Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e Computação
Departamento de Controle e Automação

Adaptação de Parâmetros em Meta-heurísticas com Sistemas Nebulosos Genéticos

Autor: Vitor Hugo Almeida Marques
Orientador: Fernando Antônio Gomide

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: **Engenharia de Computação**.

Banca Examinadora:
Prof. Dr. Fernando Antônio Gomide (presidente) - DCA/FEEC/UNICAMP
Prof. Dra. Myriam Regattieri Delgado - CPGEI/UTFPR
Prof. Dr. Vinícius Amaral Armentano - DENSIS/FEEC/UNICAMP

Campinas, SP
2011

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

M348a

Marques, Vitor Hugo Almeida
Adaptação de parâmetros em meta-heurísticas com sistemas nebulosos genéticos / Vitor Hugo Almeida Marques. --Campinas, SP: [s.n.], 2011.

Orientador: Fernando Gomide.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Estimativa de parâmetros. 2. Metaheurística. I. Gomide, Fernando. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Parameter adaptation of metaheuristic with genetic fuzzy systems
Palavras-chave em Inglês: Estimated parameters, Metaheuristics
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Myriam Regattieri De Biase da Silva Delgado, Vinícius Amaral Armentano
Data da defesa: 29-04-2011
Programa de Pós Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: Vitor Hugo Almeida Marques

Data da Defesa: 29 de abril de 2011

Título da Tese: "Adaptação de Parâmetros em Meta-heurísticas com Sistemas Nebulosos Genéticos"

Prof. Dr. Fernando Antônio Campos Gomide (Presidente): _____

Profa. Dra. Myriam Regattieri de Biase da Silva Delgado: _____

Prof. Dr. Vinícius Amaral Armentano: _____

Resumo

Esta dissertação introduz um sistema nebuloso genético (SNG) para adaptação de parâmetros em meta-heurísticas. Duas meta-heurísticas entre as mais usadas foram consideradas como exemplos, algoritmo genético e busca tabu. Os parâmetros trabalhados na busca tabu são relacionados às memórias, de curto prazo e de longo prazo. Já os parâmetros do algoritmo genético a sofrer adaptação são as taxas de reprodução e mutação. Sistemas baseados em regras nebulosas oferecem um mecanismo natural para descrever comportamentos globais como combinação de regras de controle. Eles também herdam um meio de gradualmente alternar entre regras que conjuntamente definem uma estratégia de controle. Dessa forma, esses sistemas são candidatos naturais para construir estratégias de controle de parâmetro porque eles proveem um maneira de desenvolver mecanismos baseados na natureza específica de uma região de busca e as transições entre suas fronteiras. Uma aplicação usando o problema clássico de roteamento de veículos com janela de tempo foi incluído para avaliar o desempenho do sistema nebuloso genético. Resultados experimentais mostram que meta-heurísticas com o mecanismo de adaptação com SNG melhoram o comportamento da busca e a qualidade das soluções quando comparado à versões padrões (sem SNG) e com parâmetros constantes dos algoritmos genético e busca tabu. Eles também geram boas soluções sub-ótimas mais rápidas que métodos exatos desenvolvidos para o problema e que são reportados na literatura.

Palavras-chave: Sistemas Nebulosos Genéticos, Meta-heurística, Busca Tabu, Algoritmo Genético, Definição de Parâmetros, Controle de Parâmetros, Adaptação de Parâmetros.

Abstract

This dissertation introduces a genetic fuzzy system for parameter adaptation of metaheuristics. Two metaheuristics, among the most used ones, have been considered as examples, genetic algorithm and tabu search. The considered parameters of the tabu search are related to the short and long term memories. Parameters of the genetic algorithm under adaptation are the mutation and reproduction rates. Fuzzy rule-based models offer a natural mechanism to describe global behavior as a combination of control rules. They also inherit a means to gradually shift between control rules which jointly defines a control strategy. They are a natural candidate to construct parameter control strategies because they provide a way to develop decision mechanisms based on the specific nature of search regions and transitions between their boundaries. An application using the classic vehicle routing problem with time windows is included to evaluate the genetic fuzzy system performance. Experimental results show that metaheuristics with GFS improve search behavior and solution quality when compared against standard, constant parameters genetic and tabu search approaches. It also provides reasonably good suboptimal solutions faster than specially tailored exact methods reported in the literature.

Keywords: Genetic Fuzzy Systems, Metaheuristics, Tabu Search, Genetic Algorithm, Parameter Setting, Parameter Control, Parameter Adaptation.

Agradecimentos

Ao meu orientador Prof. Fernando Gomide, sou grato pela orientação e dedicação.

Aos amigos André Luis Ogando Paraense e Vinícius Santino Alves, agradeço o apoio e sugestões.

Aos meus pais e irmãos pelo apoio incondicional e à Fernanda por fazer esta jornada muito mais fácil, dedico este trabalho com muito amor

Sumário

Lista de Figuras

Lista de Tabelas

Trabalhos Publicados Pelo Autor

1. Marques, Vitor; Gomide, Fernando. "Fuzzy Coordination of Genetic Algorithms for Vehicle Routing Problems with Time Windows". In: *Proceedings of Fourth International Workshop on Genetic and Evolutionary Fuzzy systems (GEFS)*, pp. 39-44, Mieres, Espanha, Março 2010.

2. Marques, Vitor; Gomide, Fernando. "Memory Control of Tabu Search with Genetic Fuzzy Systems". In: *Proceedings of Sixth IEEE World Congress on Computational Intelligence* (WCCI 2010), pp. 2251-2257, Barcelona, Espanha, Julho 2010.

3. Marques, Vitor; Gomide, Fernando. "Adaptive Meta-heuristics with Genetic Fuzzy Systems". *24th European Conference on Operational Research)*, Lisboa, Portugal, Julho 2010.

4. Marques, Vitor; Gomide, Fernando. "Parameter Control of Metaheuristics with Genetic Fuzzy Systems". *Evolutionary Intelligence Journal*. Submetido Março 2011, aceito Junho 2011.

Capítulo 1

Introdução

1.1 Contextualização

Um problema de otimização tem o objetivo de achar um mínimo ou máximo de uma função $f(x, y)$ onde x é uma instância do problema ($x \in I$, para I um conjunto de instâncias) e y é uma solução factível, que respeita todas as restrições do problema, ($y \in S$, sendo S o espaço das soluções factíveis). A minimização ou maximização de $f(x, y)$, define a chamada função objetivo do problema, $m(x, y)$. Portanto resolver um problema de otimização pode ser resumido como uma busca para achar um elemento $y^{'}$ do espaço de busca S que minimize ou maximize $f(x, y)$, ou seja, $f(x, y) = m\{f(x, y^{'})|y^{'} \in S\}$.

Uma das formas mais usadas de resolver um problema de otimização é criar heurísticas de busca eficientes e eficazes, tendo como passos importantes em seu desenvolvimento a especificação de uma representação e uma função de avaliação, que ligam o problema em si ao mecanismo para sua resolução. Mais ainda, quando problemas de otimização são resolvidos é necessário definir operadores de movimento consistentes com a representação, condições iniciais e critério de parada. Cada um dos componentes podem apresentar parâmetros, cujos valores ao longo da busca determinam se a solução alcançada será próxima do ótimo e se será obtida de forma eficiente. Isto ocorre, pois um problema de otimização apresenta áreas do espaço de busca com diferentes naturezas: *mínimo local ou pontos fixos*, *ciclos limitados ou órbitas fechadas* e *pontos de atração caótica* (Battiti & Tecchiolli, 2004). *Mínimos locais* são pontos de atração da dinâmica do sistema para a estratégia do gradiente. São pontos fixos até que alguma estratégia seja introduzida que force a superar os mínimos locais e continuar a busca. *Ciclos limitados* (ou *órbitas fechadas*) são uma segunda possibilidade, onde a trajetória repete infindavelmente uma sequência de estados. A terceira possibilidade é o caso onde não ocorre nenhum dos dois casos anteriores, mas a trajetória da busca fica confinada em uma limitada parte do espaço de busca. Na teoria de sistemas dinâmicos este fenômeno é descrito introduzindo o conceito

de *pontos de atração caótica*. A natureza do espaço de busca ainda envolve a suavidade da superfície das regiões e também de suas fronteiras. Idealmente, algoritmos de busca deveriam ser capazes de atravessar o espaço de busca mantendo a eficiência e efetividade das estratégias independentemente da natureza das regiões e fronteiras. Uma maneira de abordar esta questão desta forma é dotar os algoritmos de busca com estratégias de controle de parâmetro que sentem como a busca está progredindo e modificam os valores dos parâmetros de maneira adequada para reforçar diversificação ou intensificação dentro da busca.

Pensando como no dia-a-dia uma pessoa pode buscar algo, é possível criar uma analogia para esta busca na qual é possível visualizar alguns aspectos referidos. Imagine que alguém vá procurar um relógio em um armário de 3 portas, sendo duas delas somente gavetas com diversos objetos de dimensões variadas de vestuário e que a outra porta tenha somente prateleiras com livros e folhas. Quantas formas diferentes a busca pode ser realizada? Com certeza várias, para não dizer infinitas. Agora, imagine que a pessoa comece a busca em uma das portas de gavetas, que possua somente panos (camisas) e que para esta busca use uma estratégia na qual o sentido mais usado é o tato, afinal um pano e um relógio são bem diferenciáveis com o tato, permitindo inclusive um tatear rápido. A pessoa percorre as gavetas tateando rapidamente e nada é encontrado. Segue-se a busca e na sequência chega-se à porta das prateleiras. Será que a estratégia deverá ser a mesma? E a velocidade de procura será a mesma? Com certeza o tato continua ser elemento interessante na estratégia, mas no meio de livros e folhas será que tatear rapidamente tem o mesmo efeito?

O que esta analogia ilustra é que dependendo de vários fatores da configuração do espaço de busca (armário), tamanho, forma e outras características que definem sua natureza, a estratégia usada (tatear rapidamente), pode ser adequada para uma parte do espaço de busca (porta de gavetas com panos), mas, não a mais adequada para outras regiões do espaço de busca (porta de prateleiras com livros e folhas). Como podem as mudanças no espaço de busca ser percebidas? A região atual é ou não um bom lugar para procurar? Uma vez percebida uma mudança ou realizando uma mudança, qual a melhor estratégia para o novo cenário de busca? As estratégias para um relógio podem ser usadas para um outro objeto (outro problema)? Para a busca no armário, o sentido visão é um grande elemento de percepção de mudança e de decisão, a memória nem se fala, mas, mesmo neste simples cenário várias respostas são de difícil definição. E para uma meta-heurística? Começar a responder estas perguntas no âmbito apresentado constitui o grande desafio com o qual este trabalho lida.

1.2 Motivação e Relevância

Em meta-heurísticas a forma de prover mudanças na busca é com a *escolha de parâmetros*. Parâmetros estes ditos estratégicos (Eiben et al., 2007), pois são capazes de alterar a estratégia da busca.

A área de escolha de parâmetros em meta-heurística vem crescendo bastante, com um número cada vez maior de publicações. O fato da maioria dos problemas clássicos já terem sido resolvidos com as meta-heurísticas existentes direciona a comunidade científica na área a trabalhar para que os modelos e técnicas existentes gerem melhores resultados.

Teoria nebulosa e sistemas nebulosos (Zadeh, 1965) proveem um mecanismo eficiente para modelar sistemas com comportamentos não-lineares e estratégias de tomadas de decisão. Um dos elementos chave é que eles definem uma maneira intuitiva e concisa para especificar estratégias de controle e interpolar suavemente funções reais. Em particular, modelos com base de regras nebulosas oferecem um mecanismo natural para descrever comportamentos globais como combinação de regras de controle local (Pedrycz & Gomide, 2007). Eles também herdam um meio de gradualmente alternar entre regras de controle que conjuntamente definem uma estratégia. Por isto, sistemas com base de regras nebulosas são candidatos naturais para construir estratégias de controle de parâmetros ao possibilitar o desenvolvimento de estratégias de decisão baseadas nas especificidades da natureza de um espaço de busca e nas transições entre fronteiras da busca. Quando a um sistema nebuloso adiciona-se um processo de aprendizagem baseado em um algoritmo genético, ele é chamado de sistema nebuloso genético. Algoritmos genéticos proveem uma robusta capacidade de busca para lidar com problemas em espaços de busca mais complexos, portanto, constituem uma abordagem atraente para problemas que demandam efetividade e eficácia na resolução (Cordón et al., 2001).

Buscar nesta duas grandes linhas de pesquisa, escolha de parâmetros em meta-heurísticas e sistemas nebulosos genéticos, a conexão entre evidências da busca (tudo aquilo que informa sobre o estado da busca) e mudanças de direção estratégica da busca (mudanças de parâmetros estratégicos que atuam na diversificação e intensificação melhorando a busca), é o propósito central deste trabalho.

Ao longo do desenvolvimento do trabalho foram realizadas algumas escolhas importantes, o estabelecimento de com quais meta-heurísticas trabalhar e em qual problema aplicar a proposta.

As meta-heurísticas algoritmo genético e busca tabu foram adotadas neste trabalho por serem as duas meta-heurísticas mais usadas para resolver problemas de otimização e por apresentarem características de busca diferentes e complementares.

Com relação a onde aplicar o modelo, foi escolhido o problema de roteamento de veículos com janelas de tempo. O que motiva esta escolha é o fato de ser um problema desafiador que foi e continua sendo muito estudado, portanto, possui bons *benchmarks*. Ainda, sua relevância é indiscutível em diversos setores: transporte, logística e serviços. A demanda crescente nestes setores leva a um constante enriquecimento deste problema e o mantém como um dos problemas mais relevantes para estudo em otimização.

1.3 Objetivos

Os objetivos deste trabalho podem ser assim destacados:

- realizar uma pesquisa em escolha de parâmetros e sistemas nebulosos genéticos construindo um modelo que permita sistemas nebulosos serem usados para conectar *estado da busca* às mudanças estratégicas da busca via *escolha de parâmetros*.
- realizar a escolha de parâmetros em um modelo adaptativo, ou seja, com aprendizado.
- produzir resultados compatíveis com a literatura para o problema de roteamento de veículos com janela de tempo.

Estes objetivos traduzem a forma como buscaremos algumas respostas para as diversas decisões da meta-heurística ao longo da busca. Alguns benefícios são esperados com o cumprimento dos objetivos: ganho de qualidade das meta-heurísticas com custos computacionais possíveis de serem pagos, maior autonomia e menor tempo gasto com escolhas manuais de parâmetros.

1.4 Organização do texto

Além deste capítulo introdutório, a dissertação apresenta cinco capítulos organizados da seguinte forma:

- Capítulo 2 - Escolha de Parâmetros em Meta-heurísticas. Este capítulo introduz a área de meta-heurística caracterizando um pouco da evolução, apresenta o contexto de uso de meta-heurísticas e as decisões envolvidas em seu uso, como a escolha da meta-heurística para determinado problema, decisões de projeto e a decisão da escolha de parâmetros que é o foco do capítulo. Ainda, as meta-heurísticas busca tabu e algoritmo genético são introduzidas, pela relevância neste trabalho.
- Capítulo 3 - Problema de Roteamento de Veículo com Janela de Tempo. Este capítulo realiza uma abrangente, porém resumida, introdução e revisão do problema de roteamento de veículos, destacando: origem e extensões, representação e formulação matemática do problema de roteamento de veículo com janela de tempo tratado aqui e por fim passeia de forma objetiva nas diversas técnicas usadas para resolver o problema, indo de métodos exatos às meta-heurísticas híbridas.

- Capítulo 4 - Sistemas Nebulosos Genéticos para escolha de parâmetros em Meta-heurísticas. Este capítulo começa introduzindo Sistemas Nebulosos Genéticos na perspectiva de controle com uma breve revisão da literatura. Apresenta toda a proposta deste trabalho em seus pormenores: toda arquitetura do sistema nebuloso genético e suas partes, os modelos de evolução dentro da arquitetura e como toda base de conhecimento trabalhada no sistema é usada nas meta-heurísticas para realizar a adaptação de parâmetros.

- Capítulo 5 - Resultados experimentais. Este capítulo reporta os resultados alcançados apresentando primeiro metodologia e a definição inicial dos algoritmos.

- Capítulo 6 - Conclusão e Trabalhos Futuros. Este capítulo conclui o texto e apresenta evoluções e itens que podem ser desenvolvidos na continuação da pesquisa.

Capítulo 2

Escolha de Parâmetros em Meta-heurísticas

2.1 Introdução

A palavra meta-heurística é uma combinação de palavras de raízes gregas ($\mu\epsilon\tau\grave{\alpha}$, "meta"), que significa *além* e ($E\rho\acute{\upsilon}\sigma\acute{\iota}\kappa\omega$, "heurística"), que significa *encontrar*. O termo foi primeiramente usado por Glover (1986), com intuito de descrever uma estratégia que trabalha em um nível superior de uma heurística, realizando seu controle.

As meta-heurísticas são geralmente empregadas em problemas onde algoritmos específicos e eficientes não estão disponíveis e algoritmos exatos não conseguem obter respostas em tempo satisfatório para problema de dimensões maiores, ou seja, normalmente quanto mais desconhecido e maior é o problema. Dentre os problemas com estas características podem ser destacados os problemas de otimização combinatória, que têm por objetivo achar um objeto matemático discreto (vetor de bits, uma permutação) dentro de um espaço de busca discreto (espaço de estados que caracteriza todas possíveis soluções) que maximize ou minimize determinada função objetivo.

Este capítulo aborda temas relacionados a meta-heurísticas explicando aspectos importantes para um conhecimento da área: uma visão breve da área e sua evolução, o porquê de usarmos meta-heurísticas, dificuldades de escolha de uma meta-heurística para um problema e uma vez decidido a meta-heurística, as dificuldades de construção. As meta-heurísticas usadas neste trabalho, algoritmo genético e busca tabu, são mais detalhadas e por fim, o capítulo foca na escolha de parâmetros, um tema primário neste capítulo e no trabalho como um todo. Todos estes aspectos retratam como a área de meta-heurísticas vem evoluindo, mostrando sua relevância para resolver problemas práticos.

2.2 Meta-heurísticas

A área de meta-heurística tem uma história que vem desde a década de 50, quando vários métodos surgiram antes mesmo do cunho da expressão. Na linha do tempo abaixo estão relacionados os principais eventos e surgimentos na área e alguns marcos importantes no período de 1951 à 2009. Pode-se perceber mais trabalhos realizados na área a partir dos anos 80, principalmente com as meta-heurísticas evolutivas.

1951 : primeiros trabalhos em métodos de otimização estocástica (Robbins & Monro, 1951).

1954 : Barricelli apresenta as primeiras simulações do processo evolutivo e as emprega em problemas de otimização.

1965 : Rechenberg concebe o primeiro algoritmo com estratégias evolutivas.

1966 : Fogel et al. propõem a programação evolutiva.

1970 : Hastings concebe o algoritmo Metropolis-Hastings, que pode representar qualquer função de densidade de probabilidade.

1975 : Holland propõe os primeiros algoritmos genéticos.

1980 : Smith descreve a programação genética.

1983 : baseado no trabalho de Hastings. Kirkpatrick et al. criam a têmpera simulada.

1985 : independentemente, Černý propõe o mesmo algoritmo.

1986 : primeira menção ao termo “meta-heurística” por Glover, na criação da busca tabu.

1986 : Farmer et al. trabalham em sistemas imunes artificiais.

1987 : trabalho no comportamento coletivo das formigas acha uma aplicação em inteligência artificial (Moyson & Manderick, 1987).

1988 : a primeira conferência em algoritmos genéticos é organizada na Universidade de Illinois em Urbana-Champaign.

1988 : John Koza registra sua primeira patente em programação genética.

1989 : Goldberg publica um dos mais conhecidos livros sobre algoritmos genéticos.

1989 : Evolver, o primeiro software de otimização usando algoritmos genéticos é lançado pela empresa Axcelis.

1989 : o termo "algoritmo memético" é usado pela primeira vez por Moscato.

1992 : o algoritmo de colônia de formigas é proposto pela primeira vez por Dorigo, em sua tese de doutorado.

1993 : O periódico "Evolutionary Computation" começa a ser publicado pelo MIT.

1995 : Feo & Resende propõem o procedimento de busca gulosa adaptativa e aleatória.

1995 : Eberhart & Kennedy concebem otimização por enxame de partículas.

1996 : Mühlenbein & Paaß trabalham na estimação de algoritmos distribuídos.

1997 : Storn & Price propõem um algoritmo evolutivo diferencial.

1997 : Rubinstein trabalha no método da entropia cruzada.

1999 : Boettcher & Percus propõem otimização extrema.

2000 : primeiro algoritmo genético iterativo (Takagi, 2000).

2001 : Geem et al. propõem busca harmônica.

2004 : Nakrani & Tovey descrevem algoritmo de abelhas.

2005 : Krishnanand & Ghose propõem otimização por enxame de insetos luminescentes para captura simultânea de ótimos de funções multi-modais.

2005 : Karaboga propõe o algoritmo das colônias de abelhas artificiais.

2006 : Pham et al. propõem um novo algoritmo chamado de algoritmo das abelhas.

2006 : Eusuff et al. propõem uma meta-heurística memética denominada de algoritmo do pulo aleatório do sapo.

2007 : Mucherino & Seref propõem a busca de macaco.

2009 : Yang descreve o algoritmo do vaga-lume.

Um trabalho que ajuda a entender a área de meta-heurísticas é o trabalho de Blum & Roli (2003), onde meta-heurísticas conhecidas na literatura até 2003 são apresentadas de forma objetiva cobrindo características específicas de cada uma e também suas similaridades. O texto apresenta várias formas de classificar as meta-heurísticas. As taxonomias variam bastante nas características destacadas: *meta-heurísticas bio-inspiradas* vs. *não bio-inspiradas*, *baseada em população* vs. *baseado em trajetória de um ponto*, *com uso de memória* vs. *sem uso de memória*. Outro aspecto muito importante abordado neste trabalho é o posicionamento de meta-heurísticas quanto às suas características de intensificação (explorar áreas com boas soluções e com perspectiva de melhoria do incumbente, melhor solução encontrada até o momento atual da busca) e diversificação da busca (ir para novas partes dos espaço de busca, quando um ótimo local já foi encontrado). Entender estas classificações e conceitos ajuda muito nas decisões de escolha e construção de meta-heurísticas, temas abordados na sequência.

2.2.1 O Uso de Meta-heurísticas

Uma distinção importante entre teoria da complexidade computacional e análise de algoritmo é que esta se preocupa com a análise dos recursos necessários para que determinado algoritmo resolva um problema, já aquela tenta responder questões mais genéricas a respeitos de todos possíveis algoritmos para resolver um problema, em outras palavras, busca classificar problemas quanto a possibilidade de resolução com certas restrições de recursos.

Um conjunto de problemas pertence à classe *P* se existe algum algoritmo que encontra a sua solução ótima em tempo polinomial. Já os englobados pela classe *NP* são problemas computáveis cujas soluções, até o momento, somente são obtidas em um tempo exponencial, ou seja, ainda não são conhecidos algoritmos de complexidade polinomial capazes de resolvê-lo.

Um problema é *NP-difícil* se todos problemas pertencentes as classe *NP* são redutíveis a ele em tempo polinomial, se além disto o problema pertencer a classe *NP* ele é dito *NP-completo*. Assim, um problema *NP-difícil* é ao menos tão difícil de resolver quanto os problemas *NP-completos*. Para mais informações sobre teoria de complexidade computacional, vide, por exemplo, Garey & Johnson (1979).

Os problemas resolvidos com meta-heurísticas são geralmente problemas NP-completos. Classe que inclui problemas como *Caixeiro Viajante*, *Empacotamento*, *Roteamento de Veículos*, que será abordado em detalhes nesta dissertação, só para citar alguns problemas clássicos. A ideia de uso de meta-heurísticas para problemas onde não são conhecidos algoritmos que os resolvam até a otimalidade em tempo polinomial é conseguir obter boas soluções em tempo polinomial, mesmo que não ótima, sem um conhecimento muito especializado em determinado problema; logicamente, que quanto maior este conhecimento mais elaborada poderá ser a meta-heurística e melhores resultados poderão ser atingidos.

2.2.2 As Decisões no Uso de Meta-heurísticas

A primeira decisão no processo de construção de uma meta-heurística para resolver um problema é escolher a meta-heurística a ser empregada. Este primeiro desafio não é trivial, pois a natureza do problema influencia na capacidade de uma meta-heurística de conseguir boas soluções. Normalmente, o que se faz é selecionar um conjunto de meta-heurísticas potenciais, devido a estudos e casos anteriores e testar todas. Para problemas mais conhecidos quanto a natureza e características de construção, meta-heurísticas que têm maior chance de sucesso são sabidas. Para problemas com definição de melhores rotas, por exemplo, como problemas de roteamento de veículos, é sabido que *buscas tabus* (Gendreau et al., 1994; Cordeau & Laporte, 2001) apresentam bons resultados, devido ao casamento *natureza do problema-características da técnica*. Neste problema especificamente, boas soluções estão normalmente mais concentradas no espaço de busca, permitindo melhores resultados desta meta-heurística. Já problemas com muitos ótimos locais, em um espaço de busca com menor concentração de boas soluções, algoritmos evolutivos como algoritmos genéticos (AG), por exemplo, costumam se sair melhor.

Hoje, outra abordagem comum é a hibridização de meta-heurísticas, buscando-se o melhor de cada técnica. De maneira geral, uma boa forma de pensar na escolha de meta-heurísticas é verificar as características comuns e específicas de cada uma e ver a adequação à alguma característica conhecida do problema, mas, é muito difícil saber qualquer resultado a *priori*.

Uma vez escolhida a meta-heurística várias decisões permeiam a construção. As mais difíceis para o usuário da técnica estão relacionadas às decisões sobre estruturas de vizinhanças e valores de parâmetros. Quais estruturas usar? Como usar? Em que momento usar? Quais parâmetros usar? Vou fixar valores destes parâmetros? Quais serão? Várias destas perguntas têm que ser respondidas como decisão de projeto e construção das meta-heurísticas. Estas dúvidas e a forma como são tratadas são fontes das principais críticas aos pesquisadores de meta-heurísticas. Críticas que apontam meta-heurísticas como “caixas pretas especializadas”.

Um dos pontos que é criticado, diz respeito ao fato de ser sempre possível definir uma instância, com uma função objetivo que forçará determinada meta-heurística a enumerar todo o espaço de busca. A teoria do *Não há almoço gratuito* (“No Free Lunch”), Wolpert & Macready (1997), mostra que algoritmos de otimização, na média de todas instâncias de problemas definidos matematicamente, apresentam desempenho similar, ou seja, se para um conjunto de problemas e instâncias uma meta-heurística apresenta bons resultados, certamente terá resultados ruins para outro conjunto de problemas e instâncias. Outra crítica reincidente ressalta a dificuldade das meta-heurísticas na resolução de problemas não-lineares e contínuos.

Estas críticas fortalecem o estudo na área com o surgimento de propostas de meta-heurísticas mais abrangentes e adaptativas, que conseguem solucionar um maior número de problemas e instâncias.

2.2.3 Meta-heurísticas Usadas Neste Trabalho

Neste trabalho são usadas as meta-heurísticas busca tabu (BT) e algoritmo genético (AG). A escolha foi pautada no amplo uso destas meta-heurísticas. Além disto, elas apresentam características distintas como um maior poder de diversificação da busca, presente em algoritmos baseados em população, como o algoritmo genético e um maior poder de intensificação da busca, presente em algoritmos baseados em uma única trajetória de busca, como a busca tabu. No capítulo 3, outras meta-heurísticas são apresentadas no contexto de resolução do problema de roteamento de veículos com janelas de tempo.

Busca Tabu

Busca tabu foi caracterizada por Glover como uma meta-heurística, uma estratégia geral para guiar e controlar heurísticas moldadas para o problema sendo resolvido. Ela permite a métodos de buscas locais superar ótimos locais e apresenta três aspectos básicos para cumprir este objetivo: sua memória (curto e longo prazo), suas estruturas de vizinhança e seu critério de aspiração (Glover, 1989), Glover (1990).

Memória de curto prazo- É implementada por listas tabu. As listas guardam movimentos internos a busca local. Estes movimentos retiram atributos da solução corrente e inserem novos atributos (arestas em um problema representado por grafo, por exemplo). O "tempo" (geralmente rodadas do algoritmo) que estes atributos permanecem fora e dentro da solução, denominado duração tabu (ou prazo tabu) e o tamanho das listas tabu são variáveis potenciais para controle de memória.

Memória de longo prazo- A memória de longo prazo é geralmente associada a frequência com que estruturas de solução aparecem na trajetória de busca. O objetivo desta memória é explorar regiões do espaço de busca que foram menos visitadas e têm bom potencial. Por exemplo, em uma representação em grafo, a frequência com que uma aresta aparece na solução serve como uma medida de memória de longo prazo.

Espaço de busca- É o espaço de todas possíveis soluções que podem ser consideradas (visitadas) durante a busca.

Estruturas de Vizinhança- Estão ligadas à definição de espaço de busca. A cada iteração da BT, as transformações que podem ser aplicadas à solução corrente, denominada *s*, definem um conjunto de soluções vizinhas no espaço de busca, denominado *N(s)* (a vizinhança de *s*) e pode ser formalmente ser definida como subconjunto do espaço de busca:

N(s) = {soluções obtidas de s por uma simples transformação local}

Critério de aspiração- Tabus podem, às vezes, proibir movimentos atraentes, mesmo não havendo perigo de revisita de soluções. Isto pode levar à estagnação do processo de busca. O que torna

necessário um critério para cancelar tabus, chamado de critério de aspiração. O critério mais simples e um dos mais usados critérios de aspiração consiste em permitir um movimento, mesmo sendo tabu, desde que ele resulte em uma solução com valor da função objetivo melhor que a melhor solução obtida até o momento e que a nova solução seja inédita.

O algoritmo básico da busca tabu, algoritmo 1, descrito abaixo, explica como uma solução evolui ao longo da trajetória de busca. Basicamente, em cada passo, é escolhida a melhor solução ao longo da vizinhança da solução corrente, desde que não seja tabu ou atenda o critério de aspiração. O algoritmo segue até o que o critério de término seja atingido.

Algoritmo 1 - Busca tabu

1: s ← geraSolucaoInicial()
2: inicializaListasTabu(LT_1 , . . . , LT_r)
3: k ← 0
4: **enquanto** condição de término não atendida **faça**
5: ConjuntoDosPermitidos(s, k) ← {s' ∈ $N(s)$| s' não viola condição tabu ou satisfaz um critério de aspiração}
6: s ← escolheMelhorDe(ConjuntoDosPermitidos(s, k))
7: atualizaListasTabuEcriteriosdeAspiracao()
8: k ← k+1
9: **fim enquanto**

Algoritmo Genético

Desde meados dos anos 50 se tentava simular a evolução, estudos de Nils Aall Barricelli e do geneticista australiano Alex Fraser são os primeiros esforços neste sentido. Na década de 60, com Ingo Rechenberg e Hans-Paul Schwefel usando estratégias evolutivas e Lawrence J. Fogel usando programação evolucionária, começa-se a resolver problemas de otimização usando conceitos de evolução. Com os trabalhos de John Holland no início da década de 70 e principalmente o seu livro, "Adaptation in natural and artificial systems", algoritmo genético se tornou popular. Mas, somente no meio da década de 80, com a *Primeira Conferência Internacional de Algoritmos Genéticos* em Pittsburgh, Pennsylvania, é que a pesquisa em algoritmos genéticos deixou de ser majoritariamente teórica.

Em um algoritmo genético, uma população de cadeia de caracteres (chamada de cromossomos ou genótipos do genoma), que codifica soluções candidatas (chamadas indivíduos, criaturas, ou fenótipos) para um problema, evolui para melhores soluções. Tradicionalmente, soluções são representadas em cadeia binária de caracteres (0s e 1s), mas outras codificações são possíveis. Cada caracter define um gene e seu local no cromossomo é denominado locus. Variações de um gene são denominadas alelos. Em otimização, a codificação é normalmente numérica (inteira ou real). A evolução

geralmente começa de uma população de indivíduos que são gerados aleatoriamente e acontece em gerações. Em cada geração, o fitness (medida de qualidade, quão adaptado está o indivíduo) de cada indivíduo na população é avaliado. Múltiplos indivíduos são estocasticamente selecionados da população corrente (seleção estocástica baseia-se no fitness), e modificados (através de recombinação e/ou mutação) formando assim um nova população, que é usada na próxima iteração do algoritmo. Comumente, o algoritmo termina quando ou um número máximo de gerações foram produzidas, ou um nível desejado de fitness da população foi atingido. O algoritmo 2 exibe estes passos supracitados do algoritmo genético. Para funcionar bem o algoritmo genético deve ter um controle em sua convergência evitando uma convergência prematura por um lado, mas também evitando uma convergência lenta demais por outro, portanto conceitos como pressão seletiva são muito importantes no aspecto de evolução dos algoritmos. Do ponto de vista de sua natureza, os algoritmos genéticos são algoritmos de busca estocásticos, portanto, não se pode garantir que uma solução ótima será encontrada em um número finito de iterações. Entretanto, em 1996 , Aytug & Koehler mostraram que, dado um valor $0 < \delta < 1$, é possível determinar um limite superior para o número de iterações, $t(\delta)$, de modo a garantir que a solução ótima seja visitada com probabilidade δ. Esse limite teórico é uma função da probabilidade de mutação, do tamanho da população e do número de bits para armazenar um cromossomo e desenvolvido para o algoritmo genético canônico (codificação binária, mutação uniforma, crossover simples, seleção por roleta). Para mais detalhes em algoritmos genéticos e outros algoritmos evolutivos os trabalhos Goldberg (1989); Michalewicz (1996); Bäck (1996); Bäck et al. (1997) são literaturas bem ricas.

Algoritmo 2 - Algoritmo Genético

```
p ← geraPopulacaoInicial()
avalia cada indivíduo de p e determina seu fitness
determina fitness médio da população
repita
    p ← aplica operador de seleção
    p ← aplica operador de reprodução a pares cromossomo definidos com taxa de reprodução r
    p ← aplica operador de mutação com taxa de mutação m
    avalia cada indivíduo de p e determina seu fitness
até condição de parada atingida
```

2.3 Escolha de Parâmetros

Na construção de meta-heurística uma grande responsável pelo esforço e tempo gastos é a escolha de valores para parâmetros, que tem por objetivo uma melhor adequação destes ao problema e conjunto de instâncias a serem solucionados. Isto é, a busca por melhor qualidade na solução e

melhor desempenho computacional. Os parâmetros aqui tratados dizem respeito aos que a literatura chama de parâmetros estratégicos, aqueles que são capazes de modificar a estratégia de busca da meta-heurística.

O número de parâmetros de uma meta-heurística pode variar bastante, por exemplo, no artigo de Xu & Kelly (1996), em seu algoritmo de fluxo em rede para o problema de roteamento, são apresentados 28 parâmetros de penalidade e outros 4 relacionados a escolha de soluções. Os pesquisadores vêm enfrentando este desafio com ações que variam de procedimentos de experimentação com tentativas e erro a análises de sensibilidade mais elaboradas. Breedam (1995) afirmou: "Os valores atribuídos a todos estes parâmetros técnicos são, em grande parte, determinados por experimentos de tentativa e erro" [1]. Gendreau et al. (1994) afirmam: "Este algoritmo contem vários parâmetros e um número de variações pode ser facilmente vislumbrado. Diversos testes e um considerável esforço em refinamento foram realizados para chegar a versão atual" [2]. Frases que demostram as características de experimentação na escolha de valores para parâmetro e o tempo gasto nestes procedimentos. Ainda, são poucos os artigos que relatam como foi efetivamente realizada a escolha dos parâmetros.

De maneira geral, a literatura classifica a forma como a escolha de parâmetros é executada como na Fig. 2.1, Eiben et al. (2007).

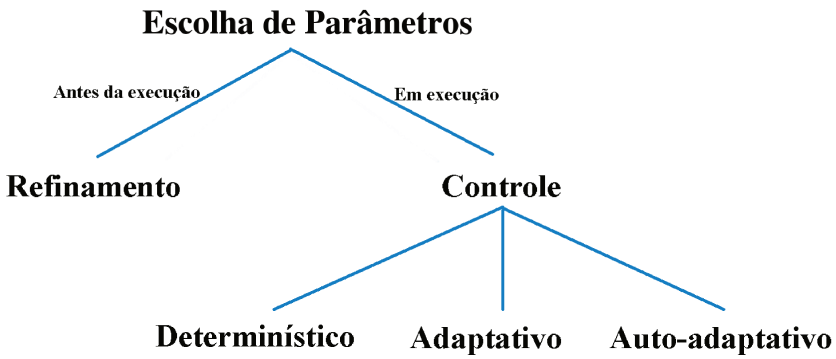

Fig. 2.1: Taxonomia na escolha de parâmetros

2.3.1 Refinamento de Parâmetros

Refinamento é a forma de definir parâmetro mais usada, por ser o caminho mais simples e por ser geralmente o passo inicial de quem quer realizar controle dos parâmetros. No refinamento de

[1] texto original: "The values that have to be assigned to all these technical parameters are for the greater part determined by trial-and-error-like experiments"

[2] texto original:"This algorithm contains several parameters and a number of variants can easily be envisaged. Several tests and a considerable amount of fine tunning were carried out in order to arrive at the current version"

parâmetros, procedimentos de experimentação e análise são executados anteriormente à execução da meta-heurística.

Esta abordagem típica apresenta vários problemas, como ressaltado em Eiben et al. (2007). O problema mais destacado é o tempo gasto com experimentações de valores. Pensando em um problema simples, com 2 parâmetros e 10 valores possíveis para cada um, há um total de 100 possibilidades de configurações. Levando em consideração que quase toda meta-heurística tem características não-determinísticas, ou seja, existe uma necessidade de ser executada mais de uma vez os testes, este número pode ficar bem maior. Além disto, os parâmetros são relacionadas entre si, testá-los independentemente pode não trazer resultados satisfatórios. Outro complicador que desestimula esta abordagem é que geralmente as variáveis estão no espaço contínuo e por mais tempo que se gaste no processo experimental, o conjunto de valores obtidos pode ficar longe de ser bom.

Quando se trata de meta-heurísticas evolutivas (algoritmos evolutivos) o cenário retratado acima piora. Estudos (Schwefel, 1981; Davis, 1989; Hesser & Männer, 1991; Smith & Fogarty, 1996; Stephens et al., 1997) mostram que em diferentes fases destes algoritmos os valores ótimos para os parâmetros são diferentes. Um exemplo para caracterizar isto é a necessidade da mutação ser mais intensa em gerações iniciais de um algoritmo genético, quando se busca maior diversificação e em estágios finais ser menor, quando se quer intensificar a busca para os melhores indivíduos obtidos no processo. A mutação não uniforme surgiu com este objetivo, aproveitar os diferentes momentos de um AG.

Várias formas de refinamento mais elaboradas foram surgindo na literatura, realizando análises descritivas e de sensibilidade. Alguns autores sob a influência de Hooker (1995), onde o autor advoga a favor de experimentos mais descritivos, adotaram metodologias de refinamento baseadas em experimentos fatoriais, características de uma análise descritiva. Desta forma, o problema real a ser tratado na verdade é o entendimento e verificação de importância de cada parâmetro dentro de um algoritmo. Como exemplo, Xu & Kelly (1996) tentam identificar a contribuição de cinco parâmetros diferentes em uma busca tabu, permutando a ativação destes e rodando com 7 instâncias testes selecionando os valores mais efetivos. Parson & Johnson (1997) usam abordagem similar. Xu et al. (1998) prosseguem com esta abordagem, mas, usando novos testes estatísticos, tais como: teste Friedman e Wilcoxon Conover (1998) .

Outra abordagem, adotada nos trabalhos Coy et al. (2001) e Beielstein & Preuss (2007), se baseia em um método estatístico chamando *metodologia de superfície de resposta*, que busca entender as relações de variáveis de entrada e saída. Ele se assemelha a um método de descida no gradiente ou busca local. O método realiza uma busca iterativa no espaço dos parâmetros, que pode assim ser descrita: define-se uma métrica qualitativa de avaliação de distância entre configurações da meta-heurística em questão. Começa uma busca em um ponto aleatório do espaço de parâmetros e analisa-

se a vizinhança, caso haja uma configuração melhor na vizinhança, esta passa ser o ponto corrente, caso contrário a busca se encerra retornando a configuração incumbente. Fica evidente a fragilidade desta abordagem necessitando estabelecer uma boa métrica, que nem sempre é possível, quando se tem categorização dos parâmetros por exemplo, isto é difícil. A configuração resultante do processo de busca é então adotada para execução das instâncias as quais deseja-se resolver.

Em proposta recente, Hutter et al. (2009) usam a meta-heurística de busca local iterativa no espaço de parâmetros, o arcabouço para configuração automática de parâmetros é denominado *ParamILS*. O algoritmo executa a busca em várias instâncias de um problema, variando um conjunto de parâmetros ordinais e categóricos, estatísticas em torno da qualidade das soluções encontradas definem a função objetivo, alguns limitantes de tempo também são adicionados para manter o tempo computacional do algoritmo tratável.

Uma abordagem nova e que parece promissora é a abordagem do livro de Birattari (2005). A proposta do autor do livro traduz as pequisas do grupo *Metaheuristics Network* que vem usando *aprendizagem de máquina*, mais especificamente, técnicas dentro de *aprendizado supervisionado*, uma das vertentes dentro de *aprendizagem de máquina*. O trabalho evoca uma abordagem estatística para o refinamento de meta-heurísticas. No texto é abordado o problema de realizar refinamento da meta-heurística para um conjunto de instâncias e esta meta-heurística ser capaz de responder satisfatoriamente para novas instâncias do mesmo problema. Logo, são tratados aspectos como *superaprendizagem*, situação onde as instâncias de treinamento são super-aprendidas perdendo o algoritmo sua capacidade de generalização. Resumidamente, o refinamento de meta-heurística é analisado sobre uma perspectiva formal em *aprendizado supervisionado* e com uma metodologia bem definida, isto por si só é uma novidade. No final alguns casos de estudo são apresentados.

Refinamento de parâmetro não almeja captar a dinâmica da busca, portanto, existe outra vertente de estudo que trabalha de forma a permitir aos algoritmos de busca mudarem de estratégia dinamicamente. Na maioria das vezes a busca "informa" sobre seu estado, possibilitando uma escolha de parâmetro em uma perspectiva de *controle*.

2.3.2 Controle

O controle do parâmetro ocorre quando, em tempo de execução de uma meta-heurística, a direção de estratégia ou mesmo a magnitude de valores de parâmetros são alterados. Três são as formas colocadas na literatura de realizar o controle.

- Determinístico. Ocorre quando o valor de determinado parâmetro é alterado por regras determinísticas segundo algum cronograma de alteração durante a execução. Não há retro-alimentação durante a busca. Não foi considerado este tipo de controle na revisão, por não ser o foco aqui.

- Adaptativo. Há retro-alimentações durante a evolução da busca que determinam direção e magnitude das alterações no parâmetro.

- Auto-adaptativo. Há uma evolução sobre a evolução. Meta-heurísticas, geralmente evolutivas, resolvem este outro problema, codificando os parâmetros como cromossomos, por exemplo, e propagando melhores valores de parâmetros.

Na literatura encontra-se menos trabalho no campo de *controle* de parâmetros de meta-heurísticas do que trabalhos em *refinamento* destes parâmetros. Os trabalhos de controle estão mais focados em algoritmos evolutivos. O livro de Lobo et al. (2007), traz uma coleção destes trabalhos.

Controle Adaptativo e Auto-Adaptativo

O ajuste de parâmetros dentro da computação remete ao início dos algoritmos evolutivos. Reed et al. (1967) realizaram testes em parâmetros probabilísticos para um jogo de poker simples atingindo planos perto de ótimos. No mesmo ano, Rosenberg (1967) propôs adaptar probabilidades de crossover. Ainda para algoritmos genéticos, Bagley (1967) considerou incorporar o controle de parâmetros a representação do indivíduo. Apesar deste início de auto-adaptação em AGs, abordagens efetivas só surgiram mais tarde. Schaffer & Morishima (1987) introduziram "crossover pontual auto-adaptativo" que ajustava o número de pontos de crossover. Poucos anos depois, Bäck (1992) propôs o primeiro método para adaptação da taxa de mutação em algoritmos genéticos.

A ideia de um meta-AG vem antes. Neste caso, um AG em um nível acima busca adaptar os parâmetros de controle de um algoritmo que resolve o problema original. Os trabalhos de Weinberg (1970) e Mercer & Sampson (1978) sugerem os primeiros passos.

Em estratégias evolutivas a necessidade de reforçar mutação em momentos específicos do processo evolutivo foi reconhecido por Rechenberg (1973), originando a conhecida regra de 1/5 de sucesso para mutação. Onde se 1/5 de taxa de mutação resulta em melhoria, a taxa de mutação é aumentada e diminuída caso contrário. Neste mesmo trabalho, foi acoplada a evolução de parâmetros estratégicos com outros parâmetros surgindo o conceito "alteração explícita", outra denominação para o controle adaptativo que veio a ser usada mais tarde.

Schwefel (1981) sugeriu um método auto-adaptativo para mudanças de parâmetros estratégicos chegando ao que hoje é chamado de controle auto-adaptativo. O controle efetuado ocorreu na matriz de covariância de uma distribuição normal multidimensional. Mesma técnica usada por Fogel et al. (1991) em programação evolutiva, com um operador para reforço de mutação.

Mais tarde, surgiram técnicas para aumentar a capacidade de controle, como a chamada controle de tamanho de caminho acumulado proposta por Ostermeier et al. (1994). Uma tentativa para "desaleatorizar" a adaptação. O trabalho desenvolvido realiza dois procedimentos que se tornaram bem

conhecidos: adaptação do tamanho do passo acumulado ("cumulative step-size adaptation", CSA) e adaptação da matriz de covariância ("covariance matrix adaptation", CMA) Gruenz & Beyer (1999).

A ideia por trás do controle do passo é reduzir a correlação entre passos sucessivos. Se passos sucessivos estão correlacionados paralelamente, produto escalar maior que zero entre as soluções obtidas, significa que caminham na mesma direção, podendo o passo ser substituído por um maior. No caso de uma correlação anti-paralela, produto escalar menor que zero, os passos se anulam. Portanto a melhor alternativa é não haver correlação. O trabalho proposto usa então um princípio para isto, reduzir a diferença da distribuição do *caminho evolutivo* atual e *caminho evolutivo* em situação de seleção aleatória reduz qualquer correlação.

Dado um caminho evolutivo $\mathbf{p}^{(g+1)} = (1-c)\mathbf{p}^{(g+1)} + \sqrt{c(2-c)}\mathbf{z}^{(g+1)}$, que acumula os passos da mutação selecionada, considere o caso onde o caminho evolutivo apresenta somente uma seleção aleatória, apresentando mutação distribuída normalmente dado por $\mathbf{u}^{(g)} = \sum_{k=1}^{g} \sigma\eta^{n(k)}(\mathbf{0},\mathbf{1})$, onde $\eta^k(\mathbf{0},\mathbf{1})$ é um vetor de distribuição normal independente com média zero e variância um. O comprimento de $u^{(g)}$ é uma distribuição χ com experança $\overline{u} = \sigma\overline{\chi}$. Uma seleção por fitness muda isto. Quando o passo da mutação está muito grande mutações menores são selecionadas resultando em um comprimento de *caminho evolutivo* menor do que $\overline{u}$, o que sugere redução do passo. No caso contrário, comprimento maior que $\overline{u}$, sugere-se o aumento do tamanho do passo. A adaptação da *matriz de covariância* é uma evolução na técnica e representa o estado-da-arte para este tipo de abordagem em estratégias evolutivas.

Fora do foco em meta-heurísticas evolutivas não são encontradas muitas contribuições para controle dos parâmetros em termos de variedade de métodos, na extensão de nosso conhecimento. Para busca tabu, por exemplo, Battiti & Tecchiolli (2004) apresentaram a primeira forma de controle que foi denominada busca tabu reativa ("Reactive Tabu Search"). Este trabalho define um controle no tamanho da lista tabu e introduz mecanismos de diversificação durante a busca para o problema quadrático de alocação (PQA), mais tarde esta mesma denominação foi empregada outras vezes para outros problemas. O principal aspecto do algoritmo é monitorar o ciclo de soluções (configurações de alocação) e dependendo do tamanho e freqüência dos ciclos, o tamanho da lista tabu é alterado e estratégias de diversificação são executadas, respectivamente. Para algoritmos de têmpera simulada, mecanismos de controle foram introduzidos quando parâmetros que definem a escala de temperatura e a seleção do passo aleatório desta, foram automaticamente ajustadas de acordo com o progresso do algoritmo. Este algoritmo foi chamado de têmpera simulada adaptativa ("Adaptive Simulated Annealing"). O algoritmo foi publicado por Ingber (1993a,b) para tornar o algoritmo original mais eficiente e menos suscetível a parâmetros definidos por usuários.

O trabalho nesta dissertação foca no controle de meta-heurísticas, seja evolutiva ou não, usando *Algoritmos Genéticos Nebulosos* acima da meta-heurística em destaque na resolução do problema ori-

ginal. O controle é especificado dentro da meta-heurística através de uma base de regra nebulosa. No monitoramento do que pesquisadores chamam de "evidências da busca", foram usadas evidências que podem ser aplicadas tanto para um algoritmo genético quanto para uma busca tabu, meta-heurísticas que este trabalho trata.

2.4 Resumo

Neste capítulo foi apresentada uma visão dos aspectos teóricos e práticos de meta-heurísticas. Foi exposta a evolução da área, mostrando as técnicas, o porquê de usarmos meta-heurísticas, dificuldades de escolha de uma meta-heurística para um problema e uma vez decidida a meta-heurística, as dificuldades de construção. A revisão de um dos principais problemas abordados na dissertação, a escolha de parâmetros estratégicos de meta-heurísticas, finaliza o capítulo.

No capítulo 3 é apresentado o caso de estudo usado neste trabalho, problema de roteamento de veículos com janelas de tempo, onde uma revisão é realizada.

Capítulo 3

Problema de Roteamento de Veículo com Janela de Tempo

3.1 Introdução

O problema de roteamento de veículos com janela de tempo (PRVJT), por possuir diversas aplicações, tem sido extensivamente estudado ao longo das últimas três décadas fazendo com que haja bastante literatura associada ao tema bem como várias técnicas de solução exploradas para resolvê-lo. Nos trabalhos de Golden & Assad (1986), Assad & Golden (1988), M. Desrochers & Soumis (1988) foram feitas as primeiras análises sobre as primeiras técnicas empregadas. Outros trabalhos como Bräysy & Gendreau (2005a) e Bräysy & Gendreau (2005b) realizam análises comparativas de métodos de resolução para o problema, focando em heurística e meta-heurísticas, já Bräysy et al. (2004a) dirigiram suas análises para algoritmos evolutivos. De forma não exaustiva, o capítulo aborda a origem e extensões do PRVJT, apresenta sua representação e formulação e por último realiza uma revisão focando em quatro nichos de técnicas de solução (heurísticas, métodos exatos, meta-heurísticas e soluções híbridas). Além disto, abordagens que não são consideradas propriamente uma técnica, mas sim formas de aprimorar técnicas, como paralelismo e cooperação, também são apresentadas.

3.2 Origem e Extensões

O problema de roteamento de veículo capacitado (PRVC) é uma combinação de caixeiro viajante (PCV) com o problema de empacotamento (PE), sendo origem de uma série de problemas diversos em transporte e logística.

O PRVC pode ser assim definido: seja um grafo $G = (V, A)$ completo, em que A é um conjunto de arcos que representam os caminhos que ligam os clientes entre si e estes ao depósito, e $V = 0, .., n+1$

denota um conjunto de $n + 2$ vértices. Por convenção, o vértice 0 e $n + 1$ representa o depósito e os outros simbolizam os clientes. A cada arco (i, j) é associado um custo não negativo, $c_{i,j}$, que representa o custo de viagem do vértice i ao vértice j. Na maioria dos casos, os custos dos arcos satisfazem a desigualdade triangular, $c_{i,k} + c_{k,j} \geq c_{i,j}$, para $i, k, j \in V$. Quando o custo do arco (i, j) é igual ao custo do arco (j, i), dizemos que o problema é simétrico, assimétrico, caso contrário.

Um conjunto K de veículos idênticos, com capacidade C, é alocado a um único depósito. A cada cliente i é associada uma demanda não negativa, d, tendo o depósito demanda zero. O PRVC consiste em encontrar um conjunto de exatamente K rotas, cada uma percorrida por um veículo, de modo a minimizar o custo total de transporte e a satisfazer algumas restrições: a rota começa e termina no depósito; cada cliente é visitado somente uma vez e por um único veículo; a soma das demandas dos clientes incluídos em uma rota não deve exceder a capacidade do veículo.

O PRVC é simples e não é capaz de representar todas as situações cotidianas e operações de diversos setores de logística de empresas de distribuição de mercadorias e serviços. Assim, muitas vezes é necessário introduzir neste problema algumas restrições associadas a diferentes elementos para se chegar perto do problema real. Alguns desses aspectos são:

- entrega de diferentes tipos de produto;
- serviço executado em tempo predefinido;
- entrega a ser realizada em período específico, atendendo limites de horário do cliente;
- restrição de demandas a determinado subconjunto de veículo;
- devolução de produtos não entregues;
- veículos de tipo e capacidade diferentes;
- existência de um conjunto de depósitos, possuindo ou não seu conjunto específico de veículos, que devam sair e/ou voltar para o depósito;
- pode haver revisão de veículos de determinado tipo após certo tempo de utilização;
- a demanda pode ser estocástica;
- o custo entre clientes pode ter custo variado em função de algum fator, como trânsito por exemplo.

Com isto novos problemas são derivados, como os apresentados, de forma sucinta, na sequência.

3.2.1 Problema de Roteamento de Veículos com Prioridade para a Entrega (PRVPE)

O Problema de Roteamento com Prioridade para a Entrega (em inglês "Vehicle Routing Problem with Backhauls" - VRPB) é uma extensão do problema capacitado na qual os clientes são divididos em dois subconjuntos. O primeiro contém os clientes aos quais é preciso entregar produtos (em inglês, "linehaul customers"). O segundo subconjunto é formado pelos clientes que desejam enviar produtos ao depósito ("backhaul costumers"). Assim, a cada cliente é associada uma demanda não negativa, que pode ser de entrega ou de coleta. Os clientes que receberão mercadorias devem ser atendidos antes dos clientes que terão seus produtos coletados, caracterizando uma relação de precedência. O PRVPE possui uma restrição de capacidade diferente daquela usada no PRVC, já que tanto a carga a ser entregue quanto a coletada não deve ultrapassar a capacidade do veículo.

3.2.2 Problema de Roteamento de Veículos com Coleta e Entrega (PRVCE)

Assim como o PRVPE, o Problema de Roteamento de Veículos com Coleta e Entrega ("Vehicle Routing Problem with Pickup and Delivery" - VRPPD) permite que parte dos clientes receba mercadorias e que parte as entregue ao veículo coletor. Entretanto, no PRVCE, os itens entregues por um cliente podem ser reaproveitados por outro cliente, não havendo, portanto, obrigatoriedade de precedência da entrega em relação à coleta. A cada cliente são associados dois valores, representando o que ele recebe e entrega. Cada cliente pode ter ou não dois clientes associados, um de quem receberá uma entrega e outro ao qual os seus itens coletados serão entregues, neste caso, se estabelece uma precedência. O depósito pode certamente assumir este papel. Neste cenário, a diferença coleta-entrega em cada visita na rota não pode ultrapassar a capacidade do veículo.

3.2.3 Problema de Roteamento de Veículos com Múltiplos Depósitos (PRVMD)

Este problema, PRVMD ("Multi-Depot Vehicle Routing Problem" - MDVP), se diferencia do problema capacitado pela existência de vários depósitos, cada qual abrigando uma frota de veículos. Entretanto, ao final de sua rota, um veículo deve sempre retornar ao seu depósito de origem, sem passar por outros depósitos. Naturalmente, para evitar a resolução do PRVMD, que é mais complexa, podemos agrupar os clientes próximos em conjuntos, e designar cada grupo de clientes a um depósito, gerando vários PRVC, um para cada depósito, que podem ser resolvidos separadamente. Como extensão a esse problema, é possível definir depósitos intermediários pelos quais um veículo pode passar para reabastecer, antes de finalizar a sua rota. Existem poucos trabalhos na literatura explorando esse problema, (em inglês, recebeu o nome de "Multi-Depot Vehicle Routing Problem with Inter-Depot

Routes" - MDVRPIR).

3.2.4 Problema de Roteamento de Veículos com Múltiplo Uso de Veículos (PRMUV)

Quando a frota de veículos é pequena, ou a duração média das rotas é bem menor que um dia, é necessário atribuir mais de uma rota a um veículo. Neste caso, temos o Problema de Roteamento com Múltiplo Uso de Veículos ("Vehicle Routing Problem with Multiple Use of Vehicle" - VRPMUV). O objetivo é o mesmo do problema capacitado, ou seja, a determinação das rotas que, somadas, forneçam o menor custo total. As restrições do PRVC também são utilizadas no PRMUV.

3.2.5 Problema de Roteamento de Veículos com Frota Heterogênea (PRVFH)

Este problema, PRVFH ("Heterogeneous Fleet Vehicle Routing Problem" - HFVRP), diferencia-se do problema capacitado pelo fato de cada veículo apresentar uma capacidade específica. Existem duas variantes para o problema com frota heterogênea. Na primeira, o número de veículos de cada tipo é pré-estabelecido, de modo que para resolver o problema basta atribuir uma rota para cada veículo. Na segunda, é possível definir características particulares de velocidade, de tempo de viagem e de custos variáveis e fixos para cada veículo. O número de veículos de cada tipo não é especificado, assim, além de construir uma rota para cada veículo, é necessário determinar o número de veículos utilizados. A esse último problema dá-se o nome de Problema de Dimensionamento e Roteamento de uma Frota Heterogênea de Veículos (em inglês, "Fleet Size and Mix Vehicle Routing Problem"). Em ambas as variantes, as restrições são similares às do PRVC, lembrando que cada veículo possui uma restrição de capacidade particular.

3.2.6 Problema de Roteamento de Veículos com Entregas Fracionadas (PRVEF)

O Problema de Roteamento de Veículos com Entregas Fracionadas ("Split Deliveries Vehicle Routing Problem" - SDVRP) foi introduzido na literatura recentemente. Nele, os clientes podem ser atendidos por mais de um veículo, contrariando uma restrição imposta ao problema capacitado. Desta forma, é preciso definir também uma nova restrição de capacidade, que deve garantir que a soma das frações de demanda dos clientes visitados por um dado veículo não exceda a capacidade deste.

3.3 PRVJT: Formulação e Representação

Este trabalho trata do problema de roteamento de veículos com janelas de tempo, PRVJT, com um depósito. Mais especificamente, existe um conjunto de clientes que possuem uma demanda de transporte específica e uma janela de tempo determinando quando a demanda deve ser atendida. Veículos com capacidades idênticas realizam o atendimento do serviço. O objetivo é achar o conjunto de rotas com custo mais baixo começando e terminando no depósito dentro do horário de trabalho. Factível significa que os veículos devem respeitar suas capacidades e atender a demanda do cliente dentro da janela de tempo especificada por este. Cada cliente é atendido uma vez. O serviço pode ser definido antes da janela de tempo (neste caso o veículo espera), mas não pode ultrapassar a janela. Essa é a abordagem rígida ("hard") para o PRVJT.

O PRVJT pode ser associado a um grafo $G(V, A)$ onde $V = C \cup \{v_0, v_{n+1}\}$ e $C = \{v_1, \ldots, v_n\}$ representam os n clientes e v_0, v_{n+1} o depósito. O conjunto $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ define arestas entre nós clientes, e v_0 e v_{n+1} representam o depósito no início e fim da rota, respectivamente. Consequentemente, não existe aresta começando em v_{n+1} nem uma aresta terminando em v_0. Cada aresta (v_i, v_j) tem um custo c_{ij} e um tempo de viagem associados t_{ij}. Um tempo de serviço s_i é associado ao cliente i que tem demanda d_i, $i \in C$. O depósito tem um conjunto K veículos com capacidade q para servir os clientes. Uma janela de tempo $[e_i, l_i]$ deve ser obedecida quando inicia-se o serviço do cliente i. As variáveis de decisão são $X^k ij$ onde:

$$X^k ij = \begin{cases} 1 & \text{se veículo } k \text{ atravessa aresta } (i,j) \\ 0 & \text{caso contrário} \end{cases}$$

$\forall k \in K, \forall (i,j) \in A$.

Seja B_i^k o tempo que o veículo k começa atender o cliente i, $\forall k \in K, \forall i \in C$. O problema PRVJT pode ser assim descrito:

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} X_{ij}^k \tag{3.1}$$

sujeito a:

$$\sum_{k \in K} \sum_{j \in C \cup \{v_{n+1}\}, j \neq i} X_{ij}^k = 1, \quad \forall i \in C \tag{3.2}$$

$$\sum_{i \in C} \sum_{j \in C \cup \{v_{n+1}\}, j \neq i} d_i X_{ij}^k \leq q, \quad \forall k \in K \tag{3.3}$$

$$\sum_{j \in C \cup \{v_{n+1}\}} X_{0j}^k = 1, \quad \forall k \in K \tag{3.4}$$

$$\sum_{i \in C} X^k_{i,n+1} = 1, \quad \forall k \in K \tag{3.5}$$

$$\sum_{i \in C \cup \{v_0\}} X^k_{ih} - \sum_{j \in C \cup \{v_{n+1}\}} X^k_{hj} = 0, \quad \forall h \in C, \forall k \in K \tag{3.6}$$

$$X^k_{ij}(B^k_i + s_i + t_{ij} - B^k_j) \leq 0, \quad \forall (i,j) \in A, \forall k \in K \tag{3.7}$$

$$e_i \leq B^k_i \leq l_i, \quad \forall i \in C \cup \{v_{n+1}\}, \forall k \in K \tag{3.8}$$

$$X^k_{ij} \in \{0,1\}, \quad \forall (i,j) \in A, \forall k \in K \tag{3.9}$$

Portanto, o objetivo é minimizar o custo geral de transporte considerando as seguintes restrições: (3.2) todos clientes devem ser atendidos por um único veículo, (3.3) toda demanda atendida por um veículo não pode ultrapassar sua capacidade de carga, (3.4) toda rota deve iniciar no nó v_0, (3.5) toda rota deve terminar no nó v_{n+1}, (3.6) um veículo deve atender e sair do cliente, (3.7) se um cliente j vem depois do cliente i em uma rota, o tempo de serviço i e o tempo total de viagem (i,j) são considerados, (3.8) um veículo deve atender o cliente dentro de sua janela de tempo.

3.4 Resolvendo com Métodos Exatos

As principais técnicas exatas empregadas para a resolução do PRVJT são branch-and-bound, branch-and-price (branch-and-bound mais geração de coluna) e branch-and-cut (branch-and-bound mais planos de corte).

3.4.1 Branch-and-Bound

O algoritmo Branch-and-Bound é um método de otimização global que é bastante empregado para a solução de problemas de programação inteira. O PRVJT é um problema de programação inteira mista. A resolução se inicia relaxando todas restrições de integralidade, começando na sequência a fase de *branching*, onde uma variável binária é fixada em 0 e 1, dando origem a dois subproblemas. Estes subproblemas definem limitantes inferiores (problema de minimização) pelo valor da função objetivo. Quanto mais baixo é o nível na árvore de subproblemas estabelecida, "menos relaxados" ficam os subproblemas e mais perto de uma solução se chega. Os nós da árvore formam os ramos ainda não explorados. Quando uma solução $x^\star_k$ factível é atingida, inicia-se a fase de *bounding*,

que elimina, poda todos os nós que ainda podem ser explorados mas têm limitante pior, no caso de minimização maior, que o gerado por $x_k^\star$.

Para resolução do PRVJT, Kolen et al. (1987) apresentaram uma solução com branch-and-bound minimizando a distância total percorrida.

3.4.2 Branch-and-price

O método Branch-and-Price é a combinação do método de Branch-and-Bound com geração de coluna, onde a cada nó da ramificação o método de geração de colunas encontra novas soluções viáveis. A geração de colunas é uma generalização da decomposição de Dantzig-Wolfe (Dantzig & Wolfe, 1960), que subdivide o problema em dois: o principal e um subproblema. Como o conjunto de soluções factíveis é muito grande, somente parte deste é considerado no problema principal (somente algumas colunas são adicionadas). Para o problema de roteamento, em cada iteração do problema principal, verifica-se pelo subproblema se existe alguma rota que possibilite a redução da distância total (melhora da função objetivo do problema principal). Neste caso a coluna correspondente a esta rota é adicionada ao modelo. O procedimento é reiniciado até que seja encontrada uma solução ótima.

A primeira resolução com branch-and-price para o PRVJT foi publicada por Desrochers et al. (1992), eles resolvem o subproblema (relaxação linear), utilizando o problema do caminho mais curto empregando restrições de capacidade e janela de tempo, para tal fim foi utilizado programação dinâmica. Com isto, bons limitantes inferiores são obtidos para o problema inteiro principal que é o de partição de conjunto.

3.4.3 Branch-and-cut

O método de Branch-and-cut é aplicado ao problema relaxado na fase de *branching* com o propósito de reduzir o espaço de busca para os subproblemas derivados do nó em questão na árvore. Este objetivo é atingido com adição de restrições que cortam o politopo factível do problema. O desafio dos cortes é definir o corte mais profundo, que produza a maior redução do espaço de busca, sem comprometer o politopo de forma a perder potenciais soluções ótimas.

Bard et al. (2002) empregam a técnica para o PRVJT. Soluções factíveis, no caso limitantes superiores, são obtidos com GRASP. Cortes são introduzidos para apertar a relaxação do problema inteiro misto original, o problema de separação para eliminar cortes violados é o desafio maior. Uma heurística com este propósito é apresentada.

3.5 Resolvendo com Heurísticas

Os algoritmos heurísticos apresentam capacidade de encontrar soluções boas com baixo custo computacional, o que os tornam bastante empregados em problemas NP-difíceis como PRVJT. Podemos ainda separar as heurísticas quanto ao momento de execução no algoritmo de busca: heurísticas de construção (início da busca) e heurística de refinamento ou busca local (fim da busca).

3.5.1 Heurísticas Construtivas

Uma heurística construtiva é um algoritmo que tem um procedimento passo a passo para chegar à uma solução factível, partindo de uma solução infactível, normalmente uma solução vazia. No caso do PRVJT existem soluções que caminham construindo soluções de forma sequencial, uma rota por vez, ou paralela, várias rotas simultaneamente. As heurísticas de construção são muito usadas para criar soluções iniciais para outros algoritmos, principalmente as meta-heurísticas.

Vizinho mais próximo

A heurística do vizinho mais próximo é uma heurística bem simples, na qual uma rota é construída inserindo um cliente por vez, neste caso o mais próximo segundo algum critério, até que não se consiga adicionar mais cliente à rota de forma a manter a factibilidade desta, neste momento inicia-se outra rota. O procedimento encerra quando todos os clientes pertencem a alguma rota. No trabalho de Solomon (1987b), a medida de proximidade, $\overline{c}_{ij}$ é definida como a combinação convexa de três medidas: distância entre nós (d_{ij}), um tempo de transição de i para atendimento em j (t_{ij}) e urgência do atendimento em j (u_{ij}), portanto:

$$\overline{c}_{ij} = w_1\, d_{ij} + w_2\, t_{ij} + w_3\, u_{ij}, \quad w_1, w_2, w_3 \geq 0 \; e \; w_1 + w_2 + w_3 = 1 \tag{3.10}$$

sendo

$$t_{ij} = B_j - (B_i + s_i) \tag{3.11}$$

$$u_{ij} = l_j - (B_i + s_i + t_{ij}) \tag{3.12}$$

Em (3.11), t_{ij} é a diferença do início do atendimento em j e o término do serviço em i. Em (3.12), u_{ij} é a diferença do fim do atendimento de j e chegada do veículo neste nó.

Economias

A heurística da economia foi desenvolvida por Clarke & Wright (1964) para resolver o problema de roteamento de veículos capacitados. Ela se baseia na união de rotas de menor custo. Na fig. 3.1, por exemplo, são apresentadas duas rotas: R_1 e R_2, unidas pela remoção das arestas (a, d) em R_1 e (c, d) em R_2 e adição da aresta (a, c). Neste caso a medida de economia pode ser definida como: $s_{ac} = c_{ad} + c_{cd} - c_{ac}$.

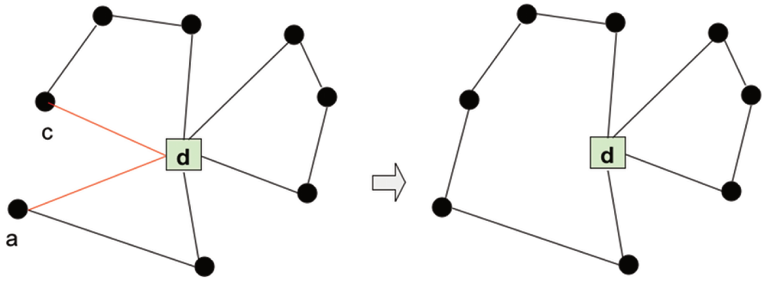

Fig. 3.1: Exemplo de união de rota na heurística da economia

De forma genérica:

$$s_{ij} = c_{id} + c_{jd} - c_{ij}, \quad \{(i, j), (i, d), (j, d)\} \in A \tag{3.13}$$

O algoritmo pode ser então ser descrito:

Algoritmo 3 - Heurística da economia de Clarke & Wright

```
1: Crie n rotas, cada uma com um cliente.
2: enquanto existir união faça
3:     para cada par de rota faça
4:         calcule segundo 3.13, a economia da união.
5:         ordene as uniões segundo a economia decrescentemente.
6:         escolha a união factível de maior economia (siga ordenação).
7:     fim para
8: fim enquanto
```

Esta heurística tem algumas variações seguindo medidas diferentes de economia, como pode ser visto em Solomon (1987b).

Varredura

O método da varredura é uma heurística criada por Gillett & Miller (1974) voltada para o problema de roteamento de veículos capacitados com restrição de distância por rota, mas foi adaptada por

Solomon (1987b) para a inclusão da restrição de janela de tempo. O funcionamento desta heurística é descrito abaixo:

Algoritmo 4 - Heurística da Varredura

1: Coloque o depósito no centro de coordenadas polares definindo uma semi-reta (vide fig. 3.2)
2: **para** cada cliente i **faça**
3: defina coordenadas polares (d_i,Θ_i), seguindo a referência no depósito.
4: **fim para**
5: Crie uma lista de clientes, l, pendentes para inserção seguindo uma ordem (crescente ou decrescente) de Θ em caso de empate, desempate com d, favorecendo menor distância.
6: Crie uma rota r vazia
7: **enquanto** $l \neq \emptyset$ **faça**
8: **se** rota r for factível com a inserção do cliente da vez **então**
9: realize inserção
10: **senão**
11: guarde a rota r e crie nova rota r a partir deste cliente.
12: **fim se**
13: **fim enquanto**
14: **para** cada rota r guardada do passo anterior **faça**
15: resolva o problema do caixeiro viajante.
16: **fim para**

A adaptação do algoritmo proposta por Solomon (1987b), ocorre no passo final, linha 14, ao invés de resolver o problema do caixeiro viajante, ele propõe a resolução de uma heurística que considera a restrição de janela de tempo. Nesta heurística pode ocorrer exclusão de clientes da rota para manter a factibilidade, neste caso o procedimento apresentado reinicia com o todos clientes excluídos.

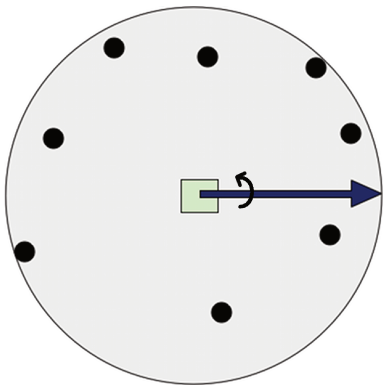

Fig. 3.2: Visão de funcionamento da heurística da varredura.

Inserção mais barata

A inserção mais barata é uma heurística sequencial onde é inserido à rota sendo construída o cliente que apresenta o menor custo de inserção em qualquer posição, mantendo a factibilidade da rota. De forma semelhante a heurística do vizinho mais próximo, se nenhum cliente pode ser inserido, uma rota nova é iniciada. Umas das heurísticas mais usada é a heurística *Push Forward Insertion Heuristic (PFIH)* de Solomon (1987b) que se enquadra nesta forma de busca de solução e será apresentada em mais detalhes abaixo, por ser base de várias heurísticas e ser a heurística usada neste trabalho para gerar solução inicial. Dois conceitos importantes para entendimento da heurística são:

- instante de início de atendimento do cliente j: $B_j = \max\{e_j, B_i + s_i + t_{ij}\}$;
- o custo da aresta (i, j) é definido como: $c_{ij} = \alpha_1 d_{ij} + \alpha_2 (B_j - B_i), \quad para\ \alpha_1, \alpha_2 \geq 0.$

Ainda é considerado na heurística um número ilimitado de veículos e a janela de tempo do depósito é [e_0,l_0]. Para ilustrar como funciona o algoritmo tomemos uma rota com a seguinte sequência de clientes $(i_0, i_1, \ldots, i_p, \ldots, i_{n+1})$, onde i_0 e i_{n+1}, representam o depósito. O cliente p representa a posição posterior a inserção, ou seja o cliente adicionado a, ficará entre a posição $p-1$ e p. Esta inserção afeta todos os tempos de atendimento da rota deste ponto para frente, portanto não basta uma verificação somente nos "arredores" da inserção. Para facilitar a verificação de factibilidade, Solomon (1987b) criou o procedimento *push forward* , que tem por objetivo acrescentar aos clientes posteriores a a, o acréscimo no instante de atendimento (PF) que a inserção acarreta. Assim para o primeiro cliente posterior p, o novo instante de atendimento é $B_{i_p}^{novo}$ e $PF_p = B_{i_p}^{novo} - B_{i_p}$. Para os clientes posteriores a p:

$$PF_{i_k} = \max\{0, PF_{i_{k-1}} - w_{i_k}\}, \quad k = p+1, \ldots, n+1,$$

w_{i_k} é o tempo de espera em i_k dado por:

$$w_{i_k} = e_{i_k} - B_{i_k},$$

ou seja é o tempo que o veículo espera caso ele chegue para o atendimento do cliente k antes do início da janela de tempo. Por isto caso o PF_{i_k} seja menor que a espera, não há mais acréscimo deste cliente em diante, a inserção na verdade está se aproveitando das "folgas" presentes no sequenciamento de atendimento dos clientes. Portanto esta medida de acréscimo é usada para verificar a factibilidade, partindo do cliente p e parando em caso de infactibilidade ou caso $PF = 0$ em algum cliente. O algoritmo apresenta dois aspectos para definição importantes:

- Escolha do primeiro cliente de uma rota

- Definição da medida de custo

O primeiro pode ocorrer de diversas formas, as que Solomon (1987b) apresenta são:

- O cliente mais distante do depósito;
- O cliente com janela de tempo de menor instante de início;
- O cliente com janela de tempo de menor instante de fim;
- Ponderações destas 3.

Esta etapa é o primeiro passo na definição de uma rota descrito no algoritmo 5, definido por Solomon para tal objetivo:

Algoritmo 5 - Heurística de Solomon para criar rota

1: $LNI \leftarrow V$; {LNI é o conjunto de clientes não inseridos}
2: **enquanto** $LNI \neq \emptyset$ **faça**
3: Crie uma rota (r) vazia.
4: Defina o primeiro cliente desta rota em LNI adotando algum critério (tais como os de Solomon supracitado) e o remova de LNI.
5: **enquanto** possível inserção em r **faça**
6: **para** cada cliente u de LNI **faça**
7: verifique a posição (p) de menor custo de inserção ($\overline{c}_{iuj}$) em r, mantendo a rota factível:
8: **se** existe p **então**
9: **se** existe algum outro cliente v de $LNI \setminus \{u\}$ que posa ser inserido na mesma posição p com menor custo **então**
10: insira v em r na posição p e o retire de LNI
11: **senão**
12: insira u em r na posição p e o retire de LNI
13: **fim se**
14: **fim se**
15: **fim para**
16: **fim enquanto**
17: **fim enquanto**
18: **retorna** r

O outro ponto de definição neste algoritmo é o custo $\overline{c}_{iuj}$, que pode ser calculado conforme abaixo:

$$\overline{c}_{iuj} = d_{iu} + d_{uj} + \mu\, d_{ij}, \quad \text{para } \mu \geq 0 \tag{3.14}$$

$$\overline{c}_{iuj} = B_j^{novo} - B_j \tag{3.15}$$

$$\tag{3.16}$$

Na Eq. (3.14) é avaliado o acréscimo de distância que a inserção de u entre i e j proporciona, já na equação (3.15) o interesse está no acréscimo de tempo para atendimento de j com a mesma inserção. Outras variações podem ser definidas, como por exemplo a combinação convexa destas duas, entre outras.

Entre as heurísticas de inserção mais barata, devemos citar ainda: Potvin & Rousseau (1993), formato paralelo da heurística do Solomon (1987b) e a proposta por Ioannou et al. (2001) que é uma variação do PFIH.

Neste trabalho foi usado o PFIH como heurística base para construção de rotas iniciais para os algoritmos criados devido seu amplo uso na literatura e bom desempenho. Como a heurística serve para criação de solução inicial, foram privilegiados melhores resultados em termos de qualidade de solução.

Outras heurísticas e estudos

Além do estudo de Solomon (1987b), Bräysy et al. (2004a) fazem uma análise comparativa de heurísticas e a proposta por Bramel & Simchi-levi (1993) é a que apresenta melhor resultado. Esta abordagem apresenta um algoritmo duas fases: a primeira é similar à uma relaxação lagrangeana, na qual é definido um número m de clientes que formam cada um, uma rota inicial e depois, como segunda etapa, uma heurística de inserção gulosa. Nesta análise estão inclusas as abordagens destacadas nesta revisão de heurísticas construtivas, que são extremamente importantes como primeiro passo de diversos algoritmos para o PRVJT.

3.5.2 Resolvendo com Heurísticas de Refinamento ou Buscas Locais

A principal característica de uma heurística de refinamento ou busca local é explorar a vizinhança de uma solução, ou seja, fazer alterações na solução seguindo algum padrão que melhore a função objetivo do problema. Várias vizinhanças, apresentadas nesta seção como uma busca local, compõem meta-heurísticas mais complexas, inclusive as apresentadas neste trabalho. Um outro aspecto importante destas buscas locais que impacta diretamente no desempenho dos algoritmos é a estratégia de exploração da vizinhança, destacando-se *First Best*, na qual é aceito o primeiro movimento que melhore a solução e a *Global Best*, onde o melhor movimento é escolhido.

K-opt

O k-opt é um procedimento criado para o problema do caixeiro viajante por Lin (1965) e sua natureza é de realizar movimentos intra-rota (dentro de uma das rotas da solução) removendo k arestas e inserindo outras k arestas entre os vértices pertencentes ao circuito, mantendo a factibilidade da rota

e reduzindo o custo. A fig. 3.3 exibe a realização de um movimento 2-opt, retirando duas arestas da rota circulada e inserindo duas arestas novas fechando o circuito.

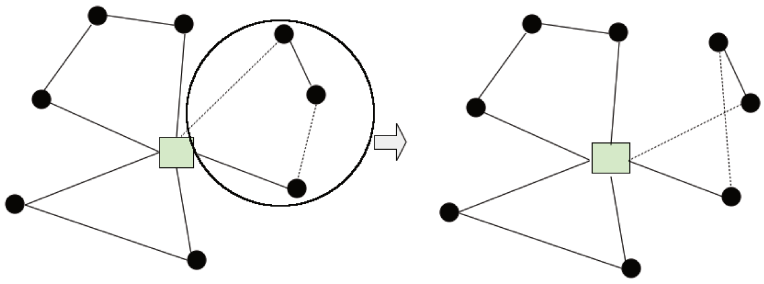

Fig. 3.3: Um movimento 2-opt

Lin & Kernighan (1973), para $k \geq 3$, criaram um procedimento onde uma operação k-opt é resultado de uma sequência finita de operações 2-opt. Como o número de possibilidades de inserção de arestas cresce exponencialmente com k, o objetivo deste procedimento é limitar o espaço de busca e conseguir boas melhoras da solução.

Or-opt

Or-opt segue a linha do procedimento de Lin (1965), neste procedimento de um a três vértices consecutivos são desconectados da rota retirando todas arestas incidentes a algum vértice ainda na rota e o vértice excluído. Depois este caminho, formado pelos vértices excluídos, é reconectado a rota com inserção de novas arestas com objetivo de manter a factibilidade e reduzir o custo da rota. A fig. 3.4 mostra os passos de um movimento or-opt com dois vértices, primeiro a retirada das arestas ligadas aos vértices depois a reconexão do caminho que ligam estes vértices ao caminho que restou formando novamente um circuito.

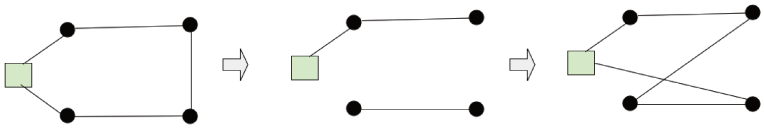

Fig. 3.4: Um movimento Or-opt com 2 vértices

2-opt*

O método de 2-opt* é uma adaptação do 2-opt realizada por Potvin & Rousseau (1995) para ser um procedimento inter-rota considerando janela de tempo e mantendo a orientação das rotas. Uma aresta é retirada das duas rotas participantes dando origem a quatro caminhos que são reconectados

formando novamente duas rotas factíveis com custo menor que o original em sua soma, conforme ilustrado na fig. 3.5. Por ser tratar de um movimento inter-rota e que não busca manter o mesmo número de elementos nas rotas alteradas, ele permite redução de rotas, pois pode ser gerado o caso da aresta desconectadas estarem ligadas ao depósito.

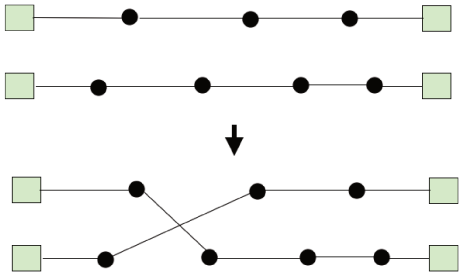

Fig. 3.5: Um movimento 2-opt*

Intertroca-λ

Osman (1993) criou este procedimento onde é realizada a troca de clientes entre rotas, gerando na soma dos custos da rota uma solução factível e de menor custo. Como neste caso é uma troca, mesmo sendo um procedimento inter-rota não existe possibilidade de redução no número de rotas.

Transferência-k cíclica b

Este procedimento consiste na transferência de k clientes de x rotas entre estas x rotas em uma determinada rotação. A fig. 3.6 ilustra um movimento de uma transferência-2 cíclica-3. De 4 rotas ($R1$,$R2$,$R3$,$R4$), 3 ($R1 - R3$) foram escolhidas para participar da rotação com 2 clientes, portanto conforme ilustrado a rota $R1$ recebeu os clientes $C4$ e $C5$ da rota $R3$, a rota $R2$ recebeu os clientes $A2$ e $A3$ da rota $R1$ e por fim, $R3$ recebeu os clientes $B1$ e $B3$ da rota $R2$. Esta técnica pode ser vista em mais detalhes em Thompson & Psaraftis (1993).

Troca-cruzada

A troca cruzada é uma técnica semelhante a 2-opt*, a diferença reside no fato da troca cruzada remover duas arestas de cada rota participante ao invés de uma. Ela foi usada originalmente em um PRVJT por Taillard et al. (1997) e continua uma das técnicas mais usadas.

Cadeias de ejeções

As cadeias de ejeções foram criadas, como muitas da heurísticas de refinamento abordadas, para o problema do caixeiro viajante, Lin & Kernighan (1973). Esta estrutura de vizinhança funciona com

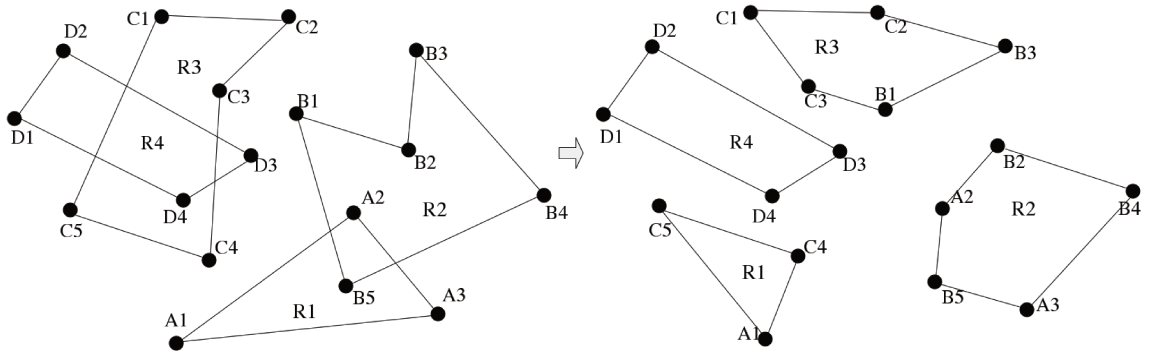

Fig. 3.6: Um movimento de uma transferência-2 cíclica-3

a realização de movimentos que retiram a solução de sua *estrutura solução*, factível para o problema, pelos ditos movimentos de ejeção, indo para uma *estrutura de referência definida*. Desta estrutura de referência, um movimento de transição, leva novamente a estrutura de referência para uma solução do problema. Mais formalmente no contexto de teoria de grafo, uma cadeia de ejeção de L níveis de sub-caminhos em um grafo G consiste na sequência de operação, movimentos de ejeção <$e_1, e_2, \ldots, e_l$>, que transforma o subgrafo G_n em G_{n+1} desconectando sub-caminhos e os reconectando a diferentes componentes. A cada nível da cadeia a estrutura de referência atingida não necessariamente é factível e geralmente não o é, mas uma solução sempre pode ser obtida por uma movimento de transição (*trial move* - "uma tentativa de obter uma solução boa"). A cadeia de ejeção como estrutura de vizinhança em uma busca é a sequência de pares (movimento de ejeção, movimento de transição), <$(e_1, t_1), \ldots, (e_m, t_m), \ldots, (e_l, t_l)$> , para o sub-escrito m significando o nível da melhor solução obtida, esta pode ser reproduzida pela sequência <$(e_1, e_2, \ldots, e_m, t_m)$>. Os procedimentos LK de Lin & Kernighan (1973), reinou por muito tempo como a melhor heurística para o PCV. A evolução das cadeias de ejeção está diretamente relacionada com a evolução das estruturas de referência, o que permitiu também o uso deste procedimento para problemas de roteamento. A fig. 3.7 mostra algumas estruturas de referência mais recentes.

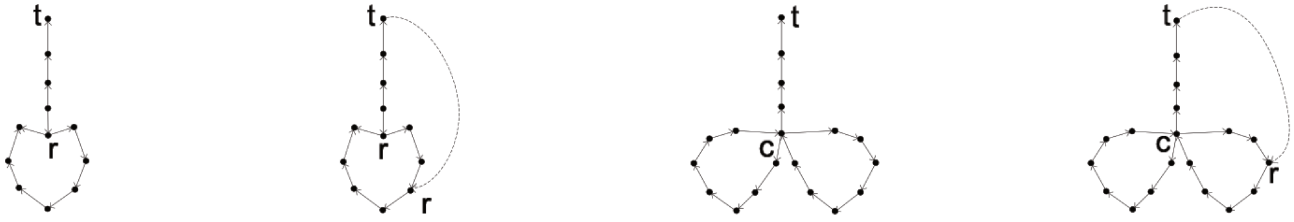

Fig. 3.7: Estruturas *SC*,*DR*, *Flower* e *CDR*

Busca em grande vizinhança

Esta busca como o próprio nome diz, define uma estrutura de vizinhança grande. Ela foi usada por Shaw (1998) para o problema de roteamento de veículos, PRV. Neste caso, a estrutura foi definida com uma remoção de um número x de clientes, relaxação e a re-inserção deste clientes buscando melhoria da função objetivo, re-optimização. Diversas podem ser tanto a retirada quanto a devolução dos clientes à solução. Os clientes podem ser retirados de forma genérica ou relacionada. Podem ainda ser retirados aleatoriamente ou seguir probabilidades de reinserção, ou mesmo usar técnicas de agrupamento, como medida de relacionamento. Dado isto, é claro que uma das desvantagens da busca pode ser o desempenho, problema este que é geralmente contornado por mecanismos mais eficientes como branch-and-bound para re-inserção dos clientes ou mesmo formas incompletas de explorar a árvore de busca, mas que conduza a bons resultados em termos de qualidade.

Este trabalho focou nas estruturas de vizinhança com melhor desempenho, uma vez que o propósito não era conseguir os melhores resultados para um problema específico e sim entender melhor a dinâmica de parâmetros na busca para aspectos conhecidos de meta-heurísticas conhecidas. Um parâmetro que defina qual vizinhança é empregada para determinado momento da busca pode ser alvo de um estudo mais elaborado e traz perspectivas de bons resultados. As vizinhanças aqui trabalhadas foram: inter-inserção (or-opt), intra-inserção (2-opt), intertroca-uma e 2-Opt*.

3.6 Resolvendo com Meta-heurísticas e Soluções Híbridas

Como já foi comentado anteriormente, diversas são as meta-heurísticas e para resolver o PRVJT, várias destas técnicas já foram empregadas. As principais meta-heurísticas usadas para este problema são apresentadas na sequência.

3.6.1 Algoritmos Evolutivos

Algoritmos evolutivos (AEs) são bastante aplicados na literatura ao PRVJT, dificilmente em algoritmos puros. Há algum tempo, o hibridismo com o uso de algoritmos evolutivos como estratégia de diversificação vem sendo adotado.

Fazendo uso de estratégias evolutivas, os trabalhos Gehring & Homberger (1999); Homberger & Gehring (2001, 2005) usam da técnica em uma primeira fase de seu algoritmo híbrido, nesta fase o número de rotas é minimizado usando uma estratégia evolutiva (μ, λ), na segunda fase a distância é minimizada com uma busca tabu. Thangiah (1995) descreve um algoritmo genético chamado GIDEON que realiza agrupamento dos clientes em setores. Cada setor define uma rota que é criada usando o algoritmo de inserção mais barata. Na sequência as rotas são melhoradas usando trocas-λ.

Estes processos são rodados iterativamente até a condição de parada por tempo ser alcançada. No algoritmo genético o cromossomo é um setor.

Tan et al. (2001) aplicam o algoritmo genético bagunçado (Goldberg et al., 1989) para o PRVJT. A solução é codificada com pares, identificação cliente e veículo. A população inicial é gerada aleatoriamente. A construção de novos indivíduos é baseado em dois operadores: *corta e une* ("cut and splice") e *seleção por limiar* ("thresholding selection"). O operador *corta e une*, similar ao crossover de um ponto, divide os clientes entre os veículos. Em seguida o roteamento é feito com a inserção mais barata e também a criação de novas rotas quando verificada infactibilidade. A *seleção por limiar* é então aplicada, preservando melhores indivíduos. Um algoritmo de reparação é então executado para corrigir clientes que possam aparecer mais de uma vez ou estarem faltando.

Para mais algoritmos evolutivos para o PRVJT, Bräysy et al. (2004a), realizam uma abrangente revisão.

3.6.2 Busca tabu

Busca tabu é uma das meta-heurísticas mais usadas para problemas de roteamento. Duas das mais conhecidas soluções na literatura para o PRVJT são os trabalhos de Gendreau et al. (1994) e Cordeau & Laporte (2001). Versões paralelas da BT para o mesmo problema foram criadas por Badeau et al. (1997). Alguns algoritmos mais refinados de busca tabu surgiram depois, como a busca granular de Toth & Vigo (2003), que restringe a vizinhança, chamada de granular, ao trabalhar com movimentos que introduzam somente "bons" elementos.

3.6.3 Têmpera Simulada

O algoritmo de Têmpera simulada se baseia em um processo metalúrgico, onde estados de baixa energia de sólidos são obtidos. O método foi criado por Metropolis et al. (1953) e usado em otimização por Kirkpatrick et al. (1983). Uma nova solução, x_n, é aceita sempre que $f(x_n) < f(x)$ (minimização), sendo x a solução corrente. Para $f(x_n) \geq f(x)$ ser aceita existe uma probabilidade $e^{\delta/T}$, onde $\delta = f(x_n) - f(x)$ e *T* é uma parâmetro de temperatura que cai, segundo uma função como $T(k) = \alpha T(k-1)$, para $0 \leq \alpha \leq 1$. Se definirmos um parâmetro determinístico K tal que $f(x_n) < f(x) + K$, x_n é aceito, esta adaptação é chamada de *TA- Threshold Acceptance*.

Para o PRVJT, o primeiro trabalho utilizando têmpera simulada foi a publicação de Chiang & Russell (1996).

3.6.4 Colônia de Formigas

O método colônia de formigas é inspirado na estratégia de busca das formigas na obtenção de alimento. Em sua busca por comida, as formigas marcam seu caminho liberando uma substância denominada feromônio. Essa substância influencia o comportamento das outras formigas, que dão preferência a trilhas com maior quantidade da substância. Caminhos menores, tendem a apresentar para outras formigas mais feromônio, uma vez que a formiga que o percorre retorna mais rapidamente ao formigueiro. Como o feromônio evapora com o tempo, este ciclo faz com que haja uma tendência para redução de caminho. No algoritmo, formigas artificiais trabalham cooperativamente, comunicando-se apenas através do feromônio deixado pelo caminho.

Cada formiga a cada iteração do algoritmo constrói uma solução para o problema de roteamento, fazendo uso de uma "memória". Dado o nó corrente c a formiga pode escolher qualquer nó *d* vizinho ($d \in N(c)$) ainda não visitado no caminho. A escolha do nó se baseia em dois componentes: da heurística verifica-se a melhor aresta (c, d) com relação à função objetivo, medida h_{cd}. E do feromônio artificial verifica-se a atratividade da aresta, medida f_{cd}. As escolhas normalmente são sorteios baseados em probabilidades definidas por estas medidas. A cada escolha de aresta o feromônio pode ser atualizado ou não.

O método foi aplicado por Bullnheimer et al. (1997) para o PRV. Uma das melhores soluções para o PRVJT é o trabalho de Gambardella et al. (1999), onde duas colônias, uma para minimização de veículos e outra para minimização de distância são usadas.

3.6.5 Grasp

O GRASP (tradução, "procedimento de busca adaptativa aleatória gulosa") foi proposta por Feo & Resende (1995). O algoritmo funciona em duas fases, executadas por iteração: a primeira fase de construção na qual uma heurística construtiva gulosa é empregada para achar uma solução. Na sequência uma busca local, segunda fase, busca a melhora da solução achando um máximo ou mínimo local. Esse método foi adaptado ao PRVJT por Kontoravdis & Bard (1995).

3.6.6 Busca com Vizinhanças Variável

A busca em vizinhança variável (BVV) foi introduzida por Hansen & Mladenovic (1997). Ela explora o espaço de soluções com mudanças sistemáticas de estruturas de vizinhança. Dado uma solução corrente s e uma lista de estruturas de vizinhança $N_1(s), N_2(s), \ldots, N_n(s)$, a cada iteração na estrutura da vez, $N_i(s)$, é definida uma solução s, na qual é aplicada uma busca local. Se a solução produzida s' for melhor que a solução s, s' passa a ser a solução corrente e a estrutura de vizinhança corrente passar a ser a primeira. Caso contrário, não sendo melhor, segue a busca na lista de estruturas

de vizinhança, com $N_{i+1}(s)$. Bräysy (2003) construiu uma solução usando BVV para o PRVJT. Nesta solução que apresenta quatro fases, a busca com vizinhança variável é a última fase, nas primeiras busca-se redução de rotas e a BVV é aplicada para reduzir a distância total da solução.

3.6.7 Soluções Híbridas

Nos últimos anos, usuários e pesquisadores de meta-heurísticas têm aplicado soluções híbridas, buscando aproveitar o melhor de diferentes métodos e os resultados produzidos são melhores que dos algoritmos tradicionais.

Gehring & Homberger (1999) trabalharam uma solução híbrida com duas fases: a primeira com algoritmos evolutivos reduzindo o número de rotas e uma segunda fase reduzindo a distância total percorrida com uma busca tabu. Estratégia semelhante foi adotada por Homberger & Gehring (2001) aplicando melhorias ao algoritmo original. Bräysy (2003) criou uma solução usando busca com vizinhança variável, com um algoritmo e quatro fases: inicialização, eliminação de rotas, minimização de distância usando quatro heurísticas de busca local e modificação da função objetivo para escapar de mínimos locais. Na sequência de trabalho com soluções híbridas, Bräysy et al. (2004b) sugerem um método de busca local com múltiplos pontos iniciais em duas fases: geração da solução inicial com uma heurística rápida seguida de uma cadeia de ejeção para redução de rotas e uma segunda fase baseada em duas heurísticas de troca-cruzada para reduzir a distância percorrida. Os algoritmos supracitados obtiveram bons resultados para o PRVJT. Geralmente, intercalam-se meta-heurísticas com maior capacidade de diversificação com meta-heurísticas com mais forte característica de intensificação para realizar a hibridização. No caso do PRVJT, ainda intercalam-se meta-heurísticas para o objetivo de redução de rotas da solução com redução de distância total percorrida.

3.7 Paralelismo e Cooperação

Paralelismo e cooperação são técnicas para melhorar algoritmos que vêm sendo muito utilizadas ultimamente, principalmente pela maior disponibilidade de hardware que permite usufruir destas técnicas. O paralelismo busca ganhos de desempenho computacional e cooperação busca, geralmente, ganho de qualidade. É comum o uso de cooperação para trocas de soluções boas entre fases de algoritmos ou mesmo entre diferentes algoritmos. Na maioria das vezes a cooperação apresenta algum paralelismo. Um exemplo de algoritmo paralelo que faz uso com sucesso de cooperação para o PRVJT é o criado por Bouthillier & Crainic (2005), um método paralelo de busca múltipla cooperativa, usando "warehouse", que armazena as melhores soluções encontradas. Vários fluxos do algoritmo (buscas), trocam assincronamente informações sobre as melhores soluções identificadas.

Homberger & Gehring (2005) usam um algoritmo paralelo cooperativo em duas fases (paralelas): a primeira com estratégia evolutiva minimizando o número de rotas e a segunda fase buscando a minimização de distância com BT.

3.8 Resumo

Este capítulo introduziu o caso de estudo deste trabalho, o problema de roteamento de veículo com janela de tempo, com uma resumida porém abrangente revisão do problema. Foram apresentados vários problemas similares, formulação, representação e várias técnicas de resolução, mostrando o estado-da-arte e vários aspectos da resolução do problema.

No próximo capítulo, Cap. 4, será exposto o uso de *Sistemas Genéticos Nebulosos* (SGN) com objetivo de adaptação de parâmetros em meta-heurísticas, apresentando o estado-da-arte. A proposta de arquitetura e como o SNG foi projetado e modelado são então detalhados.

Capítulo 4

Sistemas Nebulosos Genéticos para Escolha de Parâmetros em Meta-heurísticas

4.1 Introdução

Este capítulo apresenta todos os aspectos do *Sistema Nebuloso Genético* utilizado para prover autonomia, adaptabilidade no uso de meta-heurísticas na resolução de problemas NP-difíceis e melhoria de qualidade das soluções encontradas. A principal característica da arquitetura proposta é o uso de um meta-AG que otimiza a base de conhecimento empregada na escolha de parâmetros de uma meta-heurística que resolve o problema principal. A base de conhecimento é responsável pelo estabelecimento de controle no processo de busca visando a um balanceamento da diversificação e intensificação ao longo do processo. Neste trabalho é encontrado o termo "controle de meta-heurística", embora este seja o jargão bem utilizado na literatura, o termo "adaptação da escolha de parâmetros de meta-heurística" especifica mais o contexto do problema aqui tratado e por isto foi adotado como título da dissertação, ambas formas são encontradas no texto apresentando mesmo significado.

Ao longo do trabalho, alguns passos foram definidos para o uso da arquitetura proposta. O primeiro passo da abordagem é resolver o problema principal com meta-heurísticas escolhidas, sem preocupação excessiva com especialização de operadores e estruturas de vizinhança mais sofisticadas. As meta-heurísticas *busca tabu* e *algoritmo genético* foram utilizadas aqui, como apresentado no Cap. 2. O segundo passo procura trabalhar uma base de regras que é utilizada na escolha de parâmetros da meta-heurística e possibilita balanceamento de diversificação e intensificação da estratégia de busca. No último passo é criado um meta-AG que otimiza a base de conhecimento buscando melhores resultados.

O capítulo prossegue, a partir deste ponto, com uma revisão da literatura para modelos que estabelecem algum tipo de controle com sistemas nebulosos. Na sequência são apresentados todos os

aspectos supracitados do modelo e arquitetura de solução.

4.2 Sistemas Nebulosos Genéticos em Escolha de Parâmetros

Sistemas nebulosos genéticos atraíram a atenção da comunidade de Inteligência Computacional desde início dos anos 90 com trabalhos como os de Karr (1991), Pham & Karaboga (1991),Thrift (1991) e Velenzuela-Rendom (1991). A hibridização da lógica nebulosa com algoritmos genéticos (AG) é útil para lidar com diferentes problemas da engenharia. Durante o final dos anos 90, sistemas nebulosos incorporaram capacidade de aprendizagem dos modelos de computação flexível. Duas amplas vertentes surgiram para prover esta capacidade de aprendizagem: Sistemas Neurais Nebulosos (Nauck et al., 1997; Jang et al., 1997), SNN e Sistemas Nebulosos Genéticos, SNG. Entre os SNGs, a abordagem mais proeminente é o uso de modelos com base de regras nebulosas, chamado de *Sistema Nebuloso Genético com Base de Regras* (SNGBR) como destacado por Cordón et al. (2001).

O processo de aprendizagem genético tem uma variedade de alternativas definidas nas estruturas de seu projeto ("design"). Diferentes níveis de otimização, variando dos parâmetros do sistemas nebulosos até sua base de regras foram adotados, dando para área uma taxonomia completa e própria como apresentado por Cordón et al. (2004). Todos estes estudos recentes posicionam SNG como uma das áreas de grande destaque na academia, como foi colocado por Herrera (2008). A área apresenta uma rápida trajetória de evolução, começando com refinamentos da base de conhecimento (BC) com diferentes estruturas de aprendizagem (Michigan, Pittsburgh, IRL) e seleção de regras, a área prosseguiu evoluindo com programação genética, sistemas genéticos nebulosos hierárquicos e co-evolutivos até os mais recentes estudos de problemas com grandes dimensões, aprendizagem evolutiva multi-objetivo, aprendizagem por reforço, SNGs que lidam com dados imprecisos, aprendizagem dos componentes de inferência e estudos do "trade-off" exatidão-interpretabilidade, para mais detalhes e uma revisão completa vide o trabalho de Herrera (2008).

Em algoritmos genéticos as primeiras formas de controle foram propostas por Lee & Takagi (1993), com o uso de um sistema com base de regras nebulosas (SBRN) para controle dinâmico, mais especificamente a taxa de recombinação e tamanho da população eram dinamicamente escolhidos durante execução do AG. Essa ideia tem sido usada em diversos AGs e alguns exemplos podem ser citados. Ah King et al. (2004) adotam um AG controlado por regras nebulosas para reconfigurar redes elétricas. Neste caso, taxas de recombinação e mutação mudam de acordo com valores de fitness médio de dois passos consecutivos do algoritmo. Em sistemas de transportes existe o trabalho recente de Lau et al. (2009), que usa um controlador nebuloso dinâmico para resolver o problema de roteamento de veículo multi-objetivo com múltiplos depósitos e produtos. O controle é exercido sobre a taxa de mutação e recombinação sendo a principal variável de entrada a variação do fitness.

Para busca tabu (BT) existe pouco trabalho relacionado ao controle da trajetória de busca no formato de controle que foi caracterizado no capítulo 2, já controle adaptativo utilizando sistemas nebulosos inexiste, na extensão de nosso conhecimento.

Um elemento chave de busca tabu são as estruturas de memórias especiais que servem para determinar as vizinhanças da busca local, logo organizam a maneira como o espaço de busca é explorado. Uma distinção importante na BT surge da diferenciação entre memória de curto prazo e memória de longo prazo. A memória de curto prazo mais comumente usada mantém os atributos de solução que foram alterados durante passado recente. Isto previne ciclagem por temporariamente proibir movimentos que retornariam a uma solução recentemente visitada. Em algumas aplicações, a memória de curto prazo da BT são suficientes para produzir soluções com grande qualidade.

Em geral, a BT melhora significativamente quando é incluída memória de longo prazo e estratégias relacionadas. Estas operam introduzindo penalidades e incentivos determinados pelo "tempo" (chamado prazo tabu ou tabu tenure) que atributos fazem parte das soluções visitadas durante a busca, permitindo uma diferenciação local. Portanto um assunto chave na busca tabu é como gerenciar as memórias de curto e longo prazo. Normalmente, a gerência de memória de uma BT é especificada pelo usuário e requer esforço, experimentação e experiência consideráveis. Um mecanismo comum para lidar com gerência de memória e mudar a direção da busca é o controle do tamanho da lista tabu como Tsubakitani & Evans (1998) apresentaram e obtiveram bons resultados para o problema do caixeiro viajante. A memória da BT é responsável pela capacidade de diversificação e intensificação no comportamento da busca e diferentes valores de parâmetros de memória produzem resultados com qualidades bem distintas o que configura boa perspectiva de bons resultados com mecanismos de controle.

O modelo criado aqui explora estes aspectos chaves destes dois tipo distintos de meta-heurísticas: uma meta-heurística baseada em população com grande força de diversificação, o algoritmo genético e uma meta-heurística baseada em uma única trajetória de busca e com grande capacidade de intensificação, a busca tabu.

4.3 Construção do Modelo

Os três passos da construção do modelo referidos na introdução, resolver o problema com uma meta-heurística, realizar a escolha de parâmetros com uma base de conhecimento e por fim otimizar a base de conhecimento, constituem uma abordagem que pode ser definida como "de baixo-para cima" e está retratada na Fig 4.1.

Fig. 4.1: Como o modelo é construído

4.3.1 Resolvendo o Problema de Otimização com Meta-heurística

Como colocado no capítulo 2 o primeiro passo para resolver um problema com meta-heurística é a definição da meta-heurística que será usada. A motivação neste ponto é buscar a meta-heurística que melhor se adeqúe ao problema e ao modelo de controle, simplificar e testar mais rápido as meta-heurísticas, sem muita especialização. É interessante portanto o uso de de meta-heurísticas com características de busca diferente, como busca tabu e algoritmo genético, têmpora simulada e estratégia evolutiva. Isto evidencia a busca pela não necessidade de conhecimento profundo do problema e das técnicas usadas para obter bons resultados. Este conhecimento mais apurado geralmente está ligado à escolha dos parâmetros e sofisticação de operadores e estruturas de busca. Certamente que quanto maior este conhecimento maiores as perspectivas de resultados melhores em menor tempo.

Para validar a acurácia e qualidade do algoritmo é importante ter benchmarks e/ou especialistas do problema. Uma vez que os algoritmos estão funcionando o próximo passo é começar usar conhecimento dentro dos pontos estratégicos das meta-heurísticas.

4.3.2 Realizando Escolha de Parâmetros com uma Base de Conhecimento

Um ponto-chave na escolha de parâmetros em uma meta-heurística é usar características explícitas que direcionam a estratégia da busca no balanceamento da intensificação e diversificação. Estas

características, também chamadas de evidências da busca, podem ser comuns entre meta-heurísticas e são geralmente convertidas em variáveis de entrada para a base de regras sendo construída. A forma como esta variáveis são medidas vai depender do algoritmo e também da representação do problema.

Como criar base de regras para meta-heurísticas

Evidências como reportado por Eiben et al. (2007) podem ser caracterizadas em evidências absolutas ou relativas. As evidências absolutas normalmente geram ações diretas em pontos específicos sendo disparadas por regras. Já evidências relativas se caracterizam por mensurar o parâmetro comparativamente, baseando-se para isto na qualidade da solução, como por exemplo, usando o fitness.

Um passo importante para conhecer melhor os parâmetros estratégicos e saber usá-los para obter melhores resultados é usar evidências de acordo com o objetivo que deseja cumprir e balancear a busca quanto à diversificação e intensificação. A tabela, Tab. 4.1, exemplifica estes conceitos para as meta-heurísticas trabalhadas aqui e está disposta da seguinte forma:

- Objetivo do Controle - ressalta a intenção de uso da evidência, aumentar ou reduzir a diversificação ou intensificação. Muitas vezes estes objetivos são dependentes. Reduzir a diversificação irá aumentar a intensificação diretamente, por exemplo.
- Algoritmo - diz para qual tipo de meta-heurística esta evidência está sendo definida e analisada, uma meta-heurística baseada em trajetória de busca ou baseada em população.
- Evidência - denominação direta da evidência.
- Exemplo de medição - indica um exemplo de como a evidência pode ser mensurada para o algoritmo em questão.
- O que mais impacta? - indica quais parâmetros do algoritmo podem traduzir e influenciar mais esta evidência.

Objetivo de Controle	**Algoritmo**	**Evidência**	**Exemplo de medição**	**O que mais impacta?**
Diversificação	Baseado em população (algoritmo genético)	Diversidade	Realizar medida de distância entre os indivíduos, como *medida de jaccard* ($\frac{\|A \cap B\|}{\|A \cup B\|}$, onde $\|\ \|$ é a cardinalidade).	Taxa de mutação
Diversificação	Baseado em um único ponto (busca tabu)	Diversidade	Realizar medida de distância entre soluções (seus atributos), com a *medida de jaccard*.	Prazo tabu, Tamanho lista, Uso de memória de longo prazo.
Intensificação	Busca local (pode estar dentro de qualquer meta-heurística)	Potencial de ganho	Medir o ganho com mudança de incumbente nas últimas x iterações.	Taxa de reprodução (Algoritmo genético), taxa de busca local (proporção de indivíduos a passar por uma busca local no AG). Número máximo de movimentos a ser executado por rodada da busca local.

Objetivo de Controle	Algoritmo	Evidência	Exemplo de medição	O que mais impacta?
Intensificação	Baseado em população (algoritmo genético)	Medida de Qualidade (relativa ao incumbente)	Medir a proporção fitness dos indivíduos com relação ao fitness do incumbente e definir a média da população.	Taxa de reprodução (Algoritmo genético), taxa de busca local (proporção de indivíduos a passar por uma busca local no AG). Diminuir a diversificação.

Tab. 4.1: Quadro exemplificando evidências

Tab. 4.2: Alguns Parâmetros estratégicos de meta-heurísticas

Meta-heurística	Aspecto que influenciam divesificação e intensificação da busca
Têmpera Simulada	critério de aceitação + agenda de resfriamento
Busca tabu	vizinhanças, tamanho lista tabu, prazo tabu, critério de aspiração
Algoritmos Evolutivos	recombinação, mutação, seleção
Colônias de Formiga	atualização do ferormônio, probabilidades de construção
Busca local iterativa	busca local, movimento inicial, critério de aceitação
Busca em vizinhança grande	busca local, movimento inicial, critério de aceitação, escolha de vizinhança
GRASP	busca local, lista de candidatos
Busca local guiada	função de penalização

As evidências da Tab. 4.1 representam experimentos realizados no trabalho. Cada evidência tenta capturar "dicas" ao longo da busca que possibilitem mudanças estratégicas. As evidências podem ser mais específicas para determinado problema quanto se queira, normalmente, ganhando acurácia, mas, resultando em perda de desempenho computacional e generalidade. Portanto, é importante balancear acurácia e esforço computacional, procurando não criar outro problema tão difícil quanto o problema original ao mensurar uma evidência. Uma vez definida que evidências serão usadas, define-se, consequentemente, variáveis de entrada para a base de regras.

As variáveis de saída estão diretamente ligadas à meta-heurística e seus aspectos que influenciam a estratégia de busca quanto à diversificação e intensificação. Do trabalho de Blum & Roli (2003) foi extraída a Tab. 4.2, que exemplifica aspectos que podem definir parâmetros de controle para meta-heurísticas, nas quais as técnicas aqui usadas podem ser empregadas.

Com as variáveis de entrada e saída definidas, o próximo passo é a criação das regras que farão a escolha dos parâmetros dentro da meta-heurística. Inicialmente é importante criar regras simples e verificar e validar o mecanismo de escolha de parâmetros. A figura do especialista no problema e especialista da técnica pode ajudar criar regras que produzam resultados melhores. Uma forma de validar é fazer um planejamento de testes, criando base de regras bem diferentes e a analisando os resultados. Geralmente, uma grande variação de resultado pode significar tanto que seu controle não é efetivo quanto um grande potencial de bons resultados, vai depender de como ocorreu a variação das regras frente à variação dos resultados obtidos. Se o comportamento do resultado traduz a interpretação e direção que as regras estão dando, o caminho seguido se mostra promissor.

4.3.3 Otimizando a Base de Conhecimento

A otimização da base de conhecimento é o último passo, é um passo bem desafiador, vale salientar, pois como foi descrito no início deste capítulo, criar um sistema nebuloso genético que evolua uma

base de conhecimento passa por uma série de definições e possibilidades. Foi utilizado para tal objetivo um meta-AG que codifica sistemas nebulosos, evoluindo-os. A arquitetura como um todo é apresentada na sequência.

Arquitetura do sistema nebuloso genético

Fig. 4.2 exibe a arquitetura do sistema nebuloso genético. Duas partes se destacam: (1) o AG que evolui a base de conhecimento (BC); (2) o sistema de base de regras nebulosas (SBRN), que usa a base de conhecimento definida em (1) para escolher os parâmetros dinamicamente. O SBRN é localizado dentro da meta-heurística para coordenar e balancear a estratégia da busca. A função de fitness em (1) é diretamente ligada à função objetivo da meta-heurística que resolve um problema específico em (2).

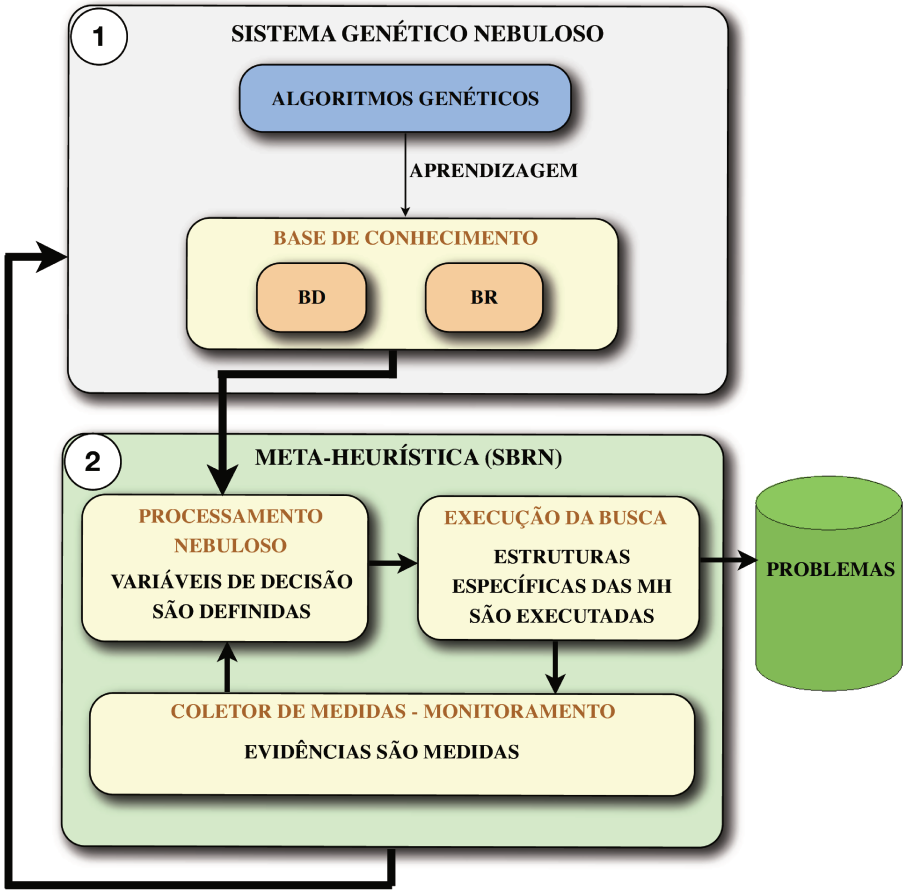

Fig. 4.2: SNG

Em mais detalhes, esta abordagem de arquitetura (Fig. 4.2), naturalmente permite duas configurações de execução: (1) e (2) podem estar conectados (modo *on-line*), onde a base de conhecimento definida em (1) é processada em (2) e (2) realimenta (1) com a avaliação da BC (valor da função

objetivo na execução da meta-heurística) ou pode estar em modo *off-line*, onde em (2), o SBRN funciona com uma base de conhecimento definida anteriormente, como por exemplo, uma base definida por conhecimento especialista ou mesmo uma base definida anteriormente em modo *on-line* e que foi guardada. Em (2), a meta-heurística usa um *pool* p, com P instâncias, para avaliar a base de conhecimento, P pode ser de uma instância até a totalidade do conjunto de instâncias. Quanto maior P melhor a generalização da base de conhecimento obtida, mas, mais caro é o procedimento em termos computacionais. Por outro lado, quanto menor P mais especializada é a base de regras, o que geralmente produz melhores resultados para as instâncias. No caso de mais de uma instância ($P > 1$), a avaliação do SNG usa a soma de valores das funções objetivos da execução da meta-heurística para definir o fitness. Esta explanação é enriquecida com o pseudocódigo, algoritmo 6.

Algoritmo 6 - Meta-heurística com SNG

Entrada: $p \leftarrow$ *pool* de P instâncias;
Entrada: $n \leftarrow$ número de gerações do SNG;
Entrada: $s \leftarrow$ tamanho da população no SNG;
Saída: $r_{best} =$ vetor da melhor solução conjunta das instâncias em p;{*menor soma de fitness*}
Saída: $kb_{best} =$ melhor base de conhecimento;

```
SNG:
i ← 0;
fitness_best ← 0;
g ← criaPopulacaoInicial();
enquanto i ≤ n ou tempo suficiente faça
    {Computa o fitness de cada indivíduo:}
    para cada base de conhecimento kb em g faça
        r, f ← Meta-heuristica_SBRN(kb, p, numIteracoes);
        fitness_kb ← defineFitnessDoVetorFuncaoObjetivo(f); {soma de fitness do pool}
        se fitness_kb > fitness_best então
            fitness_best ← fitness_kb;
            kb_best ← kb;
            r_best ← r;
        fim se
    fim para
    Aplica os operadores genéticos do SNG para formar a nova população g;
    i ← i + 1;
fim enquanto
```

No sistema nebuloso genético construído, somente a base de conhecimento (base de dados + base de regras) evolui, ou seja, a máquina de inferência nebulosa não é alterada.

No trabalho dois projetos de sistemas nebulosos genéticos foram criados, um mais tradicional com codificação binária e outro com codificação hierárquica mista. Na continuação do capítulo estes projetos são apresentados.

4.4 Projeto do Sistema Nebuloso Genético

Uma parte comum a qualquer projeto de sistema nebuloso genético é a definição das variáveis. Como abordado na seção anterior, esta definição esta diretamente ligada ao uso das evidências que são monitoradas na busca e são usadas para decidir as mudanças de direção.

A parte do projeto responsável pelo aprendizado, aqui da base de conhecimento e usando algoritmo genético, pode apresentar as mais diferentes variações. Dois algoritmos para evoluir a base de conhecimento são apresentados:

- AG com codificação binária da base de conhecimento;
- AG com codificação mista hierárquica da base de conhecimento.

4.4.1 Definição das Variáveis

As variáveis nebulosas no contexto de monitoramento da busca são medidas úteis para prover informações da meta-heurística ao longo da trajetória de busca.

A avaliação da trajetória da busca fica a cargo de um "coletor de medidas" que monitora a meta-heurística ao longo de suas iterações definindo os valores correntes das variáveis de entrada. Para o algoritmo genético, uma busca baseada em população, a medida das variáveis de entrada pode ser realizada em cada geração, como uma medida da população (média). Para busca tabu, baseado na trajetória de um ponto, as medidas devem ser realizadas em um passado recente (por exemplo as últimas x iterações).

As variáveis de entrada definidas devem portanto monitorar a busca fornecendo dicas para orientar o comportamento da busca, "mais diversificação", "mais intensificação". A escolha dos parâmetros (variáveis de saída) é definida por regras "**se** ... **então** ...", assim o controle sobre as variáveis de saída é exercido.

As evidências adotadas aqui são baseadas em três aspectos que são considerados como aspectos de influência e importantes para decidir sobre mudanças de comportamentos em busca:

- *Quão diversificadas (diferentes) estão as soluções ao longo da busca?* Diversidade da população entre gerações adjacentes para meta-heurísticas populacionais e diferença entre soluções de passado recente e solução atual para meta-heurísticas de trajetória única são indicativos de diversidade na busca.
- *Como a solução evoluiu recentemente?* Deseja-se saber aqui se a medida de qualidade evolui. Para o algoritmo genético pode ser usado o fitness médio e usar medidas relativas comparando

geração atual com gerações passadas. Para a busca tabu usa-se o valor da função objetivo atual com relação a um passado recente.

- *Qual momento da busca?* Saber se a busca está no início ou perto do fim é importante, por exemplo, para decidir em diversificar mais ou intensificar mais a busca.

As variáveis de entrada são derivadas diretamente destas três evidências *diversificação*, *evolução* (melhoria no tempo) e *momento da busca*. Quando o algoritmo genético e busca tabu forem detalhados para o problema de roteamento de veículo com janela de tempo, pormenores na definição destas medidas são apresentados.

4.4.2 Aprendizado da Base de Conhecimento - AG com Codificação Binária

Um algoritmo genético (AG) é usado para evoluir os componentes da base de conhecimento, ou seja, base de dados (BD) e base de regras nebulosas (BR). Foi adotada a abordagem de Pittsburgh para representação do cromossomo: a codificação binária representa toda base de conhecimento. Fig. 4.3 resume a estrutura do cromossomo.

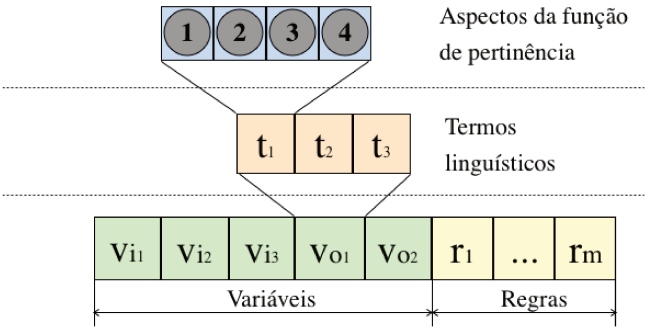

Fig. 4.3: Exemplo de estrutura do cromossomo

BD

A Fig. 4.4 exibe todos os aspectos da função de pertinência que constituem a base de dados. Eles são: (1) tipo da função de pertinência (trapezoidal ou triangular), (2) distância entre centro de funções de pertinência adjacentes, (3) distância de controle da inclinação da função, e (4) ponto de cruzamento entre funções de pertinência adjacentes. Alguns elementos da base de dados são relacionados, alterar o aspecto (2), altera o aspecto (4) ou (3). A factibilidade da cobertura, o mínimo e o máximo de cobertura, é garantida por uma função recursiva que altera os elementos (4), (3) e (2), nesta ordem. Estes aspectos estão retratados na Fig. 4.3.

Desenho da BR

A codificação binária da BR indica se uma regra faz parte ou não da base de regras nebulosa (BRN). Na Fig. 4.3, os bits r_1 à r_m definem esta codificação.

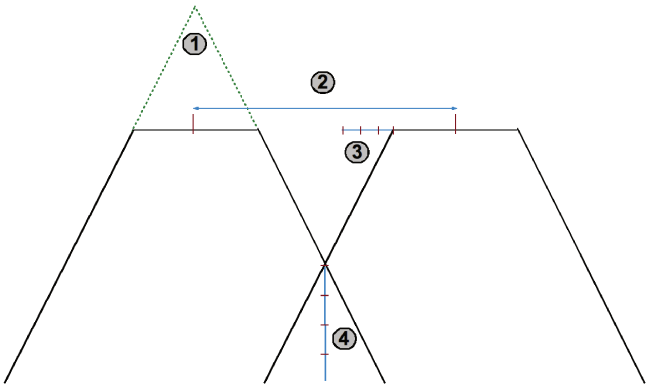

Fig. 4.4: Desenho da BD do SNG

Dimensionalidade do Espaço de Busca do Meta-AG

O espaço de busca visto pelo meta-AG, e o número de regras da BR dependem do número das variáveis de entrada e saída e suas respectivas granulações. Consequentemente a dimensionalidade deve ser tratada cuidadosamente. O comprimento do cromossomo depende tanto da BD quanto da BR. A BD tem a seguinte estrutura: um bit para o tipo da função de pertinência (aspecto 1), sete bits para distância entre centros (aspecto 2; esta escolha representa 2 casas decimais de precisão na discretização), dois bits para inclinação da função de pertinência (aspecto 3), e dois bits para ponto de cruzamento das funções (aspecto 4), totalizando doze bits para cada função de pertinências. Logo, considerando, por exemplo, quatro termos linguísticos (irrelevantes incluídos) para cada uma das três entradas e três termos linguísticas para cada uma das duas variáveis de saída, a dimensão da BD é de 216 bits ($3 \times 4 \times 12 \ + \ 2 \times 3 \times 12$). Neste caso, a BR tem cinco variáveis (três entradas e duas saídas) correspondendo a 384 ($4^3 \times 3 \times 2$) regras possíveis. Com isto, a representação completa requer um cromossomo com 600 bits. O espaço de busca enfrentado pelo AG é de grande dimensão e requer atenção, especialmente quando os SNG controla problemas complexos de otimização. Uma maneira para lidar com este problema é usar procedimentos de seleção de regras.

Seleção de regras

O propósito dos procedimentos de seleção de regras é reduzir o efeito da dimensão do espaço de busca do AG.

O procedimento de seleção de regras define as regras da base antes da execução do AG. Geralmente, este procedimento de seleção tenta mensurar quanto uma regra contribui para atingir um bom resultado. Por exemplo, em problemas de classificação como os tratado por Ishibuchi et al. (1995) e Casillas et al. (2001), a seleção de regras é realizada avaliando propriedades como consistência-k, completude-τ, frequência de regras, usando um *conjunto de dados de treino*. Abordagem similar pode ser usada no campo dos problemas de otimização, neste caso os dados de treino devem ser vistos como as instâncias para as quais resultados exatos são sabidos. Neste caso, medidas como *gap de otimalidade* pode ser usada para avaliar e comparar o desempenho das regras. Esta não é uma alternativa prática. Neste trabalho são usados valores do fitness e frequência das regras para avaliá-las e compará-las. O algoritmo de seleção de regras, algoritmo 7, usa medidas de desempenho para cada regra durante a execução do AG, e monta uma matriz M cujas entradas são valores de cada regra contra regras já presentes na base. A medida de desempenho é chamada *importância da regra* denotada por r_{ij}, a importância da regra i quando a regra j está na base de regras. Os valores r_{ij} são coletados na matriz M. Os valores ri das medidas de importância da regra i são a média do fitness das aparições da regra na base, isto é, a média das entradas da i-ésima linha de M.

O procedimento de seleção de regras usa uma seleção por rodada (execução completa do meta-AG), selecionando algumas regras por rodada até chegar ao conjunto final. O mecanismo é resumido abaixo. As entradas do procedimento são: tr, número total das regras selecionadas e rr, número de regras selecionadas por rodada.

Com este algoritmo é obtido um conjunto de regras com o qual o SNG consiga trabalhar e produzir bons resultados em um espaço de busca reduzido, cumprindo o objetivo do procedimento de seleção.

Os operadores genéticos

O algoritmo genético usa operadores binários clássicos para seleção, reprodução e mutação. A seleção é executada usando amostragem universal estocástica Baker (1987). Amostragem estocástica tenta reduzir as chances de repetir indivíduos de fitness alto na nova população, ou seja, reduz a pressão seletiva. Cruzamento em um ponto do cromossomo e mutação uniforme são usados mantendo o AG simples.

A função de fitness e escala

A função de fitness adotada pelo SNG está diretamente relacionada à função objetivo do problema de otimização. O AG usa também reescala de fitness, baseada na abordagem GENESIS 1.2ucsd $aval(x) = F - f(x),\ F > f(x)\ \forall\ x$) como sugerido em Michalewicz (1996). A reescala do fitness é usada para equilibrar a pressão seletiva evitando convergência prematura.

Algoritmo 7 - Seleção de regras

Entrada: tr, rr, p {*pool de instâncias do problema*}
Saída: RS {regras selecionadas}

1: $RT \leftarrow$ {todas as regras}
2: $RS \leftarrow \{\}$
3: **enquanto** $|RS| \leq tr$ **faça**
4: $M \leftarrow [\,]$, ($m \times m$, onde m= $|RT|$)
5: **para** cada iteração do AG **faça**
6: **para** cada cromossomo da população **faça**
7: calcula cada r_{ij} em M {*o valor do fitness médio do pool é distribuído para todas as regras presentes na base de conhecimento*}
8: **fim para**
9: **fim para**
10: **para** cada regra em *RT* **faça**
11: calcula ri (média da linha i de M)
12: **fim para**
13: ordena o conjunto de regras RT pelos valores ri
14: $RS \leftarrow RS \cup$ {seleciona as *rr* regras com maiores valores de *ri*}
15: $RT \leftarrow RT -$ {seleciona as *rr* regras com maiores valores de *ri*}
16: $RT \leftarrow RT -$ {seleciona as *rr* regras com menores valores de *ri*}
17: **fim enquanto**
18: **retorna** RS

4.4.3 Aprendizado da Base de Conhecimento - AG com Codificação Mista Hierárquica

O outro algoritmo genético utiliza uma codificação hierárquica. A motivação por trás do modelo hierárquico é não tratar partes diferentes de um sistema de maneira similar, como projetos tradicionais, mas considerar especificidades em diferentes níveis que se relacionam para evoluir a base de conhecimento como subcomponentes que se adaptam interativamente.

A abordagem utilizada é similar a Delgado et al. (2001), onde o projeto do sistema nebuloso implica em um processo evolucionário estruturado hierarquicamente em módulos que considera: a função de pertinência ou nível do conjunto de partição no primeiro nível, a população de regra individual no segundo nível, a população de conjunto de regras no terceiro nível e a população de sistemas nebulosos no quarto nível. A codificação considera tanto codificação real quanto inteira, como detalhado na sequência.

A codificação

Os cromossomos do nível de regras, conjunto de regras e sistemas nebulosos (nível II, III e IV, respectivamente) apresentam codificação inteira. O cromossomo do nível dos conjuntos de partição (nível I), possui codificação real. A relação entre os níveis pode ser visualizada na Fig. 4.5.

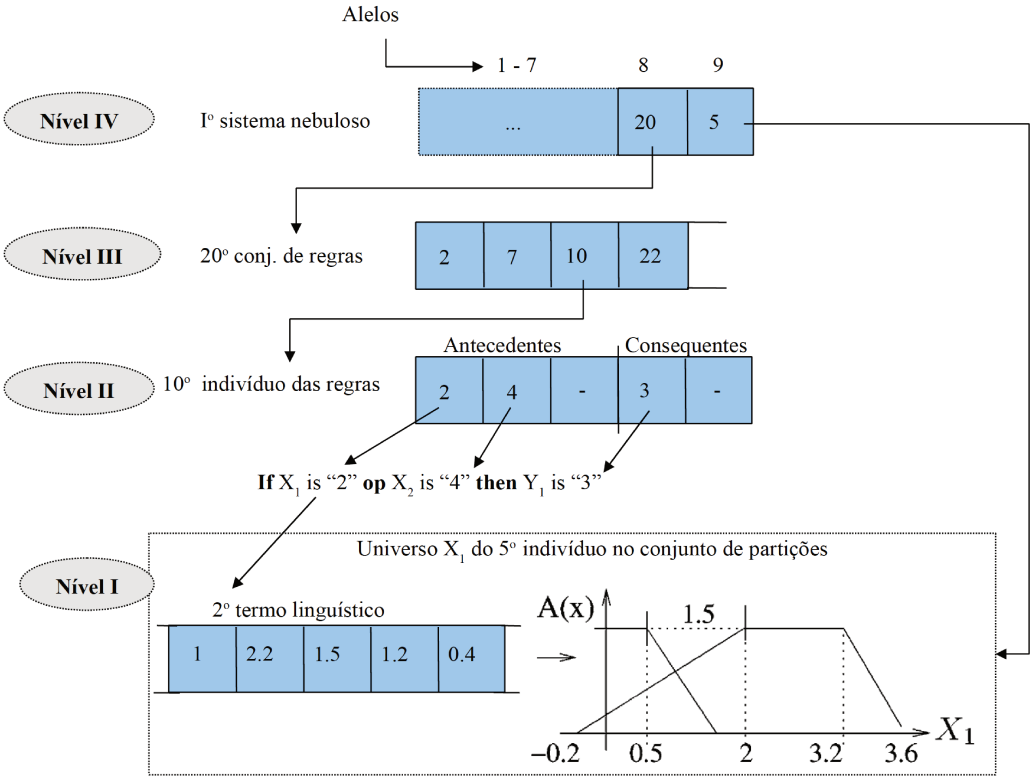

Fig. 4.5: Relação hierárquica entre os níveis da codificação

Um indivíduo no nível de conjunto de partição (nível I) contém todas as funções de pertinência definidas nos universos das variáveis usadas as quais são representadas em cromossomos com codificação real. O cromossomo é formado pela agregação na sequência dos conjuntos de partições associados às variáveis de entrada e saída. Cada membro da população de indivíduos regra (nível II em Fig. 4.5) representa uma proposição nebulosa (com antecedentes e conseqüentes). Esta população aceita diferentes combinações de função de pertinência identificadas pelo seu índice, isto é, a ordem no conjunto de partição (valores nulos indicam condições *irrelevantes*). Na população do nível de conjunto de regras (nível III), o indivíduo é formado por índices que identificam os indivíduos regras. O tamanho máximo do cromossomo define o número máximo de regras em uma base de conhecimento. Cada indivíduo no nível IV representa um sistema nebuloso. Neste nível, o código do cromossomo associa um conjunto de regras específico (alelo no locus 8) e um conjunto de partição (alelo no loci 9) e um subconjunto de alelos (loci 1-7) que definem a máquina de inferência. Neste

Tab. 4.3: Valores absolutos das funções de pertinência

Valor absoluto	como calcular
m_{1k}	$m_{2k-1} + C_{1k}$
m_{2k}	$m_{1k} + C_{2k}$
b_{1k}	$m_{1k} - L_k$
b_{2k}	$m_{2k} + R_k$
par função triangular: m_k	$\frac{(m_{2k}+m_{1k})}{2}$

trabalho somente a base de conhecimento foi evoluída, portanto estes alelos são fixados.

O i-ésimo sistema nebuloso (nível IV) na Fig. 4.5, usa, por exemplo, o 20^o conjunto de regras no nível III e o 5^o indivíduo conjunto de partição no nível I. O 10^o indivíduo regra é também exibido:

se X_1 é "2" **op** X_2 é "4" **então** Y_1 é "3",

onde o "2" (segundo), "4" (quarto) e "3"(terceiro) termos linguísticos são definidos no 5^o cromossomo do nível I para o universo das variáveis X_1, X_2 e Y_1, respectivamente.

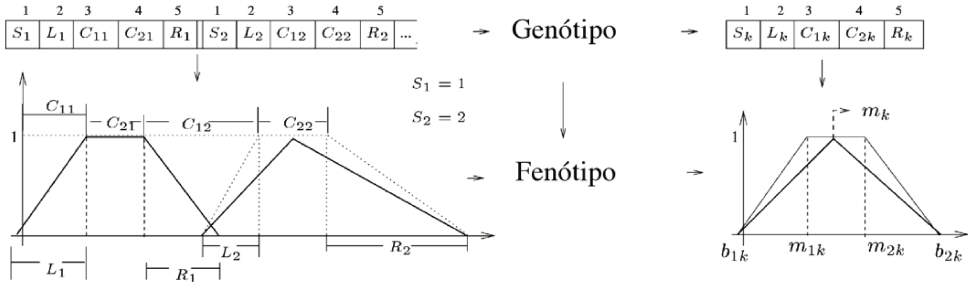

Fig. 4.6: A codificação no nível do conjunto de partições

A Fig. 4.6 detalha como os valores no nível de conjunto de partição são definidos. Os primeiros alelos S_k definem o tipo da função de pertinência: (1) triangular e (2) trapezoidal. Como a Fig. 4.6 ilustra, os alelos do loci 2-5 (L_k, C_{1k}, C_{2k}, R_k) determinam os valores relativos de uma função de pertinência com a imediatamente anterior (ou para o eixo das ordenadas no caso da primeira função de pertinência). Os valores absolutos podem ser calculados recursivamente conforme Tab. 4.3.

Do projeto supracitado o sistema nebuloso genético hierárquico é apresentado no algoritmo 8.

A fase de inicialização (passo 2) compreende a geração aleatória de uma população para cada nível. No nível de conjunto de partição (nível I), cromossomos com codificação real são gerados com partição uniforme das funções de pertinência nos universos das variáveis. No nível II, são geradas diferentes proposições nebulosas codificadas por cromossomos inteiros. Condições de irrelevância, representadas por valor nulo, são introduzidas quando regras gerais são aceitas. No nível III, cada cromossomo inteiro que representa um conjunto de regras nebulosas, é gerado aleatoriamente. Exclusões

Algoritmo 8 - SNG hierárquico

1. geração ← 1;
2. Inicializa população em cada nível;
3. Calcula o fitness para cada indivíduo de todos os níveis da seguinte forma:
3.1. Sistema nebuloso (nível IV): $fit_{SN}(i)$ é baseado no desempenho do sistema;
3.2. Base de regras (nível III): $fit_{BR(k)} = \max(fit_{SN(b)}, \ldots, fit_{SN(d)})$, onde b;...; d são os sistemas nebulosos dos quais a base de regras (k) faz parte.
3.3. Regra (nível II): $fit_{IR(j)} = media(fit_{BR(p)}, \ldots, fit_{BR(m)})$, onde p;...; m são as bases de regras das quais a regra (j) faz parte.
3.4. Conjunto de partições (nível I): $fit_{CP}(q) = \max(fit_{SN(x)}, \ldots, fit_{SN(z)})$, onde x;...; z são os sistemas nebulosos dos quais o conjunto de partições (q) faz parte.
4. Se não atingido o critério de parada, faça:
4.1. Do nível IV ao nível I aplica os operadores genéticos (seleção, reprodução e mutação) e forma a nova população;
4.2. geração ← geração + 1;
4.3. Retorna ao passo 3.

de regras são permitidas. A única restrição neste passo garante que todos os termos linguísticos iniciais das variáveis apareçam em um cromossomo. No nível IV, os alelos dos loci 8 e 9 são inicializados aleatoriamente, os demais são fixados e desconsiderados do processo.

O cálculo do fitness (passo 3) é realizado depois de definir todos os parâmetros dos sistema nebuloso. Este processo começa no nível IV indo até o nível I. No nível IV, o fitness é avaliado decodificando a representação do cromossomo e medindo a função de fitness que depende do desempenho de cada sistema nebuloso aplicado ao problema considerado. O fitness do nível III e I é computado com o máximo fitness dos sistemas nebulosos dos quais a base de regras e conjunto de partições faz parte, respectivamente. Isto é justificado porque se uma base de regras ou um conjunto de partições fazem parte de um "bom sistema nebuloso" (fitness elevado), seu fitness não deveria ser diminuído, punido por participar de alguns "sistemas nebulosos ruins". Então, o fitness neste caso é dado pelo fitness do melhor sistema nebuloso onde aparece. Para indivíduos representando regras, como há uma interação grande entre as regras, é usado a média, que faz mais sentido neste caso.

No passo 4, a condição de parada (máximo número de gerações) é verificada para continuidade ou não do algoritmo. Os operados genéticos em ordem são: seleção, reprodução e mutação, sendo aplicados do nível IV ao nível I.

Os operadores genéticos

Seleção é o primeiro operador genético aplicado, a tradicional método da roleta é usada para selecionar 80% dos indivíduos. Os 20% restantes são gerados por uma técnica que favoreça a diversidade da população. Este critério de diversidade escolhe os "mais diferentes" cromossomos com

relação àquele de maior fitness. A diferença é medida por distância euclidiana para codificações reais e número de alelos distintos para codificação inteira.

O segundo operador genético é o crossover de um ponto, onde o ponto é aleatoriamente escolhido. Mutação, o último operador, é aplicado em codificação inteira e real. Para codificação inteira, alelos com fitness mais baixo têm maior probabilidade de serem escolhidos para mutação e o novo valor é escolhido dentre todos possíveis. Para codificação real, mutação uniforme é usada por apresentar melhor desempenho.

No nível de partição, para garantir boa granularidade e manter uma sobreposição boa entre as funções de pertinência, duas condições de sobreposição devem ser atendidas:

- a completude-ϵ (Jang, 2002),
- grau máximo α de sobreposição (Jang, 2002).

Essas condições afirmam que, dado um valor x de algumas das entradas dentro da faixa do domínio, é possível sempre encontrar um termo linguístico A tal que $\mu_A(x) \geq \epsilon$, e um único termo linguístico B tal que $\mu_B(x) \geq \alpha$. Isto significa graus de sobreposição mínima (ϵ) e máxima (α) entre as funções de pertinência na evolução. Como cada alelo é relativo um ao outro, qualquer ponto pode ser escolhido para mutação e reprodução. A mutação uniforme modifica a forma e posição das funções de pertinência mas, a sobreposição mínima e máxima continuam respeitadas se assim o estavam. Para reprodução, se as condições de sobreposição passam a ser violadas, um procedimento de reparo, que desloca as funções de pertinência recursivamente enquanto necessário, é usado.

No segundo nível, reprodução e mutação são aplicadas para tentar diferentes combinações de termos linguísticos para cada proposição. Mutação muda o valor da posição escolhida com a escolha de um novo valor no conjunto $0; 1; \ldots; L_i$, onde L_i significa o número máximo de termos linguísticos para a i^{sima} variável de entrada.

No nível de base de regras, reprodução e mutação mudam os índices inteiros associados aos indivíduos regra. Valores nulos indicam eliminação de regra. Neste nível, alelos (que representam regras) com fitness mais baixo têm maior probabilidade de serem escolhidos para mutação. Os novos valores são escolhidos dentro do conjunto $0; 1; \ldots; S_{II}$, onde S_{II} significa o tamanho da população no nível II.

No nível dos sistemas nebulosos a mutação muda os alelos escolhendo o valor dentre todos valores possíveis para base de regras e o conjunto de partições. Para o alelo no locus 8 (base de regras) o valor novo fica dentro do conjunto $0; 1; \ldots; S_{III}$; e o alelo no locus 9 (conjunto de partição), tem novo valor definido em $0; 1; \ldots; S_I$, onde S_{III} e S_I são os tamanhos de população dos níveis III e I, respectivamente.

4.5 Resolvendo o PRVJT com Sistema Nebuloso Genético

Nesta seção é apresentado como o problema de roteamento de veículo é resolvido com as meta-heurísticas busca tabu e algoritmo genético e como dentro de cada meta-heurística o sistema nebuloso trabalha. Com isto o ciclo é fechado e um problema difícil é resolvido seguindo o modelo apresentado ao longo deste capítulo.

4.5.1 Busca Tabu para o PRVJT

Esta subseção detalha o uso da busca tabu para resolver o PRVJT e mostra como as variáveis nebulosas são instanciadas. Ainda é detalhado o papel do sistema de controle de memória de curto e longo prazo na trajetória de busca da BT.

Estruturas de vizinhança

A busca usa quatro estruturas de vizinhança, duas inter-rotas e duas intra-rotas. Estas escolhas se justificam permitindo que a busca alcance tanto redução de rotas quanto redução de distância. As vizinhanças são as inter-inserção, intra-inserção, intertroca-uma e 2-Opt* (Marques & Gomide, 2010). A Fig.4.7 ilustra como estas vizinhaças funcionam. Mais detalhes podem ser verificados na seção 3.5 do capítulo 3

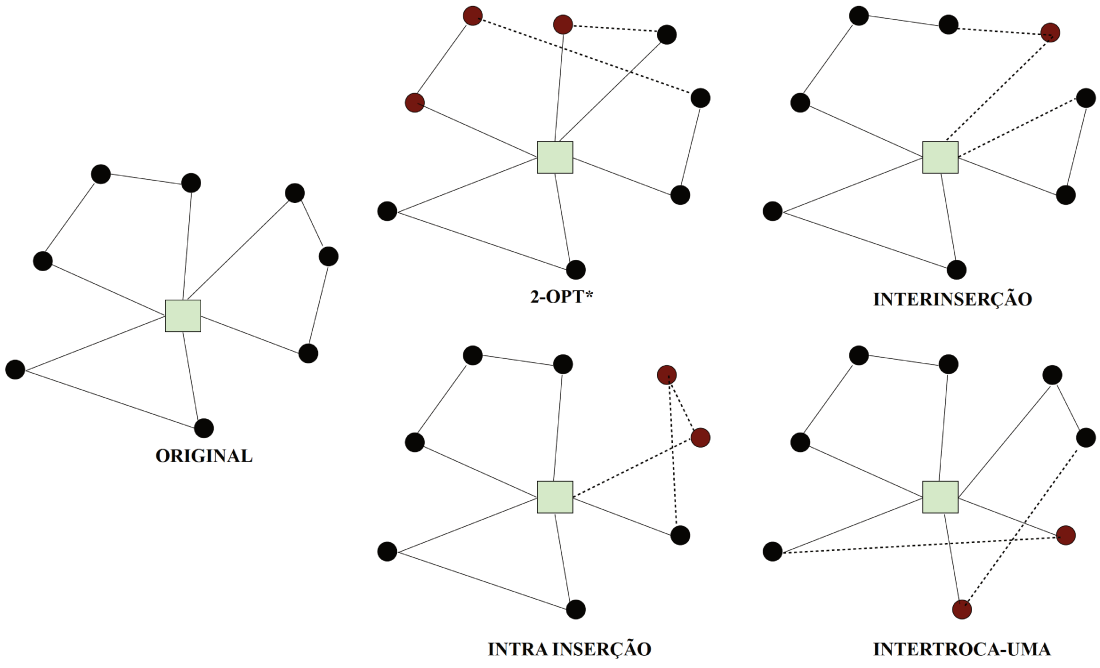

Fig. 4.7: Estruturas de vizinhança da busca tabu

Outro aspecto importante é como as estruturas de vizinhança são usadas. No trabalho, movimentos múltiplos foram usados por iteração da busca tabu. Isto pode ser feito sempre que os melhores movimentos alteram rotas distintas, podendo os movimentos ser executados simultaneamente. Esta estratégia traz ganhos de desempenho para o algoritmo. Nesta perspectiva de trabalho com as estruturas de vizinhança, a escolha da vizinhança pode ser uma variável de controle no sistema nebuloso. Aqui não foi estudado isto efetivamente, mas, existe um potencial neste aspecto.

Função objetivo

A função objetivo (fo) do problema (expressão 3.1 na seção 3.3 do capítulo 3) usa os custos das arestas de uma solução (qualquer rota da solução). No nosso algoritmo da BT, o custo das arestas inclui custos além das distâncias entre clientes. Por exemplo, sempre que um movimento reduz uma rota, um custo de movimento negativo é acrescido como incentivo de seleção (lembrando que é um problema de minimização). Esta forma de recompensa de movimentos pode ser usada para diversos propósitos tal como o uso de informação de memória de longo prazo para buscar mais diversificação da busca. Portanto, à função objetivo da seção 3.3 é acrescentado um termo de custo de incentivo ci, como ilustrado na equação abaixo:

$$fo = \min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} X_{ij}^k + ci_{ij} X_{ij}^k$$

Memória de curto prazo

A memória de curto prazo é o principal componente da memória da busca tabu. Ela usa uma lista tabu de inserção e uma lista tabu de remoção, ambas têm um parâmetro que define quantas rodadas da busca uma aresta permanece na lista. A lista tabu de inserção recebe arestas que estão sendo inseridas na solução. O parâmetro corresponde então a quantas rodadas as arestas permanecerão na solução na sequência da busca. A lista tabu de remoção recebe arestas removidas da solução corrente. Neste caso o parâmetro corresponde ao número de rodadas que as aresta removidas permanecerão fora da solução, caso movimentos tentem realizar reinserção.

Memória de longo prazo

A memória de longo prazo é o maior mecanismo para manipular a capacidade de diversificação da busca. Todas as frequências das arestas são guardadas pelo algoritmo. Cada movimento tem um conjunto de arestas para ser inseridas e um conjunto de arestas para ser removidas. As frequências de cada um destes conjuntos são somadas e a diferença destas somas é usada como incentivo de

diversificação. A estratégia do controle da memória de longo prazo usa um parâmetro que pondera o peso do incentivo (um exemplo é dado quando explicado as variáveis).

Critério de aspiração

O critério de aspiração usado neste trabalho é o mais comum: um movimento é permitido sempre que sua execução resulte em uma nova solução que seja melhor que a solução incumbente e seja uma solução inédita.

Variáveis do sistema nebuloso

As variáveis entradas definidas refletem as evidências apresentadas na seção 4.3.3. As variáveis de saída se concentram em aspectos da memória da busca tabu supracitados.

Variáveis de Entrada

As variáveis de entrada são medidas baseadas em histórico de 20 iterações e podem variar de 0 à 1 potencialmente.

- Taxa de melhoria (IR) é medida pela taxa:

$$IR = \frac{\text{média das melhorias da função objetivo no histórico recente}}{\text{melhoria máxima da função objetivo}}$$

- Diversidade (DIV) é baseado na medida de similaridade de *Jaccard*($\frac{|A \cap B|}{|A \cup B|}$), que é uma medida comparativa. Da média de duas medidas de Jaccard (med_{jac}): (solução atual contra incumbente) e (solução atual contra solução mais antiga do histórico), define-se DIV. Como jaccard é uma medida de similaridade, DIV portanto é definido como:

$$DIV = 1 - med_{jac}$$

- Momento da busca (MOM). O momento é definido pela taxa:

$$MOM = \frac{\text{iteração corrente}}{\text{máximo de iterações da busca}}$$

Variáveis de Saída

- SMP - parâmetro da memória de curto prazo (prazo tabu). Define o prazo tabu tanto para inserção quanto para remoção. Varia de 1 a 7.

- LMP - parâmetro da memória de longo prazo. Define como os movimentos da busca tabu impactam a função objetivo usando a memória de frequência de arestas nas soluções. Varia de 0 à 5. EX: Um movimento com custo $c = -10$ (ou seja melhora a solução) insere a aresta a_1 com frequência de aparição (f_{a_1}) de 5 e remove uma aresta a_2 com frequência de aparição (f_{a_2}) de 10 portanto o *custo de diversificação* ($\sum$(freqüências das arestas inseridas) $-$ $\sum$(freqüências das arestas removidas)) é (f_{a_1} - f_{a_2} = -5). Portanto este movimento introduz diversificação. Este parâmetro é exatamente o peso deste custo na composição geral do custo, ele pode então permitir que se desconsidere o custo, quando apresenta valor zero, ou até considerar com uma maior importância, quando o parâmetro vale cinco.

Exemplos de regras

Alguns exemplos de regra mostram como mudanças de direção podem ser introduzidas:

1. **se** IR é "muito baixa" **então** SMP é "alta";
2. **se** IR é "muito baixa" **então** LMP é "alta";
3. **se** DIV é "baixa" **e** IR é "baixa" **então** SMP é "muito alta";

Estas três regras exemplificam a mudança de direção de um momento de intensificação para um momento de diversificação. As duas primeiras regras mostram que se a taxa de melhoria (IR), analisando o histórico recente, é "muito baixa" então tanto o parâmetro da memória de curto prazo (SMP) e da memória de longo prazo (LMP) vão possuir valores "alta", que significa uma boa mudança e uma procura por diversificação. A terceira regra vai além, se a diversidade (DIV) é "baixa" e conjuntamente a taxa de melhoria é "baixa", uma mudança maior da busca por diversificação é definida na memória de curto prazo com SMP assumindo o valor "muito alta".

4.5.2 Algoritmo Genético para o PRVJT

Nesta subseção o algoritmo genético para resolver o PRVJT é apresentado, de maneira similar ao realizado com a busca tabu na subseção anterior.

Codificação

Um cromossomo representa a ordem dos clientes a serem visitados. Os clientes são colocados em sequência. O depósito é omitido, mas é considerado implicitamente. Fig. 4.8 mostra a estrutura do cromossomo. Note que o cromossomo é composto por subsequências, cada uma correspondendo à uma rota factível. A população inicial é gerada da seguinte maneira. O primeiro indivíduo é gerado utilizando *"push forward insertion heuristics"*, vide subseção 3.5.1. Os indivíduos restantes

são gerados das mesmas estruturas de vizinhança usadas na mutação. Diversidade populacional é garantida usando *medida de Jaccard* para evitar indivíduos similares.

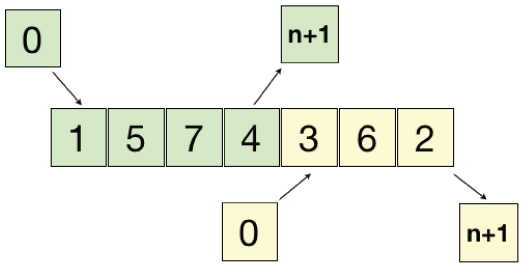

Fig. 4.8: Estrutura do cromossomo

Operadores genéticos

A *reprodução* é baseada no operador de reprodução por cópia de rotas (Fig.4.9). Este operador copia uma rota dos dois parentes para formar os filhos. Se um cliente não está em alguma rota, ele é inserido em alguma rota que permaneça factível. Se isto não for possível, novas rotas são formadas com estes clientes. O procedimento de *mutação* usa as mesmas estruturas de vizinhanças da busca tabu: *inter-inserção* , *2-opt**, *intra inserção* e *intertroca-uma*, respectivamente. Vizinhanças são escolhidas usando uma distribuição uniforme. A *seleção* é executada usando uma variação da roleta russa, chamado de procedimento de amostragem universal estocástica (Baker, 1987), que busca reduzir a chance de repetir indivíduos com alto fitness, na nova população.

Fitness

A função de fitness considera tanto distância total (dt) quanto número de veículos (nv) na solução:

$$fitness(dt, nv) = \left(p_d \times \frac{1}{dt}\right) + \left(p_v \times \frac{1}{nv}\right),$$

onde p_d e p_v são pesos com valores reais. O propósito do AG é maximizar o fitness, que acontece para os menores valores de dt e nv.

Busca local

O procedimento de busca local usa as mesmas estruturas de vizinhança da mutação. Ela sempre tenta melhorar o incumbente (mutação escolhe um movimento aleatório, que pode tanto melhorar quanto piorar a solução). A taxa da busca local define a probabilidade de um cromossomo dentro da população aplicar este operador.

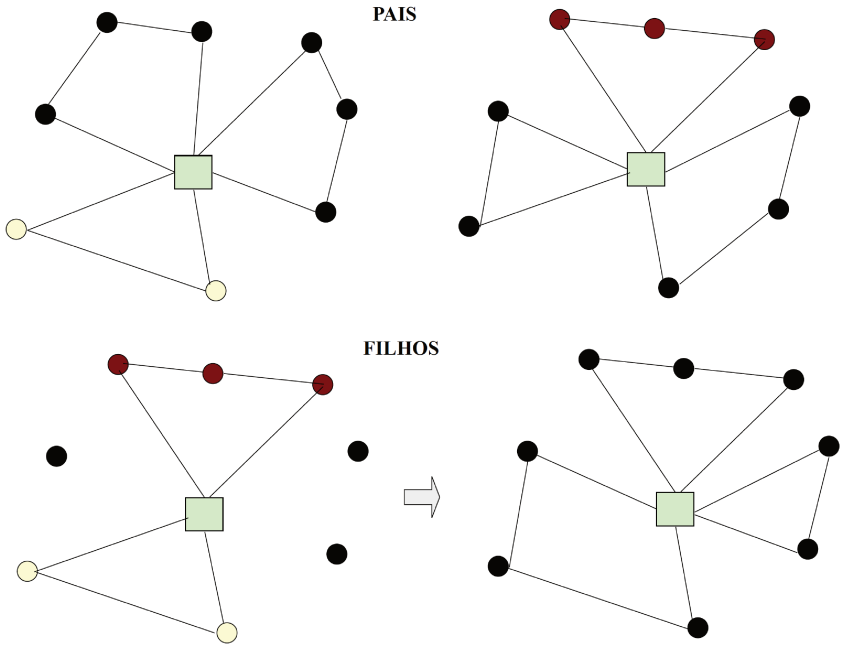

Fig. 4.9: Reprodução por Cópia de Rotas

Variáveis do sistema nebuloso

Como na busca tabu, as variáveis entradas definidas refletem as evidências apresentadas na seção 4.3.3. As variáveis de saída se concentram em aspectos tradicionais importantes, reprodução e mutação.

As variáveis de entrada são medidas baseadas em médias da população.

- Taxa de melhoria (IR) é medida pela taxa:

$$IR = \frac{\text{fitness médio da geração atual - fitness médio da geração anterior}}{\text{fitness médio da geração anterior}}$$

- Diversidade (DIV) é baseada na medida de similaridade de *Jaccard*. Da média das medidas de Jaccard da população, (med_{jac} - cada membro da população contra incumbente), define-se DIV. Como na busca tabu, DIV é definido como:

$$DIV = 1 - med_{jac}$$

- Momento da busca (MOM). O momento é definido pela taxa:

$$MOM = \frac{\text{geração corrente}}{\text{máximo de gerações do AG}}$$

Variáveis de Saída

- MR - taxa de mutação. Varia de 0.01 à 0.1.
- RR - taxa de reprodução. Varia de 0.4 à 0.9

Alguns exemplos de regras:

1. **se** IR é "muito baixa" **então** MR é "alta";
2. **se** IR é "muita alta" **então** MR é "baixa";
3. **se** IR é "muita alta" **então** RR é "alta";
4. **se** MOM é "início" **então** MR é "muita alta";
5. **se** MOM é "fim" **então** MR é "muito baixa";
6. **se** DIV é "início" **e** IR é "baixa" **então** MR é "muita alta";

Estas seis regras exemplificam alguns cenários de mudança de direção da busca. A regra (1) mostra que se a taxa de melhoria (IR) é "muito baixa" mudanças são introduzidas em um cenário de maior busca por diversificação com a taxa de mutação MR com valor "alta". As regras (2 e 3) mostram um cenário contrário, de aumento de intensificação; elas definem que se a taxa de melhoria (IR) é "muita alta" a taxa de mutação assume o valor "baixa" e a taxa de reprodução (RR) assume o valor "alta". As regras (4 e 5) trabalham com o momento da busca (MOM), se a busca está no "início" uma maior diversificação é buscada com a taxa de mutação "muita alta", acontecendo o contrário quando a busca está no "fim", neste caso não interessa reduzir a diversificação com a taxa de mutação com o valor "muito baixa". A última regra (6) exemplifica um uso da diversidade (DIV), se a diversidade é "baixa" e conjuntamente a taxa de melhoria é "baixa", uma mudança maior da busca por diversificação é definida com MR assumindo o valor "muita alta".

4.6 Resumo

Este capítulo introduziu uma revisão de controle em meta-heurística usando sistemas nebulosos genéticos e mostrou um pouco da evolução na área. Na sequência do capítulo foi destacada a abordagem "de baixo-para cima" : resolução de um problema com meta-heurística escolhida e a criação e otimização de uma base de regras. O capítulo é finalizado detalhando cada uma dessas etapas dentro da proposta. Todos os aspectos da arquitetura do sistema nebuloso genético proposto se encaixam nas duas últimas etapas.

No capítulo 5, os resultados experimentais obtidos são apresentados e analisados, assim como metodologia e definições dos algoritmos para alcançá-los.

Capítulo 5

Resultados experimentais

5.1 Introdução

Este capítulo destaca os principais resultados obtidos na pesquisa para a resolução do problema de roteamento de veículos com janela de tempo fazendo uso de sistemas nebulosos genéticos. O conteúdo aqui presente contempla: metodologia dos experimentos, definição inicial dos algoritmos, os resultados alcançados e as análises destes resultados.

5.2 Metodologia

Para testar o sistema nebuloso na adaptação de parâmetros de meta-heurísticas na resolução do PRVJT foram usadas as instâncias de Solomon (Solomon (1987a)), *benchmark* clássico para o PRVJT.

Estas instâncias são compostas por 3 classes, baseadas na geração e localização dos clientes:

- instâncias de clientes com localização aleatória - R (ramdom)
- instâncias de clientes agrupados - C (clustering)
- instâncias que misturam as duas anteriores - RC (random and clustering)

Foram usadas as maiores instâncias de Solomon que apresentam 100 clientes. Além das instâncias de Solomon, foi usado um conjunto de instâncias maiores, instâncias do *benchmark* Gehring & Homberger (G & H), apresentadas no trabalho Gehring & Homberger (1999). Neste trabalho as instâncias são apresentadas em classes similares a Solomon, mas com 200, 400, 600, 800 e 1000 clientes. Aqui foi usado o conjunto com maior número de clientes, 1000.

A Tab. 5.1, resume o uso das instâncias:

Tab. 5.1: Instâncias usadas

Conjunto de instâncias	Propósito de uso	Rótulo
56 instâncias de Solomon de 100 clientes	usado na evolução da base de conhecimento	T1
60 instâncias de G & H de 1000 clientes	usado para testar base conhecimento em grandes instâncias	T2

Os algoritmos foram implementados em Java e a biblioteca para processamento nebuloso JFuzzyLogic (JFuzzyLogic). Os modelos nebulosos usaram o procedimento de inferência clássico (max-min com defuzificação por centro de gravidade).

No sistema nebuloso as variáveis de entrada ficam na faixa 0 a 1 ou -1 a 1. No entanto, muitas vezes certos valores não são trabalhados na execução da meta-heurística. Com isto os domínios podem ser reduzidos, proporcionando redução do espaço de busca no algoritmo genético que evolui a base de conhecimento. Para tal foi adotada uma metodologia simples que definiu tanto o domínio quanto a granularidade máxima das variáveis de entradas:

- Executa a meta-heurística sem nenhum mecanismo de controle (sem a base de conhecimento), mas variando as variáveis controladas (variáveis de saída), independentemente, dentro de sua faixa seguindo um degrau de mudança (indo e voltando os valores subindo unitariamente). É usado para isto o conjunto de instâncias T1;
- São apurados os valores mínimo e máximo para cada variável de entrada e a variação a longo da execuções com a aplicação das mudanças nas variáveis de saída;
- dado os resultados supracitados, o domínio para uma variável de entrada é definido como: $[\max(0\ ou\ -1, \text{mínimo}\ -10\%)\ ;\ \min(\text{máximo}\ +\ 10\%; 1)]$;
- a granularidade da variável é definida por: $\max(10, \frac{\text{valor máximo no domínio - valor mínimo do domínio}}{\text{variação média apurada}})$.

Este procedimento simples antes da execução permitiu redução no espaço de busca, aumentando a eficiência dos meta-AGs que evoluem a base de conhecimento.

5.3 Definição Inicial dos Algoritmos

A definição inicial dos algoritmos está em Tabs. 5.2, 5.3, 5.4, onde são mostrados respectivamente, os parâmetros do SNG que evolui a base de conhecimento, o AG que usa a base de conhecimento para resolver o PRVJT e a busca tabu que usa a base de conhecimento para resolver o PRVJT.

Tab. 5.2: Parâmetros finais do SNG

Parâmetro	Valor definido
taxa de mutação	0.05
taxa de reprodução	0.9
números de gerações	100
tamanho da população	40

Tab. 5.3: Parâmetros do AG

Parâmetro	Valor definido
taxa de mutação	controlado ou 0.05
taxa de reprodução	controlado ou 0.9
taxa da busca local	0.25
número de gerações	$2 \times$ número de clientes
tamanho da população	100

Tab. 5.4: Parâmetros da Busca Tabu

Parâmetro	Valor definido
número de iterações	$8 \times$ número de clientes
prazo tabu	controlado ou 3
peso do custo da memória de longo prazo	controlado ou 1

Todos os valores fixados foram definidos baseados em testes empíricos simples, testando individualmente cada parâmetro, ou seja, não houve cruzamento de valores ou uso de mecanismos de refinamento. No SNG, os conjuntos de valores testados foram os seguintes: a *taxa de mutação* (0,025; 0,05; 0,075; 0,1), a *taxa de reprodução* (0,6;0,7;0,8; 0,9), o *número de gerações* (50; 100; 150; 200), o *tamanho da população* (30; 40; 50; 60). Na BT os valores testados foram para o *número de iterações* e foi baseado no fato de quanto maior o número de clientes maior o espaço de busca (exponencialmente maior), foi estabelecida, pelo menos, uma dependência linear entre *número de iterações* e *número de clientes*. A constante 8 foi um compromisso *convergência vs. tempo*. No AG os valores são: a *taxa de mutação* (0,025; 0,05; 0,075), a *taxa de reprodução* (0,6; 0,7; 0,8; 0,9), o *tamanho da população* (50; 100; 150). O *número de gerações* foi definida de forma similar ao número de iterações da busca tabu.

5.4 Resultados Obtidos e Análises

Os resultados dos experimentos reportados resumem primeiro a escolha de qual algoritmo dentro do sistema nebuloso foi adotado: codificação binária tradicional ou codificação mista hierárquica. Com a codificação do AG definida foi usado o mesmo conjunto de teste T1 para evoluir as bases de conhecimento e comparar os algoritmos, tanto busca tabu quanto algoritmo genético, em suas versões que utilizam a base de regras, ou seja, possui um sistema nebuloso por base de regras embutido na meta-heurística, contra as versões sem a base de regras. Ainda foi realizado um teste para verificar como esta base de conhecimento evoluída para o conjunto de teste T1 pode ser usada para resolver o conjunto de teste T2, usado o SNG no modo *off-line*, deste modo foi analisado como bases de

conhecimento evoluídas com instâncias médias podem ser aproveitadas para instâncias grandes. Por fim, foi feito uma validação estatística dos resultados experimentais.

5.4.1 AG com Codificação Hierárquica Mista vs. AG com Codificação Binária

Para analisar as codificações dos algoritmos genéticos foi usada a meta-heurística busca tabu. Em termos de desempenho computacional o AG com codificação hierárquica mista se mostrou para o conjunto de instâncias T1 aproximadamente 10% mais lento do que o AG com codificação binária. Em termos de qualidade de solução no entanto o AG com codificação hierárquica mista se mostrou melhor. Isto pode ser percebido no gráficos que mostram a evolução do fitness ao longo das gerações para algumas instâncias do conjunto de teste T1 (Figs. 5.1, 5.2, 5.3, 5.4). Nestes gráficos são mostrados o fitness médio da população e o fitness máximo dentro da população.

Fig. 5.1: Exemplo de execução do AG com codificação binária (instância r101)

Os gráficos exemplificam que a codificação hierárquica mista resulta em maior qualidade, com um fitness médio final em um patamar um pouco maior na convergência que se mostra um pouco mais rápida, o que poderia ser usado para reduzir o número de iterações e melhorar o desempenho.

Fig. 5.2: Exemplo de execução do AG com codificação hierárquica mista (instância r101)

Uma questão interessante da codificação hierárquica é a dificuldade de ajustar os níveis, por conta de suas interdependências via fitness. Em alguns testes realizados, grande variação de resultado e no tempo de convergência são causados quando o tamanho da população varia muito entre níveis adjacentes. Os melhores resultados foram alcançados tornando os níveis adjacentes mais uniformes quanto a tamanho populacional, não foram realizadas investigações mais detalhadas nesta questão, mas foi um aspecto importante percebido.

Com estes resultados, em todo o restante dos procedimentos experimentais foi adotado o algoritmo genético com codificação hierárquica mista para evoluir a base de conhecimento.

Ao longo das execuções das meta-heurísticas foram definido gráficos para verificar a evolução dos parâmetros, os gráficos (Figs. 5.5, 5.6, 5.7, 5.8) exemplificam como os parâmetros mudam na execução do SBRN tanto para o algoritmo genético quanto para a busca tabu. Cada gráfico mostra as 5 varíaveis durante a execução (3 variáveis de entrada: DIV, IR, MOM; 2 variáveis de saída: TMP, LMP para BT e RR, MR para o AG). O gráfico é exibido em dois momentos diferentes da execução do SNG: um na primeira iteração do SNG e outro na última iteração do SNG, isto é, o primeiro usa uma base de conhecimento inicial (gerada aleatoriamente) e o último uma base de conhecimento

Fig. 5.3: Exemplo de execução do AG com codificação binária (instância c101)

evoluída (passou por todo algoritmo, BC final). Não é objetivo aqui, avaliar a interpretabilidade da base de conhecimento e como as bases de regras são relativamente grandes (entre 30 e 50 regras), as bases de conhecimento não são apresentadas (no capítulo apêndice A colocamos alguns exemplos de base de conhecimento trabalhada). Os gráficos mostram como os parâmetros mudam respeitando os intervalos definidos. Outra característica apontada é a diferença entre algoritmos baseados em população e algoritmos baseados em trajetória. A busca tabu tem suas variáveis definidas na história recente da busca, últimas 20 iterações. Já as variáveis no algoritmo genético são baseadas nas médias na população a cada geração, o que permite um maior responsividade a mudanças na entrada, ou seja, mudanças de direção mais frequentes configurando uma maior flexibilidade da busca.

Nos experimentos que prosseguem, além de comparar os algoritmos com SNG e sem o SNG, eles foram comparados contra os melhores resultados reportados na literatura para o conjunto de instâncias supracitados, T1 e T2. Os algoritmos deste trabalho estão dispostos na Tab. 5.5 e os resultados da literatura na Tab 5.6.

Tab. 5.7 e Tab. 5.8 exibem os resultados para as instâncias de Solomon (conjunto T1) e G & H (conjunto T2), respectivamente. As tabelas apresentam os resultados pelo rótulo definido para cada

Fig. 5.4: Exemplo de execução do AG com codificação hierárquica mista (instância c101)

Tab. 5.5: Algoritmos deste Trabalho

Algoritmo	Rótulo
Busca tabu com SNG	GTS
Busca tabu sem SNG	PTS
Algoritmo Genético com SNG	GGA
Algoritmo Genético com SNG	PGA

Tab. 5.6: Resultados da Literatura

Resultado	Rótulo
Rochat & Taillard (1995)	RT
Gehring & Homberger (1999)	GH99
Cordeau & Laporte (2001)	CLM
Bräysy et al. (2004b)	BHD
Homberger & Gehring (2001)	GH01
Bräysy (2003)	B01
Homberger & Gehring (2005)	GH05
Bouthillier & Crainic (2005)	LC
Prescott-Gagnon et al. (2009)	PGDR

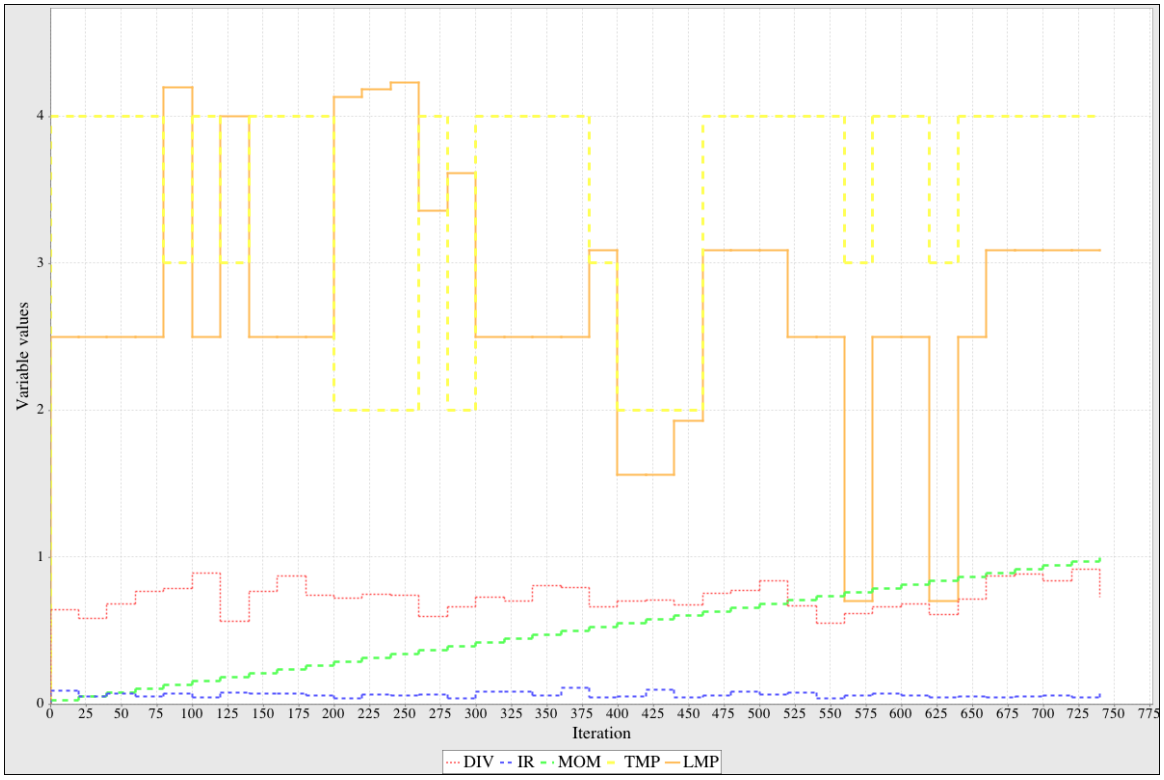

Fig. 5.5: Evolução dos Parâmetros na BT com BC Inicial

algoritmo/resultado da literatura. O resultado é apresentado como a média do número de veículos e da distância percorrida para cada grupo de instâncias dentro do conjunto mostrado. Ainda é apresentado o número de veículos acumulados (NVA) e a distância total acumulada (DTA) para todas instâncias do grupo. No final (última coluna) é mostrado o arcabouço experimental (máquina de execução, número de processadores, tempo de CPU) para o algoritmo/resultado da literatura (representado em cada linha das tabelas). As tabelas estão ordenadas decrescentemente de acordo com o número de veículos acumulados.

Os algoritmos com o SNG deste trabalho apresentam os tempos computacionais de acordo com o modo de trabalho: tempo modo *on-line* e/ou tempo modo *off-line*. Foram trabalhados os dois modos no caso do conjunto T1, ou somente o modo *off-line*, no caso do conjunto teste T2. Portanto, por exemplo, na Tab. 5.7, o algoritmo *GTS* apresenta tempo computacional médio de 45 minutos para evoluir a base de conhecimento (trabalho *on-line*) e 0,17 minuto para executar em modo *off-line* com uma base de conhecimento estabelecida. Como pode ser visto, estes tempos computacionais se mostram competitivos com os reportados na literatura, vendo ordem de grandeza.

O conjunto de teste T1 (resultados Tab. 5.7) usa somente uma instância no *pool* de instâncias do SBRN (meta-heurística com $P = 1$), ou seja, a base de conhecimento é evoluida para cada instância, mas, para criar as BCs para serem usadas com o conjunto de teste T2, cada tipo de grupo de instâncias

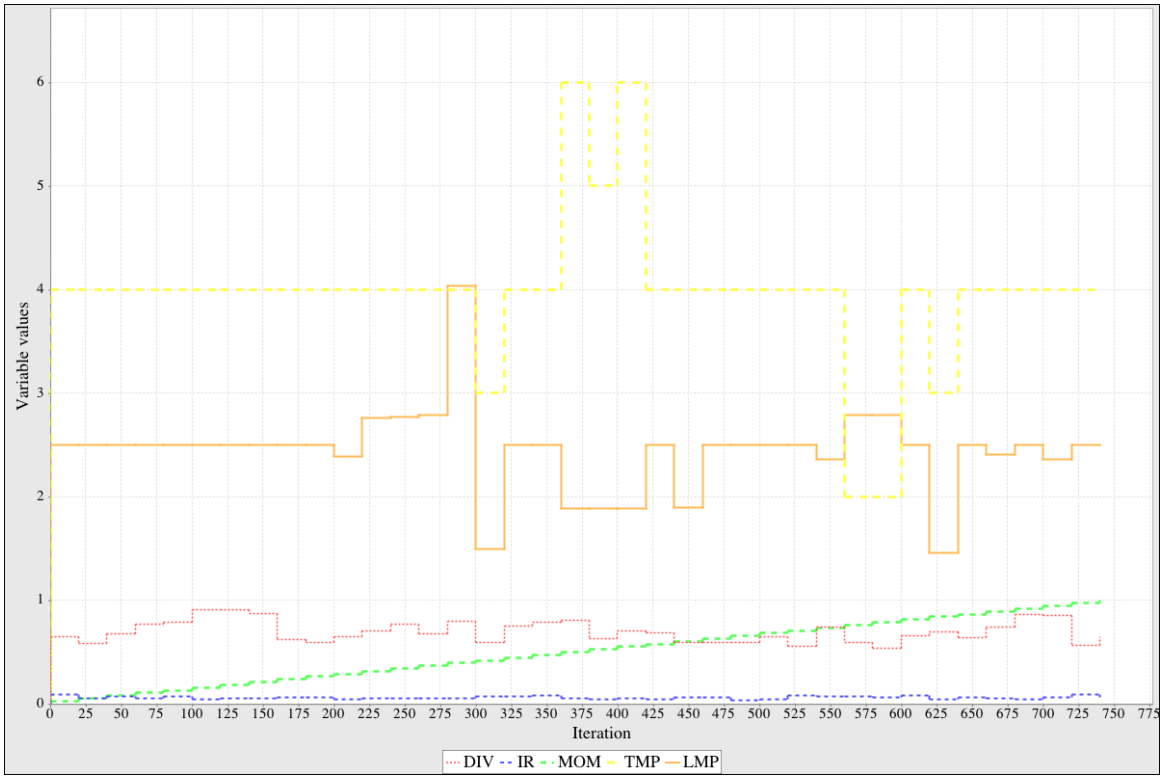

Fig. 5.6: Evolução dos Parâmetros na BT com BC Final

constituem um *pool*, isto é, um *pool* para instâncias do tipo R, um *pool* para instâncias tipo C e um *pool* para instâncias tipo RC, logo três bases de conhecimentos são criadas. Cada uma delas é usada para executar o mesmo tipo de instância no conjunto de teste T2.

Nos resultados do conjunto de instâncias T1, a busca tabu padrão (*PTS*) apresentou $6,67\%$ mais veículos que o melhor resultado da literatura (GH05) para o número de veículos acumulados (NVA) e $1,6\%$ a mais para distância total acumulada (DTA). A busca tabu com SNG (*GTS*) melhorou o NVA em $3,93\%$ e a DTA em $2,42\%$ na média, quando comparado ao *PTS*. Quando comparado ao melhor resultado da literatura, o *GTS* apresenta $2,46\%$ a mais em NVA e $0,88\%$ menos em DTA na média.

Para o AG quando comparado o algoritmo padrão (*PGA*) contra o melhor resultado da literatura apresentado (GH05), na média há $8,64\%$ a mais de veículos para NVA e $12,76\%$ a mais de distância para a DTA. Para o AG com SNG houve uma melhora (redução) do número de veículos em NVA de $4,12\%$ e uma redução da DTA em $8,08\%$ na média, quando comparado contra o AG padrão (*PGA*). Quando comparado ao melhor resultado da literatura apresentado, o *GGA* apresenta $3,45\%$ mais veículos para NVA e mais $3,49\%$ para DTA na média.

É possível perceber destas comparações que mesmo com as meta-heurísticas não sofrendo grandes sofisticações para especializá-las e buscar melhores resultados para o problema de roteamento de veículos com janela de tempo e ainda sem uso de refinamento no SNG, resultados competitivos foram

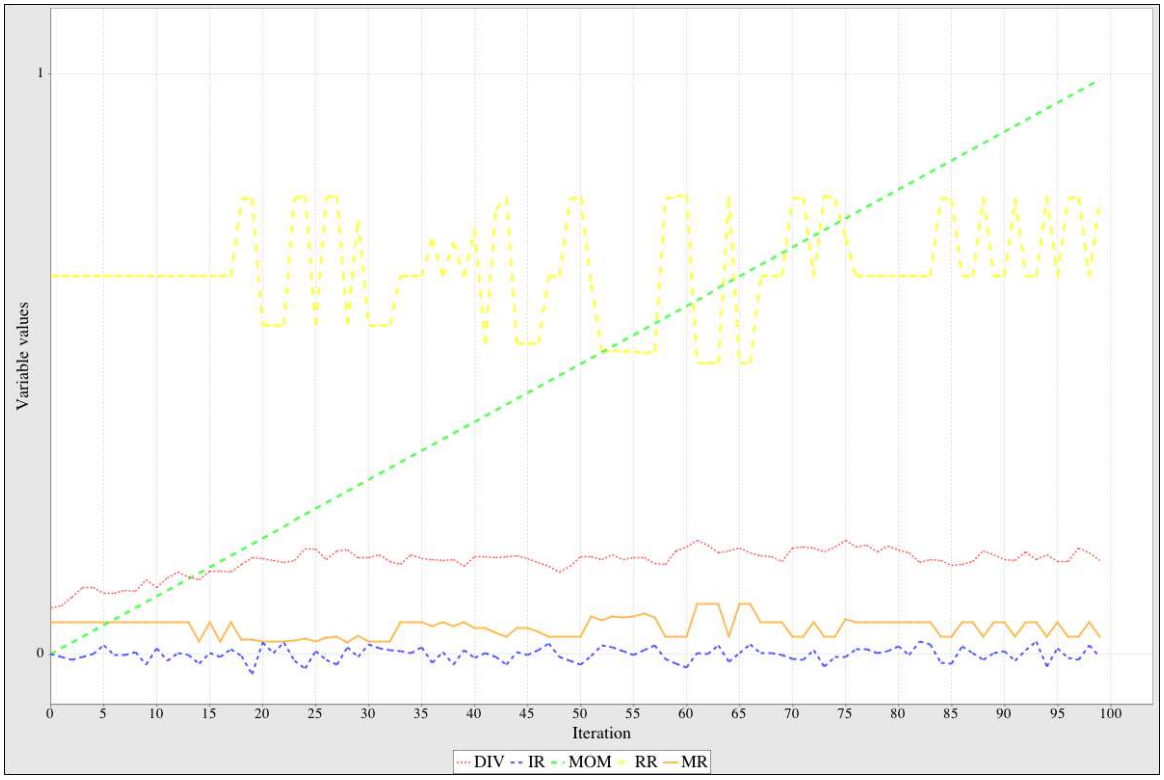

Fig. 5.7: Evolução dos Parâmetros no AG com BC Inicial

alcançados. Em ambas meta-heurísticas (busca tabu e algoritmo genético) embutir um mecanismo de adaptação de parâmetros aumentou a qualidade da solução, dentro de um processo autônomo e que possibilita redução de tempo gasto com escolha manual de parâmetros. Para o algoritmo genético esta melhora foi mais significativa. Isto ocorre por dois motivos: o algoritmo genético apresenta piores resultados para o PRVJT comparativamente a busca tabu e o segundo motivo é o fato do AG ser um algoritmo populacional, onde mecanismos de adaptação podem causar maiores impactos em termos de mudança de estratégia ao longo da busca, não só pelo trabalho com mais soluções, mas, também pelo fato do AG trabalhar dentro de um histórico mais curto, possuindo assim mais flexibilidade dentro da busca. Isto, provavelmente, é o que motiva mais estudos de escolha de parâmetros em algoritmos evolutivos do que em algoritmos não populacionais.

Para o conjunto de teste T2 (G & H com 1000 clientes), foram usadas as bases de conhecimento evoluídas na execução com T1, portanto a operação é *off-line*. As bases de conhecimento são usadas dentro da mesma classe (por exemplo, bases evoluídas com instâncias da classe R são usados em instâncias da classe R) e são escolhidas de forma aleatória. Comparando os resultados, a busca tabu com SNG (*GTS*) melhorou (reduziu) o NVA em $1,71\%$ e a DTA em $2,05\%$ na média, quando comparado ao *PTS*. Quando comparado ao melhor resultado da literatura (PGDR), o *GTS* apresenta $1,79\%$ a mais em NVA e $5,74\%$ a mais em DTA na média. Quando comparado com o melhor

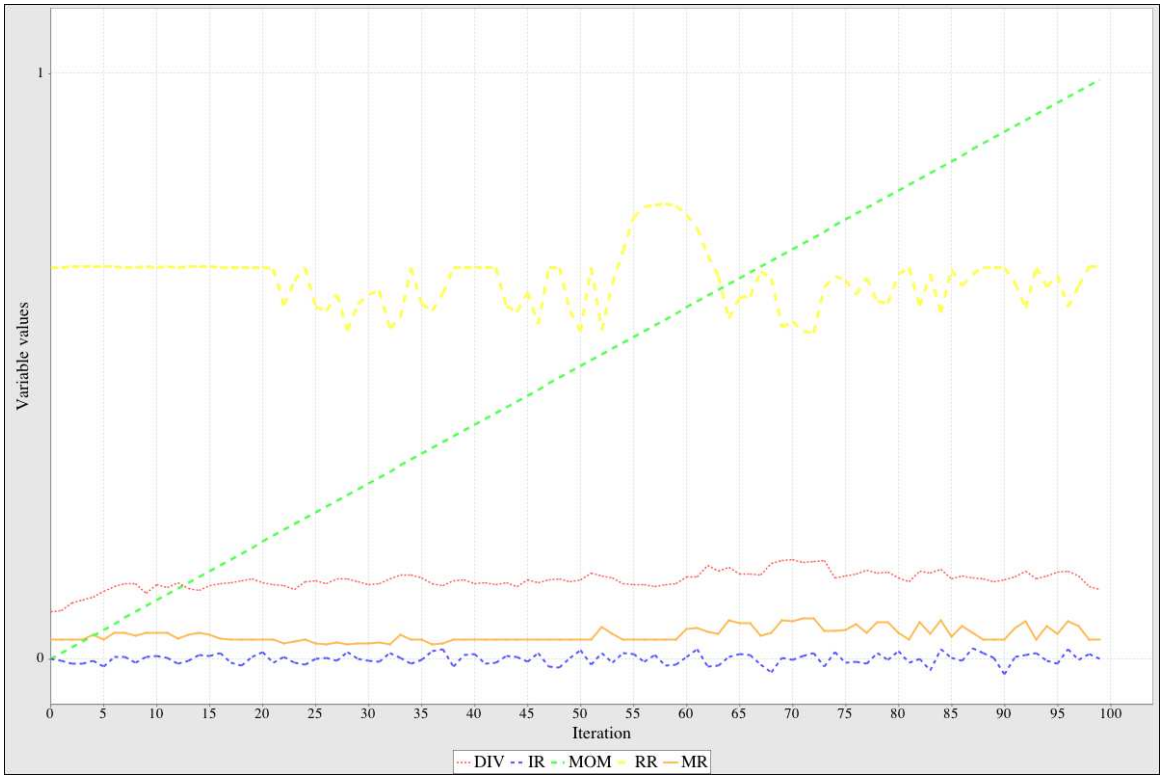

Fig. 5.8: Evolução dos Parâmetros no AG com BC Final

resultado da literatura (GH05) para conjunto de teste T1, o algoritmo *GTS* apresenta $1,18\%$ a mais em NVA e $2,49\%$ a menos em DTA na média.

Para o AG com SNG houve uma melhora (redução) do número de veículos em NVA de $2,01\%$ e uma redução da DTA em $2,29\%$ na média, quando comparado contra o AG padrão (*PGA*). Quando comparado a melhor resultado da literatura apresentado (PGDR), o *GGA* apresenta $2,71\%$ mais veículos para NVA e mais $7,04\%$ para DTA na média. Quando comparado ao melhor resultado da literatura (GH05) para conjunto de teste T1, o algoritmo *GGA* apresenta $2,08\%$ a mais em NVA e $1,31\%$ a menos em DTA na média.

O importante destes resultados para o conjunto de instâncias T2 é que ele corrobora com os resultados do conjunto de teste T1, mostrando que a abordagem pode trazer ganhos para instâncias grandes trabalhando em modo off-line.

Esse conjunto de testes T2 continuou a apresentar ganhos do algoritmo padrão com SNG com relação ao algoritmo padrão. Além disto, a algoritmo genético mantém o maior ganho com a presença do SNG quando comparado à busca tabu, mas neste caso foi inferior à diferença percebida com o conjunto de teste T1.

Ainda, neste conjunto de testes foi usado o algoritmo com método exato (PGDR) para comparação. Ele apresenta um tempo computacional maior, mas, proporciona um grande ganho de qualidade

Tab. 5.7: Comparação da média para o conjunto de teste T1

Autor	R1	R2	C1	C2	RC1	RC2	NVA/DTA	Experimento
PGA	13,16	3,18	10,00	3,00	12,875	3,375	437	P2,4G
	1299,77	1125,06	955,99	657,54	1516,82	1304,02	64404	10 execs., 0,2min
PTS	12,83	3,18	10,00	3,00	12,75	3,37	432	P2,4G
	1213,28	973,90	852,91	596,20	1412,87	1138,65	58130	1 exec., 0,15min
GGA	12,58	2,82	10,00	3,00	12,12	3,25	419	P2,4G
	1226,55	1038,61	845,64	596,53	1413,97	1168,06	59182	10 execs., 55/0,23min
RT	12,25	2,91	10,00	3,00	11,88	3,38	415	SG100M
	1208,50	961,72	828,38	589,86	1377,39	1117,44	57231	1 exec., 92,2min
GH99	12,42	2,82	10,00	3,00	11,88	3,25	415	4xP200M
	1198	947	829	590	1356	1144	56946	1 exec., 5min
GTS	12,33	2,82	10,00	3,00	12,00	3,25	**415**	P2,4G
	1207,78	958,93	828,46	589,86	1362,74	1072,49	**56698**	10 execs.,45/0,17min
CLM	12,08	2,73	10,00	3,00	11,50	3,25	407	Sun U2 300M
	1210,14	969,57	828,38	589,86	1389,78	1134,52	57555	n/a
LC	12,08	2,73	10,00	3,00	11,50	3,25	407	5xP850M
	1209,19	963,62	828,38	589,86	1389,22	1143,70	57412	1 exec., 12min
BHD	12,00	2,73	10,00	3,00	11,50	3,25	407	AMD 700M
	1239,77	1016,86	832,06	590,68	1417,79	1199,95	59210	1 exec., 2,6min
GH01	12,00	2,73	10,00	3,00	11,50	3,25	406	4xP400M
	1217,57	961,29	828,63	590,33	1395,13	1139,37	57641	5 execs., 13,5min
B01	11,92	2,73	10,00	3,00	11,50	3,25	405	P200M
	1222,12	975,12	828,38	589,86	1389,58	1128,39	57710	1 exec., 82,5min
GH05	11,92	2,73	10,00	3,00	11,50	3,25	**405**	P400M
	1212,73	955,03	828,38	589,38	1386,44	1108,52	**57192**	5 execs., 17,26min

com relação as resultados anteriores da literatura. Comparando o algoritmo *GTS* e *GGA* com o GH05, reportado para as instâncias de Solomon (T1), foram obtidos valores maiores para NVA, mas, menores para DTA, similar ao que tinha sido obtido para o conjunto de teste T1.

5.4.2 Testes Estatísticos dos Resultados

Os resultados alcançados foram testados para verificar a relevância estatística. Os conjuntos de testes (T1 e T2) foram usados como população e o algoritmo usado constitui o fator dos experimentos estatísticos. A mesma população é submetida a diferentes tratamentos (os algoritmos que são comparados), portanto se trata de experimentos com grupos casados. A variável dependente usada para avaliar as amostras é uma medida usando *número de veículos (nv)* e *distância total (dt)* para cada sujeito, uma instância do conjunto de teste. Em mais detalhes, a medida de qualidade (q) é definida como: $q = nv \times dt$. Dado que o PRVJT é um problema de minimização, quanto menor o valor de q melhor é o resultado. Aqui, não é assumida a normalidade das amostras de dados. No entanto, os

Tab. 5.8: Comparação da média para o conjunto de teste T2

Autor	R1	R2	C1	C2	RC1	RC2	NVA/DTA	Experimento
PGA	94,76	19,48	99,12	34,24	92,74	20,02	3603	P2.4G
	55453,55	34156,25	44045,20	18630,80	51518,65	27358,28	2311627	3 execs.,79min
PTS	93,83	19,27	97,52	33,77	91,83	19,84	3560	P2.4G
	54696,13	33711,75	43323,29	18376,67	50669,22	27036,26	2278133	1 exec.,62min
GGA	92,90	19,10	96,70	33,60	91,10	19,70	3531,00	P2.4G
	54101,02	33420,99	42970,93	18283,42	50262,10	26835,00	2258734	3 execs., 84min
GTS	92,11	19,00	95,97	33,10	90,28	19,50	**3499**	P2.4G
	53019,00	33220,46	42584,19	17990,89	49759,48	26566,65	**2231406**	3 execs., 65min
BHD	92,10	19,00	95,80	30,60	90,00	19,00	3465	AMD 700M
	50025,64	31458,23	42086,77	17035,88	46736,92	25994,12	2133376	1 exec.,39,6min
GH99	91,90	19,00	96,00	30,20	90,00	19,00	3461	4xP200M
	57186	31930	43273	17570	50668	27012	2276390	1 exec., 50min
GH05	91,90	19,00	96,10	29.90	90.10	18.90	3459	P400M
	57072,15	32320,68	43524,95	17566,99	51337,25	27059,89	2288819	3 execs.,20.6min
GH01	91,90	19,00	95,40	29,70	90,10	18,50	3446	4xP400M
	58069,61	31873,62	43392,59	17574,72	50950,14	27175,98	2290367	5 execs.,30,1min
PGDR	91,90	19,00	94,50	29,56	90,00	18,82	**3438**	O2.3G
	50168,00	30730,35	41913,42	16817,76	45924,74	25464,40	**2110187**	5 exec., 162min

conjuntos de testes têm mais de 25 elementos, 56 para o conjunto de teste T1 de Solomon e 60 para o conjunto de teste T2 de G & H, o que define as amostras como grandes, ajudando assim a aumentar a chance desta escolha ser adequada. Dadas estas características descritas acima, o teste *Wilcoxon* foi escolhido para os experimentos Lehmann (1986). O projeto do experimento é mostrado Tab. 5.9 e o conjunto de hipóteses testadas são colocadas na Tab. 5.10. Para não estender muito o texto, somente a amostra usada para o teste de uma hipótese (H_1, Tab. 5.11) é exemplificada. De uma tabela de distribuição gaussiana, o valor crítico de z para 0,05 de grau de significância para um teste direcional é 1,645, então se z é menor que -1,645 (*one-tailed* para $<$) no teste *Wilcoxon*, a hipótese nula pode ser rejeitada. Todos resultados dos testes de *Wilcoxon* e suas interpretações estão na Tab. 5.12.

Tab. 5.9: Projeto de Experimento

Algoritmo 1	Algoritmo 2
medida q	medida q

Instância	***PTS***			***PGA***		
	NV	**DT**	q $(NV \times DT)$	**NV**	**DT**	q $(NV \times DT)$
r101	19,000	1655,387	31452,358	19,000	1715,295	32590,605
r102	18,000	1473,649	26525,674	18,000	1495,909	26926,358
r103	14,000	1229,058	17206,810	14,000	1356,864	18996,095

r104	10,000	1054,579	10545,786	11,000	1126,099	12387,084
r105	14,000	1390,324	19464,536	15,000	1448,782	21731,731
r106	12,000	1266,112	15193,350	13,000	1297,082	16862,064
r107	11,000	1094,544	12039,986	12,000	1191,764	14301,171
r108	10,000	989,840	9898,398	11,000	1143,846	12582,301
r109	12,000	1186,890	14242,679	13,000	1288,193	16746,515
r110	12,000	1128,091	13537,094	12,000	1264,654	15175,853
r111	11,000	1115,751	12273,259	12,000	1135,861	13630,338
r112	10,000	979,705	9797,052	10,000	1048,753	10487,529
c101	10,000	828,937	8289,369	10,000	828,937	8289,369
c102	10,000	842,700	8427,003	10,000	1000,895	10008,955
c103	10,000	839,545	8395,454	10,000	964,597	9645,972
c104	10,000	856,967	8569,671	10,000	1090,423	10904,232
c105	10,000	828,937	8289,369	10,000	837,619	8376,194
c106	10,000	828,937	8289,369	10,000	828,937	8289,369
c107	10,000	828,937	8289,369	10,000	1011,594	10115,943
c108	10,000	828,937	8289,369	11,000	905,032	9955,348
c109	10,000	828,937	8289,369	10,000	1093,126	10931,255
rc101	15,000	1639,162	24587,423	15,000	1710,629	25659,442
rc102	13,000	1532,715	19925,295	14,000	1578,671	22101,387
rc103	12,000	1352,275	16227,301	12,000	1473,104	17677,251
rc104	11,000	1203,719	13240,910	11,000	1290,743	14198,174
rc105	15,000	1534,542	23018,131	15,000	1808,880	27133,203
rc106	13,000	1421,891	18484,582	13,000	1590,476	20676,182
rc107	11,000	1332,484	14657,324	12,000	1371,326	16455,910
rc108	11,000	1182,992	13012,915	11,000	1253,052	13783,570
r201	4,000	1300,937	5203,746	4,000	1458,749	5834,995
r202	4,000	1121,368	4485,474	4,000	1256,633	5026,531
r203	3,000	1012,197	3036,591	3,000	1094,172	3282,516
r204	3,000	809,790	2429,371	3,000	995,193	2985,579
r205	3,000	1065,047	3195,141	3,000	1176,319	3528,957
r206	3,000	961,508	2884,524	3,000	1110,857	3332,570
r207	3,000	889,583	2668,749	3,000	1113,594	3340,782
r208	2,000	747,420	1494,841	2,000	889,346	1778,693
r209	4,000	911,369	3645,475	3,000	1168,858	3506,574

r210	3,000	1033,818	3101,454	3,000	1194,956	3584,867
r211	3,000	823,402	2470,206	3,000	988,426	2965,278
c201	3,000	591,557	1774,670	3,000	623,570	1870,711
c202	3,000	591,557	1774,670	3,000	619,159	1857,477
c203	3,000	591,173	1773,520	3,000	747,138	2241,414
c204	3,000	626,099	1878,296	3,000	742,984	2228,953
c205	3,000	588,876	1766,628	3,000	588,876	1766,628
c206	3,000	588,493	1765,479	3,000	731,149	2193,448
c207	3,000	591,732	1775,196	3,000	596,313	1788,940
c208	3,000	588,324	1764,971	3,000	593,421	1780,264
rc201	5,000	1415,152	7075,762	4,000	1686,344	6745,375
rc202	4,000	1173,030	4692,118	4,000	1436,880	5747,518
rc203	3,000	1172,407	3517,222	4,000	1225,730	4902,921
rc204	3,000	893,268	2679,804	3,000	1043,339	3130,018
rc205	4,000	1365,378	5461,514	4,000	1675,887	6703,547
rc206	3,000	1295,054	3885,163	4,000	1376,645	5506,580
rc207	4,000	1026,250	4104,998	3,000	1311,324	3933,972
rc208	3,000	932,537	2797,610	3,000	1090,609	3271,826
média			**8813,614**			**9847,435**

Tab. 5.11: Amostra dos dados para o teste de H_1

5.5 Resumo

Este capítulo apresentou toda metodologia dos experimentos realizados, definição inicial dos algoritmos propostos e todos o resultados experimentais obtidos. Os resultados mostram que o uso de sistemas nebulosos genéticos proporciona ganho de qualidade nas meta-heurísticas trabalhadas e a arquitetura usada permite ganho de escala. No próximo capítulo o trabalho é concluído e itens para desenvolvimento futuro são apresentados.

Tab. 5.10: Definição das hipóteses

Hipóteses	**Descrição da hipótese alternativa**
$H_0 : \mu_1 = \mu_2$ $H_1 : \mu_1 < \mu_2$	o algoritmo de busca tabu padrão(*PTS*) é melhor do que o algoritmo genético padrão (*PGA*)
$H_0 : \mu_1 = \mu_2$ $H_2 : \mu_1 < \mu_2$	o algoritmo de busca tabu com SNG(*GTS*) é melhor do que o algoritmo de busca tabu padrão (*PTS*)
$H_0 : \mu_1 = \mu_2$ $H_3 : \mu_1 < \mu_2$	o algoritmo genético com SNG(*GGA*) é melhor do que o algoritmo genético padrão (*PGA*)
$H_0 : \mu_1 = \mu_2$ $H_4 : \mu_1 < \mu_2$	o algoritmo de busca tabu com SNG(*GTS*) é melhor do que o algoritmo de busca tabu padrão (*PTS*) para grandes instâncias (1000 clientes) em modo *off-line*
$H_0 : \mu_1 = \mu_2$ $H_5 : \mu_1 < \mu_2$	o algoritmo genético com SNG(*GGA*) é melhor do que o algoritmo genético padrão (*PGA*) para grandes instâncias (1000 clientes) em modo *off-line*

Tab. 5.12: Resultados dos testes estatísticos de Wilcoxon

Resultado do teste Wilcoxon	**Interpretação**
$p_1 = 8,758 \times 10^{-10}$ $z_1 = -6,131$	z_1 é menor do que -1,645, logo a hipótese nula é rejeitada e a hipótese alternativa H_1 é verificada
$p_2 = 7,616 \times 10^{-9}$ $z_2 = -5,777$	z_2 é menor do que -1,645, logo a hipótese nula é rejeitada e a hipótese alternativa H_2 é verificada
$p_3 = 9,894 \times 10^{-11}$ $z_3 = -6,469$	z_3 é menor do que -1,645, logo a hipótese nula é rejeitada e a hipótese alternativa H_3 é verificada
$p_4 = 1,630 \times 10^{-11}$ $z_4 = -6,736$	z_4 é menor do que -1,645, logo a hipótese nula é rejeitada e a hipótese alternativa H_4 é verificada
$p_5 = 1,630 \times 10^{-11}$ $z_5 = -6,736$	z_5 é menor do que -1,645, logo a hipótese nula é rejeitada e a hipótese alternativa H_5 é verificada

Capítulo 6

Conclusão e Trabalhos Futuros

6.1 Conclusão

Esse trabalho introduziu uma arquitetura de um sistema nebuloso genético para realizar adaptação de parâmetros estratégicos de meta-heurísticas motivado pelo fato de durante a busca, diferentes valores de parâmetros produzem melhores resultados de acordo com a região do espaço de busca que se encontra a meta-heurística. Sistemas nebulosos possuem grande flexibilidade e adaptabilidade, constituindo assim uma ferramenta promissora para capturar o dinamismo das características do espaço de busca ao longo de sua execução. Ainda, a arquitetura proposta possibilitou mais autonomia para o usuário das meta-heurísticas, com menos definições manuais dos parâmetros estratégicos.

Nos testes realizados foram usadas duas meta-heurísticas, busca tabu (com adaptação do prazo tabu e de parâmetro da memória de longo prazo) e algoritmo genético (com adaptação da taxa de reprodução e mutação), que foram usadas para resolver o problema de roteamento de veículos com janela de tempo (PRVJT), adotado como caso de estudo. Além destes, foram testados dois outros algoritmos dentro do processo de aprendizado do sistema nebuloso genético, um algoritmo genético com codificação binária e um algoritmo genético com codificação hierárquica mista.

O algoritmo genético com codificação hierárquica produziu resultados melhores, porém se mostrou um pouco menos eficiente, 10% a mais de tempo computacional, do que o algoritmo genético com codificação binária. Apesar do poder de trabalhar com níveis funcionando como subcomponentes que se relacionam, apresentando com isto uma maior flexibilidade e autonomia que potencializa os resultados, este algoritmo tem maior dificuldade de implementação e ajuste, exatamente por este mesmo motivo. Ele, no entanto, foi adotado baseando-se nos melhores resultados em termos de qualidade e sem que acarretasse perdas impactantes de tempo computacional.

Nos testes realizados, as versões com SNG das meta-heurísticas propocionaram resultados melhores do que as versões sem SNG para resolver o PRVJT. Houve melhorias tanto para o número

de veículos acumulados (NVA) quanto para a distância total acumulada (DTA). Para as instâncias de Solomon de 100 clientes, por exemplo, usou-se a base de conhecimento sendo evoluída para as instância em questão e a busca tabu com o SNG melhorou o NVA em $3,93\%$ e a DTA em $2,42\%$ na média, quando comparado ao a busca tabu sem o SNG. Já o algoritmo genético com o SNG reduziu o NVA em $4,12\%$ e uma redução da DTA em $8,08\%$ na média, quando comparado contra o algoritmo genético sem SNG. Os resultados mostram que a introdução do sistema nebuloso genético resultou em uma melhoria maior para o algoritmo genético que para busca tabu, isto se deve a dois fatores: o fato da busca tabu padrão proporcionar melhores resultados que o algoritmo genético padrão, dado um mesmo nível de esforço na construção destes algoritmos e o fato do algoritmo genético ser populacional enquanto a busca tabu ser baseada em um única trajetória, este dois fatores potencializaram o ganho do AG.

Os resultados alcançados com instâncias maiores de Gehring e Homberger com 1000 clientes, fazendo uso de uma base de conhecimento evoluída para instâncias de 100 clientes de Solomon, corroboraram com os resultados obtidos para as instâncias de Solomon de 100 clientes. Isto ainda demonstra que é possível fazer uso do sistema nebuloso em uma abordagem mais parecida com aprendizado de máquina resolvendo instâncias maiores com ganho de qualidade.

Em todos os testes, de maneira geral, foram obtidos resultados bem competitivos com os melhores resultados reportados na literatura para o PRVJT havendo ganhos com relação a resultados importantes como Homberger & Gehring (2005). Neste caso, foi produzido resultado com distância total acumulada menor, mas, apresentando o número de veículos maior, seguindo o *trade-off* destes dois pontos passíveis de minimização, número de veículos e distância total percorrida. Vale ressaltar que os resultados foram alcançados sem realizar refinamento no SNG e sem grande esforço em pormenores das meta-heurísticas de forma a especializá-las e sofisticá-las para o problema tratado, sendo estas mantidas simples e rápidas.

De forma resumida, os resultados mostram que SNG é um caminho promissor para que se alcance um elo entre o espaço de parâmetros e o espaço de busca, permitindo que as meta-heurísticas tenham mais equilíbrio com melhor desempenho.

6.2 Trabalhos futuros

Alguns pontos de investigação e melhorias puderam ser verificados ao longo deste trabalho. Pode-se destacar que há uma possibilidade de ganho significativo com uso de técnicas de refinamento de parâmetros no sistema nebuloso genético empregado. Outro ponto com boa perspectiva de ganho é no teste de outros parâmetros estratégicos, como tamanho populacional para o algoritmo genético ou qual vizinhaça ser usada em cada passo da busca, no caso da busca tabu.

Como forma de estender a pesquisa é bem interessante testar novos problemas e também testar outras meta-heurísticas. Ainda, tratar o PRVJT como problema multi-objetivo e explorar esta característica para analisar o modelo proposto é um caminho possível.

De uma maneira geral, bastante pesquisa surge na área de escolha de parâmetros e resultados começam a aparecer e ser consolidados, tanto na perspectiva de refinamento quanto de controle. Uma outra área de pesquisa que vem crescendo e pode apresentar uma sobreposicão em alguns aspectos é a área de hiper-heurísticas. O objetivo das hiper-heurísticas é selecionar, adaptar e combinar heurísticas/meta-heuristícas para resolver problemas com melhor desempenho, ou seja, a escolha neste caso não está nos parâmetros e sim nos próprios métodos ou características do método em si, mas os objetivos são similares. Evolução na área de controle de meta-heurística pode beneficiar a área de hiper-heurísticas e vice-versa.

Referências Bibliográficas

Ah King, R.; Radha, B.; Rughooputh, H. (2004). A fuzzy logic controlled genetic algorithm for optimal electrical distribution network reconfiguration. In *Networking, Sensing and Control, 2004 IEEE International Conference on*, volume 1, pages 577–582.

Assad, A.; Golden, Bruce L. (1988). *Vehicle routing : methods and studies / edited by Bruce L. Golden and Arjang A. Assad*. North-Holland ; Sole distributors for the U.S.A. and Canada, Elsevier Science Pub. Co., Amsterdam ; New York : New York, N.Y., U.S.A. :. ISBN 0444704078.

Aytug, Haldun; Koehler, Gary J. (1996). Stopping Criteria for Finite Length Genetic Algorithms. *INFORMS JOURNAL ON COMPUTING*, 8(2):183–191.

Bäck, Thomas (1992). Self-adaptation in genetic algorithms. In *Proceedings of the First European Conference on Artificial Life*, pages 263–271. MIT Press.

Bäck, Thomas (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, Oxford, UK. ISBN 0-19-509971-0.

Bäck, Thomas; Fogel, David B.; Michalewicz, Zbigniew, editors (1997). *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, UK.

Badeau, P.; et al. (1997). A parallel tabu search heuristic for the vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 5(2):109 – 122.

Bagley, John Daniel (1967). *The behavior of adaptive systems which employ genetic and correlation algorithms*. Ph.D. thesis, Ann Arbor, MI, USA.

Baker, J. (1987). Reducing bias and inefficiency in the selection algorithm. In *Proceeding of the Second International Conference on Genetic Algorithms and their application*, pages 14–21. L. Erlbaum Associates Inc.

Bard, Jonathan F.; Kontoravdis, George; Yu, Gang (2002). A Branch-and-Cut Procedure for the Vehicle Routing Problem with Time Windows. *TRANSPORTATION SCIENCE*, 36(2):250–269.

Barricelli, N. (1954). Esempi numerici di processi di evoluzione. *Methodos*, pages 45–68.

Battiti, Roberto; Tecchiolli, Giampietro (2004). The reactive tabu search. *ORSA Journal on Computing*, 6(2):126–140.

Beielstein, Thomas B.; Preuss, Mike (2007). Experimental research in evolutionary computation. In *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, pages 3001–3020. ACM.

Birattari, Mauro (2005). *Tuning Metaheuristics (Studies in Computational Intelligence)*. Springer.

Blum, Christian; Roli, Andrea (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308.

Boettcher, Stefan; Percus, Allon G. (1999). Extremal optimization: Methods derived from coevolution.

Bouthillier, A.; Crainic, T. (2005). A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. 32(7):1685–1708.

Bramel, Julien; Simchi-levi, David (1993). Probabilistic analyses and practical algorithms for the vehicle routing problem with time windows. *Operations Research*, 44:501–509.

Bräysy, Olli (2003). A reactive variable neighborhood search for the vehicle-routing problem with time windows. *Informs J. on Computing*, 15(4):347–368.

Bräysy, Olli; Dullaert, Wout; Gendreau, Michel (2004a). Evolutionary algorithms for the vehicle routing problem with time windows. *Journal of Heuristics*, 10(6):587–611. ISSN 1381-1231. doi:http://dx.doi.org/10.1007/s10732-005-5431-6.

Bräysy, Olli; Gendreau, Michel (2005a). Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118. ISSN 1526-5447. doi:http://dx.doi.org/10.1287/trsc.1030.0057.

Bräysy, Olli; Gendreau, Michel (2005b). Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39(1):119–139. ISSN 1526-5447. doi:http://dx.doi.org/10.1287/trsc.1030.0057.

Bräysy, Olli; et al. (2004b). A multi-start local search algorithm for the vehicle routing problem with time windows. *European Journal of Operational Research*, 159(3):586 – 605.

Breedam, Alex Van (1995). Improvement heuristics for the vehicle routing problem based on simulated annealing. *European Journal of Operational Research*, 86(3):480 – 490. ISSN 0377-2217.

Bullnheimer, Bernd; Hartl, Richard F.; Strauss, Christine (1997). Applying the ant system to the vehicle routing problem. In *Second International Conference on Metaheuristics, Sophia-Antipolis, France*.

Casillas, J.; et al. (2001). Genetic feature selection in a fuzzy rule-based classification system learning process for high-dimensional problems. *Inf. Sci.*, 136(1-4):135–157.

Chiang, Wen-Chyuan; Russell, Robert (1996). Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63(1):3–27.

Clarke, G.; Wright, J. W. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *OPERATIONS RESEARCH*, 12(4):568–581. doi:10.1287/opre.12.4.568.

Conover, W. J. (1998). *Practical Nonparametric Statistics*. John Wiley & Sons.

Cordeau, J.; Laporte, Gilbert (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52:928–936.

Cordón, O.; et al. (2001). *Genetic Fuzzy Systems Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. World Scientific, Singapore.

Cordón, O.; et al. (2004). Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets and Systems*, 141(1):5 – 31.

Coy, Steven P.; et al. (2001). Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, 7(1):77–97.

Dantzig, George B.; Wolfe, Philip (1960). Decomposition Principle for Linear Programs. *OPERATIONS RESEARCH*, 8(1):101–111.

Davis, Lawrence (1989). Adapting operator probabilities in genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*, pages 61–69. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Delgado, Myriam Regattieri; Von Zuben, Fernando; Gomide, Fernando (2001). Hierarchical genetic fuzzy systems. *Inf. Sci.*, 136(1-4):29–52.

Desrochers, Martin; Desrosiers, Jacques; Solomon, Marius (1992). A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *OPERATIONS RESEARCH*, 40(2):342–354.

Dorigo, M. (1992). *Optimization, learning and natural algorithms*. Ph.D. thesis, Milão, Itália.

Eberhart, R.; Kennedy, J. (1995). A new optimizer using particle swarm theory. pages 39–43.

Eiben, A. E.; et al. (2007). *Parameter Control in Evolutionary Algorithms*, pages 19–46. Springer.

Eusuff, Muzaffar; Lansey, Kevin; Pasha, Fayzul (2006). Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Engineering Optimization*, 38(2):129–154.

Farmer, J. D.; Packard, N. H.; Perelson, A. S. (1986). The immune system, adaptation, and machine learning. *Phys. D*, 2(1-3):187–204.

Feo, Thomas A.; Resende, Mauricio G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.

Fogel, D. B.; Fogel, L. J.; Atma, J. W. (1991). Meta-evolutionary programming. In *Proc. of 25th Asilomar Conference on Signals, Systems & Computers*, pages 540–545.

Fogel, L. J.; Owens, A. J.; Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. John Wiley, New York, USA.

Gambardella, L.; Taillard, E.; Agazzi, G. (1999). Macs-vrptw: A multiple ant colony system for vehicle routing problems with time windows. Technical report, Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale.

Garey, Michael R.; Johnson, David S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman & Co Ltd.

Geem, Zong W.; Kim, Joong H.; Loganathan, G. V. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation*, 76(2):60–68.

Gehring, H.; Homberger, J. (1999). Two evolutionary metaheuristics for the vehicle routing problem with time windows. *Infor*, 37:297–318.

Gendreau, Michel; Hertz, Alain; Laporte, Gilbert (1994). A tabu search heuristic for the vehicle routing problem. *Manage. Sci.*, 40(10):1276–1290.

Gillett, Billy E.; Miller, Leland R. (1974). A Heuristic Algorithm for the Vehicle-Dispatch Problem. *OPERATIONS RESEARCH*, 22(2):340–349.

Glover, F. (1989). Tabu search-part I. *ORSA Journal on Computing*, 1(3):190–206.

Glover, F. (1990). Tabu search-part II. *ORSA Journal on Computing*, 2(1):4–32.

Glover, Fred (1986). Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, 13(5):533–549.

Goldberg, David; Deb, K.; Korb, B. (1989). Messy genetic algorithms: motivation, analysis, and first results. *Complex Systems*, (3):493–530.

Goldberg, David E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1 edition.

Golden, B. L.; Assad, A. A. (1986). Perspectives on vehicle routing: Exciting new developments. *Oper. Res.*, 34(5):803–810. ISSN 0030-364X. doi:http://dx.doi.org/10.1287/opre.34.5.803.

Gruenz, Lothar; Beyer, Hans-georg (1999). Some observations on the interaction of recombination and self-adaptation in evolution strategies. In *Proc. of Congress on Evolutionary Computation (CEC'99)*, pages 639–645. IEEE Press.

Hansen, Pierre; Mladenovic, Nenad (1997). Variable neighborhood search. *Computers & Operations Research*, 24:1097–1100.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109. doi:10.1093/biomet/57.1.97.

Herrera, F. (2008). Genetic fuzzy systems: taxonomy, current research trends and prospects. *Evolutionary Intelligence*, 1(1):27 – 46.

Hesser, Jürgen; Männer, Reinhard (1991). Towards an optimal mutation probability for genetic algorithms. In *PPSN I: Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, pages 23–32. Springer-Verlag, London, UK.

Holland, J. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press.

Homberger, J.; Gehring, H. (2001). A parallel two-phase metaheuristic for routing problems with time windows. *Asia-Pacific Journal of Operational Research*, 13(1):35–47.

Homberger, J; Gehring, H. (2005). A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, 162(1):220–238.

Hooker, J. N. (1995). Testing heuristics: We have all wrong! *Journal of Heuristics*, 52:33–42.

Hutter, Frank; et al. (2009). ParamILS: An Automatic Algorithm Configuration Framework. *Journal of Artificial Intelligence Research (JAIR)*.

Ingber, L. (1993a). Adaptive simulated annealing (asa). Technical report, Pasadena, CA.

Ingber, L. (1993b). Simulated annealing: Practice versus theory. *Mathematical Computer Modelling*, 18(11):29–57.

Ioannou, G.; Kritikos, M.; Prastacos, G. (2001). A greedy look-ahead heuristic for the vehicle routing problem with time windows. *The Journal of the Operational Research Society*, 52(5):523–537.

Ishibuchi, H.; et al. (1995). Selecting fuzzy if-then rules for classification problems using genetic algorithms. *IEEE Trans. Fuzzy Syst.*, 3:260 – 270.

Jang, J. S. R. (2002). Anfis: adaptive-network-based fuzzy inference system. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(3):665–685.

Jang, J.-S.R.; Sun, C.-T.; Mizutani, E. (1997). *Foundations of Neuro-Fuzzy Systems*. Prentice-Hall, Englewood Cliffs, NJ.

JFuzzyLogic (). Open source fuzzy logic library and fcl language implementation.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical report, Erciyes University, Engineering Faculty, Computer Engineering Department.

Karr, C. (1991). Genetic algorithms for fuzzy controllers. *AI Expert*, 6(2):26–33.

Kennedy, J.; Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4.

Kirkpatrick, S.; et al. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

Kolen, A. W. J.; Rinnooy Kan, A. H. G.; Trienekens, H. W. J. M. (1987). Vehicle Routing with Time Windows. *OPERATIONS RESEARCH*, 35(2):266–273.

Kontoravdis, G. A.; Bard, J. F. (1995). A grasp for the vehicle routing problem with time windows. *INFORMS J. on Computing*, 7(1):10–23.

Koza, John R. (1988). Non-linear genetic algorithms for solving problems. United States Patent 4,935,877.

Krishnanand, K.; Ghose, D. (2005). Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. In *IEEE Swarm Intelligence Symposium proceedings*, pages 84–91.

Krishnanand, K.; Ghose, D. (2009). Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm Intelligence*, 3(2):87–124.

Lau, H; et al. (2009). A fuzzy guided multi-objective evolutionary algorithm model for solving transportation problem. *Expert Systems with Applications*, 36(4):8255 – 8268.

Lee, M.; Takagi, Hideyuki (1993). Dynamic control of genetic algorithms using fuzzy logic techniques. In *Proc. of the Fifth International Conference on Genetic Algorithms*, pages 76–83.

Lehmann, E.L. (1986). *Testing statistical hypotheses*. Wiley.

Lin, Shen (1965). Computer solutions of the traveling salesman problem. 44:2245–2269.

Lin, Shen; Kernighan, Brian W. (1973). An effective heuristic algorithm for the travelling-salesman problem. *Operations Research*, 21:498–516.

Lobo, F.J.; Lima, Cláudio F.; Michalewicz, Zbigniew (2007). *Parameter Setting in Evolutionary Algorithms (Studies in Computational Intelligence)*. Springer.

M. Desrochers, J.K. Lenstra, M.W.P. Savelsbergh; Soumis, F. (1988). *Vehicle Routing: Methods and Studies*, chapter Vehicle routing with time windows: Optimization and approximation, pages 65–84. North-Holland, Amsterdam.

Marques, Vitor; Gomide, Fernando (2010). Memory control of tabu search with genetic fuzzy systems. In *Sixth IEEE World Congress on Computational Intelligence (WCCI), Barcelona, Spain*, pages 2251–2257.

Mercer, R.E.; Sampson, J.R. (1978). Adaptive search using a reproductive meta-plan. *Kybernetes*, 7(3):215–228.

Metropolis, N.; et al. (1953). Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092.

Michalewicz, Z. (1996). *Genetic algorithm + data structures =evolution programs.* Springer.

Moscato, Pablo (1989). On evolution, search, optimization, genetic algorithms and martial arts - towards memetic algorithms.

Moyson, F.; Manderick, B. (1987). The collective behaviour of ants : an example of self-organization in massive parallelism. In *Actes de AAAI Spring Symposium on Parallel Models of Intelligence*.

Mucherino, A.; Seref, O. (2007). Monkey search: A novel meta-heuristic search for global optimization. In *Proceedings of the Conference: Data Mining, System Analysis and Optimization in Biomedicine*.

Mühlenbein, H.; Paaß, G. (1996). From recombination of genes to the estimation of distributions i.binary parameters. volume 496/1991 of *Lecture Notes in Computer Science*, pages 178–187.

Nakrani, Sunil; Tovey, Craig (2004). On honey bees and dynamic server allocation in internet hosting centers. *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, 12(3-4):223–240.

Nauck, D.; Klawoon, F.; Kruse, R. (1997). *Foundations of Neuro-Fuzzy Systems*. Wiley, Chichester, New York.

Osman, Ibrahim Hassan (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Ann. Oper. Res.*, 41(1-4):421–451.

Ostermeier, Andreas; Gawelczyk, Andreas; Hansen, Nikolaus (1994). A derandomized approach to self-adaptation of evolution strategies. *Evol. Comput.*, 2(4):369–380.

Parson, R.; Johnson, M. (1997). A case study in experimental design applied to genetic algorithms with applications to dna sequence assembly. *American Journal of Mathematical and Management Sciences*, 17(3):369–396.

Pedrycz, W.; Gomide, F. (2007). *Fuzzy Systems Engineering: Toward Human-Centric Computing*. Wiley Interscience/IEEE, Roboken-NJ, USA, first edition.

Pham, D. T.; et al. (2006). The bees algorithm, a novel tool for complex optimisation problems. In *Proceedings of 2nd Virtual International Conference on Intelligent Production Machines and Systems* ., pages 454–459. Elsevier (Oxford).

Pham, DT; Karaboga, D. (1991). Optimum design of fuzzy logic controllers using genetic algorithms. *Journal of Systems Engineering*, 1:114–118.

Potvin, Jean-Yves; Rousseau, Jean-Marc (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340.

Potvin, Jean-Yves; Rousseau, Jean-Marc (1995). An exchange heuristic for routing problems with time windows. *Journal of the Operational Research Society*, 46:1433–1446.

Prescott-Gagnon, Eric; Desaulniers, Guy; Rousseau, Louis-Martin (2009). A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks*, 54(4):190–204.

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. Technical report, Royal Air Force Establishment.

Rechenberg, Ingo (1973). *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Ph.D. thesis, Stutggart - Germany.

Reed, J.; Toombs, R.; Barricelli, N.A. (1967). Simulation of biological evolution and machine learning. i. selection of self-reproducing numeric patterns by data processing machines, effects of hereditary control, mutation type and crossing. *Journal of Theoretical Biology*, pages 319–342.

Robbins, H.; Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407.

Rochat, Yves; Taillard, E. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1(1):147–167.

Rosenberg, R.S. (1967). *Simulation of genetic populations with biochemical properties.* Ph.D. thesis, Ann Arbor, MI, USA.

Rubinstein, R. Y. (1997). Optimization of computer simulation models with rare events. *European Journal of Operations Research*, 99:89–112.

Schaffer, J. David; Morishima, Amy (1987). An adaptive crossover distribution mechanism for genetic algorithms. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 36–40. L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

Schwefel, Hans-Paul (1981). *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., New York, NY, USA.

Shaw, Paul (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *CP '98: Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*, pages 417–431. Springer-Verlag, London, UK.

Smith, Jim; Fogarty, Terence C. (1996). Self adaptation of mutation rates in a steady state genetic algorithm. In *Proceedings of International Conference on Evolutionary Computation*, pages 318–323.

Smith, Stephen Frederick (1980). *A learning system based on genetic adaptive algorithms*. Ph.D. thesis, Pittsburgh, PA, USA.

Solomon, M. (1987a). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.*, 35(2):254–265.

Solomon, Marius M. (1987b). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265. ISSN 0030364X.

Stephens, C.; et al. (1997). Self-adaptation in evolving systems.

Storn, R.; Price, K. (1997). Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

Taillard, E.; et al. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2):170–186.

Takagi, H. (2000). Active user intervention in an ec search. In *Int. Conf. on Information Sciences (JCIS2000)*, pages 995–998.

Tan, K.C.; et al. (2001). A messy genetic algorithm for the vehicle routing problem with time window constraints. volume 1, pages 679–686.

Thangiah, S. (1995). Vehicle routing with time windows using genetic algorithms. In *Application Handbook of Genetic Algorithms: New Frontiers, Volume II*, pages 253–277. Lance Chambers, CRC Press.

Thompson, Paul M.; Psaraftis, Harilaos N. (1993). Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Oper. Res.*, 41(5):935–946.

Thrift, P. (1991). Fuzzy logic synthesis with genetic algorithms. In *Proc. of the fourth International Conference on Genetic Algorithms.*, pages 509–513.

Toth, Paolo; Vigo, Daniele (2003). The granular tabu search and its application to the vehicle-routing problem. *INFORMS J. on Computing*, 15(4):333–346. ISSN 1526-5528.

Tsubakitani, S.; Evans, J. (1998). Optimizing tabu list size for the traveling salesman problem. *Computers & Operations Research*, 25(2):91–97.

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51.

Velenzuela-Rendom, M. (1991). The fuzzy classifier system: a classifier system for continuously varying variables. In *Proc. of the fourth International Conference on Genetic Algorithms.*, pages 509–513.

Weinberg, Roger (1970). *Computer simulation of a living cell*. Ph.D. thesis, Ann Arbor, MI, USA.

Wolpert, David H.; Macready, William G. (1997). No free lunch theorems for optimization. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 1(1):67–82.

Xu, J.; Chiu, S.Y.; Glover, F. (1998). A case study in experimental design applied to genetic algorithms with applications to dna sequence assembly. *International Transactions in Operational Research*, 5(3):233–244.

Xu, Jiefeng; Kelly, James P. (1996). A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation Science*, 30(4):379–393.

Yang, X. S. (2009). *Research and Development in Intelligent Systems*, chapter 8 (Firefly Algorithm, Lévy Flights and Global Optimization), pages 209–218. Springer London.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8:338–353.

Apêndice A

Exemplos de Soluções

Este apêndice exemplifica soluções alcançadas: as base de conhecimentos evoluídas nos SNGs, arquivos do *JFuzzyLogic* JFuzzyLogic e soluções alcançadas para o problema de roteamento de veículos com janela de tempo, formato de arquivo criado aqui.

A.1 Exemplos de Bases de Conhecimento

Os exemplos das bases de conhecimento estão colocadas no formato do arquivo definido para o *JFuzzyLogic*, que basicamente define em ordem: declaração das variáveis nebulosas, definição das funções de pertinência (fuzzificação e defuzzificação) e por fim, a definição da base de regras.

A.1.1 Base de conhecimento para Busca Tabu

```
FUNCTION_BLOCK fb

VAR_INPUT
DIV : REAL;
IR : REAL;
MOM : REAL;
END_VAR

VAR_OUTPUT
LMP : REAL;
TMP : REAL;
END_VAR

FUZZIFY DIV
TERM T1 :=  (0.5378713011741638, 0.0) (0.5559309720993042, 1.0) (0.5895894765853882, 0.0) ;
TERM T2 :=  (0.5553107857704163, 0.0) (0.6031814217567444, 1.0) (0.6364570260047913, 0.0) ;
TERM T3 :=  (0.6136822700500488, 0.0) (0.6440770626068115, 1.0) (0.6591590642929077, 1.0) (0.6706597208976746, 0.0) ;
TERM T4 :=  (0.6569637060165405, 0.0) (0.6796522736549377, 1.0) (0.7153223752975464, 0.0) ;
TERM T5 :=  (0.6966161131858826, 0.0) (0.7052025198936462, 1.0) (0.8068106770515442, 1.0) ;
END_FUZZIFY

FUZZIFY IR
TERM T1 :=  (0.0013607405126094818, 0.0) (0.02818853035569191, 1.0) (0.04337954893708229, 0.0) ;
TERM T2 :=  (0.01907617785036564, 0.0) (0.04729953780770302, 1.0) (0.0734911561012268, 0.0) ;
TERM T3 :=  (0.0341777428984642, 0.0) (0.07098808139562607, 1.0) (0.1059882789850235, 0.0) ;
TERM T4 :=  (0.08141255378723145, 0.0) (0.11408253014087677, 1.0) (0.2002541720867157, 1.0) ;
```

```
END_FUZZIFY

FUZZIFY MOM
TERM T1 :=  (0.017628485336899757, 0.0) (0.04251024127006531, 1.0) (0.05744064599275589, 1.0) (0.06998763978481293, 0.0) ;
TERM T10 :=  (0.5968688726425171, 0.0) (0.6236191391944885, 1.0) (1.0, 1.0) ;
TERM T2 :=  (0.04005260020494461, 0.0) (0.10955370962619781, 1.0) (0.18241436779499054, 0.0) ;
TERM T3 :=  (0.1270984709262848, 0.0) (0.1839655339717865, 1.0) (0.2427385449409485, 1.0) (0.2620736062526703, 0.0) ;
TERM T4 :=  (0.2556348443031311, 0.0) (0.26849365234375, 1.0) (0.28596562147140503, 1.0) (0.3034188747406006, 0.0) ;
TERM T5 :=  (0.27218908071517944, 0.0) (0.3121049106121063, 1.0) (0.3737240135669708, 0.0) ;
TERM T6 :=  (0.3460111916065216, 0.0) (0.4014427065849304, 1.0) (0.46054142713546753, 0.0) ;
TERM T7 :=  (0.41554683446884155, 0.0) (0.4258103668689728, 1.0) (0.48224586248397827, 1.0) (0.5012402534484863, 0.0) ;
TERM T8 :=  (0.4908718466758728, 0.0) (0.5237193703651428, 1.0) (0.5458729267120361, 1.0) (0.5588918924331665, 0.0) ;
TERM T9 :=  (0.5109994411468506, 0.0) (0.5821772813796997, 1.0) (0.6294604539871216, 0.0) ;
END_FUZZIFY

DEFUZZIFY LMP
TERM T1 :=  (0.35580989718437195, 0.0) (0.6293826103210449, 1.0) (0.9099254608154297, 0.0) ;
TERM T2 :=  (0.7958711385726929, 0.0) (1.228539228439331, 1.0) (1.6413847208023071, 1.0) (2.2165839672088623, 0.0) ;
TERM T3 :=  (1.3435420989990234, 0.0) (1.9599982500076294, 1.0) (2.4004626274108887, 0.0) ;
TERM T4 :=  (2.2599692344665527, 0.0) (2.6515815258026123, 1.0) (2.7786977291107178, 1.0) (3.3163788318634033, 0.0) ;
TERM T5 :=  (3.0148184299468994, 0.0) (3.209944248199463, 1.0) (5.0, 1.0) ;
METHOD : COG;
DEFAULT := 2.5;
RANGE := (0.35580989718437195 .. 5.0);
END_DEFUZZIFY

DEFUZZIFY TMP
TERM T1 :=  (1.2621963024139404, 0.0) (1.6045031547546387, 1.0) (2.0377917289733887, 0.0) ;
TERM T2 :=  (1.6338844299316406, 0.0) (2.3657259941101074, 1.0) (2.8293232917785645, 0.0) ;
TERM T3 :=  (2.6122899055480957, 0.0) (2.9245595932006836, 1.0) (3.1964237689971924, 1.0) (3.7423341274261475, 0.0) ;
TERM T4 :=  (2.996488332748413, 0.0) (3.5160717964172363, 1.0) (4.0087409019470215, 1.0) (4.503442764282227, 0.0) ;
TERM T5 :=  (4.091209411621094, 0.0) (4.572991371154785, 1.0) (6.0, 1.0) ;
METHOD : COG;
DEFAULT := 3.5;
RANGE := (1.2621963024139404 .. 6.0);
END_DEFUZZIFY

RULEBLOCK Rules
ACT : MIN;
ACCU : MAX;
AND : MIN;
RULE R0 : IF DIV IS T4 AND (MOM IS T4 AND IR IS T3) THEN TMP IS T1 , LMP IS T3;
RULE R1 : IF IR IS T1 AND MOM IS T5 THEN LMP IS T2;
RULE R2 : IF DIV IS T3 AND (MOM IS T3 AND IR IS T1) THEN TMP IS T4 , LMP IS T1;
RULE R3 : IF DIV IS T4 AND (MOM IS T7 AND IR IS T4) THEN TMP IS T1 , LMP IS T2;
RULE R4 : IF DIV IS T1 AND (MOM IS T10 AND IR IS T2) THEN TMP IS T1 , LMP IS T4;
RULE R5 : IF DIV IS T4 AND (MOM IS T10 AND IR IS T2) THEN TMP IS T2 , LMP IS T1;
RULE R6 : IF IR IS T3 AND MOM IS T5 THEN TMP IS T4 , LMP IS T4;
RULE R7 : IF DIV IS T1 AND (MOM IS T7 AND IR IS T2) THEN TMP IS T3;
RULE R8 : IF DIV IS T1 AND (MOM IS T3 AND IR IS T2) THEN LMP IS T5;
RULE R9 : IF DIV IS T4 AND (MOM IS T8 AND IR IS T1) THEN LMP IS T4;
RULE R10 : IF DIV IS T5 AND (MOM IS T1 AND IR IS T1) THEN TMP IS T3 , LMP IS T5;
RULE R11 : IF DIV IS T5 AND (MOM IS T1 AND IR IS T1) THEN TMP IS T4 , LMP IS T1;
RULE R12 : IF IR IS T2 AND MOM IS T5 THEN TMP IS T3 , LMP IS T4;
RULE R13 : IF DIV IS T4 AND (MOM IS T4 AND IR IS T1) THEN TMP IS T5 , LMP IS T4;
RULE R14 : IF DIV IS T1 AND (MOM IS T9 AND IR IS T4) THEN LMP IS T5;
RULE R15 : IF DIV IS T2 AND (MOM IS T8 AND IR IS T1) THEN LMP IS T5;
RULE R16 : IF DIV IS T5 AND (MOM IS T10 AND IR IS T3) THEN TMP IS T3 , LMP IS T3;
RULE R17 : IF DIV IS T5 AND (MOM IS T8 AND IR IS T4) THEN TMP IS T4 , LMP IS T3;
RULE R18 : IF DIV IS T5 AND (MOM IS T6 AND IR IS T2) THEN TMP IS T3 , LMP IS T5;
RULE R19 : IF DIV IS T2 AND MOM IS T9 THEN TMP IS T4 , LMP IS T3;
RULE R20 : IF DIV IS T3 AND (MOM IS T3 AND IR IS T4) THEN TMP IS T4;
RULE R21 : IF DIV IS T1 AND (MOM IS T1 AND IR IS T1) THEN TMP IS T3;
RULE R22 : IF DIV IS T5 AND (MOM IS T4 AND IR IS T3) THEN LMP IS T2;
RULE R24 : IF DIV IS T5 AND MOM IS T10 THEN LMP IS T4;
RULE R25 : IF IR IS T3 AND MOM IS T8 THEN TMP IS T5 , LMP IS T3;
RULE R26 : IF DIV IS T4 AND (MOM IS T3 AND IR IS T3) THEN TMP IS T2 , LMP IS T4;
RULE R27 : IF DIV IS T4 AND MOM IS T4 THEN TMP IS T4 , LMP IS T4;
RULE R28 : IF DIV IS T4 AND (MOM IS T8 AND IR IS T4) THEN TMP IS T1 , LMP IS T2;
RULE R29 : IF DIV IS T2 AND MOM IS T7 THEN TMP IS T2 , LMP IS T2;
END_RULEBLOCK

END_FUNCTION_BLOCK
```

A.1.2 Base de conhecimento para Algoritmo Genético

```
FUNCTION_BLOCK fb

VAR_INPUT
DIV : REAL;
IR : REAL;
MOM : REAL;
END_VAR

VAR_OUTPUT
MR : REAL;
RR : REAL;
END_VAR

FUZZIFY DIV
TERM T1 :=  (0.0, 0.0) (0.0556882806122303, 1.0) (0.06850478053092957, 1.0) (0.11904223263263702, 0.0) ;
TERM T2 :=  (0.0668274387717247, 0.0) (0.10730782151222229, 1.0) (0.15888309478759766, 1.0) (0.21886253356933594, 0.0) ;
TERM T3 :=  (0.15138065814971924, 0.0) (0.18098947405815125, 1.0) (0.21241553127765656, 0.0) ;
TERM T4 :=  (0.1881265640258789, 0.0) (0.2527223229408264, 1.0) (0.2956776022911072, 0.0) ;
TERM T5 :=  (0.24167534708976746, 0.0) (0.32746148109436035, 1.0) (0.38325390219688416, 0.0) ;
TERM T6 :=  (0.3537887632846832, 0.0) (0.4030688405036926, 1.0) (0.6000000238418579, 1.0) ;
END_FUZZIFY

FUZZIFY IR
TERM T1 :=  (-0.17459486424922943, 0.0) (-0.1503818929195404, 1.0) (-0.12727773189544678, 0.0) ;
TERM T2 :=  (-0.14407409727573395, 0.0) (-0.12506385147571564, 1.0) (-0.11817323357168198, 1.0) (-0.11194128543138504, 0.0) ;
TERM T3 :=  (-0.12105657160282135, 0.0) (-0.09856346249580383, 1.0) (-0.07364852726459503, 1.0) (-0.05317902937531471, 0.0) ;
TERM T4 :=  (-0.0712115615606308, 0.0) (-0.054554082453250885, 1.0) (-0.04136333614587784, 1.0) (-0.026004519313573837, 0.0) ;
TERM T5 :=  (-0.03823797404766083, 0.0) (-0.02249978855252266, 1.0) (-0.0020237360149621964, 0.0) ;
TERM T6 :=  (-0.03276166319847107, 0.0) (6.465474143624306E-5, 1.0) (0.028016025200486183, 0.0) ;
TERM T7 :=  (-0.007142158225178719, 0.0) (0.020051002502441406, 1.0) (0.04956946149468422, 0.0) ;
TERM T8 :=  (0.036166105419397354, 0.0) (0.05221359059214592, 1.0) (0.18000000715255737, 1.0) ;
END_FUZZIFY

FUZZIFY MOM
TERM T1 :=  (0.0, 0.0) (0.08113371580839157, 1.0) (0.11178684234619140, 1.0) (0.14252294600009918, 0.0) ;
TERM T2 :=  (0.1313726305961609, 0.0) (0.15638810396194458, 1.0) (0.2058895230293274, 1.0) (0.2668330669403076, 0.0) ;
TERM T3 :=  (0.1844257116317749, 0.0) (0.2728729844093323, 1.0) (0.32929688692092896, 0.0) ;
TERM T4 :=  (0.2877807021141052, 0.0) (0.3448846936225891, 1.0) (0.3865920305252075, 1.0) (0.41953539848327637, 0.0) ;
TERM T5 :=  (0.3995676040649414, 0.0) (0.4664633274078369, 1.0) (0.5042929649353027, 1.0) (0.5858935117721558, 0.0) ;
TERM T6 :=  (0.4826459586620331, 0.0) (0.5706310272216797, 1.0) (0.6649048924446106, 0.0) ;
TERM T7 :=  (0.5997334718704224, 0.0) (0.6603589653968811, 1.0) (1.0, 1.0) ;
END_FUZZIFY

DEFUZZIFY MR
TERM T1 :=  (0.009999999776482582, 0.0) (0.017237646505236626, 1.0) (0.023664824664592743, 1.0) (0.03201485052704811, 0.0) ;
TERM T2 :=  (0.020021159201860428, 0.0) (0.028804976493120193, 1.0) (0.03449716791510582, 1.0) (0.039056021720170975, 0.0) ;
TERM T3 :=  (0.033928703516721725, 0.0) (0.04674821346998215, 1.0) (0.055019546300172806, 0.0) ;
TERM T4 :=  (0.0517357736825943, 0.0) (0.05654815956950188, 1.0) (0.06324116885662079, 1.0) (0.06888318061828613, 0.0) ;
TERM T5 :=  (0.06640089303255081, 0.0) (0.06985749304294586, 1.0) (0.07751846313476562, 0.0) ;
TERM T6 :=  (0.07226401567459106, 0.0) (0.08132647722959518, 1.0) (0.10000000149011612, 1.0) ;
METHOD : COG;
DEFAULT := 0.054999999701976776;
RANGE := (0.009999999776482582 .. 0.10000000149011612);
END_DEFUZZIFY

DEFUZZIFY RR
TERM T1 :=  (0.41016656160354614, 0.0) (0.4512222409248352, 1.0) (0.4932943880558014, 0.0) ;
TERM T2 :=  (0.4744355380535126, 0.0) (0.5016734600067139, 1.0) (0.5272425413131714, 0.0) ;
TERM T3 :=  (0.5005731582641602, 0.0) (0.5283388495445251, 1.0) (0.5460017323493958, 1.0) (0.5675213932991028, 0.0) ;
TERM T4 :=  (0.5266833305358887, 0.0) (0.5635037422180176, 1.0) (0.5771095156669617, 1.0) (0.6043714284896851, 0.0) ;
TERM T5 :=  (0.5875033140182495, 0.0) (0.6176853775978088, 1.0) (0.6372928023338318, 1.0) (0.6708590388298035, 0.0) ;
TERM T6 :=  (0.6368640661239624, 0.0) (0.676466703414917, 1.0) (0.7119913101196289, 0.0) ;
TERM T7 :=  (0.6661206483840942, 0.0) (0.7011991143226624, 1.0) (0.8999999761581421, 1.0) ;
```

```
METHOD : COG;
DEFAULT := 0.6499999761581421;
RANGE := (0.41016656160354614 .. 0.8999999761581421);
END_DEFUZZIFY

RULEBLOCK Rules
ACT : MIN;
ACCU : MAX;
AND : MIN;
RULE R0 : IF IR IS T2 AND MOM IS T7 THEN MR IS T6 , RR IS T3;
RULE R1 : IF DIV IS T6 AND (MOM IS T6 AND IR IS T5) THEN MR IS T3 , RR IS T4;
RULE R2 : IF IR IS T4 AND MOM IS T4 THEN MR IS T2 , RR IS T5;
RULE R3 : IF DIV IS T3 AND (MOM IS T6 AND IR IS T7) THEN MR IS T6 , RR IS T2;
RULE R4 : IF IR IS T5 AND MOM IS T2 THEN MR IS T1;
RULE R5 : IF DIV IS T6 AND IR IS T1 THEN MR IS T2 , RR IS T4;
RULE R6 : IF DIV IS T4 AND IR IS T5 THEN MR IS T6 , RR IS T1;
RULE R7 : IF DIV IS T3 AND (MOM IS T3 AND IR IS T7) THEN MR IS T1 , RR IS T4;
RULE R8 : IF DIV IS T6 AND (MOM IS T2 AND IR IS T5) THEN MR IS T5 , RR IS T3;
RULE R9 : IF DIV IS T6 AND (MOM IS T1 AND IR IS T2) THEN MR IS T1 , RR IS T5;
RULE R10 : IF DIV IS T3 AND IR IS T5 THEN MR IS T2 , RR IS T7;
RULE R11 : IF DIV IS T1 AND (MOM IS T2 AND IR IS T8) THEN MR IS T2 , RR IS T6;
RULE R12 : IF DIV IS T6 AND (MOM IS T3 AND IR IS T5) THEN MR IS T5;
RULE R13 : IF DIV IS T5 AND (MOM IS T2 AND IR IS T7) THEN MR IS T3 , RR IS T1;
RULE R14 : IF DIV IS T6 AND (MOM IS T1 AND IR IS T4) THEN MR IS T2 , RR IS T6;
RULE R15 : IF DIV IS T5 AND (MOM IS T6 AND IR IS T7) THEN MR IS T3;
RULE R16 : IF DIV IS T4 AND (MOM IS T5 AND IR IS T4) THEN MR IS T1 , RR IS T5;
RULE R17 : IF DIV IS T4 AND (MOM IS T2 AND IR IS T2) THEN MR IS T1 , RR IS T2;
RULE R18 : IF DIV IS T1 AND MOM IS T2 THEN MR IS T3 , RR IS T5;
RULE R19 : IF DIV IS T3 AND (MOM IS T5 AND IR IS T7) THEN MR IS T5 , RR IS T3;
RULE R20 : IF IR IS T5 AND MOM IS T4 THEN MR IS T5 , RR IS T3;
RULE R21 : IF IR IS T2 THEN MR IS T4 , RR IS T6;
RULE R22 : IF DIV IS T1 AND (MOM IS T3 AND IR IS T4) THEN MR IS T2;
RULE R23 : IF DIV IS T1 AND IR IS T3 THEN MR IS T2 , RR IS T5;
RULE R24 : IF DIV IS T5 AND (MOM IS T7 AND IR IS T7) THEN MR IS T6 , RR IS T7;
RULE R25 : IF DIV IS T1 AND (MOM IS T6 AND IR IS T8) THEN MR IS T3 , RR IS T4;
RULE R26 : IF DIV IS T1 AND MOM IS T4 THEN MR IS T4 , RR IS T4;
RULE R27 : IF DIV IS T6 AND IR IS T5 THEN MR IS T3 , RR IS T4;
RULE R28 : IF IR IS T7 AND MOM IS T5 THEN MR IS T2;
RULE R29 : IF DIV IS T6 AND (MOM IS T4 AND IR IS T3) THEN MR IS T4 , RR IS T6;
END_RULEBLOCK

END_FUNCTION_BLOCK
```

A.2 Exemplos de Solução para PRVJT

As soluções alcançadas são gravadas em arquivos pós execução dos algoritmos e possuem o seguinte formato: total de veículos (NV), apresentação de cada rota com a sequência dos clientes atendidos, e por fim o número total de clientes e a distância total percorrida (dt) no conjunto das rotas. A sequência das rotas possui informações das instâncias dos *benchmarks* e acrescenta a variável s que indica o momento do início do atendimento de cada cliente, portanto, o sequenciamento tem o seguinte formato:

ID CLIENTE (s:*INÍCIO ATENDIMENTO* # TW-*JANELA* # COOR-*LOCALIZAÇÃO DO CLIENTE*) -> *PRÓXIMO*. O depósito possui id cliente igual a zero.

O exemplo mostrado é para uma execução de uma instância de Solomom (R101 de 100 clientes).

```
Total vehicles: 15
R1: 0 (s: 0.0 #  TW 0 <-> 240 #  COORD 40.0 - 50.0 )  ->  5 (s: 41.0 #  TW 41
```

<-> 71 # COORD 20.0 - 85.0) -> 45 (s: 54.0 # TW 37 <-> 67 # COORD 20.0 - 82.0) -> 2 (s: 71.280106 # TW 50 <-> 80 # COORD 22.0 - 75.0) -> 8 (s: 91.0 # TW 91 <-> 121 # COORD 15.0 - 80.0) -> 6 (s: 106.83095 # TW 95 <-> 125 # COORD 18.0 - 75.0) -> 46 (s: 121.83095 # TW 113 <-> 143 # COORD 18.0 - 80.0) -> 3 (s: 138.23407 # TW 109 <-> 139 # COORD 22.0 - 85.0) -> 1 (s: 151.23407 # TW 145 <-> 175 # COORD 25.0 - 85.0) -> 4 (s: 168.30515 # TW 141 <-> 171 # COORD 20.0 - 80.0) -> 100 (s: 195.33453 # TW 180 <-> 210 # COORD 31.0 - 67.0) -> 70 (s: 209.80667 # TW 180 <-> 210 # COORD 35.0 - 69.0) -> 0 (s: 239.45355 # TW 0 <-> 240 # COORD 40.0 - 50.0)

R2: 0 (s: 0.0 # TW 0 <-> 240 # COORD 40.0 - 50.0) -> 11 (s: 59.0 # TW 59 <-> 89 # COORD 8.0 - 40.0) -> 12 (s: 74.0 # TW 64 <-> 94 # COORD 8.0 - 45.0) -> 73 (s: 100.155495 # TW 77 <-> 107 # COORD 2.0 - 60.0) -> 78 (s: 117.36659 # TW 90 <-> 120 # COORD 8.0 - 56.0) -> 60 (s: 155.0 # TW 155 <-> 185 # COORD 15.0 - 60.0) -> 0 (s: 191.92583 # TW 0 <-> 240 # COORD 40.0 - 50.0)

R3: 0 (s: 0.0 # TW 0 <-> 240 # COORD 40.0 - 50.0) -> 98 (s: 46.0 # TW 46 <-> 76 # COORD 26.0 - 52.0) -> 88 (s: 67.0 # TW 67 <-> 97 # COORD 24.0 - 58.0) -> 7 (s: 96.23538 # TW 79 <-> 109 # COORD 15.0 - 75.0) -> 79 (s: 117.63714 # TW 89 <-> 119 # COORD 6.0 - 68.0) -> 55 (s: 152.93536 # TW 140 <-> 170 # COORD 30.0 - 60.0) -> 68 (s: 172.93536 # TW 144 <-> 174 # COORD 40.0 - 60.0) -> 0 (s: 192.93536 # TW 0 <-> 240 # COORD 40.0 - 50.0)

R4: 0 (s: 0.0 # TW 0 <-> 240 # COORD 40.0 - 50.0) -> 92 (s: 14.764823 # TW 14 <-> 44 # COORD 53.0 - 43.0) -> 33 (s: 61.479942 # TW 51 <-> 81 # COORD 85.0 - 25.0) -> 28 (s: 80.08227 # TW 62 <-> 92 # COORD 92.0 - 30.0) -> 30 (s: 94.08227 # TW 74 <-> 104 # COORD 88.0 - 30.0) -> 50 (s: 120.84532 # TW 116 <-> 146 # COORD 72.0 - 35.0) -> 89 (s: 161.25912 # TW 144 <-> 174 # COORD 67.0 - 5.0) -> 0 (s: 223.73769 # TW <-> 240 # COORD 40.0 - 50.0)

R5: 0 (s: 0.0 # TW 0 <-> 240 # COORD 40.0 - 50.0) -> 14 (s: 35.35534 # TW 35 <-> 65 # COORD 5.0 - 45.0) -> 47 (s: 48.35534 # TW 45 <-> 75 # COORD 2.0 - 45.0) -> 16 (s: 72.0 # TW 72 <-> 102 # COORD 0.0 - 40.0) -> 15 (s: 84.0 # TW 58 <-> 88 # COORD 2.0 - 40.0) -> 9 (s: 103.43398 # TW 91 <-> 121 # COORD 10.0 - 35.0) -> 10 (s: 119.0 # TW 119 <-> 149 # COORD 10.0 - 40.0) -> 13 (s: 142.0 # TW 142 <-> 172 # COORD 5.0 - 35.0) -> 17 (s: 163.18034 # TW 149 <-> 179 # COORD 0.0 - 45.0) -> 0 (s: 213.49164 # TW 0 <-> 240 # COORD 40.0 - 50.0)

R6: 0 (s: 0.0 # TW 0 <-> 240 # COORD 40.0 - 50.0) -> 63 (s: 39.051247 # TW 39 <-> 69 # COORD 65.0 - 20.0) -> 76 (s: 75.0 # TW 75 <-> 105 # COORD 60.0 - 12.0) -> 49 (s: 104.0 # TW 104 <-> 134 # COORD 42.0 - 12.0) -> 22 (s: 117.60555 # TW 92 <-> 122 # COORD 40.0 - 15.0) -> 20 (s: 129.60556 # TW 122 <-> 152 # COORD 42.0 - 15.0) -> 24 (s: 148.0 # TW 148 <-> 178 # COORD 38.0 - 15.0) -> 91 (s: 187.15475 # TW 167 <-> 197 # COORD 49.0 - 42.0) -> 80 (s: 202.53992 # TW 192 <-> 222 # COORD 47.0 - 47.0) -> 0 (s: 220.15569 # TW 0 <-> 240 # COORD 40.0 - 50.0)

R7: 0 (s: 0.0 # TW 0 <-> 240 # COORD 40.0 - 50.0) -> 69 (s: 41.0 # TW 41 <-> 71 # COORD 31.0 - 52.0) -> 90 (s: 86.0 # TW 86 <-> 116 # COORD 37.0 - 47.0) -> 53 (s: 113.26268 # TW 91 <-> 121 # COORD 20.0 - 50.0) -> 0 (s: 143.26268 # TW 0 <-> 240 # COORD 40.0 - 50.0)

```
R8: 0 (s: 0.0 #  TW 0 <-> 240 # COORD 40.0 - 50.0 )  ->  82 (s: 42.0 #  TW 42
<-> 72 #  COORD 27.0 - 43.0 )  ->  52 (s: 65.15295 #  TW 52 <-> 82 #  COORD
25.0 - 30.0 )  ->  99 (s: 80.25197 #  TW 77 <-> 107 #  COORD 26.0 - 35.0 )  ->
86 (s: 102.335014 #  TW 87 <-> 117 #  COORD 21.0 - 24.0 )  ->  57 (s:
121.390396 #  TW 96 <-> 126 #  COORD 30.0 - 25.0 )  ->  74 (s: 142.57074 #  TW
141 <-> 171 #  COORD 20.0 - 20.0 )  ->  0 (s: 188.62625 #  TW 0 <-> 240 #
COORD 40.0 - 50.0 )

R9: 0 (s: 0.0 #  TW 0 <-> 240 #  COORD 40.0 - 50.0 )  ->  31 (s: 61.0 #  TW 61
<-> 91 #  COORD 88.0 - 35.0 )  ->  29 (s: 73.0 #  TW 67 <-> 97 #  COORD 90.0 -
35.0 )  ->  27 (s: 88.0 #  TW 62 <-> 92 #  COORD 95.0 - 35.0 )  ->  26 (s:
115.0 #  TW 115 <-> 145 #  COORD 95.0 - 30.0 )  ->  34 (s: 136.18034 #  TW 111
<-> 141 #  COORD 85.0 - 35.0 )  ->  32 (s: 151.5655 #  TW 131 <-> 161 #  COORD
87.0 - 30.0 )  ->  93 (s: 195.62428 #  TW 178 <-> 208 #  COORD 61.0 - 52.0 )
->  0 (s: 226.7193 #  TW 0 <-> 240 #  COORD 40.0 - 50.0 )

R10: 0 (s: 0.0 #  TW 0 <-> 240 #  COORD 40.0 - 50.0 )  ->  83 (s: 35.0 #  TW 35
<-> 65 #  COORD 37.0 - 31.0 )  ->  23 (s: 71.019226 #  TW 65 <-> 95 #  COORD
38.0 - 5.0 )  ->  21 (s: 83.019226 #  TW 67 <-> 97 #  COORD 40.0 - 5.0 )  ->
19 (s: 98.40439 #  TW 72 <-> 102 #  COORD 42.0 - 10.0 )  ->  18 (s: 113.78955 #
TW 87 <-> 117 #  COORD 44.0 - 5.0 )  ->  48 (s: 151.0 #  TW 151 <-> 181 #
COORD 42.0 - 5.0 )  ->  25 (s: 168.0 #  TW 154 <-> 184 #  COORD 35.0 - 5.0 )
->  0 (s: 223.27692 #  TW 0 <-> 240 #  COORD 40.0 - 50.0 )

R11: 0 (s: 0.0 #  TW 0<-> 240 #  COORD 40.0 - 50.0 )  ->  65 (s: 11.18034 #  TW
11 <-> 41 #  COORD 35.0 - 40.0 )  ->  59 (s: 53.19596 #  TW 42 <-> 72 #  COORD
10.0 - 20.0 )  ->  75 (s: 79.00735 #  TW 74 <-> 104 #  COORD 5.0 - 5.0 )  ->
87 (s: 109.255806 #  TW 90 <-> 120 #  COORD 12.0 - 24.0 )  ->  97 (s: 129.2558
#  TW 120 <-> 150 #  COORD 4.0 - 18.0 )  ->  58 (s: 152.85727 #  TW 152 <-> 182
#  COORD 15.0 - 10.0 )  ->  77 (s: 173.48741 #  TW 150 <-> 180 #  COORD 23.0 -
3.0 )  ->  0 (s: 233.4674 #  TW 0 <-> 240 #  COORD 40.0 - 50.0 )

R12: 0 (s: 0.0 # TW 0 <-> 240 #  COORD 40.0 - 50.0 )  ->  72 (s: 30.0 #  TW 30
<-> 60 #  COORD 63.0 - 65.0 )  ->  39 (s: 55.29706 #  TW 37 <-> 67 #  COORD
60.0 - 80.0 )  ->  36 (s: 72.368126 #  TW 43 <-> 73 #  COORD 65.0 - 85.0 )  ->
40 (s: 87.368126 #  TW 85 <-> 115 #  COORD 60.0 - 85.0 )  ->  38 (s: 102.75329
#  TW 75 <-> 105 #  COORD 62.0 - 80.0 )  ->  41 (s: 119.15641 #  TW 92 <-> 122
#  COORD 58.0 - 75.0 )  ->  37 (s: 139.05591 #  TW 124 <-> 154 #  COORD 65.0 -
82.0 )  ->  35 (s: 152.66145 #  TW 139 <-> 169 #  COORD 67.0 - 85.0 )  ->  0
(s: 206.86552 #  TW 0 <-> 240 #  COORD 40.0 - 50.0 )

R13: 0 (s: 0.0 #  TW 0 <-> 240 #  COORD 40.0 - 50.0 )  ->  42 (s: 33.54102 #  TW
33 <-> 63 #  COORD 55.0 - 80.0 )  ->  44 (s: 64.0 #  TW 64 <-> 94 #  COORD 55.0
- 82.0 )  ->  61 (s: 93.72308 #  TW 66 <-> 96 #  COORD 45.0 - 65.0 )  ->  81
(s: 111.78534 #  TW 86 <-> 116 #  COORD 49.0 - 58.0 )  ->  43 (s: 149.44397 #
TW 128 <-> 158 #  COORD 55.0 - 85.0 )  ->  0 (s: 197.52283 #  TW 0 <-> 240 #
COORD 40.0 - 50.0 )

R14: 0 (s: 0.0 #  TW 0 <-> 240 #  COORD 40.0 - 50.0 )  ->  95 (s: 34.0 #  TW 34
<-> 64 #  COORD 56.0 - 37.0 )  ->  62 (s: 53.219543 #  TW 52 <-> 82 #  COORD
65.0 - 35.0 )  ->  67 (s: 70.29061 #  TW 70 <-> 100 #  COORD 64.0 - 42.0 )  ->
71 (s: 93.32902 #  TW 65 <-> 95 #  COORD 65.0 - 55.0 )  ->  94 (s: 113.95917 #
TW 95 <-> 125 #  COORD 57.0 - 48.0 )  ->  96 (s: 132.0 #  TW 132 <-> 162 #
COORD 55.0 - 54.0 )  ->  54 (s: 148.0 #  TW 139 <-> 169 #  COORD 55.0 - 60.0 )
```

```
->  0 (s: 176.02776 #  TW 0 <-> 240 #  COORD 40.0 - 50.0 )

R15: 0 (s: 0.0 #  TW 0 <-> 240 #  COORD 40.0 - 50.0 )  ->  64 (s: 53.0 #  TW 53
<-> 83 #  COORD 45.0 - 30.0 )  ->  51 (s: 83.0 #  TW 83 <-> 113 #  COORD 55.0 -
20.0 )  ->  85 (s: 101.54401 #  TW 87 <-> 117 #  COORD 63.0 - 23.0 )  ->  84
(s: 120.02929 #  TW 96 <-> 126 #  COORD 57.0 - 29.0 )  ->  56 (s: 139.24884 #
TW 130 <-> 160 #  COORD 50.0 - 35.0 )  ->  66 (s: 158.46838 #  TW 133 <-> 163 #
COORD 41.0 - 37.0 )  ->  0 (s: 181.50679 #  TW 0 <-> 240 #  COORD 40.0 - 50.0 )

Total nodes: 100
Total distance: 1714.69264793396
```

Para mais resultados envie email para: vmarques@dca.fee.unicamp.br.