CENTRO DE INTEGRAÇÃO LOGÍSTICA

ETAPA 5

DESENVOLVIMENTO DA METODOLOGIA DE LOCALIZAÇÃO DE CILs

TOMO III – APÊNDICE II

ESTRATÉGIAS DE RESOLUÇÃO EXATA E HEURÍSTICA PARA O PROBLEMA DE LOCALIZAÇÃO-ALOCAÇÃO DE CILs

Universidade Federal do Rio de Janeiro
COPPE UFRJ
Ministério dos Transportes
Governo Federal BRASIL País rico é país sem pobreza

ESTUDOS E PESQUISAS PARA DESENVOLVIMENTO DE METODOLOGIA PARA IMPLEMENTAÇÃO DE CENTROS DE INTEGRAÇÃO LOGÍSTICA COM VISTAS A SUBSIDIAR POLÍTICAS PÚBLICAS VOLTADAS À PROMOÇÃO DA INTERMODALIDADE NO TRANSPORTE DE CARGAS

Etapa 5

Desenvolvimento da Metodologia de Localização de CILs

Tomo III – Apêndice II

Estratégias de Resolução Exata e Heurística para o Problema de Localização-Alocação de CILs

(Termo de Cooperação nº 01/2013/SPNT/MT)

Janeiro de 2016

República Federativa do Brasil
Dilma Rousseff
Presidência da República

Ministério dos Transportes
Antonio Carlos Rodrigues
Ministro de Estado dos Transportes

Edson Giroto
Secretário-Executivo - Substituto

Secretaria de Política Nacional Transportes
Herbert Drummond
Secretário de Política Nacional de Transportes

Francisco Luiz Baptista da Costa
Diretor do Departamento de Planejamento de Transportes

Katia Matsumoto Tancon
Coordenador-Geral de Avaliação

Eimair Bottega Ebeling
Coordenador-Geral de Planejamento

Equipe Técnica
Artur Monteiro Leitão Junior
Analista de Infraestrutura

Everton Correia do Carmo
Coordenador de Informação e Pesquisa

Francielle Avancini Fornaciari
Analista de Infraestrutura

Luiz Carlos de Souza Neves Pereira
Engenheiro, M.Sc.

Mariana Campos Porto
Analista de Infraestrutura

Universidade Federal do Rio de Janeiro – UFRJ
Professor Roberto Leher
Reitor

Professor Fernando Luis Bastos Ribeiro
Decano do Centro de Tecnologia

Professor Edson Watanabe
Diretor da COPPE

Professor Fernando Rochinha
Diretor de Tecnologia e Inovação

Professor Rômulo Dante Orrico Filho
Coordenação Geral

Equipe Técnica
Professor Abilio Pereira de Lucena Filho
Professor Glaydston Mattos Ribeiro
Professor Hostilio Xavier Ratton Neto
Beatriz Berti da Cóstä
Geraldo Regis Mauri
Gerusa Ravache
Giselle Ferreira Borges
José do Egypto Neirão Reymão
Marcus Vinicius Oliveira Camara
Mariam Tchepurnaya Daychoum
Saul Germano Rabello Quadros
Vanessa de Almeida Guimarães
Vinicius Leal do Forte

Equipe de Apoio
Maria Lucia de Medeiros
Natália Portella Santos Parra Viegas

SUMÁRIO

APÊNDICE II – ESTRATÉGIAS DE RESOLUÇÃO EXATA E HEURÍSTICA PARA O PROBLEMA DE LOCALIZAÇÃO-ALOCAÇÃO DE CILS 2

I. PESQUISA OPERACIONAL E OTIMIZAÇÃO 2

I.1. MÉTODOS EXATOS 8
I.2. MÉTODOS HEURÍSTICOS E META-HEURÍSTICOS 11

I.2.1 Meta-Heurística GRASP 14
I.2.2. Meta-Heurística Iterated Local Search 17
I.2.3 Meta-Heurística Variable Neighborhood Search 18
I.2.4 Meta-Heurística Simulated Annealing 19
I.2.5 .Meta-Heurística Algoritmos Genéticos 19
I.2.6 Meta-Heurística Tabu Search 22

II. CONSIDERAÇÕES FINAIS 23

BIBLIOGRAFIA 24

LISTA DE FIGURAS

Figura I.1 – Interpretação gráfica de um problema de otimização. 7
Figura I.2 – Espaço de busca de soluções. 13
Figura I.3 – Métodos de Trajetória. 13
Figura I.4 – Métodos Populacionais. 13
Figura I.5 – Pseudocódigo para o GRASP. 15
Figura I.6 – Heurística Construtiva. 16
Figura I.7 – Heurística de Refinamento ou Busca Local. 16
Figura I.8 – Pseudocódigo do ILS. 17
Figura I.9 – Pseudocódigo do VNS. 18
Figura I.10 – ***Pseudocódigo do SA.*** 20
Figura I.11 – Etapas de um Algoritmo Genético. 21
Figura I.12 – Cruzamento de cromossomos em um único ponto. 21
Figura I.13 – Pseudocódigo do AG. 21
Figura I.14 – Pseudocódigo da Busca Tabu. 22

LISTA DE QUADROS

Quadro I.1 – Informações sobre o problema de produção das companhias. 5

LISTA DE SIGLAS E ABREVIATURAS

AG – Algoritmos Genéticos
CIL – Centro de Integração Logística
GRASP – *Greedy Randomized Adaptative Search Procedures*
ILS – *Iterated Local Search*
LRC – Lista Restrita de Candidatos
PO – Pesquisa Operacional
SA – *Simulated Annealing*
SC – *Scatter Search*
SOBRAPO – Sociedade Brasileira de Pesquisa Operacional
VNS – *Variable Neighborhood Search*

APÊNDICE II – ESTRATÉGIAS DE RESOLUÇÃO EXATA E HEURÍSTICA PARA O PROBLEMA DE LOCALIZAÇÃO-ALOCAÇÃO DE CILS

APÊNDICE II – ESTRATÉGIAS DE RESOLUÇÃO EXATA E HEURÍSTICA PARA O PROBLEMA DE LOCALIZAÇÃO-ALOCAÇÃO DE CILS

Este apêndice tem como objetivo subsidiar a elaboração do Relatório 5 por meio da descrição das principais técnicas encontradas na literatura pesquisada associadas à resolução de problemas de otimização combinatória, em especial, ao problema de localização-alocação de centros de integração logística. Isso se torna fundamental para que se compreenda os conceitos que embasaram a elaboração do modelo matemático usado, no relatório 5, para a obtenção dos resultados apresentados.

Dessa forma, antes de apresentar tais técnicas, faz-se necessário elucidar alguns conceitos relacionados à área de Pesquisa Operacional e Otimização, na qual os problemas de localização-alocação de CILs estão inseridos.

Assim, o presente documento, Relatório 5 – Tomo III – Apêndice II, foi dividido em três capítulos, o primeiro traz considerações iniciais, o segundo trata sobre conceitos básicos de pesquisa operacional e otimização, falando de modelos exatos e modelos heurísticos e meta-heurísticos. E finalmente a terceira seção faz as considerações finais sobre o documento.

I. PESQUISA OPERACIONAL E OTIMIZAÇÃO

O termo ‘otimização’ refere-se ao estudo de problemas reais representados por meio de modelos matemáticos, no qual se deve escolher, sistematicamente, valores de variáveis reais (que podem assumir valores fracionários, tais como o quanto transportar de produtos que podem ser contabilizados em toneladas) ou inteiras (para os casos em que valores fracionários não fazem sentido, tais como o número de pessoas ou de objetos) dentro de um conjunto viável, de maneira a minimizar ou maximizar uma função objetivo.

É o caso, por exemplo, da escolha sobre quais as quantidades de produtos que compõem um mix de produção que determinada empresa deve produzir e fabricar a fim de alcançar o maior lucro possível. Pode ser aplicada também para designação de funcionários às tarefas, de forma a minimizar o custo, o tempo de fabricação de algum produto ou maximizar o lucro.

Aplicado ao contexto logístico, pode ser usada para determinar quais produtos devem ser enviados por quais modos de transporte entre pontos de origens e destinos, de maneira a satisfazer toda demanda ao menor custo possível. Ainda pode estar associado à localização de facilidades como centros de distribuição, novas unidades fabril e, no âmago do relatório 5, para localização de centros de integração logística.

Cabe mencionar, portanto, que o modelo desenvolvido no projeto congrega diferentes áreas de estudo da pesquisa operacional, uma vez que busca determinar a localização de centros de integração logística no território nacional, indicando quais cargas utilizarão cada CIL (alocação) e quais as conexões modais serão usadas (problema de transporte).

A Otimização, portanto, está intrinsecamente relacionada à Pesquisa Operacional que, segundo a Sociedade Brasileira de Pesquisa Operacional – SOBRAPO, pode ser definida como: uma ciência aplicada voltada para a resolução de problemas de Otimização, aplicando conceitos e métodos de várias áreas científicas na concepção, planejamento ou operação de sistemas, tendo como foco a tomada de decisões.

A Pesquisa Operacional permite, então, avaliar alternativas e encontrar as soluções que melhor servem aos objetivos predefinidos. Segundo Hillier e Lieberman (2006), o processo de solução de um problema de Pesquisa Operacional pode ser dividido nas seguintes fases:

1. Definição do problema;
2. Formulação do modelo matemático que representa o problema;
3. Desenvolvimento de um procedimento computacional para obter soluções a partir do modelo matemático;
4. Teste do modelo e redefinição se necessário;
5. Preparação para implantação do modelo; e
6. Utilização.

A Fase 1 de definição do problema é muito importante, pois permite estabelecer: (a) os parâmetros daquilo que se pretende resolver e suas relações, as quais matematicamente se dão, normalmente, por meio de funções lineares; e (b) uma função objetivo a ser otimizada (por exemplo, maximização do lucro ou minimização do custo), que vai determinar como o objetivo do problema será influenciado pelos valores

das variáveis de decisão (o que se deseja decidir sobre, por exemplo, quantidade a ser produzida ou transportada).

Na confecção do modelo matemático, na Fase 2, busca-se representar o problema e seus parâmetros por meio de símbolos e expressões matemáticas, dando origem ao que se chama de "modelo matemático" representativo do problema que se pretende resolver. Nesse caso, consideram-se variáveis de decisão, uma função objetivo e restrições (pressupostos que precisam ser atendidos, como por exemplo, quantidade máxima de matéria prima disponível para produção, demanda quantidade máxima de funcionários em uma empresa, capacidade dos caminhões e dos links ferroviários, entre outros). Para exemplificar, apresenta-se a seguir um modelo matemático genérico de otimização:

Minimizar

$$z = CX \qquad (1)$$

Sujeito a:

$$AX \geq B \qquad (2)$$

$$X \geq 0 \qquad (3)$$

Onde:

- C é um vetor-linha n dimensional que representa uma medida como, por exemplo, custo;
- A é uma matriz $m\ x\ n$ de coeficientes normalmente positivos que permitem fazer associações entre, por exemplo, o consumo de alguns recursos em relação aos seus valores totais disponíveis representados pelo vetor-coluna B de dimensão m; e
- X é a variável de decisão.

A Equação (1) define a função objetivo que neste caso precisa ser minimizada. As Restrições (2) garantem que as disponibilidades mínimas devem ser respeitadas e as Restrições (3) indicam o domínio das variáveis de decisão, no caso, reais e positivas. As Restrições (2) e (3) definem o espaço de busca ou espaço de soluções. Ou seja, considera que existem restrições, sejam ela de ordem operacional ou não, que não permitem que você alcance qualquer resultado. Por exemplo, há quantidade limitada de matéria-prima, funcionários e tempo não permitindo que determinada empresa produza infinitas unidades.

Normalmente o espaço de busca possui infinitos pontos possíveis de solução, devendo-se buscar aquele que minimiza a Função Objetivo (1), ou seja, o ponto ótimo. A função objetivo, neste caso, poderia ser a minimização do custo.

Como um exemplo ilustrativo, têm-se: uma determinada companhia fabrica dois diferentes produtos Prod1 e Prod2 baseados em dois materiais principais M_1e M_2. O objetivo consiste em encontrar o melhor mix de produção, ou seja, aquele mais lucrativo. O quadro 2.1 apresenta a disponibilidade de cada material, o lucro unitário de cada produto e quanto que cada produto necessita de cada material para ser manufaturado. Consideram-se as variáveis de decisão x_1e x_2, que representam a quantidade de Prod1 e Prod2 a serem produzidos, respectivamente.

A interpretação deste modelo é a seguinte: deseja-se maximizar o lucro por meio da venda dos produtos 1 e 2, os quais estão representados na função objetivo com seus preços. Portanto $5x_1$ significa que a cada unidade de produto 1 vendida, a empresa ganha R$5. Já as restrições estão associadas à disponibilidade de matéria prima para produção.

Quadro I.1 – Informações sobre o problema de produção das companhias.

	Prod1	**Prod2**	**Material disponível**
M_1	6	4	24
M_2	1	2	6
Lucro	$ 5 / un	$ 4 / un	-

Maximizar

$$lucro = 5x_1 + 4x_2$$

Sujeito a:

$$6x_1 + 4x_2 \leq 24$$

$$1x_1 + 2x_2 \leq 6$$

$$x_1, x_2 \geq 0$$

A primeira indica, então, o quanto se usa da matéria prima para a produção das duas mercadoria, o que, multiplicado pela quantidade produzida, precisa ser menor do que a quantidade total que se tem em estoque desta matéria-prima. A segunda tem a mesma interpretação. O última, por sua vez, indica que só poderão ser produzidas quantidades positivas, ou seja, 1 tonelada, 1 tonelada e meia etc, tendo em vista que não faz sentido a produção de quantidades negativas (Figura 2.1).

Cabe mencionar que existem diferentes classificações para um problema de otimização, dependendo das características do espaço de busca (contínuo ou discreto) e dos valores que as variáveis de decisão do problema podem assumir (reais, inteiros ou binários). Dependendo de sua classificação, um problema de otimização pode ser resolvido algebricamente, na forma fechada, ou iterativamente, empregando métodos exatos ou aproximados (Gomes, 2006). A definição da metodologia a ser emprega pertence à Fase 3. Assim, a Fase 3 tem como objetivo encontrar métodos de solução para o problema de otimização em foco, por meio de técnicas de otimização. Na literatura científica, verificam-se diversos métodos que podem ser classificados como exatos, aproximativos e heurísticos ou meta-heurísticos (Hillier e Lieberman, 2006; Arenales *et al*., 2007). Tais técnicas serão detalhadas nas seções subsequentes.

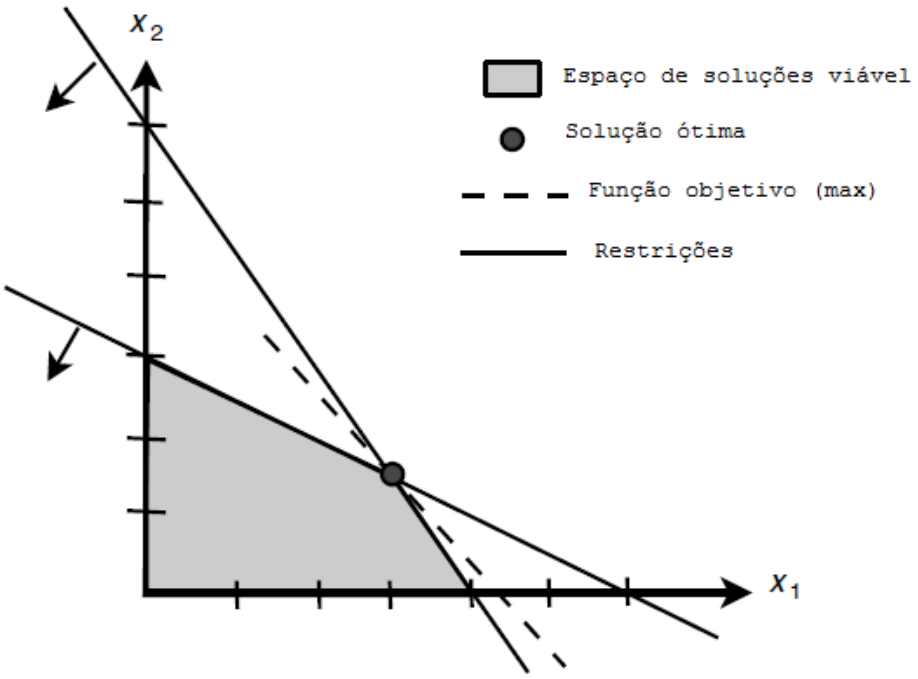

Figura I.1 – Interpretação gráfica de um problema de otimização.

Identificada a melhor técnica de otimização, procede-se a Fase 4 que busca testar o modelo matemático a partir de dados reais para fins de validação. Com as soluções produzidas é possível avaliar se os resultados gerados estão de acordo com o que se esperava, a partir do conhecimento de um especialista do problema estudado.

Com o problema bem definido e a técnica de otimização validada, busca-se preparar um ambiente computacional para implementação do modelo. Esta é a Fase 5 que deve elaborar guias que auxiliem o usuário no processo de entrada de dados, solução e análise de resultados, incluindo, se possível, análises de pós-otimização como, por exemplo, qual a variação de custo ou de lucro esperada (variação da Função Objetivo) caso a disponibilidade de um dado recurso seja alterada.

Na Fase 6 busca-se aplicar, repetidamente, o processo de solução conforme os dados de entrada. Nesta etapa considera-se importante auxiliar as equipes operacionais para eliminar dúvidas. A partir desta base conceitual, a seguir, serão apresentadas algumas das principais técnicas de otimização que podem ser aplicadas para resolver problemas complexos como o problema de localização-alocação de centros de integração logística.

I.1. MÉTODOS EXATOS

Para elucidação dos métodos exatos, faz-se necessário resgatar o modelo (1)-(3) exposto na seção anterior, o qual apresenta relações lineares entre variáveis de decisão, restrições e função objetivo. Problemas com estas características podem ser resolvidos com técnicas de Programação Linear ou Programação Inteira (Hillier e Lieberman, 2006).

No que concerne aos Problemas de Programação Linear, assume-se que estes possuem as seguintes propriedades (Bazaraa *et al*., 2009):

- Proporcionalidade: A contribuição de qualquer variável de decisão na função objetivo ou nas restrições é proporcional à variável. Isso implica que não existe desconto ou economias de escala;
- Divisibilidade: As variáveis de decisão podem ser fracionadas;
- Aditividade: A contribuição de qualquer variável de decisão à função objetivo ou às restrições é independente dos valores das demais variáveis de decisão;
- Certeza: Assume-se que todos os parâmetros do modelo (todos os coeficientes da função objetivo e das restrições) são constantes conhecidas.

Entre os métodos exatos, o método Simplex (Dantzig, 1960) apresenta um bom desempenho para resolver Problemas de Programação Linear (Hillier e Lieberman, 2006). Esse método consiste em investigar o espaço de soluções a partir de uma solução viável. De maneira iterativa, novas soluções, melhores que as anteriores identificadas, são analisadas e quando nenhuma outra solução melhor que a atual pode ser obtida, o processo é interrompido.

Outras técnicas importantes para a resolução de Problemas de Programação Linear são: método de pontos interiores, método projetivo e método dual-simplex. A escolha do método de solução mais eficiente depende da estrutura do problema.

Quando as variáveis de decisão de um problema de otimização estão restritas a números inteiros, denomina-se Problema de Programação Linear Inteira, ou simplesmente Problema de Programação Inteira. Nesse contexto, são definidos os problemas de otimização combinatória e o método exato mais intuitivo para resolver

tais problemas é a busca exaustiva. Esta técnica consiste na enumeração e avaliação de todas as soluções possíveis, selecionando aquelas que produzem os melhores valores para a função objetivo a ser otimizada.

Entretanto, a abordagem exaustiva é computacionalmente intratável para a grande maioria de problemas inteiros, uma vez que a quantidade de combinações cresce exponencialmente com o tamanho do problema a ser resolvido. Isso faz com que o problema não consiga ser resolvido por meio do uso de softwares.

As técnicas de *branch-and-bound* (Lawler e Wood, 1966; Wolsey 1998) e de *branch-and-cut* (Caprara e Fischette, 1997; Jüngler e Thienel, 1998; Wolsey e Nemhauser, 1999) pertencem à classe dos métodos exatos voltados para Problemas de Programação Inteira. Tais métodos apresentam garantia de convergência e fornecem a solução ótima para um determinado problema de otimização combinatória. Entretanto, dependendo do tamanho do problema a serem resolvido, os requisitos computacionais desses métodos tendem a ser proibitivos.

Tanto o método de *branch-and-bound* quanto o método de *branch-and-cut* utilizam uma árvore de busca. Cada ramificação nessa árvore implica na fixação de variáveis. Esses métodos buscam a solução ótima por meio de uma enumeração parcial do espaço de soluções, recursivamente particionando o espaço e eliminando regiões que não são promissoras. Eles ainda consideram o cálculo de um limitante inferior (superior) no caso de minimização (maximização) de uma função objetivo, que normalmente é obtido por meio de técnicas de relaxação e auxiliam no processo de ramificação e poda da árvore de busca.

Os problemas de otimização combinatória que possuem solução, por sua vez, podem ser representados a partir de hiperplanos derivados das restrições do modelo, constituindo assim, a região ou espaço de solução do problema. A eficiência desses algoritmos deve-se ao fato de que a região viável do espaço de soluções é um conjunto convexo e a função objetivo é uma função convexa (Talbi, 2009). Assim, a solução ótima global é necessariamente um ponto na região viável do problema.

A função objetivo indica a direção de crescimento de seu valor, assim, auxilia na busca pela melhor solução. Dependendo da estrutura do problema de otimização, novas restrições podem ser derivadas a partir dos nós da árvore de representação, gerando novas restrições. Essas novas restrições são denominadas cortes (*cuts*) que, por sua vez, reduzem o número de ramificações a delimitar o espaço de busca.

Outra técnica útil para resolução de Problemas de Programação Inteira é a Relaxação Lagrangeana (Wolsey, 1998). Essa técnica consiste em relaxar, no sentido Lagrangeano, um conjunto de restrições consideradas “difíceis”, retirando-as da formulação, mas aplicando uma penalidade na função objetivo para compensar essa retirada. A utilização desta técnica cresceu a partir dos trabalhos de Held e Karp (1970) e Held e Karp (1971) e tem se mostrado importante para vários problemas clássicos de otimização como o Problema do Caixeiro Viajante e o Problema de Particionamento de Conjuntos.

Por último, a Decomposição Dantzig-Wolfe (Dantzig e Wolfe, 1960) consiste em outra técnica importante para resolução de Problemas de Programação Linear Inteira. O método consiste em decompor o problema original em subproblemas tal que restrições de acoplamento são empregadas para garantir a união. Vários problemas clássicos de otimização vêm sendo resolvidos com a Decomposição Dantzig-Wolfe como, por exemplo, o Problema de Roteamento de Veículos Capacitado e o Problema de Alocação.

As técnicas desenvolvidas para o Problema de Programação Linear Inteira também podem ser aplicados aos Problemas de Programação Inteira Mista, que consideram tanto varáveis inteiras quanto variáveis contínuas na modelagem matemática do problema. Salienta-se, neste ponto, que o problema tratado no âmbito do presente projeto é considerado um Problema de Programação Inteira Mista, pois envolve tanto variáveis inteiras (CILs) quanto fracionárias (quantidade de produtos).

Outra técnica para resolução exata de problemas de otimização é a programação dinâmica. Nesta técnica, "sub instâncias" menores do problema original são resolvidas de maneira exata, armazenando os resultados em uma tabela de resultados parciais para que possam ser utilizados posteriormente. Desta maneira, evita-se o cálculo de uma mesma chamada diversas vezes. No decorrer do algoritmo, soluções parciais são combinadas de modo que, a cada passo, obtém-se resposta para "sub instâncias" cada vez maiores. Com isso, uma sequência de decisões inter-relacionadas é obtida até que se chegue a uma solução do problema original. Esse tipo de técnica deve ser aplicado em problemas específicos, que obedecem a certas características como a divisão em subproblemas. Por isso, sua aplicação é restrita e depende de uma análise prévia do problema sendo abordado.

Embora os métodos exatos garantam, em tese, a resolução exata de problemas de otimização, na prática isso nem sempre é constatado. Isso acontece, pois, na maioria dos casos, trata-se de problemas reais de grande porte (e importância),

fazendo com que as decisões empresariais associadas devam ser tomadas o mais breve possível. Isso faz com que o tempo de resolução de um problema desta magnitude não atenda à urgência da decisão (dependendo do número de variáveis e de restrições, pode levar dias para se encontrar a solução ótima). Assim, são usadas técnicas heurísticas que não garantem o resultado ótimo, mas encontra um resultado considerado adequado em um menor tempo de solução.

Na sequência serão apresentados métodos heurísticos e meta-heurísticos que, cada vez mais, despontam na literatura científica como alternativas viáveis aos métodos exatos.

I.2. MÉTODOS HEURÍSTICOS E META-HEURÍSTICOS

Problemas de Otimização Combinatória, como mencionado na seção anterior, são problemas recorrentes nas áreas de transporte, logística, administração e localização, em que se busca minimizar ou maximizar uma função por meio de escolhas sistemáticas dos valores das variáveis envolvidas.

Tais problemas podem ser definidos por um conjunto base, $E = \{1, \dots, n\}$, um conjunto de soluções viáveis dado por $F \subset 2^E$ e uma função objetivo $f : 2^E \rightarrow R$, tal que $f(S) = \sum c(e), \forall\, S \in 2^E$, onde $c(e)$ é o custo associado à inclusão do elemento $e \in E$ na solução S (Gendreau e Potvin, 2010). Na versão de minimização, procura-se uma solução ótima $S^* \in F$ tal que $f(S^*) \leq f(S), \forall\, S \in F$. O conjunto E, o vetor de custos c e o conjunto de soluções viáveis F são definidos para cada problema específico (Cook *et al.* , 1997).

Para os problemas de otimização combinatória conhecidos como NP-Difíceis, a obtenção de soluções ótimas por técnicas exatas (ver item 2.1), considerando pelo menos um critério de otimização, exige o emprego de algoritmos de alta complexidade que consomem elevados tempos de processamento. Isso inviabiliza, em alguns casos, a resolução de forma exata de problemas de grande porte que ocorrem em situações práticas reais. Nesse contexto, a aplicação de algoritmos aproximados, também denominados heurísticos, torna-se uma alternativa viável. Tais algoritmos, além de poderem ser combinados com os métodos exatos para acelerar seu tempo de processamento, também podem ser utilizados de maneira isolada para se encontrar

soluções próximas às ótimas (podendo até mesmo ser ótimas, embora não se possa garantir com exatidão).

Segundo Silver *et al*. (1980), heurísticas são procedimentos dedicados à resolução de problemas cuja estrutura pode ser interpretada e explorada de forma inteligente, permitindo assim que uma solução razoável seja obtida por meio de uma abordagem intuitiva. Recentemente, algoritmos baseados em meta-heurísticas ganharam notoriedade por conseguirem resolver diversos problemas de Otimização Combinatória de maneira quase ótima, em um tempo de execução consideravelmente menor que os tempos exigidos por métodos exatos. Segundo Osman e Laporte (1996), meta-heurísticas são procedimentos genéricos para a solução aproximada de problemas de otimização combinatória computacionalmente difíceis.

A adaptação de uma meta-heurística para um determinado problema fornece uma heurística para esse problema e o processo de busca de uma boa solução aproximada consiste em aplicar heurísticas projetadas de acordo com cada problema específico. Os dois tipos de heurísticas mais comumente encontradas em trabalhos que envolvem problemas de otimização combinatória, são as heurísticas construtivas e de busca local. A ideia básica de uma heurística construtiva é iterativamente adicionar componentes individuais a uma solução parcial até obter uma solução completa viável. Por outro lado, as heurísticas de busca local têm por objetivo tentar melhorar uma solução viável ao manipular os elementos que a compõe, até que se encontre um ótimo local.

Outro tipo de heurística bastante utilizada são as estratégias gulosas, que buscam sempre a solução parcial mais atraente a cada passo. Essas estratégias baseiam-se em informações locais e cada decisão tomada será irreversível até o final do algoritmo. Uma grande quantidade de problemas pode ser resolvidos dessa forma, embora não há garantias de que a solução encontrada é a melhor possível.

Originalmente, as meta-heurísticas apenas controlavam a interação entre estratégias de alto nível com heurísticas de refinamento para criar um processo capaz de escapar de ótimos locais (Figura I.2), executando uma profunda e criteriosa busca no espaço de soluções. Com o tempo, incluíram também procedimentos que exploram uma ou mais estrutura de vizinhança das soluções, métodos que mantinham uma população de soluções interagindo entre si, e também técnicas de destruição e reconstrução de soluções.

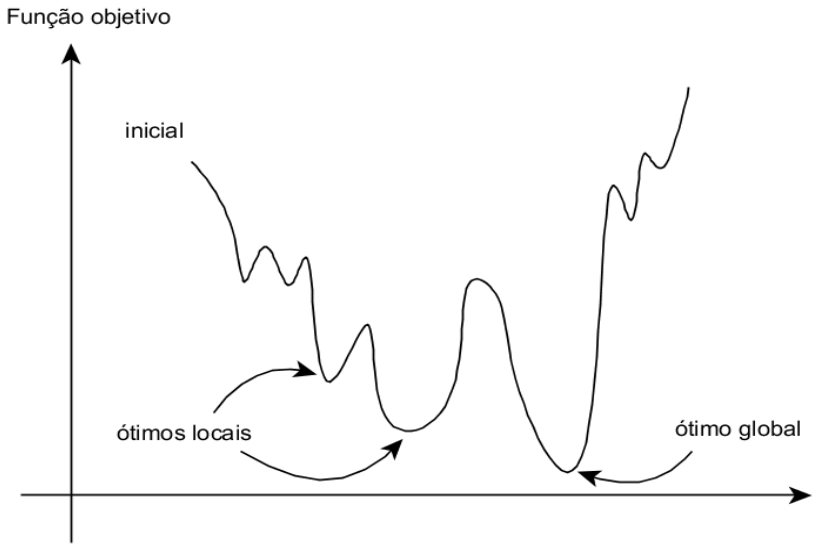

Figura I.2 – Espaço de busca de soluções.

Em geral, as meta-heurísticas baseiam-se na combinação desses dois tipos de heurísticas entre si, bem como outros procedimentos que os auxiliem. É comum classificar as meta-heurísticas de acordo com o número de soluções em utilização ao mesmo tempo, dividindo-se em métodos de trajetória, onde uma única solução é considerada (Figura I.3), e métodos populacionais, onde várias soluções são consideradas (Figura I.4).

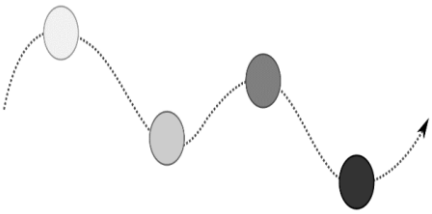

Figura I.3 – Métodos de Trajetória.

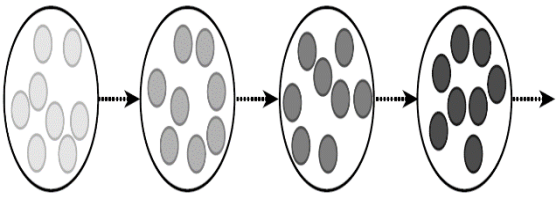

Figura I.4 – Métodos Populacionais.

As meta-heurísticas se tornaram populares por serem flexíveis, fáceis de entender e, ainda, por utilizarem, em alguns casos, comportamentos da natureza para justificarem suas abordagens.

Na literatura, existe diversas meta-heurísticas já consolidadas e comprovadamente eficientes na resolução de problemas de otimização, destacando-se: *Greedy Randomized Adaptive Search Procedure* – GRASP (Feo e Resende, 1995), Busca Tabu (Glover, 1986), Algoritmos Genéticos – AG (Goldberg, 1989; Holland, 1975), *Scatter Search* – SC (Glover, 1977), *Simulated Annealing* – SA (Cerny, 1985; Kirkpatrick *et al*., 1983), *Variable Neighborhood Search* – VNS (Mladenovic e Hansen, 1997) e *Iterated Local Search* – ILS (Baxter, 1981; Lourenço *et al*., 2003). Cada uma delas é baseada em um paradigma distinto e oferece diferentes mecanismos que permitem escapar de ótimos locais.

As seções que se seguem apresentam algumas das meta-heurísticas mais citadas na literatura científica.

I.2.1 Meta-Heurística GRASP

Greedy Randomized Adaptive Search Procedure – GRASP é uma meta-heurística proposta por Feo e Resense (1995) que já foi aplicada com sucesso a diversos problemas de otimização. O GRASP é composto por um processo iterativo, no qual cada iteração é dividida em duas fases: construção e busca local.

A cada iteração, uma solução viável s é construída por um procedimento guloso e aleatório e avaliada por uma função $f(s)$. Essa função fornece o custo da solução construída e deve ser definida de acordo com o problema abordado. Em seguida, a solução s é submetida a uma busca no espaço vizinho de soluções, com o objetivo de melhorá-la até encontrar um ótimo local. A melhor solução geral é mantida e retornada como resultado final. O pseudocódigo da versão original proposta do GRASP pode ser visualizado na Figura I.5.

O primeiro passo do algoritmo é inicializar as soluções s^* (melhor solução) e s (solução corrente) como vazias e o valor do custo de s^*, representado por $f(s^*)$, com valor infinito. Assim, o laço principal começa e uma solução s é construída (linha 4). Essa solução é submetida à heurística de busca local (linha 5) e, caso o custo da solução s seja melhor (no caso, menor) que o custo da melhor solução s^*, então s^* é

atualizada com a solução corrente s (linha 7). Esses passos se repetem até que se completem a quantidade máxima de iterações ($maxIter$).

Algoritmo GRASP

```
1:  GRASP(α, E, maxIter)
2:  s* ← s ← ∅; f(s*) ← ∞;
3:  Para iter = 1 até maxIter faça
4:      s ← ConstruçãoGulosaAleatória(α, E);
5:      s ← BuscaLocal(s);
6:      Se f(s) < f(s*) então
7:          s* ← s;
8:      Fim-se
9:  Fim-para
10: Retorne s*;
```

Figura I.5 – Pseudocódigo para o GRASP.

A etapa de construção é também iterativa e baseada em um procedimento guloso e aleatório, que considera o conjunto de todos os elementos E. Inicialmente, todos os itens $i \in E$ que podem ser inseridos na solução são avaliados de acordo com uma função $c(i)$, que determina o quanto aquele elemento contribui para a solução s. Assim, são inseridos em uma lista denominada Lista Restrita de Candidatos (LRC) todos os elementos i que atendem ao seguinte critério $c(i) \in [\, c^{min}, c^{min} + \alpha\,(c^{max} - c^{min})\,]$ com $\alpha \in [0,1]$ e considerando c^{min} e c^{max}, respectivamente, os valores máximo e mínimo de contribuição dentre os elementos avaliados.

Após a criação da LRC, um dos elementos é escolhido aleatoriamente e inserido na solução. Em seguida, repetem-se esses mesmos passos até que todos os elementos possíveis sejam inseridos. No critério anteriormente apresentado, pode-se notar que o parâmetro α funciona como uma componente de aleatoriedade. Quando α se aproxima de zero, a escolha tende a ser mais gulosa (primeiro elemento tende a ser escolhido), ao passo que quando se aproxima de um, a seleção tende a ficar mais aleatória.

A Figura I.6 representa o pseudocódigo da fase de construção do GRASP. A solução a ser construída s é inicializada como uma solução vazia (linha 2) e, após avaliar cada elemento i, $\forall\, i \in E$ (linha 3), constrói-se a LRC (linha 5) de acordo a função $c(i)$ e o parâmetro α. Dentre os elementos da LRC, um é escolhido aleatoriamente (linha 6) e inserido na solução (linha 7). Esses passos (linhas 5, 6, 7 e 8) se repetem até que uma solução seja construída.

Algoritmo Construção

1: **ConstruçãoGulosaAleatória**(α, E);
2: $s \leftarrow \emptyset$;
3: Avaliar $c(i)$ $\forall$ $i \in E$;
4: **Enquanto** $\neg$ SoluçãoCompleta(s) **faça**
5: $LRC \leftarrow$ ConstroiLRC(α);
6: $i \leftarrow$ EscolhaAleatória(LRC);
7: $s \leftarrow s \cup \{i\}$;
8: Avaliar $c(i)$ $\forall$ $i \in E \backslash s$;
9: **Fim-enquanto**
10: **Retorne** s;

Figura I.6 – Heurística Construtiva.

Após a construção, a solução s é submetida a uma intensificação, com o objetivo de melhorá-la. A fase de busca local é composta por uma estratégia de aprimoramento iterativa, que busca soluções de melhor qualidade em um conjunto de soluções próximas a s. Esse conjunto é denominado vizinhança de s, representado por $N(s)$, e pode ser obtido aplicando simples operações que alteram s. Assim, a busca local visa encontrar o melhor vizinho de s com respeito a uma vizinhança $N(s)$.

A Figura I.7 apresenta o pseudocódigo da fase de busca local para o GRASP. A linha 2 define o conjunto H, composto por todas as soluções y presentes na vizinhança $N(s)$ que possuem qualidade melhor que a solução s, de acordo com a função de avaliação $f(s)$. Assim, a cada iteração, seleciona-se um vizinho s' que seja melhor que s (linha 4) e um novo conjunto H é estruturado com base na solução s' (linha 5). A busca local termina quando não existirem vizinhos capazes de melhorar a qualidade da solução corrente s'. Nesse caso, afirma-se que s' é ótimo local com respeito à vizinhança $N(s)$.

Algoritmo Busca Local

1: **BuscaLocal**(s);
2: $H = \{y \in N(s) | f(y) < f(s)\}$;
3: **Enquanto** $|H| > 0$ **faça**
4: Selecionar $s' \in H$;
5: $H = \{y \in N(s') | f(y) < f(s')\}$;
6: **Fim-enquanto**
7: **Retorne** s';

Figura I.7 – Heurística de Refinamento ou Busca Local.

I.2.2 Meta-Heurística Iterated Local Search

A meta-heurística Iterated Local Search é um método de trajetória que explora o espaço de soluções por meio da aplicação sucessiva de perturbações a ótimos locais, obtidos através de um procedimento de busca local.

Inicialmente, um procedimento construtivo é utilizado para geração de uma solução inicial, cuja vizinhança é explorada exaustivamente até que se encontre um ótimo local (relativo à estrutura de vizinhança que está sendo empregada neste momento). Em seguida, é aplicado à solução corrente um mecanismo de perturbação, forçando o método a escapar do ótimo local, e, logo após, a vizinhança da solução perturbada é explorada pela heurística de refinamento a fim de encontrar um novo ótimo local.

Ao desenvolver uma heurística baseada em Iterated Local Search, quatro procedimentos devem ser especificados de acordo com o problema que está sendo abordado (ver Figura I.8). O primeiro, GeraSolucaoInicial(), é a heurística construtiva responsável por criar uma solução inicial para o problema. A segunda componente é a BuscaLocal() que deve ser capaz de explorar a vizinhança da solução gerada pelo primeiro procedimento.

Algoritmo ILS

```
1 Procedimento ILS
2 s_0 ← GeraSolucaoInicial();
3 s* ← BuscaLocal(s_0);
4 enquanto critério de parada não satisfeito faça
5     s' ← Perturba(s*, historico);
6     s'* ← BuscaLocal(s');
7     s* ← CriterioAceitacao(s*, s'*, historico);
8 fim enquanto
9 fim ILS
```

Figura I.8 – Pseudocódigo do ILS.

O terceiro procedimento é o Pertuba() que tem como objetivo modificar as soluções de modo que se escape do ótimo local encontrado pela busca local. O último elemento CriterioAceitacao() é responsável por determinar se a solução gerada pela busca local será descartada ou não como ponto de partida da próxima iteração.

I.2.3 Meta-Heurística Variable Neighborhood Search

Variable Neighborhood Search – VNS é uma meta-heurística utilizada na resolução de problemas de otimização baseada na ideia de trocar sistematicamente de estruturas de vizinhança enquanto realiza uma busca por soluções de boa qualidade. Semelhante à meta-heurística Iterated Local Search, essa estratégia também dispõe de uma fase de perturbação (denominada aqui de *shake*), responsável por tentar escapar de ótimos locais que eventualmente surgem ao decorrer no algoritmo.

A Figura I.9 inicia com uma solução inicial como parâmetro, que pode ser gerada por uma heurística construtiva qualquer, devidamente adaptada ao problema que está sendo abordado. A cada iteração, a função Shake() provoca uma alteração na solução corrente ao aplicar um movimento aleatório da k-ésima estrutura de vizinhança, i.e. $x' \in N_k(x)$. Em seguida, a função BestImprovement() tem por objetivo encontrar um ótimo local relativo a uma dada vizinhança que será aplicada à solução corrente.

Por fim, a função NeighborhoodChange() compara a função objetivo da solução fornecida pela etapa anterior com a solução corrente e, caso seja menor, ocorre a cópia de solução e a sequência de vizinhanças é reinicializada. Caso contrário, a próxima estrutura de vizinhança é selecionada para ser aplicada à solução corrente.

Algoritmo VNS

```
1:  VNS (x, k_max, t_max )
2:  t ← 0;
3:  Enquanto t ≤ t_max faça
4:     k ← 1;
5:     Repetir
6:        x' ←Shake(x,k);
7:        x'' ←BestImprovement(x');
8:        x, k ←NeighboorhoodChange(x,x'',k);
9:     até que k = k_max;
10:    t ← CpuTime();
11: Fim-enquanto;
12: Retorne x;
```

Figura I.9 – Pseudocódigo do VNS.

I.2.4 Meta-Heurística Simulated Annealing

Simulated Annealing – SA é também uma meta-heurística largamente estudada com aplicações em diversos problemas de otimização. Sua principal ideia é fornecer, em um dado momento do método, mecanismos para escapar de ótimos locais ao permitir movimentos que piorem a função objetivo da solução (diferente dos movimentos denominados *hill-climbing*, em que apenas melhorias da função objetivo são aceitas) na esperança de encontrar um ótimo global.

A cada iteração desta meta-heurística, os valores de função objetivo de duas soluções (a corrente e uma nova solução) são comparadas. Soluções com melhores custos de função objetivo são sempre aceitas, ao passo que soluções cujo custo é pior podem ser aceitas com certa probabilidade.

Essa probabilidade é calculada a partir de um parâmetro denominado temperatura, que vai sendo decrementado no decorrer do algoritmo. Inicialmente, a temperatura possui um alto valor e, consequentemente, a probabilidade de aceitar uma solução de pior custo é alta. À medida que o algoritmo é executado, a temperatura é resfriada, diminuindo também a probabilidade de aceitação das soluções com pior custo.

A função probabilidade de aceitação é dada pela seguinte equação $P(aceitar x) = \begin{cases} e^{-\frac{f(x')-f(x)}{t_k}}, & se f(x') - f(x) > 0 \\ 1, & c.c. \end{cases}$, onde t_k é a temperatura corrente na iteração k. Com isso, o pseudocódigo da meta-heurística SA pode ser observado na Figura I.10.

I.2.5 .Meta-Heurística Algoritmos Genéticos

Algoritmos Genéticos – AG são estratégias populacionais que se tornaram interessantes na resolução de problemas de otimização. A ideia principal de tais algoritmos foi derivada da teoria da evolução das espécies, baseando-se nos conceitos de cruzamento, mutação genética de alguns indivíduos de uma determinada população e na sobrevivência do mais adaptado. Alguns outros termos, como Computação Evolutiva ou Algoritmos Evolucionários, vêm sendo utilizado para se referir a algoritmos que seguem essas características.

Os AG foram primeiramente propostos por Holland (1975). Em clara analogia à biologia, soluções para um dado problema são chamadas de cromossomos, ao passo que um grupo de cromossomos é denominado de população. A avaliação de um cromossomo é chamada de aptidão ou *fitness* que pode representar a função objetivo do problema.

Algoritmo *Simulated Annealing*

1: SA (x, t_0, t_r)
2: **Repetir**
3: $m \leftarrow 0$;
4: **Repetir**
5: $x' \leftarrow$N(x);
6: $\Delta_{x,x'} = f(x') - f(x)$;
7: **Se** $\Delta_{x,x'} \leq 0$ **então**
8: $x \leftarrow x'$;
9: **senão**
10: $x \leftarrow x'$ com probabilidade $e^{\frac{-\Delta_{x,x'}}{t_k}}$;
11: **Fim-se**
12: $m \leftarrow m + 1$;
13: **até que** $m = M_k$;
14: $k \leftarrow k + 1$;
15: **até que** critério de parada;
16: $t \leftarrow CpuTime()$;
17: **Retorne** x;

Figura I.10 – *Pseudocódigo do SA.*

A ideia, então, consiste em gerar um conjunto (população) de soluções para o problema em questão e, a partir desse conjunto, gerar novas soluções (cromossomos) realizando algumas operações de cruzamento com as soluções já existentes no conjunto, dando prioridade na escolha das soluções com melhor avaliação (Figura I.11). Espera-se, com isso, que a qualidade das soluções melhore com o passar das gerações e o algoritmo convirja para um ótimo local.

A cada geração, mutações podem ocorrer em um ou mais cromossomos, o que fornece o componente de diversificação do algoritmo, e pode ser representada como uma simples alteração de algum elemento daquela solução. O cruzamento ou *crossover* de duas soluções pode ser representado por uma combinação de duas ou mais soluções, a fim de aproveitar componentes promissoras de soluções distintas, conforme mostra a Figura I.12.

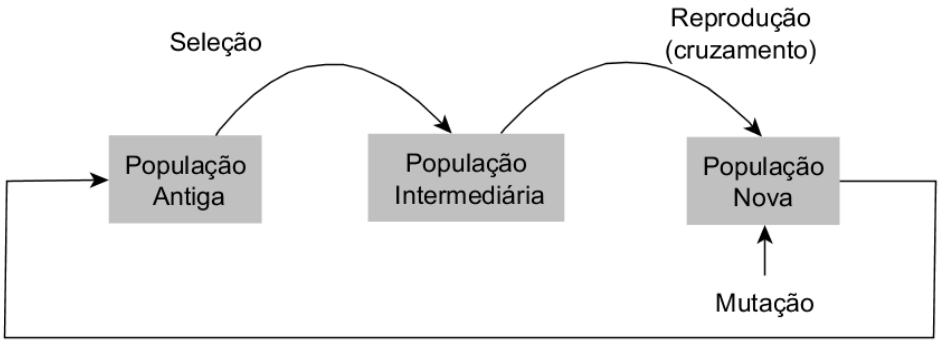

Figura I.11 – Etapas de um Algoritmo Genético.

Figura I.12 – Cruzamento de cromossomos em um único ponto.

A escolha das soluções para aplicação das operações é definida por critérios. Elitismo, por exemplo, considera apenas o cromossomo mais adaptado, i.e., que possuir o melhor fitness. Pode-se também realizar uma pré-seleção de alguns elementos e realizar um torneio entre eles, onde só o vencedor será escolhido para aquela operação. Outra possibilidade, denominada roleta, é definir probabilidades a um grupo de cromossomos, dando uma chance maior àqueles que possuírem melhor aptidão. A Figura I.13 exibe um pseudocódigo do AG.

Algoritmo Algoritmo Genético

```
1:  Enquanto critério de parada não satisfeito faça
2:    Repetir
3:      Se CondiçãoDeCrossover então
4:        Selecionar cromossomos para o cruzamento(critérioSeleção);
5:      Fim-se
6:      Se CondiçãoDeMutação então
7:        Selecionar pontos de mutação;
8:        Selecionar cromossomos para a mutação;
9:      Fim-se
10:     Avaliar os cromossomos da nova população;
11:   até que critério de parada;
12:   GerarNovaPopulacao();
13: Fim-enquanto
```

Figura I.13 – Pseudocódigo do AG.

I.2.6 Meta-Heurística Tabu Search

Essa meta-heurística, proposta por (Glover, 1986), consiste em um procedimento de busca local que faz o uso do conceito de tabu para encontrar a melhor função objetivo do problema. O tabu consiste em uma lista de informações (ou lista tabu), no caso, uma lista de soluções candidatas visitadas recentemente, que serão evitas durante a busca em um predeterminado intervalo de tempo, a fim de proibir revisitar tais soluções e assim, diversificar o exame do espaço de soluções. Assim como o Simulated Annealing, a Busca tabu também aceita movimentos de piora na função objetivo.

A Lista Tabu pode proibir visitas a soluções atraentes. Como forma de equilibrar as escolhas, existe o critério de Aspiração, que permite que soluções sejam visitadas mesmo que estejam na lista Tabu. Esse critério é comumente satisfeita se uma determinada solução da vizinhança é melhor do que todas as anteriores obtidas (Figura I.14).

Algoritmo Busca Tabu

1: Busca Tabu (s)
2: $s^* \leftarrow s$; $Iter \leftarrow 0$; $MelhorIter \leftarrow 0$;
3: Seja BTmax o número máximo de iterações sem melhora em s^*;
4: Inicialize a função de aspiração A;
5: $T \leftarrow \emptyset$; *{Lista Tabu}*
6: **Enquanto** $Iter - MelhorIter \leq Btmax$ **faça**
7: $Iter \leftarrow Iter + 1$;
8: Seja $s' \leftarrow s \bigoplus m$ o melhor elemento de $V \subseteq N(s)$ tal que $m \notin T$ ou $(f(s') < A(f(s)))$;
9: Atualize a Lista Tabu T;
10: $s \leftarrow s'$;
11: **Se** $f(s) < f(s^*)$ **então**
12: $s^* \leftarrow s$;
13: MelhorIter ← Iter;
14: **Fim-se**
15: Atualize a função de aspiração A;
16: **Fim-enquanto**;
17: **Retorne** s;

Figura I.14 – Pseudocódigo da Busca Tabu.

II. CONSIDERAÇÕES FINAIS

Com este apêndice buscou-se apresentar uma série de métodos de resolução que permitisse a compreensão da área de estudo em que o modelo matemático desenvolvido para a obtenção dos resultados do Relatório 5 se enquadra. A ideia é dar um embasamento geral sobre os tipos de problemas existentes (em complemento ao já apresentado no Tomo I do Relatório 5), explicando como eles são tratados matematicamente e quais os métodos de solução disponíveis na literatura científica.

Cabe, elucidar, contudo, que o modelo matemático proposto está detalhado no corpo do relatório 5, sendo este apêndice apenas ilustrativo.

BIBLIOGRAFIA

ARENALES, M. N.; ARMENTANTO, V. A.; MORABITO, R.; YANASSE, H. H. 2007. Pesquisa Operacional. Campus/Elsevier, Rio de Janeiro/RJ.

BAZARAA M.; JARVIS J. J.; SHERALI H. F. 2009. Linear Programming and Network Flows. Wiley.

BAXTER, J. 1981. Local Optima Avoidance in Depot Location. The Journal of the Operational Research Society, n. 32, p. 815-819.

CAPRARA, A.; FISCHETTI, M. 1997. Branch-and-Cut Algorithms. In: DELL'AMICO, M.; MAFFIOLI, F.; MARTELLO, S. (Eds). Annotated Bibliographies in Combinatorial Optimization, Wiley, p. 45-64.

CERNY, V. 1985. Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm. Journal of Optimization Theory and Applications, n. 45, p. 41-51.

COOK, W. J.; CUNNINGHAM, W. H.; PULLEYBLANK W. R.; SCHRIJVER A. 1997. Combinatorial Optimization. John Wiley & Sons, Inc., New York, NY, USA.

DANTZIG, G. B.; WOLFE P. 1960. Decomposition principle for linear programs. Operations Research, n. 8, p. 101–111.

FEO, T. A.; RESENDE, M. G. C. 1995. Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization, n. 6, p. 109-133.

GENDREAU, M.; POTVIN, J. Y. 2010. Handbook of Metaheuristics, International In: Operations Research & Management Science Series, 2ª Ed., vol. 146, Morgan Kaufmann Publishers.

GLOVER, F. 1977. Heuristics for Integer Programming using Surrogate Constraints. Decision Sciences, n. 8, p. 156-166.

_____. 1996. Future Paths for Integer Programming and Links to Artificial Intelligence. Computers & Operations Research, n. 13, p. 533-549.

GOLDBERG, D. E. 1989. Genetic Algorithms in Search, Optimization and Machine Learning, 1ª Ed., Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

GOMES, L. C. T. 2006. Inteligência Computacional na Síntese de Meta-heurísticas para Otimização Combinatória e Multimodal. Tese de Doutorado. Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação, Campinas/SP.

HELD M.; KARP, R. M. 1970. The traveling salesman problem and minimum spanning trees. Operations Research, n. 18, p. 1138–1162.

_____. 1971. The traveling salesman problem and minimum spanning trees: part II. Mathematical Programming, n. 1, p. 6–25.

HIGGINS, A. J. 2006. Scheduling of road vehicles in sugarcane transport: a case study. European Journal of Operational Research, n. 170, p. 987-1000.

HILLIER, F.; LIEBERMAN, G. 2006. Introdução a Pesquisa Operacional. Editora Campus.

HOLLAND, J. H. 1975. Adaptation in Natural and Artificial Systems. Michigan University Press, Ann Arbor, MI, USA.

IBM. 2013. ILOG CPLEX. IBM Software Group. User - Manual CPLEX, v. 12.5.

JENA, S. D. A. 2009. Mixed Integer Programming approach for sugar cane cultivation and harvest planning. Dissertação de Mestrado. Programa de Pós-Graduação em Informática. Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro/RJ.

JENA, S. D.; POGGI, M. 2013. Harvest planning in the Brazilian sugar cane industry via mixed integer programming. European Journal of Operational Research, n. 230, p. 374-384.

JÜNGER, M.; THIENEL, S. 1998. Introduction to ABACUS – A Branch-And-Cut System. Operations Research Letters, n. 22, p. 83-95.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. 1983. Optimization by Simulated Annealing Science, n. 220, p. 671-680.

LAWLER, E. L.; WOOD, D. E. 1966. Branch-And-Bound Methods: A Survey. Operations Research, n. 14, p. 699-719.

LOPEZ MILAN, E.; FERNANDEZ, S. M.; ARAGONES, L. M. P. 2006. Sugar cane transportation in Cuba, a case study. European Journal of Operational Research, n. 174, p. 374-386.

LOURENÇO, H. R.; MARTIN, O. C.; STTUTZLE, T. 2003. Iterated Local Search. In: GLOVER, F.; KOCHENBERGER, G. (Eds). Handbook of Metaheuristics. Kluwer Academic Publishers, p. 321-354.

MLADENOVIC N.; HANSEN, P. 1997. Variable Neighborhood Search. Computers & Operations Research, n. 24, p. 1097-1100.

OSMAN, I.; LAPORTE, G. 1996. Metaheuristics: A Bibliography. In: Annals of Operations Research, n. 63, p. 511-623.

SILVER, E. A.; VICTOR, R.; VIDAL, V.; DE WERRA, D. 1980. A tutorial on heuristic methods. European Journal of Operational Research, n. 5, p. 513-520.

TALBI, E. 2009. Metaheuristics From Design to Implementation. Wiley.

WOLSEY, L. A. 1998. Integer Programming. Wiley-Interscience.

WOLSEY, L. A.; NEMHAUSER, G. L. 1999. Integer and Combinatorial Optimization. Wiley-Interscience.