarrow{d:\Gamma\times\Gamma\Rightarrow\mathbb{R}} \begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d(n,1) & d(n,2) & \vdots & d(n,n-1) & 0 \end{bmatrix}$$

Para que uma função d seja uma distância é necessário e suficiente que as seguintes condições sejam satisfeitas:

1. $d_{ii} = 0, \forall i \in \Gamma$
2. $d_{ij} \geq 0, \forall i, j \in \Gamma$
3. $d_{ij} = d_{ji}, \forall i, j \in \Gamma$ (simetria)

Assim, qualquer função que satisfaz às três propriedades acima é chamada de distância. As mais importantes funções nesta categoria são:

- Distância Euclidiana:
  $d_{ij} = \sqrt[2]{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots + |x_{in} - x_{jn}|^2}$
- Distância de Manhattan (City Block):
  $d_{ij} = \sqrt[1]{|x_{i1} - x_{j1}|^1 + |x_{i2} - x_{j2}|^1 + \ldots + |x_{in} - x_{jn}|^1}$
- Distância de *Minkowski*:
  $d_{ij} = \sqrt[q]{|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \ldots + |x_{in} - x_{jn}|^q}$

onde $q \geq 1$. Logo, a distância de *Minkowski* generaliza tanto a distância euclidiana (caso especial onde $q = 2$) quanto a distância de Manhattan (caso especial onde $q = 1$). Existem ainda, a distância de *Chebychev*, também conhecida como métrica máxima ou distância do tabuleiro de xadrez, que é quando o valor de q vai ao infinito, neste caso a fórmula da distância é dada por:$d_{ij} = max(|i_k - j_k|)$. A figura 5.3 ilustra o formato dos grupos de acordo com a métrica utilizada para o calculo das distâncias.

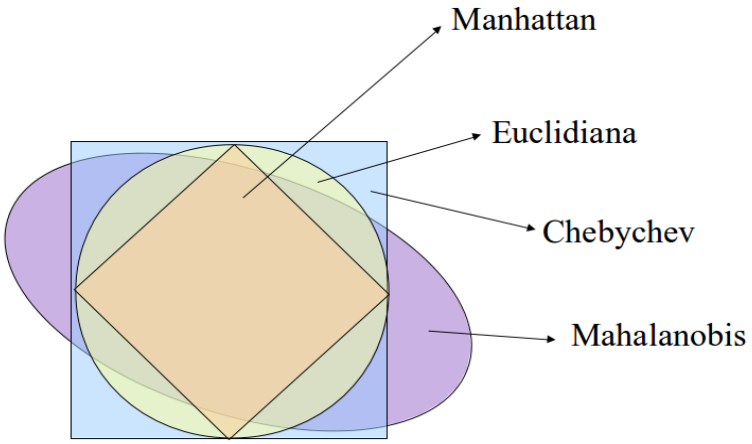


Figura 5.3. Formatos dos *clusters* de acordo com a função de distância utilizada Jain et al. [1999].

Os algoritmos de *clustering* podem ser classificados considerando diferentes aspectos. Uma classificação bastante aceita é fornecida por Jain et al. [1999], na qual os algoritmos são classificados de acordo com o método adotado para definir os *clusters*: particionais, grade, densidade e hierárquicos. A seguir são apresentadas essas abordagens.

***Clustering* Particional:** dado o número de grupos k, o método cria k partições iniciais e, em seguida, usa uma técnica de realocação iterativa, de forma a melhorar a posição dos objetos dentro da cada grupo.

***Clustering* Baseado em Grade:** Essa abordagem define uma grade para o espaço de objetos e realiza todas as operações nesse espaço quantizado. Em termos gerais, essa abordagem é capaz de encontrar *clusters* de formatos arbitrários, não apresenta alta sensibilidade aos *outliers* e é muito eficiente para grandes conjuntos de dados.

***Clustering* Baseado em Densidade:** Essa abordagem assume que os *clusters* são regiões de alta densidade separadas por regiões com baixa densidade no espaço de exemplos. A ideia dessa abordagem é que cada exemplo do *cluster* deve manter uma vizinhança com um número mínimo de vizinhos dentro de uma esfera com raio R. Os exemplos que possuem uma vizinhança com densidade mínima, e estão numa distância menor que R, pertencem ao mesmo *cluster*. Os métodos baseados em densidade permitem a geração de grupos com formatos arbitrários (não somente esféricos).

***Clustering* Baseado em Modelos:** Criam um modelo para cada grupo e encontram o melhor ajuste dos dados para cada modelo. Os algoritmos baseados em modelo alocam os grupos construindo uma função de densidade que reflete a distribuição espacial dos objetos. Eles também permitem uma forma de determinação automática do número de grupos baseados em padrões estatísticos.

***Clustering* Hierárquico:** Produzem uma hierarquia que consiste em uma série de sucessivos agrupamentos ou sucessivas divisões de elementos, onde os elementos são agregados ou desagregados. São representados por uma árvore, onde nós folhas são objetos e os nós intermediários representam os agrupamentos que contêm todos os objetos de seus descendentes. Os métodos hierárquicos são subdivididos em métodos aglomerativos e divisivos.

- ***Bottom-Up* (Aglomerativos):** Produzem uma sequência de agrupamentos com um número decrescente de *clusters*, onde cada elemento inicia-se representando um grupo, e a cada passo, um grupo ou elemento é ligado a outro de acordo com sua similaridade, até o último passo, onde é formado um grupo único com todos os elementos.
- ***Top-Down* (Divisivos):** Atuam na direção oposta, isto é, um grupo inicial contendo todos os elementos é dividido em dois subgrupos, de tal forma que os elementos em um subgrupo estejam distantes dos elementos do outro subgrupo. Esses subgrupos são então divididos em subgrupos dissimilares e o processo continua até cada elemento formar um grupo.

A análise de agrupamento é uma ferramenta útil para a análise de dados em muitas situações diferentes. Esta técnica pode ser usada para reduzir a dimensão de um conjunto de dados, reduzindo uma ampla gama de objetos à informação do centro do seu conjunto[Clarke et al., 2009]. Tendo em vista que *clustering* é uma técnica de aprendizado não supervisionado, pode servir também para extrair características escondidas dos dados e desenvolver as hipóteses a respeito de sua natureza.

5.1.1 Algoritmo *K-Means*

O *K-Means* é um método de agrupamento por repartição, baseado em protótipo onde cada *cluster* é representado por um centroide. Centroides representam o ponto p médio mais central do *cluster*, sendo calculado pela média de todos os objetos pertencentes ao *cluster*. Ele tem como critério de agrupamento minimizar a distância entre os elementos a um conjunto de k centros dada por $C = c_1, c_2, \ldots, c_k$ de forma iterativa

[Mirkin, 2005]. Ou seja, o objetivo é minimizar o erro quadrático da partição, definido como:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} d(p - c_i)^2$$

Para medir a similaridade entre os objetos usou-se a distância euclidiana que é uma medida de proximidade. O número de *clusters* onde os objetos serão atribuídos é definida previamente de forma *ad hoc* pelo usuário. A ideia por trás deste algoritmo é escolher k objetos (aleatoriamente ou com alguma heurística) que serão a base de cada grupo (os centroides), os demais objetos são associados ao centroide mais próximo. A cada passo os centroides são recalculados dentre os objetos de seu próprio grupo e os objetos são realocados para o centroide mais próximo. Este procedimento é repetido até que o nível de convergência seja satisfatório.

Algoritmo 4: *K-Means*

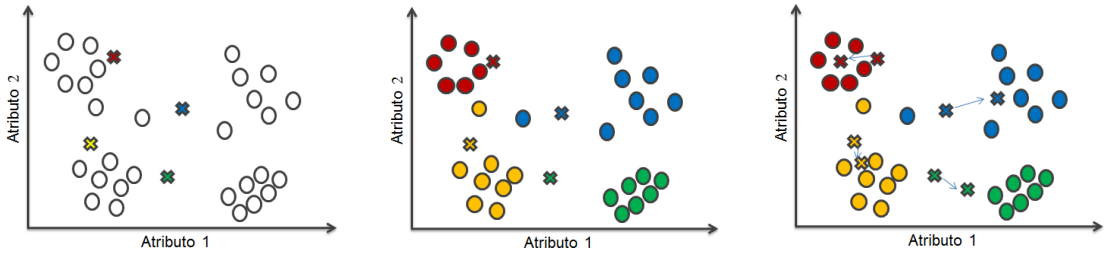
Entrada: Conjunto de dados com n objetos; O número k de grupos

Saída: Um conjunto de k grupos

1 Selecionear arbitrariamente K pontos como centróides iniciais;
2 **repita**
3 | Atribua cada objeto ao cluster mais próximo;
4 | Re-calcula cada centróide de cada cluster;
5 **até** *até que os centróides permaneçam estáveis*;

O *K-Means*, algoritmo 4, possui uma complexidade computacional equivalente a $O(N \times k \times T)$, pois, a cada iteração (T), é calculada a distância entre os N objetos até cada um dos k centroides. Sua vantagem é a eficiência em tratar grandes conjuntos de dados e suas desvantagens são o fato do usuário ter que fornecer o número de *clusters* k , o fato de não descobrir *clusters* de formatos não-convexos e sobretudo o fato de ser sensível a ruídos, já que objetos com valores altos podem causar uma grande alteração no centro de gravidade dos *clusters* e assim, distorcer a distribuição dos dados nos mesmos.

A Figura 5.4 ilustra o funcionamento do método *K -Means*. Na primeira iteração, figura 5.4(a), os objetos com $\mathbb{X}$ representam os centroides que foram escolhidos aleatoriamente. Em 5.4(b), temos a execução da linha 3 do algoritmo, a atribuição de cada objeto ao grupo cujo protótipo possui maior similaridade ao objeto. A figura 5.4(c) representa a linha 4, onde os valores dos protótipos são recalculados como sendo a média dos objetos atualmente presentes em cada grupo. Em 5.5(a) temos a preparação para a iteração do algoritmo no qual os objetos são desvinculados dos grupos ao qual pertenciam, 5.5(b) e 5.5(c) representa o laço de iterações do algoritmo.

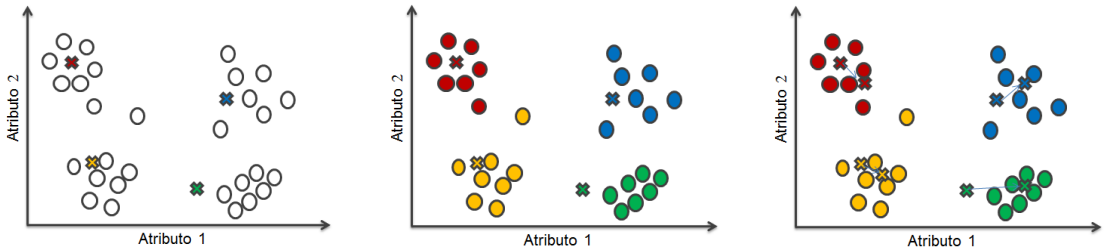


(a) Passo 1: Sorteio dos valores iniciais dos centróides.

(b) Passo 2: Atribuição dos objetos aos grupos de menor distância.

(c) Passo 3: Re-calcular os centros de gravidade.

Figura 5.4. Primeira iteração do Algoritmo *K-Means*



(a) Preparação para repetição dos passos 2 e 3.

(b) Repetição do passo 2.

(c) Repetição do passo 3.

Figura 5.5. Etapas do Algoritmo *K-Means*

5.1.2 Algoritmo *Ward*

O método de Ward foi proposto por [Ward, 1963] e é também chamado de " Mínima Variância " [Mingoti, 2005], pois ele forma grupos de maneira a atingir sempre o menor erro interno entre os vetores que compõem cada grupo e o vetor médio do grupo. Isto equivale a buscar o mínimo desvio padrão entre os dados de cada grupo.

Ele consiste em um procedimento hierárquico no qual a medida de similaridade usada para juntar agrupamentos é calculada como a soma de quadrados(SQ) entre os dois agrupamentos feita sobre todas as variáveis. Isto é, ele tenta minimizar a soma dos quadrados dentro do grupo e com isso, maximizar a homogeneidade dentro dos grupos. Os grupos formados em cada passo são resultantes de grupo solução com a menor soma de quadrados [Sharma, 1996].

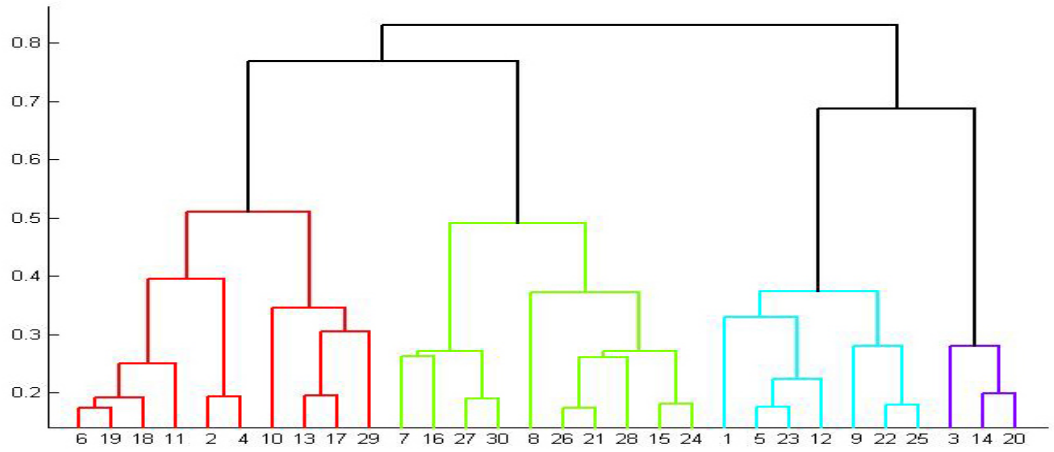


Figura 5.6. Dendograma do método *Ward* [Kaufman and Rousseeuw, 2005].

Esse método tende a resultar em agrupamentos de tamanhos aproximadamente iguais devido a sua minimização de variação interna ilustrado na figura 5.6. Em cada estágio, combinam-se os dois agrupamentos que apresentarem menor aumento na soma global de quadrados dentro dos agrupamentos. A soma dos erros quadráticos para cada agrupamento é definida como:

$$SQ_k = \sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2$$

onde k é o agrupamento em questão, n é o número total de objetos do agrupamento k e x_i é o i-ésimo objeto do agrupamento k.

Algoritmo 5: Algoritmo *Ward*

Entrada: Conjunto de dados com n objetos

Saída: Um conjunto de k grupos

1 Calcular as $m \times m$ combinações de possiveis agrupamentos;
2 Calcular as médias das variáveis para cada grupo;
3 **repita**
4 Atribua cada objeto ao grupo que causa menor aumento no erro interno;
5 Calcular novas combinações possiveis;
6 **até** $m - 1$ *vezes*;

O algoritmo 5 mostra o funcionamento do método *Ward*. No princípio, têm-se m grupos cada um com tamanho 1; ou seja, um grupo para cada vetor componente da base de dados. Neste estágio inicial o erro interno é nulo para todos os grupos pois cada vetor que compõe cada grupo é o próprio vetor médio do grupo. Igualmente o desvio padrão para cada grupo é nulo.

Na etapa subsequente, cada possibilidade de aglutinação entre os grupos 2 a 2 é verificada, e é escolhido o agrupamento que causa o menor aumento no erro interno do grupo. São $m \times m$ verificações. Desta forma, para uma base de dados com m elevado, estas verificações exigem um grande esforço computacional caso o método seja implementado em computador. Nota-se que a cada iteração tem-se $m - i$ grupos (i =número de iterações), no entanto, como o número de elementos pertencentes a cada grupo aumenta, é maior o número de cálculos para o erro interno de cada grupo. O algoritmo para quando todas as unidades da amostra são combinados num único grande aglomerado de tamanho m.

Algumas das características desse método são:

- Apresenta bons resultados tanto para distâncias euclidianas quanto para outras distâncias;
- Pode apresentar resultados insatisfatórios quando o número de elementos em cada grupo é praticamente igual;
- Tendência a combinar grupos com poucos elementos;
- Sensível à presença de *outliers*.

5.2 Considerações Finais

O uso da Mineração de Dados neste trabalho tem como objetivo encontrar conhecimento a partir de um conjunto de dados para ser utilizado em um processo decisório. Como sabe-se pouco sobre o comportamento dos dados a melhor forma de obter conhecimento sobre os mesmos é usar uma técnica preliminar de mineração de dados. Técnica esta chamada de Regras de Agrupamento. Os algoritmos de agrupamento podem ser utilizados para reduzir o conjunto grande de soluções sem perder sua característica de diversidade, agrupando soluções similares em grupos. Neste capítulo foram apresentadas as características do *clustering*, entre elas, as abordagens mais utilizadas, as medidas de similaridade frequentemente utilizadas e os algoritmos utilizados para a hibridização do método proposto.

Capítulo 6

Estratégia Híbrida Proposta e Resultados Experimentais

Neste capítulo, apresenta-se a contribuição central deste trabalho: a versão híbrida das meta-heurísticas AG e Colônia de Formiga com os métodos de Agrupamento, além de uma versão que incorpora uma busca local. Essa meta-heurística híbrida é composta basicamente de duas fases, descritas a seguir.

A primeira fase consiste na geração de dois grupos. Desses dois grupos, encontra-se o grupo mais promissor: o grupo que possui as melhores soluções, ou seja, as soluções com o menor valor de *fitness* . Por sua vez, esta é dividida em duas etapas. Inicialmente, os algoritmos para o Problema do Caixeiro Viajante, descritos em 4.1.1 e o 4.2.1, são executados até gerar uma população, um conjunto S de indivíduos ou formigas. Em seguida, um algoritmo de Agrupamento (*K-means* ou *Ward*) é executado sobre essa população afim de extrair um grupo considerado promissor.

Na segunda fase, encontram-se conjuntos de itens que aparecem com frequência nestas soluções. Esta fase consiste basicamente na contagem de frequência das posições das cidades. E a partir dai, constrói-se a solução, fazendo-se uma busca local na mesma.

6.1 Contagem de Frequência

A contagem de frequência é um passo muito importante para esse método, pois é a partir dela que geramos uma solução. Primeiramente contamos quantas vezes cada cidade aparece em cada posição, gerando a matriz de frequência, o histograma. A partir da matriz de frequência, geramos duas soluções da seguinte forma: Na primeira encontra-se a cidade com maior frequência em cada posição e na segunda é encontrada a posição na qual a cidade aparece mais vezes.

As duas soluções foram geradas para responder a questões como: qual cidade aparece mais vezes na posição A, no vetor de solução; e em qual posição do vetor de solução a cidade 4, aparece mais vezes. Essas questões foram elaboradas devido a forma com a qual é feita a contagem, que é sequencialmente, ocorrendo assim, a seguinte situação: a cidade mais frequente em uma posição pode ter uma frequência muito maior em outra posição, ou seja, uma cidade pode ser a mais frequente em duas ou mais posições no vetor de soluções.

O exemplo a seguir ilustra a contagem de frequência. A primeira matriz é a população gerada a partir das Meta-heurística utilizadas, com seis indivíduos e quatro cidades e a segunda é a matriz de frequência. As linhas da matriz de população representam os indivíduos que são soluções para o problema tratado, e as colunas a posição em que cada cidade irá ficar para formar a solução. Na matriz de frequência as linhas correspondem às posições no vetor de solução, ou seja, linha 1 é a primeira posição do vetor de solução e as colunas as cidades que irão compor a solução.

$$
\begin{array}{l}
 \\
\textit{individuo } 1 \\
\textit{individuo } 2 \\
\textit{individuo } 3 \\
\textit{individuo } 4 \\
\textit{individuo } 5 \\
\textit{individuo } 6
\end{array}
\begin{array}{c}
\begin{array}{cccc} p_1 & p_2 & p_3 & p_4 \end{array} \\
\left[\begin{array}{cccc}
c_2 & c_3 & c_1 & c_4 \\
c_3 & c_4 & c_1 & c_2 \\
c_1 & c_2 & c_4 & c_3 \\
c_4 & c_2 & c_3 & c_1 \\
c_2 & c_1 & c_3 & c_4 \\
c_4 & c_2 & c_3 & c_1
\end{array}\right]
\end{array}
\xrightarrow{\text{Frequência}}
\begin{array}{c}
\\
p_1 \\
p_2 \\
p_3 \\
p_4
\end{array}
\begin{array}{c}
\begin{array}{cccc} c_1 & c_2 & c_3 & c_4 \end{array} \\
\left[\begin{array}{cccc}
1 & 2 & 1 & 2 \\
1 & 3 & 1 & 1 \\
2 & 0 & 3 & 1 \\
2 & 1 & 1 & 2
\end{array}\right]
\end{array}
$$

Note que a matriz de frequência pode ser lida de duas formas. A primeira, é percorrendo as linhas, isto é, a forma que gera a primeira solução e respondendo a primeira pergunta. A segunda, é percorrendo as colunas, gerando-se assim, a segunda solução e respondendo então, a segunda pergunta. Observando a matriz de frequência a cidade c_2, aparece duas vezes na posição p_1 e três vezes na posição p_2, gerando a situação citada anteriormente.

Para este exemplo a primeira solução gerada seria percorrendo a matriz de frequência linha a linha. A cidade com maior frequência na posição p_1 é a c_2, a cidade com maior frequência na posição p_2 é também a cidade c_2, mas como ela já foi alocada para a posição p_1 a cidade com a segunda maior frequência será escolhida. Como nessa posição todas as cidades têm a mesma frequência, a escolha será aleatória. Escolhemos então a cidade c_3 para alocar a posição p_2. Na próxima posição, p_3, a cidade com maior frequência é a cidade c_3 que já foi selecionada anteriormente, não podendo ser escolhida novamente. Então, a próxima cidade com maior frequência é a cidade c_1. Para ocupar

a posição p_4, restou apenas a a cidade c_4, formando assim a primeira solução: c_2, c_3, c_1 e c_4.

A segunda solução é gerada percorrendo a matriz de frequência coluna a coluna. A cidade c_1 e tem a maior frequência em duas posições: a posição p_3 e a posição p_4, dessa forma, escolhemos de forma aleatória a posição que a cidade c_1 irá ocupar. Escolhemos a posição p_3. A cidade c_2 aparece mais vezes na posição p_2, então é nessa posição que ela será alocada. Já a cidade c_3 é mais frequente na posição p_3, mas infelizmente essa posição já foi alocada pela cidade c_1, restando então escolher a segunda posição na qual a cidade c_3 aparece mais. Como houve empate entre a posição p_1 e p_4, pois as posições p_2 e p_3 já foram alocadas, a escolha é feita de forma aleatória. Escolhemos a posição p_1 restando apenas a posição p_4 para a cidade c_4. A segunda solução gerada é: c_3,c_2,c_1 e c_4.

De posse das duas soluções, escolhe-se a solução com o melhor *fitness* para ser usada. Optou-se por essa abordagem, devido ao baixo custo computacional que ela proporciona, pois buscando-se a cidade com a maior frequência de todas e encontrando a posição da mesma sucessivamente, acarretaria em um tempo de execução muito elevado.

6.2 Algoritmo Genético com Agrupamento

A importância desta abordagem com AG, está no fato de ter sido a primeira a ser testada e sua simplicidade se justifica por ser uma investigação inicial para corroborar a suspeita de que a introdução de algoritmo de agrupamento poderia trazer bons resultados para meta-heurísticas populacionais. A partir da comprovação do potencial deste tipo de hibridação, através dos bons resultados obtidos, apresentados na seção 6.4, foi possível dar sequência aos estudos e testar outros algoritmos de agrupamento. Além disso, a maioria dos métodos vistos nos trabalhos relacionados, capítulo 2, utilizam AG's em suas técnicas.

Inicialmente, executa-se o AG, gerando um conjunto de soluções distintas. Em seguida, executa-se o algoritmo de agrupamento sobre o conjunto de soluções a fim de dividi-lo em dois grupos de maior proximidade. Cada individuo, na população será tratado como um ponto, então gera-se dois grupos onde os indivíduos são mas próximos. Como não sabemos qual o grupo que contém os melhores resultados, calculamos a média dos *fitness* de cada grupo. O grupo que tinha menor média é considerado como o grupo que está mais próximo da solução ótima, ou seja, o grupo promissor. No subconjunto promissor encontra-se a cidade com a maior frequência que aparece em cada posição e

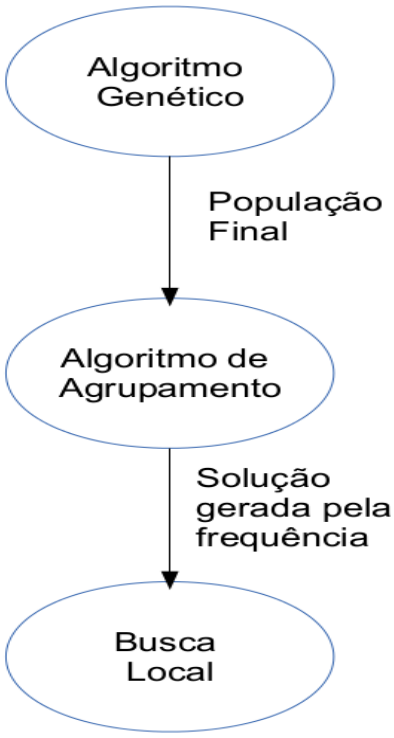


Figura 6.1. Esquema do AG Híbrido

gera-se uma solução com esta sequência de cidades. Depois executa-se o algoritmo de Busca Local *VNS* sobre essa solução. Na figura 6.1 é apresentado o esquema do AG híbrido proposto.

A frequência é utilizada para reconhecer as sequências de cidades que mais aparecem no grupo promissor de soluções viáveis. O gráfico 6.2 mostra o histograma de frequência na instância ftv44 com 44 cidades. As primeiras e as últimas posições obtiveram frequências altas para o grupo de soluções boas. A primeira posição é sempre fixa, pois o caixeiro sempre começa da cidade 1. Primeiramente encontra-se a cidade mais frequente em cada posição, que compõe o caminho da solução. Depois, para cada cidade, verifica-se em qual posição ela aparece mais vezes.

A geração da população inicial do AG é bastante simples. A escolha das cidades que farão parte da solução é realizada de maneira aleatória. A função de aptidão de uma solução é medida pela função objetivo do PCV. Os parâmetros usados para o AG foram $p = 0,02$, população de 256 indivíduos (cromossomos) com a quantidade de gerações variável, com no máximo 100000. Condição de término é que o melhor indivíduo encontrado seja o mesmo em 5.000 interações. Esses são os parâmetros otimizados que foram retirados do artigo [Tao and Michalewicz, 1998]. O parâmetro p é uma probabilidade (baixa) que é usada como controle do operador *Inver-Over*.

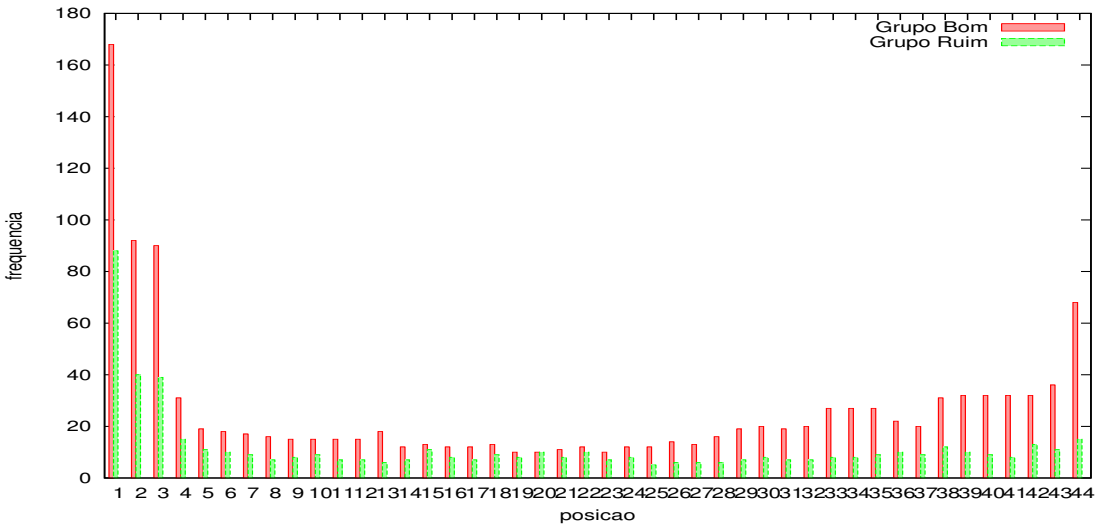


Figura 6.2. Instância *ftv44* com 44 cidades.

6.3 *Ant System* com Agrupamento

Assim como o AG, o *Ant System (AS)* também trabalha com um conjunto de soluções, mas esse conjunto de soluções não é considerado na iteração seguinte. Em cada iteração o algoritmo testa qual é a melhor formiga, ou seja, a formiga que obteve o melhor caminho e guarda esse caminho para a iteração seguinte. E todas as formigas depositam feromônio nas arestas e essa matriz de feromônio será levada para o próximo passo. Apesar da natureza estocástica do algoritmo, a forte concentração de feromônio nas arestas força a formiga a fazer sempre o mesmo percurso tornando seu desempenho insuficiente em relação a outras técnicas, por isso, a tentativa de melhorar esse método através de técnicas de MD.

A base para o agrupamento é constituída pelas melhores formigas de cada iteração do algoritmo. Essa base é chamada de Conjunto Elite. Depois da construção da base o procedimento é exatamente o mesmo que ocorre no AG. Executa-se o algoritmo de Agrupamento sobre esse conjunto dividindo em dois conjuntos. Descobre-se, através do erro médio, qual o grupo mais promissor, ou seja, o grupo que contém as melhores soluções e esse grupo será usado para a fase de contagem de frequência.

No AS existem vários parâmetros que devem ser ajustados para obter um bom desempenho e o ajuste correto desses parâmetros é fundamental para se obter um bom equilíbrio entre intensificação e diversificação. Dorigo propõe que β fique em torno de 5. Geralmente, $m = n$, isto é, a quantidade de formigas igual a quantidade de cidades. Por fim, inicializa o feromônio de todas as arestas com $\frac{1}{nL_{nn}}$, onde L_{nn} é o custo de uma construção puramente gulosa.

A heurística gulosa utilizada foi o do Vizinho Mais Próximo descrita no algoritmo 6 [Bang-Jensen et al., 2004]. Nesta heurística, parte-se da cidade origem, selecionada aleatoriamente, e adiciona-se a cada passo a cidade j ainda não visitada cuja distância à última cidade visitada é a menor possível. O procedimento de construção termina quando todas as cidades forem visitadas, situação na qual é feita a ligação entre a última cidade visitada e a cidade origem [Gutin et al., 2002].

Algoritmo 6: Vizinho Mais Próximo para o PCV.

```
1 Escolha vertice inicial v;
2 Faca S ← v;
3 enquanto nao completar a solucao faça
4 |   Seja C = (v_1, ..., v_i);
5 |   Escolha um vertice u mais proximo do extremo de C;
6 |   Acrescente u ao estremo de C;
7 fim enqto
8 Retorne C;
```

6.4 Resultados Experimentais

Neste seção, são apresentados os resultados obtidos a partir dos experimentos realizados com as diferentes versões do Algoritmo Genético e *Ant System* e ambos com Agrupamento, descritas no capítulo anterior. Objetiva-se avaliar o resultado da incorporação de métodos de Agrupamento nas meta-heurísticas Algoritmo Genético e Colônia de Formigas aplicada ao Problema do Caixeiro Viajante. Esta seção esta organizado da seguinte forma. Inicialmente, as características das instâncias utilizadas para a validação dos algoritmos são apresentadas. Em seguida, são descritos o ambiente computacional onde os testes foram executados e os parâmetros utilizados para a execução dos algoritmos. Por fim, as estratégias híbridas são comparadas.

6.4.1 Instâncias Utilizadas

Foi utilizado um conjunto de 26 instâncias retiradas da base de dados do TSPLIB[1], que é um repositório de dados com várias fontes e tipos de exemplos relacionados ao PCV. Os arquivos fornecidos seguem uma padronização específica, com extensão .tsp, onde o ótimo global e o melhor caminho são conhecidos. As características das instâncias

[1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Tabela 6.1. Características das instâncias utilizadas.

Instância	Nº cidade	Valor Ótimo
gr17	17	2085
br17	17	39
ftv33	33	1286
ftv35	35	1473
swiss42	42	1273
p43	43	5620
ftv44	44	1613
ftv47	47	1776
ry48p	48	14422
berlin52	52	7542
ft53	53	6905
ftv55	55	1608
brazil58	58	25395
ftv64	64	1839
ft70	70	38673
ftv70	70	1950
pr76	76	108159
gr120	120	6942
kro124p	124	36230
ftv170	170	2755
si175	175	21407
brg180	180	1950
rbg323	323	1326
si535	535	48450
d1655	655	62128
u2152	2152	64253

utilizadas na investigação são apresentadas na tabela 6.1, que inclui ainda o valor da solução ótima.

6.4.2 Ambiente de Execução

Os algoritmos testados foram implementados na linguagem C, compilados com o compilador **gcc**, no sistema operacional Linux, com *kernel* versão e os experimentos foram conduzidos em um PC com processador I5 2.5GHz e memória de 3GB. Os experimentos foram realizados com o objetivo de validar a abordagem propostae mostrar que a estratégia híbrida pode ser competitiva para resolver o PCV.

Para todas as execuções do GA para o PCV realizados, os parâmetros usados foram $p = 0,02$, população de 256 indivíduos (cromossomos) com a quantidade de gerações variável, com no máximo 100000. Condição de término do algoritmo é que

o melhor indivíduo encontrado seja o mesmo em 5.000 interações. Esses parâmetros foram retirados do artigo [Tao and Michalewicz, 1998]. Como algumas instâncias são muito grandes e os algoritmos demoravam muito, foi estipulada uma quantidade de tempo para tornar-lo viável. Vale ressaltar neste ponto que decidiu-se não variar os parâmetros do AG nem do ACO pois o foco da pesquisa consiste em isolar a contribuição do Agrupamento na meta-heurística. A variação desses parâmetros consiste em avaliar o desempenho dos algoritmos propriamente dito, quando aplicado ao problema.

Para a obtenção dos resultados apresentados, todas as estratégias foram executadas, para cada instância, vinte vezes durante 120 minutos, utilizando-se vinte sementes diferentes no caso do AG. São reportados para análise: o desvio padrão, a média e o melhor valor obtido nestas vinte execuções de cada estratégia, para cada instância.

6.4.3 Comparação entre as estratégias

A ordem dos resultados apresentados é decorrente da sequência de desenvolvimento descrito anteriormente. Desta forma, a Tabela 6.2 compara o comportamento do AG simples para o PCV e a estratégia com o algoritmo de Agrupamento *K-means*. Ela possui apenas dezesseis instâncias, pois as demais o AG chegou no Ótimo, não havendo, portanto, motivos para a comparação.

As estatísticas usadas para avaliar o desempenho dos métodos foram a média e o desvio padrão. A primeira é definida como o valor que aponta para onde mais se concentram os dados de uma distribuição. Ela é obtida através do seguinte fórmula: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, onde n é a quantidade de amostras. A segunda é a medida que mostra o quanto de variação ou dispersão existe em relação a média. Um baixo desvio padrão indica que os dados tendem a estar próximos da média; um desvio padrão alto indica que os dados estão espalhados por uma gama de valores. O desvio padrão para uma amostra de n números pode ser calculado da seguinte forma: $s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$, onde $\bar{x}$ é a média já definida anteriormente.

A primeira coluna da tabela identifica as instâncias utilizadas. A segunda e terceira colunas trazem, respectivamente, a média entre os valores encontrados pelas vinte execuções do GA e o melhor resultados entre elas. A quarta coluna traz o desvio padrão calculado para os mesmos dezesseis valores utilizados para o calculo da média. A quinta, sexta e sétima colunas trazem a média, melhor solução e desvio padrão para os resultados da estratégia com Agrupamento. Para todas as instâncias, os melhores valores de cada critério (média, melhor valor e desvio padrão) estão destacados em negrito. Deste ponto do texto em diante, este recurso será utilizado em todas as tabelas comparativas envolvendo as estratégias implementadas.

Considerando os valores das médias para as dezesseis instâncias da tabela, a estratégia híbrida AG com Agrupamento obteve melhores resultados para doze delas, enquanto que o AG simples obteve melhor resultado em apenas quatro instâncias. Para os melhores valores, o AG com Agrupamento obteve melhor desempenho em onze instâncias, enquanto que o AG simples conseguiu superá-lo em apenas quatro delas. E em apenas uma instância, ambas estratégias obtiveram resultados idênticos quanto a esse critério. Em praticamente todas as instâncias, o desvio padrão observado na estratégia proposta foi bem pequeno, o que demonstra estabilidade na qualidade do desempenho da técnica.

Tabela 6.2. Comparação entre as estratégias AG simples e AG com Agrupamento e Busca Local.

Instância	AG simples			AG com *K-means* e Busca Local		
	Média	Melhor	Desvio Padrão	Média	Melhor	Desvio Padrão
p43	5734.00	5727	25.651	**5727.00**	5727	0.000
ftv44	1746.40	1681	132.334	**1624.50**	**1623**	10.700
berlin52	15106.10	14969	319.585	**14922.70**	**14914**	22.452
ftv64	2094.80	1948	282.031	**1849.20**	**1839**	14.408
ft70	40723.60	40234	963.621	**39112.90**	**38926**	321.283
ftv70	2237.600	1982	341.889	**1964.80**	**1959**	9.359
pr76	279112.00	275897	13207.952	**275462.50**	**275325**	769.705
gr120	10540.00	9857	959.311	**7933.80**	**7797**	258.773
kro124p	40635.70	38111	5297.701	**36992.10**	**36612**	874.352
ftv170	**4000.40**	**3680**	648.976	4300.70	3769	1040.135
si175	22105.00	21822	528.740	**21473.10**	**21446**	66.580
brg180	**4742.00**	**4660**	188.573	4753.00	4670	182.236
rbg323	**3113.40**	**3099**	34.733	3113.57	3104	23.952
si535	**55597.00**	**54583**	1955.273	57613.00	55887	3120.167
d1655	1477178.00	1433143	67585425	**738309.80**	**729888**	17388.597
u2152	2320791.4	2284392	50700000	**1368206.00**	**1357223**	17841.804

A análise desta tabela evidencia a eficácia da utilização das técnicas de Agrupamento e Busca Local no Algoritmo Genético, principalmente se levando em consideração o desempenho médio de cada algoritmo. Nota-se portanto uma sensível melhora no desempenho global, considerando-se o conjunto completo de instâncias quando da introdução de MD na técnica tradicional. Foram realizados experimentos utilizando o AG com Busca Local, no qual não obtivemos nenhum resultado satisfatório.

Uma vez que a estratégia *K-means* e Busca Local trouxe uma contribuição positiva sobre o AG, a tabela 6.3 traz uma análise comparativa da primeira, confrontada com o AG utilizando o algoritmo *Ward*, que é uma técnica de Agrupamento Hierárquico aglomerativo que utiliza a soma dos quadrados entre os agrupamentos como forma de minimização do erro interno. O intuito de tal análise é investigar os efeitos da introdução de diversos algoritmos. Os significados das colunas desta tabela são os mesmos da tabela anterior, assim o significado dos valores destacados em negrito.

Tabela 6.3. Comparação entre as estratégias AG com *k-means* e Busca Local e AG com *Ward* e Busca Local.

Instância	AG com *k-means* e busca Local			AG com *Ward* e Busca Local		
	Média	Melhor	Desvio Padrão	Média	Melhor	Desvio Padrão
p43	**5727.00**	5727	0.000	5727.4	5727	3.794
ftv44	1624.50	1623	10.700	**1624.1**	1623	10.435
berlin52	14922.70	14914	22.452	**14922.3**	14914	27.966
ftv64	1849.20	1839	14.408	1849.20	1839	15.671
ft70	39112.90	38926	321.283	**38915.9**	**38719**	388.732
ftv70	1964.80	1959	9.359	**1961.8**	**1950**	19.120
pr76	**275462.50**	275325	769.705	275476.5	275325	913.922
gr120	7933.80	7797	258.773	**7438.0**	**7227**	386.988
kro124p	36992.10	36612	874.352	**36654.8**	**36345**	772.574
ftv170	4300.70	3769	1040.135	**2932.9**	**2856**	157.514
si175	21473.10	21446	66.580	**21438.6**	**21427**	35.866
brg180	4753.00	4670	182.236	**4503.0**	**4500**	14.491
rbg323	**3113.57**	**3104**	23.952	4886.5	4790	176.461
si535	57613.00	55887	3120.167	**48970.5**	**48846**	201.302
d1655	738309.80	729888	17388.597	**737160.30**	**721880**	27787.252
u2152	1368206.00	1357223	17841.804	**1361873.90**	**1346494**	38912.884

No confronto entre os desempenhos médios destas duas estratégias, o algoritmo de Agrupamento *Ward* com Busca Local apresentou melhores resultados para doze instâncias, enquanto o *K-means* com Busca Local se mostrou melhor em três, havendo ainda um empate. Considerando-se os melhores valores obtidos, o *Ward* com Busca Local também alcançou bom desempenho em dez instâncias, contra apenas uma instância do *k-means* e havendo empate em cinco. Quanto ao desvio padrão, os valores encontrados são pequenos, com ligeira vantagem para o *K-means*.

A tabela 6.4 traz, em seguida, a comparação entre as estratégias *K-means* com

Tabela 6.4. Comparação entre as estratégias AS simples e AS com *K-means* e Busca Local.

Instância	AS simples			AS com *K-means* e Busca Local		
	Média	Melhor	Desvio Padrão	Média	Melhor	Desvio Padrão
gr17	**2125.90**	2085	87.160	2323.00	**2323**	0.00
br17	39.00	39	0.000	39.00	39	0.00
ftv33	1437.90	1399	67.800	**1406.60**	**1343**	98.27
ftv35	1636.80	1556	134.601	**1641.00**	**1544**	198.48
swiss42	1410.60	1375	61.598	**321.75**	**1313**	29.62
p43	**5648.10**	**5642**	13.524	5756.40	5743	25.22
ftv44	1898.20	1852	102.535	**1849.30**	**1775**	159.81
ftv47	2145.00	2078	129.065	**2055.40**	**1946**	260.02
ry48p	16384.70	15733	976.882	**15111.50**	**14635**	1103.85
berlin52	16352.90	16007	577.980	**14817.60**	**14608**	392.85
ft53	8734.90	8516	384.063	**7952.10**	**7544**	825.58
ftv55	**1895.40**	1839	128.508	1910.00	**1810**	178.49
brazil58	27255.20	26815	1100.377	**26488.22**	**25690**	1884.71
ftv64	2263.00	2160	172.029	**2198.00**	**1950**	391.21
ft70	48148.90	47182	1627.066	**41001.10**	**40204**	1128.91
ftv70	2437.30	2378	123.264	**2396.10**	**2190**	372.73
pr76	329269.00	325547	6254.661	**290272.10**	**281787**	10797.71
kro124p	46453.10	45266	1900.108	**43232.80**	**40553**	4433.92
ftv170	**3794.30**	**3543**	309.030	4007.50	3641	649.35
si175	28875.20	28568	477.400	**22731.40**	**22404**	696.88
brg180	**2467.00**	**2410**	64.884	4726.00	4660	145.05
rbg323	**1773.50**	**1743**	43.891	3126.70	3111	38.10
si535	70475.60	69793	936.623	**57342.40**	**56380**	1962.36

Busca Local com a Meta-heurística Populacional Otimização por Colônia de Formiga através do algoritmo *Ant System*, no intuito de investigar os efeitos dessa combinação com outras técnicas. O *K-means* obteve melhores resultados médios em dezesseis instâncias, das vinte e três instâncias utilizadas. Enquanto que o AS simples obteve melhor resultado em seis, havendo empate em uma instância. Quanto aos melhores valores encontrados, o *K-means* superou o AG simples em dezoito instâncias, havendo apenas um empate. O desvio padrão mostrou-se equilibrado em ambos os casos.

A tabela 6.5 compara os valores do *Ward* com *K-means* utilizados no algoritmo *Ant System*. Em relação a média o *Ward* superou o *K-Means* em dezesseis instâncias, havendo um empate. E em relação aos melhores valores o *Ward* também superou o *K-Means*, em quinze instâncias.

Tabela 6.5. Comparação entre as estratégias AS com *K-means* e Busca Local e AS com *Ward* e Busca Local.

Instância	AS *Ward* e Busca Local			AS com *K-means* e Busca Local		
	Média	Melhor	Desvio Padrão	Média	Melhor	Desvio Padrão
gr17	**2085.00**	**2085**	0.000	2323.00	2323	0.000
br17	39.00	39	0.000	39.00	39	0.000
ftv33	**1402.60**	1362	107.491	1406.60	**1343**	98.277
ftv35	**1564.20**	**1514**	134.6463	1641.00	1544	198.484
swiss42	1337.90	**1284**	145.316	**1321.75**	1313	29.622
p43	**5745.30**	**5733**	32.651	5756.40	5743	25.226
ftv44	**1782.10**	**1683**	126.106	1849.30	1775	159.818
ftv47	**1952.80**	**1883**	180.237	2055.40	1946	260.023
ry48p	**14984.70**	**14446**	1331.435	15111.50	14635	1103.857
berlin52	15191.10	15051	349.415	**14817.60**	**14608**	392.854
ft53	**7708.60**	**7309**	902.285	7952.10	7544	825.581
ftv55	**1823.90**	**1726**	258.121	1910.00	1810	178.499
brazil58	**26419.10**	**25454**	1998.273	26488.22	25690	1884.713
ftv64	**2182.20**	2115	194.970	2198.00	**1950**	391.210
ft70	**40810.90**	40284	1194.377	41001.10	**40204**	1128.912
ftv70	**2286.70**	**2172**	313.850	2396.10	2190	372.731
pr76	**279907.70**	**276811**	10096.608	290272.10	281787	10797.716
kro124p	**42810.00**	40722	5357.431	43232.80	**40553**	4433.925
ftv170	4033.60	3696	791.712	**4007.50**	**3641**	649.356
si175	**22659.50**	**22079**	1035.532	22731.40	22404	696.880
brg180	4761.00	4680	143.143	**4726.00**	**4660**	145.051
rbg323	**3103.80**	**3097**	22.441	3126.70	3111	38.106
si535	57712.70	**56189**	2883.455	**57342.40**	56380	1962.361

A tabela 6.6, compara o melhor valor obtido pelas técnicas AS e AG convencional e por cada versão híbrida com o algoritmo de Agrupamento *Ward*, pois obtivemos o melhor resultado, com o valor Ótimo da instância utilizada. As colunas apresentam a diferença percentual entre os valores, isto é, o quanto a melhor solução de cada estratégia está superior ao valor Ótimo conhecido, em termos percentuais. Os resultados demonstraram que todas as estratégias híbridas encontraram uma melhor solução um número superior de vezes do que o algoritmo convencional, o que evidencia que em termos globais a hibridação trouxe melhorias às técnicas. O AG Híbrido encontrou

os melhores valores (representados por 0.000) onze vezes entre as vinte seis instâncias. Já o AS Híbrido teve apenas dois valores ótimos, mas em relação ao AS simples seu percentual é menor, ou seja, ele chegou mais próximo ao ótimo.

Tabela 6.6. Diferença percentual(%) entre o Melhor valor das estratégias convencionais e AG com *Ward* e Busca Local e AS com *Ward* e Busca Local e o valor Ótimo da instância.

Instância	AG	AG Híbrido	AS	AS Híbrido
gr17	0.000000	0.000000	0.000000	0.000000
br17	0.000000	0.000000	0.000000	0.000000
ftv33	0.000000	0.000000	9.020218	5.909798
ftv35	0.000000	0.000000	9.436524	2.783435
swiss42	0.000000	0.000000	2.513747	0.864101
p43	1.903915	1.903915	2.651246	2.010676
ftv44	4.215747	0.619963	9.671420	4.339740
ftv47	0.000000	0.000000	15.596847	6.024775
ry48p	0.000000	0.000000	8.701983	0.166412
berlin52	98.475206	97.745956	115.141872	99.562450
ft53	0.000000	0.000000	23.287473	5.850833
ftv55	0.000000	0.000000	12.562189	7.338308
brazil58	0.000000	0.000000	5.099429	0.232329
ftv64	5.927134	0.000000	17.672648	15.008157
ft70	4.036408	0.654203	24.533913	4.165697
ftv70	1.641026	0.461538	19.641026	11.384615
pr76	155.084644	154.555793	199.328766	155.929696
gr120	41.990781	12.316335	45.899831	40.90091
kro124p	5.191830	1.054375	23.430858	12.398565
ftv170	33.575318	36.805808	31.651543	34.156080
si175	1.938618	0.182183	32.690242	3.139160
brg180	138.974359	139.487179	157.948718	140.000000
rbg323	133.710407	134.087481	142.006033	133.559578
si535	12.658411	15.349845	44.701754	15.973168
d1655	2206.758627	1074.813289	-	-
u2152	3455.307923	2012.310709	-	-

Embora a hibridização tenha melhorado os métodos originais, essa melhora não foi tão significativa no algoritmo *Ant System*. O algoritmo não foi capaz de gerar uma

população diversificada a ponto de encontrar grupos distintos que poderiam conter boas soluções através da contagem de frequência.

6.5 Considerações Finais

Neste capítulo apresentou-se a proposta de uma estratégia híbrida utilizando Meta-heurísticas Populacionais aplicadas ao Problema do Caixeiro Viajante e Mineração de Dados através das técnicas de Agrupamento. Foram utilizados dois algoritmos de Agrupamento, o *K-means* e o *Ward* e as Meta-heurísticas AG e ACO. A implementação de tal estratégia introduz procedimentos novos para melhorar a performance das Meta-heurísticas, tal como descobrir regiões promissoras e através da contagem das frequências gerar uma solução e a partir dessa solução fazer uma busca local na mesma.

Além da apresentação da estratégia proposta, apresentamos também os resultados experimentais acerca dos benefícios que o método proporcionou para as técnicas tradicionais. Foram apresentados experimentos que validam a utilização de MD em Meta-heurísticas Populacionais, através de agrupamentos de soluções próximas às melhores e à análise de suas frequências. Pode-se verificar padrões que estejam associados a boas soluções, através das frequências das cidades em cada posição. O objetivo foi avaliar o impacto da hibridação na qualidade das soluções obtidas com algoritmos originais, quando aplicados a este problema.

A técnica de hibridização, mostrou melhores resultados no Algoritmo Genético, isso se deve pela capacidade de diversificação que a técnica apresentou, além disso a inicialização dos feromônios no Algoritmo *Ant System* estagnou o método prendendo-o a ótimos locais.

Capítulo 7

Considerações Finais e Trabalhos Futuros

Meta-heurísticas híbridas são cada vez mais estudadas e pesquisas demonstraram que este método tem o potencial de melhorar o desempenho e a robustez das técnicas tradicionais. Muitos trabalhos propõem a combinação de duas ou mais meta-heurísticas ou métodos de otimização exatas e meta-heurísticas Talbi [2002].

Outra abordagem promissora é a utilização de técnicas de Mineração de Dados (MD) para melhorar meta-heurísticas. A MD é um processo de extração de informações úteis a partir de bases de dados, na forma de regras e padrões. A fim de alcançar este objetivo, a MD utiliza técnicas de estatística, aprendizado de máquina, reconhecimento de padrões ou otimização combinatória.

A ideia que motiva a aplicação de MD e Meta-heurísticas Populacionais é a possibilidade de se encontrar padrões, a partir do histórico da mesma, que representem características das melhores soluções geradas, o que pode auxiliar a busca por soluções mais próximas do valor ótimo em menos tempo computacional.

Com base nos resultados experimentais realizados no capítulo anterior ficou evidente que a proposta de incorporar técnicas de MD em Meta-heurísticas Populacionais trouxe melhorias consideráveis em seu desempenho em relação à qualidade das soluções geradas pelas técnicas convencionais, principalmente em instâncias com elevadas dimensões. As técnicas hibridas, em geral, apresentaram melhorias em termos de valores médios obtidos, de melhores valores encontrados e em desvio padrões nas soluções, o que demonstra estabilidade e robustez das propostas.

Isto pode ser justificado pelo fato da MD reduzir significativamente a região de busca em problemas altamente combinatórios. Com isso, podemos melhorar a convergência do algoritmo como também podemos, simultaneamente, permitir a exploração

de ótimos locais de melhor qualidade, aumentando com isso as chances de se atingir um ótimo global em problemas de otimização.

Neste trabalho, o processo de Mineração de Dados é executado uma única vez. Propostas futuras podem alternar iterações dos algoritmos convencionais com o procedimento de Agrupamento mais vezes ao longo do processo. Paralelizações cooperativas poderiam implementar sucessivos processos de MD sobre as soluções geradas por diversos processadores executando iterações híbridas, alimentado-os novamente com padrões obtidos sobre bases de soluções continuamente refinadas.

Em um âmbito mais amplo, diversas outras investigações poderiam ser propostas para hibridização de Meta-heurísticas com técnicas de Mineração de Dados, como introduzir estas mesmas técnicas de MD a ACO e AG aplicados a outros problemas da literatura, ou introduzir outras técnicas de MD ou MD em diferentes Meta-heurísticas.

As ideias e parte dos resultados apresentados nesta dissertação foram publicados nos seguintes meios de divulgação:

- **Trabalho no WoPI 2013 - III *Workshop* de Pesquisa em Informática**.
  - *Algoritmo Genético com Agrupamento e Busca Local para o Problema do Caixeiro Viajante* [Protásio and Nakamura, 2013a].

- **Trabalho no SBPO 2013 - Simpósio Brasileiro de Pesquisa Operacional**.
  - *Meta-heurística Hibrida para o Problema do Caixeiro Viajante* [Protásio and Nakamura, 2013b].

Referências Bibliográficas

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In *Proc. of 20th Intl. Conf. on VLDB*, pages 487–499.

Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2007). *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA.

Bang-Jensen, J., Gutin, G., and Yeo, A. (2004). When the greedy algorithm fails. *Discrete Optimization*, 1(2):121–127.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308.

Boaventura, P. O. (1996). *Grafos - Teoria, Modelos e Algoritmos*. EDGARD BLUCHER, São Paulo, 1nd edition.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Inc., New York, NY, USA, 1nd edition.

Branke, J. and Schmidt, C. (2005). Faster convergence by means of fitness estimation. *Soft Comput.*, 9(1):13–20.

Campello, R. and Maculan, N. (1994). *Algoritmos e heuristicas: desenvolvimento e avaliacao de performance*. EDUFF - UNIVERSIDADE FEDERAL FLUMINENSE, 1nd edition.

Chaves, A. A., de Assis Correa, F., and Lorena, L. A. N. (2007). Clustering search heuristic for the capacitated p-median problem. *Innovations in Hybrid Intelligent Systems*, 44:136–143.

Chaves, A. A. and Lorena, L. A. N. (2008). Hybrid metaheuristic for the prize collecting travelling salesman problem. *8th European Conference (EvoCOP)*, 4972:123–134.

Chaves, A. A. and Nogueira Lorena, L. A. (2011). Hybrid evolutionary algorithm for the capacitated centered clustering problem. *Expert Syst. Appl.*, 38(5):5013–5018.

Clarke, B., Fokoué, E., and Zhang, H. H. (2009). *Principles ad Theory for Data Mining and Machine Learning*. Springer, 1nd edition.

Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393–410.

Darwin, C. (1872). *On the Origin of Species*. JOHN MURRAY, 6nd edition.

de Melo, V. V., Delbem, A. C. B., Junior, D. L. P., and Federson, F. M. (2007). Improving global numerical optimization using a search-space reduction algorithm. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 1195–1202, New York, NY, USA. ACM.

De Melo, V. V., Delbem, A. C. B., Júnior, D. L. P., and Federson, F. M. (2007). Discovering promising regions to help global numerical optimization algorithms. In *Proceedings of the artificial intelligence 6th Mexican international conference on Advances in artificial intelligence*, MICAI'07, pages 72–82, Berlin, Heidelberg. Springer-Verlag.

de Oliveira, A. C. M. and Lorena, L. A. N. (2004). Detecting promising areas by evolutionary clustering search. *17th Brazilian Symposium on Artificial Intelligence (SBIA2004)*, 3171:385–394.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Itália.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART B*, 26(1):29–41.

Dorigo, Marco; Stützle, T. (2004). *Ant Colony Optimization*. A Bradford Book, 1nd edition.

dos Santos, J. P. Q., de Lima, F. C., Magalhães, R. M., de Melo, J. D., and Neto, A. D. D. (2009). A parallel hybrid implementation using genetic algorithm, grasp and reinforcement learning. In *Proceedings of the 2009 International Joint Conference on Neural Networks*, IJCNN'09, pages 2502–2507, Piscataway, NJ, USA. IEEE Press.

dos Santos, J. P. Q., de Lima, F. C., Magalhães, R. M., de Melo, J. D., and Neto, A. D. D. (2010). A parallel hybrid implementation using genetic algorithms, grasp

and reinforcement learning for the salesman traveling problem. In Tenne, Y. and Goh, C.-K., editors, *Computational Intelligence in Expensive Optimization Problems*, volume 2 of *Adaptation Learning and Optimization*, pages 345–369. Springer Berlin Heidelberg.

Fonseca, E. R., Richard Fuchshuber, Santos, L. F. M., Plastino, A., and Martins, S. d. L. (2008). Explorando a metaheurística híbrida md-grasp para o problema de multicast confiável. *Simpósio Brasileiro de Pesquisa Operacional*, pages 1284 – 1295.

Goldbarg, Marco Cesar; Luna, H. P. L. (2005). *Otimização Combinatória e programação linear: modelos e algoritmos*. Elsevier, 2nd edition.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. ADDISON-WESLEY, 1nd edition.

Goos, S., A. S. D. J. and Pasteels, J. (1999). Self-organized shortcuts in the argentine ant. (76):579–581.

Gutin, G. and Punnen, A. P. (2002). *The Traveling Salesman Problem and Its Variations*. Combinatorial Optimization. Springer.

Gutin, G., Yeo, A., and Zverovich, A. (2002). Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the tsp. *Discrete Applied Mathematics*, 117(1-3):81–86.

Han, Jiawei; Kamber, M. (2006). *Data Mining: Concepts and Techniques*. Elsevier, 2nd edition.

Hansen, P. and Mladenović, N. (1997). Variable neighborhood search. *Comput. Oper. Res.*, 24(11):1097–1100.

Hansen, P., Mladenović, N., and Moreno Pérez, J. (2008). Variable neighbourhood search: methods and applications. *4OR*, 6(4):319–360.

Hartigan, J. A. (1975). *Clustering Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 9th edition.

Herrera, F. and Lozano, M. (2001). Adaptive genetic operators based on coevolution with fuzzy behaviors. *Trans. Evol. Comp*, 5(2):149–165.

Hoffman, A. and Wolfe, P. (1985). chapter History, pages 1–16. John Wiley & Sons.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The MIT Press, 2nd edition.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323.

Jin, Y., Olhofer, M., and Sendhoff, B. (2000). On evolutionary optimization with approximate fitness functions. In et al., D. W., editor, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, pages 786–793. Morgan Kaufmann.

Jin, Y., Olhofer, M., and Sendhoff, B. (2002). A framework for evolutionary optimization with approximate fitness functions. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 6:481–494.

Jin, Y. and Sendhoff, B. (2004). Reducing fitness evaluations using clustering techniques and neural networks ensembles. In *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO, volume 1 of LNCS*, pages 688–699. Springer.

Jourdan, L., Dhaenens, C., and Talbi, E.-G. (2006). Using datamining techniques to help metaheuristics: a short survey. In *Proceedings of the Third international conference on Hybrid Metaheuristics*, HM'06, pages 57–69, Berlin, Heidelberg. Springer-Verlag.

Karp, R. M. (1972). Reducibility among combinatorial problems. *Complexity of Computer Computations*, 40(4):85–103.

Kaufman, L. and Rousseeuw, P. J. (2005). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience, 9th edition.

Keogh, E. (2003). A gentle introduction to machine learning and data mining for the database community. Technical report.

Kim, H.-S. and Cho, S.-B. (2001). An efficient genetic algorithm with less fitness evaluation by clustering. *Proceedings of the 2001 Congress on Evolutionary Computation*, 2:887–894.

Linden, R. (2005). *Algoritmos Genéticos: Teoria e Implementação*. Brasport, 1 edition.

Linden, R. (2009). Técnicas de agrupamento. *Revista de Sistemas de Informação da FSMA*, (4):18–36.

Louis, S. (2003a). Genetic learning from experience. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 3, pages 2118–2125.

Louis, S. (2003b). Learning for evolutionary design. In *Evolvable Hardware, 2003. Proceedings. NASA/DoD Conference on*, pages 17–20.

Martins, D. (2012). Making heuristics faster with data mining. *XLIV SBPO (Simpósio Brasileiro de Pesquisa Operacional)*, (4).

Martínez-estudillo, A., Hervás-martínez, C., and Martínez-estudillo, F. (2006). Hybridization of evolutionary algorithms and local search by means of a clustering method. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 36:534–545.

Melo, V. V. (2009). *Técnicas de Aumento de Eficiência para metaheurísticas aplicadas a otimização global contínua e discreta*. PhD thesis, Universidade São Paulo, São Carlos.

Mingoti, S. (2005). *Análise de dados através de métodos de estatística multivariada: uma abordagem aplicada*. UFMG, 1th edition.

Mirkin, B. G. (2005). *Clustering for Data Mining: A Data Recovery Approach*. Taylor and Francis Group, 1nd edition.

Ochi, L. S., Santos, H. G., Marinho, E. H., and de A. Drummond, L. M. (2006). Combining an evolutionary algorithm with data mining to solve a single-vehicle routing problem. *Neurocomputing*, 70(1-3):70–77.

Oliveira, A. C. M. and Lorena, L. A. N. (2007). Hybrid evolutionary algorithms and clustering search. In *Hybrid Evolutionary Algorithms*, volume 75 of *Studies in Computational Intelligence*, pages 77–99. Springer Berlin Heidelberg.

Oliveira, D. G. (2008). Estudo comparativo entre metaheurísticas populacionais tamanho da população variável.

Osman, I. H. and Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operational Research*, 63:513–628.

Papadimitriou, C. and Steiglitz, K. (1982). *Combinatorial Optimization: Algorithm and Complexity*. Prentice Hall.

Plastino, A., Fonseca, E. R., Fuchshuber, R., Freitas, A. A., Santos, L. F. M., Martins, S. d. L., and Salhi, S. (2011). A hybrid data mining metaheuristic for the p-median problem. *Journal Statistical Analysis and Data Mining*, 4(3):313–335.

Protásio, I. A. and Nakamura, F. G. (2013a). Algoritmo genético com agrupamento e busca local para o problema do caixeiro viajante. *WoPI 2013 - III Workshop de Pesquisa em Informática*, 1:4.

Protásio, I. A. and Nakamura, F. G. (2013b). Meta-heurística hibrida para o problema do caixeiro viajante. *SBPO 2013 - Simpósio Brasileiro de Pesquisa Operacional*, 1:1 –12.

Reinelt, G. (1991). Tsplib—a traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384.

Ribeiro, M. H. F., Trindade, V., Plastino, A., and Martins, S. (2006). Hybridization of grasp metaheuristic with data mining techniques. *Journal of Mathematical Modelling and Algorithms*, 5(1):23 – 41.

Santos, L. F., Plastino, A., and Martins, S. (2008). Applications of the dm-grasp heuristic: A survey. *International Transactions in Operational Research*, 15(1):387 – 416.

Sharma, S. (1996). *Applied Multivariate Techniques*. JOHN WILEY and SONS INC., 1th edition.

Szwarcfiter, J. (1986). *Grafos e algoritmos computacionais*. Campus, 2nd edition.

Talbi, E. G. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5):541–564.

Tao, G. and Michalewicz, Z. (1998). Inver-over operator for the tsp. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature - PPSN V*, PPSN V, pages 803–812, London, UK, UK. Springer-Verlag.

Wang, Y., Sun, J., Li, J., and Gao, K. (2012). A modified inver-over operator for the traveling salesman problem. In *Proceedings of the 7th International Conference on Advanced Intelligent Computing Theories and Applications: With Aspects of Artificial Intelligence*, ICIC'11, pages 17–23, Berlin, Heidelberg. Springer-Verlag.

Ward, J. H. (1963). Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301):236–244.

Witten, Ian H.; Frank, E. H. M. A. (2011). *Data Mining Practical Machine Learning Tools and Techniques*. Elsevier, 3nd edition.

Yoo, S.-H. and Cho, S.-B. (2004). Partially evaluated genetic algorithm based on fuzzy c-means algorithm. In Yao, X., Burke, E., Lozano, J., Smith, J., Merelo-Guervós, J., Bullinaria, J., Rowe, J., Tiňo, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 440–449. Springer Berlin Heidelberg.

Yoon, J.-W. and Cho, S.-B. (2011). An efficient genetic algorithm with fuzzy c-means clustering for traveling salesman problem. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 1452–1456. IEEE.

Apêndice A

Testes com os algoritmos do WEKA

WEKA (Waikato Environment for Knowledge Analysis) é um *software* livre que contém uma coleção de algoritmos de Aprendizagem de Máquina para resolução de problemas de Mineração de Dados, desenvolvido por um grupo de pesquisadores da Universidade de Waikato (Nova Zelândia) e foi implementado pela primeira vez em sua forma moderna em 1997. O *software* é escrito na linguagem Java™ dentro das especificações da GPL (*GNU Public License*), a figura da ferramenta *WEKA* é mostrado na figura A.1. O *software* contém uma coleção de ferramentas de visualização e algoritmos para análise de dados e modelagem preditiva, juntamente com interfaces gráficas para facilitar o acesso a estas funcionalidades. Também possui uma *API* geral, de modo que você pode incorporar o *WEKA*, como qualquer outra biblioteca, em suas próprias aplicações que utilizam tarefas de mineração de dados. Para trabalhar de *WEKA* não é necessário conhecimento profundo de mineração de dados que é razão para a ferramenta ser bastante popular.

Figura A.1. Tela inicial da ferramenta *WEKA*.

A escolhas dos algoritmo de *Clusterização* utilizados neste trabalho, basearam-

se nos testes realizados com o *software WEKA* utilizando os Métodos Hierárquicos de Agrupamento. O critério de escolha baseou-se no balanceamento da divisão das instâncias, ou seja, na divisão onde os grupos continham quantidades aproximadas de instâncias. Grupo 0 e Grupo 1 são os rótulos dados pela ferramenta *WEKA*, nessa fase não sabemos qual o grupo está têm as soluções mais próximas ao ótimo.

A tabela A.1 mostra como as instâncias foram dividas utilizando os algoritmos *MakeDensityBasedClusterer* e *Complete Linkage*. O Algoritmo *MakeDensityBasedClusterer* utiliza é baseado no Método de Densidade. Estes métodos consideram grupos como sendo regiões densas de objetos no espaço de dados que são separados por regiões de baixa densidade, que geralmente representam ruídos. O algoritmo *Complete Linkage* ou Vizinho mais Distante é um algoritmo de agrupamento Hierárquico que emprega a distancia de valor máximo, ou seja, após agrupar os dois indivíduos mais semelhantes, de menor distância, verifica a distância máxima deste primeiro grupo para os objetos restantes. Desta forma procura garantir que os objetos de um grupo guardem a máxima distância de outros grupos.

Tabela A.1: Resultados com os Algoritmos *MakeDensityBasedClusterer* e *Complete Linkage*.

Instância	*MakeDensityBasedClusterer*		*Complete Linkage*	
	Grupo 0	Grupo 1	Grupo 0	Grupo 1
gr17	175 (68%)	81 (32%)	191 (75%)	65 (25%)
br17	134 (52%)	122 (48%)	55 (21%)	201 (79%)
ftv33	109 (43%)	147 (57%)	129 (50%)	127 (50%)
ftv35	139 (54%)	117 (46%)	190 (74%)	66 (26%)
swiss42	137 (54%)	119 (46%)	203 (79%)	53 (21%)
p43	164 (64%)	92 (36%)	195 (76%)	61 (24%)
ftv44	126 (49%)	130 (51%)	91 (36%)	165 (64%)
ftv47	161 (63%)	95 (37%)	75 (29%)	181 (71%)
ry48p	187 (73%)	69 (27%)	92 (36%)	164 (64%)
berlin52	115 (45%)	141 (55%)	128 (50%)	128 (50%)
ft53	78 (30%)	178 (70%)	124 (48%)	132 (52%)
ftv55	144 (56%)	112 (44%)	139 (54%)	117 (46%)
brazil58	131 (51%)	125 (49%)	130 (51%)	126 (49%)
ftv64	169 (66%)	87 (34%)	139 (54%)	117 (46%)
ft70	140 (55%)	116 (45%)	134 (52%)	122 (48%)
ftv70	135 (53%)	121 (47%)	111 (43%)	145 (57%)

pr76	87 (34%)	169 (66%)	73 (29%)	183 (71%)
gr120	104 (41%)	152 (59%)	84 (33%)	172 (67%)
kro124p	140 (55%)	116 (45%)	93 (36%)	163 (64%)
ftv170	124 (48%)	132 (52%)	183 (71%)	73 (29%)
si175	163 (64%)	93 (36%)	29 (11%)	227 (89%)
brg180	123 (48%)	133 (52%)	109 (43%)	147 (57%)
rbg323	127 (50%)	129 (50%)	89 (35%)	167 (65%)
si535	106 (41%)	150 (59%)	118 (46%)	138 (54%)
d1655	120 (47%)	136 (53%)	125 (49%)	131 (51%)
u2152	120 (47%)	136 (53%)	136 (53%)	120 (47%)

A tabela A.2 mostra os resultados com o algoritmos *Average Linkage* e *Ward Linkage*. Ambos são algoritmos de agrupamento Hierárquico. No método *Average Linkage* ou Distância Média, a função de distância entre *clusters* é dada pela distância média entre cada par de objetos (um de cada *clusters*). Ele procede, inicialmente, da mesma maneira que os métodos anteriores, ou seja, principia por agrupar os dois objetos mais semelhantes. A seguir utiliza a média aritmética das distâncias dos objetos de cada grupo para confeccionar a nova matriz de distâncias. O Método *Ward Linkage*, diferente dos anteriores, tem como característica, a obtenção da soma dos quadrados, a qual chamaremos de SQ, para todos os possíveis grupos. A reunião definitiva dos objetos irá contemplar os menores valores de SQ. Este método pode ser usado diretamente na matriz de dados iniciais .

Tabela A.2: Resultados com os Algoritmos *Average Linkage* e *Ward Linkage*.

Instância	*Average Linkage*		*Ward Linkage*	
	Grupo 0	Grupo 1	Grupo 0	Grupo 1
gr17	191 (75%)	65 (25%)	74 (29%)	182 (71%)
br17	206 (80%)	50 (20%)	183 (71%)	73 (29%)
ftv33	137 (54%)	119 (46%)	145 (57%)	111 (43%)
ftv35	109 (43%)	147 (57%)	101 (39%)	155 (61%)
swiss42	203 (79%)	53 (21%)	204 (80%)	52 (20%)
p43	150 (59%)	106 (41%)	167 (65%)	89 (35%)
ftv44	254 (99%)	2 (1%)	247 (96%)	9 (4%)
ftv47	96 (38%)	160 (63%)	98 (38%)	158 (62%)

ry48p	248 (97%)	8 (3%)	196 (77%)	60 (23%)
berlin52	250 (98%)	6 (2%)	209 (82%)	47 (18%)
ft53	218 (85%)	38 (15%)	98 (38%)	158 (62%)
ftv55	138 (54%)	118 (46%)	155 (61%)	101 (39%)
brazil58	127 (50%)	129 (50%)	129 (50%)	127 (50%)
ftv64	234 (91%)	22 (9%)	30 (12%)	226 (88%)
ft70	98 (38%)	158 (62%)	91 (36%)	165 (64%)
ftv70	109 (43%)	147 (57%)	103 (40%)	153 (60%)
pr76	240 (94%)	16 (6%)	200 (78%)	56 (22%)
gr120	238 (93%)	18 (7%)	128 (50%)	128 (50%)
kro124p	213 (83%)	43 (17%)	215 (84%)	41 (16%)
ftv170	203 (79%)	53 (21%)	138 (54%)	118 (46%)
si175	238 (93%)	18 (7%)	179 (70%)	77 (30%)
brg180	193 (75%)	63 (25%)	131 (51%)	125 (49%)
rbg323	212 (83%)	44 (17%)	140 (54%)	116 (46%)
si535	250 (98%)	6 (2%)	170 (66%)	86(34%)
d1655	226 (88%)	30 (12%)	156 (60%)	100 (40%)
u2152	242 (95%)	14 (5%)	133 (51%)	123 (49%)

A tabela A.3 mostra os resultados com os algoritmos *Centroid Linkage* e *Expectation-Maximization (EM)* ou Maximização de Esperança. O algoritmo *Centroid Linkage* também é um Método Hierárquico. Nesse método, a distância entre duas classes é dada pela distância entre os respectivos centros de gravidade (ponto central obtido através da média) ou outros pontos representativos das classes . O Algoritmo *EM* é uma extensão do algoritmo *k-means* e é baseado em modelos, esses métodos criam um modelo para cada agrupamento e tentam identificar o melhor modelo para cada objeto. Este método parte da ideia de que os dados são gerados por uma série de probabilidade de distribuições.

Tabela A.3: Resultados com os Algoritmos *Centroid Linkage* e *Expectation-Maximization (EM)*.

Instância	*Centroid Linkage*		*EM*	
	Grupo 0	Grupo 1	Grupo 0	Grupo 1
gr17	243 (95%)	13 (5%)	121 (47%)	135 (53%)
br17	254 (99%)	2 (1%)	106 (41%)	150 (59%)

ftv33	255 (100%)	1 (0%)	126 (49%)	130 (51%)
ftv35	255 (100%)	1 (0%)	118 (46%)	138 (54%)
swiss42	254 (99%)	2 (1%)	35 (14%)	221 (86%)
p43	255 (100%)	1 (0%)	169 (66%)	87 (34%)
ftv44	255 (100%)	1 (0%)	162 (63%)	94 (37%)
ftv47	255 (100%)	1 (0%)	174 (68%)	82 (32%)
ry48p	255 (100%)	1 (0%)	189 (74%)	67 (26%)
berlin52	255 (100%)	1 (0%)	107 (42%)	149 (58%)
ft53	255 (100%)	1 (0%)	166 (65%)	90 (35%)
ftv55	255 (100%)	1 (0%)	189 (74%)	67 (26%)
brazil58	255 (100%)	1 (0%)	127 (50%)	129 (50%)
ftv64	255 (100%)	1 (0%)	31 (12%)	225 (88%)
ft70	255 (100%)	1 (0%)	122 (48%)	134 (52%)
ftv70	255 (100%)	1 (0%)	120 (47%)	136 (53%)
pr76	243 (95%)	13 (5%)	116 (45%)	140 (55%)
gr120	255 (100%)	1 (0%)	144 (56%)	112 (44%)
kro124p	255 (100%)	1 (0%)	184 (72%)	72 (28%)
ftv170	255 (100%)	1 (0%)	130 (51%)	126 (49%)
si175	255 (100%)	1 (0%)	125 (49%)	131 (51%)
brg180	255 (100%)	1 (0%)	105 (41%)	151 (59%)
rbg323	255 (100%)	1 (0%)	165 (64%)	91 (36%)
si535	255 (100%)	1 (0%)	154 (60%)	102 (40%)
d1655	255 (100%)	1 (0%)	120 (47%)	136 (53%)
u2152	255 (100%)	1 (0%)	149 (58%)	107 (42%)

Os resultados com os algoritmo *Single Linkage*, *Mean Linkage*, *ADJComplet Linkage* e *Neighbor_joining Linkage*, foram tão insatisfatórios quanto o algoritmo *Centroide Linkage* considerando o critério de seleção utilizado. Entre os Métodos Hierárquicos, o que teve o melhor desempenho, ou seja, o que dividiu equilibradamente as instâncias utilizadas, foram os algoritmos *Ward Linkage* e *Average Linkage*. Optou-se por escolher o Método *Ward Linkage* por seus resultados não destoar tanto em cada instância. Como queremos reduzir o espaço de busca e não sabemos qual grupo possui as melhores soluções, o ideal, então, é equilibrar a quantidade de soluções em cada grupo.