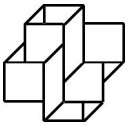


Laboratório Nacional de Computação Científica

Programa de Pós-Graduação em Modelagem Computacional

Metaheurísticas para Problemas de Otimização em Dois Níveis

Por

Jaqueline da Silva Angelo

PETRÓPOLIS, RJ - BRASIL

SETEMBRO DE 2014

METAHEURÍSTICAS PARA PROBLEMAS DE OTIMIZAÇÃO EM DOIS NÍVEIS

Jaqueline da Silva Angelo

TESE SUBMETIDA AO CORPO DOCENTE DO LABORATÓRIO NACIONAL DE COMPUTAÇÃO CIENTÍFICA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM MODELAGEM COMPUTACIONAL

Aprovada por:

Prof. Helio J. C. Barbosa, D.Sc
(Presidente)

Prof. Laurent E. Dardenne, D.Sc.

Prof. Fernanda M. P. Raupp, D.Sc.

Prof. Ricardo H. C. Takahashi, D.Sc.

Prof. Nelson F. F. Ebecken, D.Sc

Prof. Alexandre G. Evsukoff, D.Sc

PETRÓPOLIS, RJ - BRASIL
SETEMBRO DE 2014

Angelo, Jaqueline da Silva

S586u Metaheurísticas para problemas de otimização em dois níveis / Jaqueline da Silva Angelo. Petrópolis, RJ. : Laboratório Nacional de Computação Científica, 2014.

xvii, 183 p. : il.; 29 cm

Orientador: Helio J. C. Barbosa

Tese (D.Sc.) – Laboratório Nacional de Computação Científica, 2014.

1. Otimização matemática 2. Otimização em dois níveis 3. Metaheurísticas 4. Metamodelos. I. Barbosa, Helio J. C.. II. LNCC/MCTI. III. Título.

CDD 003

"Vai, ó preguiçoso, ter com a formiga,
observa seu proceder e torna-te sábio;
Ela não tem chefe, nem inspetor, nem
mestre;
Prepara no verão sua provisão, apanha no
tempo da ceifa sua comida."
Provérbio 6:6

À Daniel Giglio, meu maior incentivador e companheiro inseparável.

Agradecimentos

Primeiramente gostaria de agradecer aos meus pais, Nelson e Cleonice, por todo apoio, amor e carinho que sempre me deram.

Agradeço aos professores e colegas do LNCC que contribuíram de alguma forma com minha formação durante o período do doutorado.

Ao meu noivo Daniel Giglio, que sempre está do meu lado, me apoiando, alegrando minha vida e me ajudando a ser uma pessoa melhor. Sem ele eu não teria chegado onde cheguei.

À Eduardo Krempser, pela sua amizade e colaboração nos diversos trabalhos em que estivemos envolvidos, e nos muitos outros que estão por vir.

Agradeço imensamente aos membros da banca pela participação e contribuição para o aperfeiçoamento desta tese.

Um agradecimento especial ao meu orientador Helio Barbosa. Uma pessoa admirável que me surpreende a cada dia com seu conhecimento e inteligência, e que sempre me faz ver o quanto eu ainda tenho que aprender.

Por último, mas não menos importante, agradeço a Deus por me dar o privilégio de conhecer e fazer parte da vida de todas essas pessoas, mesmo que, para algumas, por um curto período de tempo.

Resumo da Tese apresentada ao LNCC/MCTI como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

METAHEURÍSTICAS PARA PROBLEMAS DE OTIMIZAÇÃO EM DOIS NÍVEIS

Jaqueline da Silva Angelo

Setembro, 2014

Orientador: Helio J. C. Barbosa, D.Sc

Este trabalho visa o desenvolvimento e implementação computacional de algoritmos robustos e eficientes para tratar problemas de otimização multinível, particularmente os de dois níveis. Problemas desta natureza são caracterizados por possuírem um problema de otimização dentro das restrições de outro problema de otimização, sendo considerados mais difíceis de serem tratados do que os problemas clássicos de otimização, pois, em geral, não são convexos e nem diferenciáveis, mesmo quando as funções envolvidas são todas lineares. Para resolver tais problemas, diferentes técnicas de otimização foram desenvolvidas, utilizando como base as metaheurísticas de Otimização por Colônia de Formigas e a Evolução Diferencial. Desenvolveu-se também um modelo de substituição (metamodelo), baseado no Método dos Vizinhos mais Próximos, na tentativa de reduzir o custo computacional em um dos métodos propostos. Uma diversidade de problemas em dois níveis foi utilizada para validar os algoritmos desenvolvidos, incluindo: (i) problemas de otimização no espaço contínuo (restritos e irrestritos); (ii) uma aplicação em Pesquisa Operacional envolvendo o planejamento de produção e distribuição; e (iii) problemas envolvendo múltiplos seguidores no nível inferior. A análise da aplicabilidade e do desempenho das metodologias propostas mostraram que estas foram capazes de resolver com sucesso todos os problemas, onde resultados competitivos foram obtidos na linha dos problemas abordados.

Abstract of Thesis presented to LNCC/MCTI as a partial fulfillment of the requirements for the degree of Doctor of Sciences (D.Sc.)

METAHEURISTICS FOR BILEVEL OPTIMIZATION PROBLEMS

Jaqueline da Silva Angelo

September, 2014

Advisor: Helio J. C. Barbosa, D.Sc

This work aims at the development and implementation of robust and efficient computational algorithms to treat multilevel optimization problems, particularly bilevel problems. Those problems are characterized by an optimization problem within the constraints of another optimization problem, and are considered more difficult to treat than classical optimization problems, since, in general, they are non-convex nor differentiable, even when the functions involved are all linear. To solve those problems, different techniques were developed which are based on Ant Colony Optimization and Differential Evolution metaheuristics. Besides those, a surrogate model (metamodel) was also developed, based on the Nearest Neighbors Method, in an attempt to reduce the computational cost of one of the proposed methods. A variety of bilevel problems were addressed to validate the proposed algorithms, including: (i) optimization problems in continuous space with and without constraints; (ii) an application in Operational Research involving the production and distribution planning problem; and (iii) bilevel problems containing multiple followers in the lower level. The analysis of the applicability and the performance of the proposed methodologies showed that they were able to successfully solve all problems, in which competitive results were obtained concerning the applications addressed.

Sumário

Lista de Figuras

Figura

Lista de Tabelas

Tabela

Capítulo 1

Introdução

Ao longo dos anos, um ramo da programação matemática que vem se tornando uma importante área de pesquisa, cheia de desafios, é o desenho e implementação de métodos computacionais eficientes, em particular as metaheurísticas de inspiração natural, para tratar problemas de otimização *multinível*.

A programação multinível é utilizada para modelar problemas descentralizados relacionando de forma hierárquica diversos tomadores de decisão. Neste modelo, as decisões são tomadas de forma sequencial sem nenhum tipo de cooperação. Cada decisor controla um subconjunto de variáveis, sendo que a decisão de cada nível depende das decisões dos demais níveis. No caso de dois níveis, pode-se pensar num líder –nível superior– que otimiza sua função objetivo considerando tanto suas restrições quanto a reação do seguidor –nível inferior–, que por sua vez toma sua decisão a fim de otimizar seu próprio objetivo, levando em conta o que foi imposto pelo líder.

Existem diversos problemas práticos que naturalmente envolvem decisões hierárquicas. Exemplos de aplicações podem ser encontradas nas áreas de (i) transporte: planejamento de redes [Costantino et al., 2013], estratégia ótima de definição de preços [Labbé and Violin, 2013; Pimentel and Fampa, 2012], planejamento de cronograma ferroviário [Meng et al., 2014]; (ii) gerenciamento: localização de instalações [Beresnev, 2013; Huijun et al., 2008], planejamento de

produção e distribuição [Calvete et al., 2011]; (iii) engenharias: desenho ótimo de estruturas [Friedlander and Gomes, 2011; Ghotbi and Dhingra, 2012; Barbosa, 1997], equilíbrio ótimo de substâncias químicas [Dempe, 2002]; (iv) economia: políticas sociais, de tributação e ambientais [Sinha et al., 2014, 2013], planejamento econômico [Wei et al., 2014; Safari et al., 2014]; entre outras [Colson et al., 2007; Dempe, 2003a, 2002].

Os problemas de programação com dois níveis são difíceis de serem resolvidos pois caracterizam-se por conter um problema de otimização como parte das restrições de outro problema de otimização. Além disso, eles são considerados mais difíceis de serem tratados que os problemas clássicos de otimização, pois, em geral, não são convexos e nem diferenciáveis, mesmo quando as funções envolvidas são todas lineares. Mostrou-se em [Hansen et al., 1992; Ben-Ayed and Blair, 1990] que problemas desta natureza são classificados como $\mathcal{NP}$-difíceis, ou seja, ainda não são conhecidos algoritmos capazes de garantidamente determinar seu ótimo em tempo polinomial.

Embora existam muitas aplicações em que a modelagem multinível seja adequada, na prática, exemplos simples e ilustrativos são considerados devido principalmente à falta de algoritmos eficientes para tratar as dificuldades inerentes deste tipo de aplicação, como não-convexidade e não-diferenciabilidade [Colson et al., 2005a]. Assim, não é de se surpreender que grande parte dos métodos desenvolvidos focam nos casos considerados "simples", que possuem propriedades bem comportadas, como nos problemas em que as funções objetivo e/ou restrições envolvidas são lineares, quadráticas ou convexas, com conjuntos viáveis convexos. Em particular, a instância de problemas de programa multinível mais estudada foi por um longo tempo aquela envolvendo apenas o caso linear [Colson et al., 2007].

Boa parte dos trabalhos encontrados nesta área de pesquisa são dedicados ao estudo das restrições, condições de otimalidade, e existência e unicidade de soluções. A técnica de solução mais frequentemente utilizada consiste na transformação do problema em dois níveis em um problema de um único nível,

através das condições de otimalidade, visando utilizar métodos de otimização clássicos para resolver o problema "transformado". Contudo, nem sempre é possível aplicar esta abordagem, como por exemplo, quando as funções objetivo e/ou restrições não são conhecidas explicitamente mas sim obtidas via simulação. Além disso, o problema de único nível resultante pode não ser equivalente ao problema original multinível [Dempe, 2002; Bard, 1998].

Uma características destes problemas é que, por definição, para cada conjunto de variáveis fixado pelo líder, a reação racional do seguidor é responder de maneira ótima de acordo com sua função objetivo. Uma das dificuldades que surgem na resolução de tais problemas é que, se a resposta do seguidor não for unicamente valorada, o líder pode não alcançar o valor mínimo da sua função objetivo, uma vez que o seguidor possui múltiplas mínimas soluções a responder. Neste caso, não existe garantia de que a resposta do seguidor será a melhor do ponto de vista do líder, levando a soluções sub-ótimas para o problema líder [Bard, 1998].

Outra dificuldade recai no fato de que, a menos que a solução do nível inferior seja ótima, ela não será viável para o problema como um todo. Este fato sugere que métodos aproximativos não poderiam ser utilizados para resolver o problema do seguidor, uma vez que não garantem a obtenção da solução ótima exata, mas uma boa aproximação desta. No entanto, do ponto de vista prático, soluções quase-ótimas ou *satisfatórias*[1] são plenamente aceitáveis [Deb and Sinha, 2009], especialmente quando a solução exata do problema seguidor for muito custosa, tornado o uso de métodos exatos impraticável.

Com isso, identifica-se a necessidade de se investir em técnicas mais robustas, que forneçam soluções de qualidade em tempo computacional viável, capazes de lidar de forma eficiente com as diversas dificuldades deste tipo de aplicação, como as metaheurísticas.

Embora muitas pesquisas venham sendo feitas na área de programação multinível com uma série de materiais científicos publicados [Colson et al., 2005a;

[1] Uma solução satisfatória não necessariamente é ótima porém atende a necessidade do tomador de decisão.

Dempe, 2003b], existem apenas cinco livros dedicados à área: [Shimizu et al., 1997; Bard, 1998; Dempe, 2002; Sakawa and Nishizaki, 2009; Talbi, 2013], e dentre estes apenas um (editado) dedica seus capítulos a propostas envolvendo metaheurísticas [Talbi, 2013], que é o objeto de nosso interesse aqui.

1.1 Por que metaheurísticas?

A pesquisa em procedimentos matemático-computacionais inspirados na Natureza, em especial na Biologia, já fornece subsídios relevantes para a resolução de problemas práticos em diversas áreas de pesquisa. Tem-se como exemplos as redes neuronais artificiais, inspiradas originalmente na pesquisa sobre o comportamento dos neurônios do cérebro, os algoritmos géneticos, inspirados pela Genética e o pelo processo de evolução das espécies (Darwinismo), e a otimização por colônia de formigas, inspirada na formação de trilhas quando em busca de alimentos em uma colônia de formigas.

As metaheurísticas em geral constituem uma ferramenta muito versátil e robusta para a solução de diversos problemas de otimização, distinguindo-se das técnicas usuais de programação matemática por: (i) empregar uma *população* de indivíduos, ou soluções candidatas; (ii) empregar regras de transição *probabilísticas*; e (iii) não exigir informações adicionais (derivadas, por exemplo) sobre a função a otimizar.

Assim, a busca de soluções pode se dar em conjuntos não-convexos e mesmo disjuntos, com funções objetivo também não-convexas e não-diferenciáveis, podendo-se trabalhar simultâneamente com variáveis reais, lógicas e inteiras. É de se ressaltar também que as metaheurísticas possuem artifícios para não se prenderem a mínimos locais, como é o caso dos métodos clássicos de otimização [Gaspar-Cunha et al., 2013].

Desta forma, este trabalho busca o desenvolvimento, implementação computacional e análise da aplicabilidade e desempenho de metaheurísticas que sejam eficazes na resolução de problemas de otimização em dois níveis. A motivação

para o uso destas técnicas se deve a diversos fatores:

- pode-se construir metaheurísticas para a obtenção *simultânea* de várias soluções para um dado problema, no caso de multimodalidade da função objetivo, proporcionando assim a possibilidade de comparação e escolha entre possíveis soluções para um dado problema.

- elas podem ser facilmente adaptadas para problemas de otimização com múltiplos objetivos, possibilitando obter uma aproximação do conjunto ótimo de Pareto (fronteira eficiente) contendo as soluções não-dominadas.[2] Nesta área, a computação evolucionista tem se mostrado bastante eficaz, em parte por sua natureza populacional, que possibilita a aproximação de toda a fronteira de Pareto numa só execução do algoritmo.

- está sempre presente nas MHIN a possibilidade da *hibridização*: pode-se sempre acoplar a elas um algoritmo já existente –especializado para o problema em questão– para efetuar uma busca local mais agressiva. O algoritmo híbrido resultante mantém entretanto a exploração global do espaço de soluções, característica da metaheurística original.

1.2 Proposta do trabalho

Pretende-se desenvolver nesta tese um conjunto de métodos computacionais eficazes para resolver diversos problemas de otimização multinível, em particular, os de dois níveis.

Dentre as metaheurísticas existentes, duas técnicas serão utilizadas para a resolução dos problemas em dois níveis abordados: a Otimização por Colônia de Formigas (ACO da sigla inglês para *Ant Colony Optimization*) e o método de Evolução Diferencial (DE da sigla em inglês para *Differential Evolution*). A fim de melhorar o desempenho de um dos métodos propostos, tanto na busca por melhores soluções quando na redução do custo computacional, um modelo de substituição,

[2] Soluções não-dominadas são aquelas em que só é possível melhorar um objetivo introduzindo uma piora em algum outro objetivo.

ou metamodelo, será desenvolvido para substituir avaliações computacionalmente intensivas do nível inferior por uma aproximação relativamente barata dentro do processo de otimização.

Os métodos desenvolvidos serão testados e analisados em diferentes problemas de otimização em dois níveis, a fim de avaliar seu desempenho e sua capacidade de obter soluções de qualidade.

Os problemas de otimização abordados, incluem funções objetivo e de restrições diversas (lineares e não-lineares) bem como um problema presente na área de Pesquisa Operacional envolvendo o esquema de planejamento de produção e distribuição. Além destes, problemas envolvendo múltiplos seguidores no nível inferior também serão abordados.

1.3 Organização da Tese

Os conceitos, metodologias, problemas abordados e experimentos computacionais estão organizados da seguinte forma:

No capítulo 2, serão apresentados os conceitos envolvendo o problema de otimização em dois níveis, onde serão descritas: a definição de solução ótima, conhecida como solução de equilíbrio de Stackelberg; as principais dificuldades envolvendo problemas desta natureza; as diferentes abordagens para o tratamento deste tipo de problema; a relação da programação em dois níveis com outros importantes problemas de programação matemática; e uma visão geral sobre alguns métodos de resolução, incluindo as metaheurísticas, utilizados para resolver este tipo de problema.

No capítulo 3, o problema de programação com múltiplos seguidores será apresentado, onde os casos com seguidores dependentes e independentes serão discutidos e analisados. Assim como no problema com um único seguidor, esta classe de aplicação também se mostra desafiadora tanto do ponto de vista da obtenção de soluções quanto no desenvolvimento de métodos de otimização eficazes.

As metaheurísticas de Otimização por Colônia de Formigas e Evolução

Diferencial serão descritas no capítulo 4. Utilizando estas técnicas, seis algoritmos foram desenhados na tentativa de resolver eficientemente os problemas de interesse.

O capítulo 5 apresenta todos os problemas multinível utilizados para validar os algoritmos propostos. Estes problemas foram divididos em três classes: 1. problemas de otimização linear e não-linear, com ou sem restrições; 2. problemas de planejamento de produção e distribuição; e 3. problemas contendo múltiplos seguidores. Assim, duas técnicas diferentes foram desenvolvidas para tratar cada classe de problemas, totalizando seis métodos.

Os três capítulos seguintes são dedicados exclusivamente à descrição, análise da aplicabilidade, e desempenho dos métodos desenvolvidos, onde o capítulo 6 apresenta os métodos propostos e os experimentos realizados na abordagem dos problemas da classe 1, o capítulo 7 apresenta os métodos e experimentos para a abordagem dos problemas da classe 2 e o capítulo 8 apresenta os métodos e experimentos para a abordagem dos problemas da classe 3.

Por fim, o capítulo 9 apresenta as conclusões do trabalho, bem como perspectivas para futuros desenvolvimentos em torno dos temas abordados e de novos temas.

Capítulo 2

Problema de Otimização em Dois Níveis

O problema de otimização em dois níveis tem em sua origem duas linhas de investigação. A primeira vem da teoria de jogos[1] quando, em 1934, o economista alemão Heinrich von Stackelberg utilizou o problema para descrever situações de oligopólio[2] em economia de mercado [Stackelberg, 1934]. Porém, o problema de *programação em dois níveis ou multinível* foi apresentado à comunidade científica apenas em 1973/74 por Bracken e McGill [Bracken and McGill, 1973, 1974].

O modelo de Stackelberg foi baseado no modelo descrito pelo economista francês Antoine A. Cournot (1838), onde empresas competidoras, que fabricam produtos homogêneos, decidem sobre as quantidades de produtos a serem produzidos. A diferença entre estes dois modelos é que no modelo de Cournot as empresas decidem simultaneamente as quantidades, enquanto que no de Stackelberg as empresas decidem sequencialmente, ou seja, uma após a outra. Neste último, a empresa que assume a posição de líder decide primeiro, levando em conta o efeito desta decisão no comportamento de sua rival, que assume o papel de seguidora. A empresa seguidora então observa a ação da líder e em seguida decide sobre suas quantidades.

A empresa líder tem total conhecimento sobre a reação da seguidora, que age

[1] A teoria de jogos é uma teoria matemática utilizada para modelar fenômenos que se manifestam quando dois ou mais agentes de decisão/jogadores interagem entre si. Ou ainda, como a teoria dos modelos matemáticos que estuda a escolha de decisões ótimas sob condições de conflito.

[2] O oligopólio ocorre quando um grupo de empresas promove o domínio de determinada oferta de produtos e/ou serviços.

de forma racional visando seu objetivo. Com isso, o líder pode fazer bom uso desta informação, no sentido de aumentar seus ganhos. De fato, ao contrário do modelo descrito por Cournot, o líder poderá ter lucro superior aos da empresa seguidora, pois se beneficia do fato de conhecer sua concorrente e saber exatamente como ela irá reagir dada sua estratégia. Esta situação de mercado favorece à empresa líder, pois permite que esta ganhe uma vantagem competitiva em relação à seguidora.

O modelo de Stackelberg pode ser visto como o resultado de uma estratégia de antecipação da empresa líder, criando assim uma situação de assimetria entre diferentes empresas. Uma situação de mercado onde tal assimetria pode existir é quando existe uma empresa já instalada no mercado e outra que deseja entrar nesse mesmo mercado. Nesse contexto, é natural pensar que a empresa já estabelecida tenha a iniciativa e decida primeiro sobre a produção/oferta de determinado produto, tomando a posição de líder. É razoável então que a empresa que deseja entrar no mercado decida sobre suas quantidades assumindo a produção da empresa estabelecida como um dado, tornando-se a empresa seguidora.

A outra linha de investigação da otimização em dois níveis vem da programação matemática e data de 1973/74, onde Bracken e McGill descreveram a programação em dois níveis como um problema de otimização que tem em suas restrições um outro problema de otimização [Bracken and McGill, 1973, 1974]. Neste caso, um subconjunto das variáveis de decisão do problema é restrito a ser solução de um outro problema de otimização parametrizado pelas variáveis restantes. Assim, o conjunto das variáveis é dividido em duas partes (x, y), no caso de dois níveis. As variáveis x são controladas pelo líder e as variáveis y são controladas pelo seguidor, parametrizadas por x. Desta forma, o líder otimiza sua função objetivo levando em consideração suas restrições e a reação do seguidor, que escolhe sua melhor estratégia tendo como base a decisão imposta pelo líder.

A principal característica de problemas desta natureza é que a decisão de um nível pode influenciar a tomada de decisão dos demais níveis, mas não controlar completamente suas ações. Assim, a função objetivo de um nível é geralmente,

em parte, determinada por variáveis controladas por um outro nível da hierarquia, que pode operar em paralelo ou em níveis subordinados.

Segundo Bard, os problemas de otimização multinível compartilham das seguintes particularidades [Bard, 1998]:

- a decisão de cada nível é feita de forma interativa onde predomina uma estrutura de hierarquia;
- cada nível subordinado estabelece sua política de decisão apenas após, e em função de, o nível superior definir suas ações;
- cada nível busca otimizar uma função objetivo própria que é, no entanto, afetada por ações externas provenientes de um outro nível; e
- ações externas que afetam o problema do tomador de decisão são refletidas em sua função objetivo e no conjunto de soluções viáveis.

O modelo de programação multinível pode ser incorporado a qualquer problema que siga um padrão de hierarquia, contendo um ou mais tomadores de decisão no nível superior e um ou mais decisores, subordinados ou não, no nível inferior, onde as ações e recompensas de um nível são afetadas pelas ações dos demais níveis.

A complexidade do problema de programação multinível pode ser avaliada considerando o caso mais simples do problema com dois níveis. Os problemas mais simples de programação em dois níveis, aqueles com funções contínuas e lineares, são classificados como $\mathcal{NP}$-difícil [Hansen et al., 1992; Ben-Ayed and Blair, 1990]. Obviamente que a complexidade destes problemas aumenta significativamente quando o número de níveis aumenta e quando o problema envolve mais de um líder e/ou seguidor [Blair, 1992].

2.1 Modelo matemático

O problema de programação em dois níveis (PDN) pode ser formulado da seguinte forma:

$$\begin{aligned} \min_{x \in X, y \in Y} \quad & F(x, y(x)) \\ \text{sujeito a} \quad & G(x, y(x)) \leq 0 \\ & y(x) \in \arg\min_{y \in Y} f(x, y) \\ & \text{sujeito a} \quad g(x, y) \leq 0 \end{aligned} \tag{2.1}$$

As variáveis do problema (2.1) são divididas em duas classes, *variáveis do nível superior* $x \in X \subset \mathbb{R}^n$ e *variáveis do nível inferior* $y \in Y \subset \mathbb{R}^m$. De forma similar, as funções $F : \mathbb{R}^{n+m} \to \mathbb{R}$ e $f : \mathbb{R}^{n+m} \to \mathbb{R}$ são as *funções objetivo* do *nível superior* e *inferior* respectivamente, enquanto que $G : \mathbb{R}^{n+m} \to \mathbb{R}^p$ e $g : \mathbb{R}^{n+m} \to \mathbb{R}^q$ são as restrições do *nível superior* e *inferior* respectivamente.

Neste problema, o tomador de decisão do nível superior –*líder*– tem controle sobre as variáveis x, enquanto que o tomador de decisão do nível inferior –*seguidor*– controla as variáveis y, sendo $y = y(x)$ parametrizado por x. Reagindo à decisão do líder, as variáveis y são definidas em resposta às decisões x do líder. Desta forma, a decisão do líder pode influenciar a tomada de decisão do seguidor, porém não pode controlar completamente suas ações.

As seguintes definições relativas ao problema em dois níveis podem ser apresentadas [Bard, 1998]:

- Conjunto das restrições do problema de dois níveis:
  $\Omega := \{(x, y) : x \in X, y \in Y, G(x, y) \leq 0, g(x, y) \leq 0\}$
- Conjunto viável do seguidor, para cada $x \in X$:
  $\Omega_y := \{y \in Y : g(x, y) \leq 0\}$
- Projeção de Ω no espaço de decisão do líder:
  $P(X) := \{x \in X : \exists y \in Y, G(x, y(x)) \leq 0, g(x, y) \leq 0\}$

- Reação racional do seguidor para $x \in P(X)$:

  $R(x) := \{y \in Y : y \in \arg\min\{f(x, \hat{y}) \mid \hat{y} \in \Omega_y\}\}$

- Região induzida[3] (ou conjunto viável) da PDN:

  $RI_{PDN} := \{(x, y) : (x, y) \in \Omega, y \in R(x)\}$

Qualquer solução $(x, y) \in RI_{PDN}$ é chamada de solução viável. Para garantir que o problema (2.1) seja bem posto, assume-se que para todo $x \in X$, o problema inferior irá apresentar pelo menos uma resposta, ou seja, dado um x, $R(x) \neq \emptyset$. Porém, mesmo sob esta condição, o problema (2.1) pode não ter solução. Em particular, se $R(x)$ não for unicamente valorado, ou seja, se o seguidor possuir mais de uma solução ótima, para todos os possíveis x, o líder pode não alcançar seu valor ótimo, pois não há garantias de que o seguidor irá responder com a melhor solução do ponto de vista do líder. Um exemplo deste caso pode ser visto em [Bard and Falk, 1982].

Por este fato, na maioria dos trabalhos encontrados na literatura, assume-se que o seguidor possui uma única solução ótima para cada x fixado pelo líder, ou seja, define-se $R(x)$ como uma aplicação ponto-ponto, e assim $y = R(x)$. Do contrário, $R(x)$ pode ser definido como uma aplicação ponto-conjunto, e neste caso o seguidor terá múltiplas soluções ótimas a responder para cada x. Nesta situação, o líder pode assumir um comportamento pessimista ou otimista diante da resposta do seguidor, gerando assim as abordagens *otimista* e *pessimista* da programação em dois níveis, que serão brevemente apresentadas na seção 2.3.

2.1.1 Existência de solução

Diferentemente dos problemas de programação matemática clássicos, os problemas de programação em dois níveis podem não ter solução, mesmo quando F, f, G e g são funções contínuas e os conjuntos X e Y são compactos [Bard, 1998, 1991]. Uma outra característica que deve ser levada em consideração é que, por

[3] Este conjunto é geralmente não-convexo e pode ser descontínuo ou mesmo vazio, na presença de restrições no nível superior. Mais detalhes serão vistos na seção 2.2.

definição, a PDN requer que o líder estabeleça suas variáveis primeiro e o seguidor em sequência, ou seja, a ordem dos decisores também influência a existência de solução [Bard, 1998].

A garantia de existência de solução de um problema de PDN depende basicamente da caracterização da reação do seguidor, isto é, da caracterização da aplicação $R(x)$ como sendo uma aplicação ponto-ponto ou ponto-conjunto. Quando $R(x)$ é uma aplicação ponto-ponto, i.e., se a solução ótima do problema seguidor for unicamente determinada para qualquer valor de x, é possível utilizar o Teorema de Bolzano-Weierstrass[4] para provar a existência de solução ótima no problema de PDN [Dempe, 2002; Harker and Pang, 1988]. No entanto, esta condição só é válida para os problemas em dois níveis que não possuem restrições no nível superior.

O teorema a seguir indica uma condição suficiente para que a aplicação $R(x)$ seja unívoca, contínua e fechada.

Teorema 2.1 *Se para cada $x \in X$, f e g são funções duas vezes continuamente diferenciáveis para todo $y \in \Omega_y$, f é estritamente convexa para todo y em Ω_y e o conjunto Ω_y é compacto e convexo, então $R(x)$ é uma aplicação unívoca, contínua e fechada [Vicente, 1992].*

Já o próximo teorema estabelece condições suficientes para a existência de solução de um problema de PDN, sem restrição no nível superior.

Teorema 2.2 *Se além de serem válidas as hipóteses do teorema anterior, F é contínua em x e em y e X é um conjunto compacto, então existe sempre uma solução para o problema [Vicente, 1992].*

Para o caso de $R(x)$ não ser unicamente valorado, i.e., o seguidor têm mais de uma solução ótima para qualquer x fixado pelo líder, estudos também foram realizados tanto para a abordagem otimista quanto para a pessimista [Dempe, 2002; Lucchetti et al., 1987].

[4] O Teorema de Bolzano-Weierstrass afirma que qualquer função contínua num conjunto compacto assume um mínimo e um máximo.

2.1.2 Condições de otimalidade

Na literatura podem ser encontrados diversos autores que estabeleceram condições de otimalidade para o problema de PDN, cada um considerando uma determinada formulação do problema [Vicente, 1992; Fernandes, 2005]. Iniciaremos pela definição de solução ótima de um problema em dois níveis, conhecida como *solução de equilíbrio de Stackelberg*:

Definição 2.1 *No problema (2.1), a solução minimizante $y^*(x)$ do seguidor em resposta à estratégia x do líder satisfaz à seguinte relação:*

$$f(x, y^*(x)) \leq f(x, y) \quad \forall y \in \Omega_y \tag{2.2}$$

Para tal $y^*(x)$, se existir um $x^* \in X$ tal que

$$F(x^*, y^*) \leq F(x, y^*(x)) \quad \forall x \in \Omega \tag{2.3}$$

então o par (x^*, y^*), onde $y^* = y^*(x^*)$, é chamado de solução de equilíbrio de Stackelberg e y^* é a solução ótima do seguidor em resposta a x^* [Shimizu and Aiyoshi, 1981].

Uma abordagem bastante utilizada no estudo de condições de otimalidade, consiste em transformar o problema em dois níveis em um único nível. Esta abordagem consiste em substituir o problema do nível inferior pela suas condições necessárias e suficientes de otimalidade de Karush-Kuhn-Tucker (KKT) e adicioná-las às restrições do problema líder, transformando o problema com dois níveis em um único nível. No entanto, esta abordagem só é possível quando o nível inferior envolve apenas funções convexas e diferenciáveis. Neste caso, para garantir que a resposta do seguidor seja única, e que portanto $R(x)$ é uma aplicação unívoca, considera-se que $F(x, y)$ é estritamente convexa em y para cada x. Entretanto, é importante destacar que mesmo sob estas condições o problema transformado pode não ser equivalente ao problema original [Dempe, 2002; Bard, 1998].

Teorema 2.3 *Uma condição necessária e suficiente para que* (x^*, y^*) *seja solução do problema (2.1) é que exista um* $\lambda^* \in \mathcal{R}^m$ *tal que* (x^*, y^*, λ^*) *solucione o seguinte problema:*

$$\begin{aligned} \min_{x,y,\lambda} \quad & F(x,y) \\ \textit{sujeito a} \quad & G(x,y) \leq 0 \\ & \bigtriangledown_y f(x,y) + \lambda \bigtriangledown_y g(x,y) = 0 \\ & g(x,y) \leq 0 \\ & \lambda g(x,y) = 0 \\ & \lambda \geq 0 \\ & (x,y) \in X \times Y \end{aligned} \tag{2.4}$$

onde $\lambda \in \mathcal{R}^m$ *é o vetor (linha) de multiplicadores de Lagrange e* $\bigtriangledown_y$ *é o operador gradiente com relação às variáveis y [Fernandes, 2005].*

É importante destacar que o problema (2.4) de único nível é não-convexo, e por isso, mínimos locais podem existir. Mesmo quando todas as funções consideradas são lineares, as condições de qualificação que surgem são dificilmente verificáveis [Vicente, 1992]. Ainda assim, essa formulação tem um papel importante no desenvolvimento de novos métodos de resolução pois possibilita o uso de técnicas tradicionais da programação matemática.

2.2 Dificuldades

Na PDN mesmo problemas considerados simples podem apresentar dificuldades, como é o caso do problema de PDN linear, onde todas as funções envolvidas são lineares. Um exemplo pode ser visto no seguinte problema descrito em [Bard, 1998]:

$$
\begin{aligned}
\min_{x\geq 0, y\geq 0} \quad & F(x,y) = x - 4y \\
\text{sujeito a} \quad & \min_{y\geq 0} f(y) = y \\
& \text{sujeito a} \quad -x - y \leq -3 \\
& \qquad\qquad -2x + y \leq 0 \\
& \qquad\qquad 2x + y \leq 12 \\
& \qquad\qquad -3x + 2y \geq -4
\end{aligned}
\tag{2.5}
$$

A Figura 2.1 ilustra a geometria do problema (2.5), indicando a região viável Ω, representada por todos os possíveis pontos dentro poliedro, e a região induzida RI, representada pelas linhas em negrito.

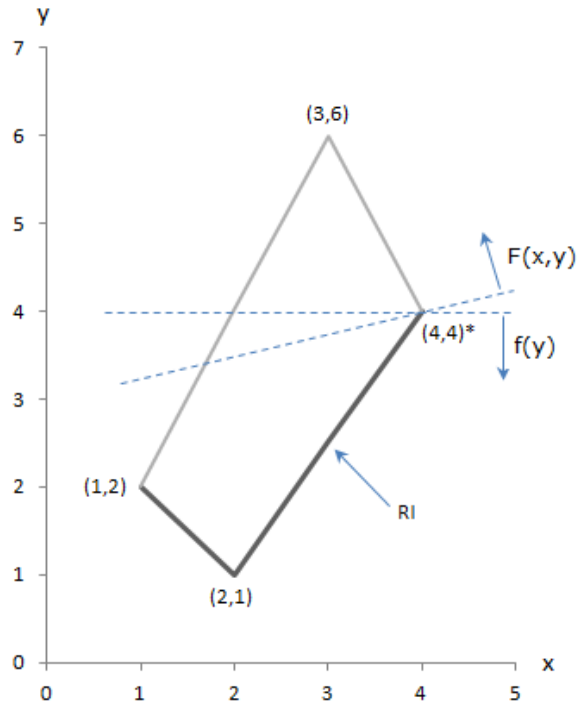


Figura 2.1: Conjunto viável e região induzida do problema (2.5).

Para um dado x, em que $1 \leq x \leq 4$, a região viável do seguidor Ω_y é representada pelos pontos na linha vertical, "em cima" da variável x em particular, contidos dentro do poliedro. Como o objetivo do seguidor é minimizar y, então a reação racional do seguidor é o menor ponto viável nesta linha vertical, dada por $R(x) = \max\{-x + 3, (3x - 4)/2\}$.

A solução de equilíbrio de Stackelberg deste exemplo ocorre no ponto $(x^*, y^*) = (4, 4)^*$ com $F^* = -12$ e $f^* = 4$. O líder poderia atingir um melhor valor em sua função objetivo no ponto $(3, 6)$, contudo este ponto está fora da região induzida. Analisando melhor este caso, se o líder escolhe $x = 3$, o seguidor irá responder com $y = 2.5$ levando a $F(3, 2.5) = -7$ e $f(2.5) = 2.5$, o que claramente é melhor para o seguidor, porém não para o líder, comparando-se com a solução de equilíbrio de Stackelberg. Além disso, observa-se que este problema possui dois minimizadores locais no problema líder: um no ponto $(1, 2)$ e outro em $(3, 2.5)$, que fornecem $F = -7$. Isso mostra que, mesmo um problema em dois níveis linear pode não ser convexo e, portanto, possuir minimizadores locais. Mais ainda, se a restrição $y \geq 1.5$ for incluída no problema líder, a região induzida torna-se desconexa, conforme ilustrado na Figura 2.2.

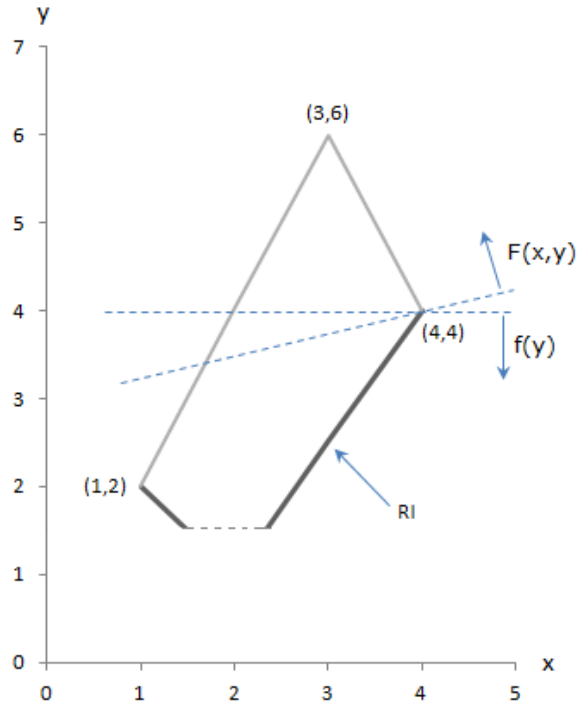


Figura 2.2: Região induzida desconexa do problema (2.5).

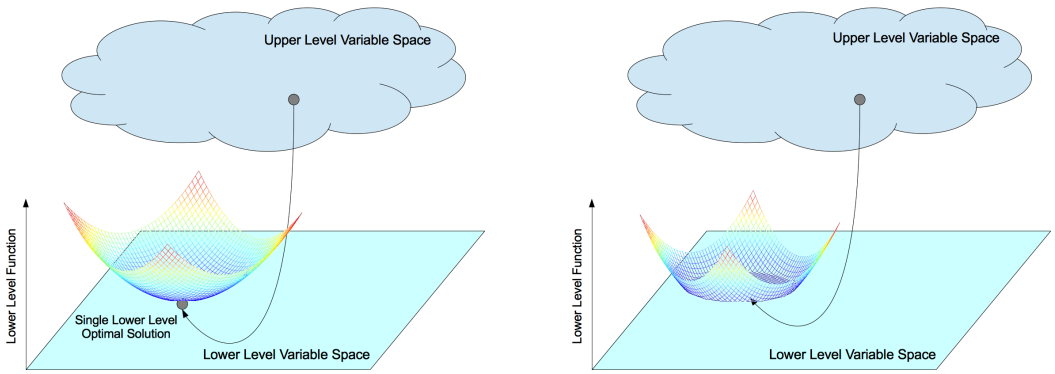
Isso mostra que problemas de programação em dois níveis são, em geral, sensíveis à localização das restrições. Mesmo a adição de uma restrição que não está ativa na solução ótima pode mudar drasticamente o problema [Dempe, 2002;

Macal and Hurter, 1997], o que não ocorre no caso do problema de um nível.

2.3 Abordagens otimista e pessimista

Como mencionado anteriormente, uma dificuldade que surge na PDN é que se a resposta do seguidor não for unicamente valorada para todos os possíveis x, então o líder pode não alcançar o valor ótimo de sua função objetivo, uma vez que o seguidor têm múltiplas soluções ótimas, que para ele são indiferentes, mas podem afetar de maneira positiva ou negativa a função objetivo do líder. Neste caso, não há garantias de que o seguidor irá retornar a melhor solução para o líder, levando a soluções sub-ótimas para o problema líder [Bard, 1998].

Uma ilustração da relação entre os níveis superior e inferior é apresentada pela Figura 2.3 [Sinha et al., 2012]. Na Figura 2.3a o problema de nível inferior é representado por uma parábola, com um único valor mínimo na função objetivo, correspondente à cada valor de x fixado no nível superior. A Figura 2.3b apresenta um cenário onde a função objetivo do nível inferior é representado por uma parábola "achatada" na base, indicando múltiplas soluções ótimas para o problema do nível inferior.



(a) Seguidor responde com uma única solução mínima.

(b) Seguidor têm múltiplas mínimas soluções para responder.

Figura 2.3: Relação entre o nível superior e inferior [Sinha et al., 2012].

Para tratar situações como estas, duas abordagens podem ser consideradas a *otimista* e a *pessimista*. Na abordagem dita otimista, o líder assume que o

seguidor está disposto a apoiá-lo, ou seja, que o seguidor irá selecionar uma solução $y \in R(x)$, que é a melhor do ponto de vista do líder. Esta abordagem resulta no chamado *problema de dois níveis fraco* ou *otimista* [Dempe, 2002]:

$$\begin{array}{lll} (L) & \min\limits_{x \in X} \min\limits_{y \in R(x)} & F(x,y) \\ & \text{sujeito a} & G(x,y) \leq 0 \\ & (S) & y \in R(x) := \arg\min\limits_{y \in Y} f(x,y) \\ & & \text{sujeito a} \quad g(x,y) \leq 0 \end{array} \tag{2.6}$$

Por outro lado, na abordagem pessimista, o líder se protege contra a pior situação possível, levando ao chamado *problema de dois níveis forte* ou *pessimista* [Dempe, 2002]:

$$\begin{array}{lll} (L) & \min\limits_{x \in X} \max\limits_{y \in R(x)} & F(x,y) \\ & \text{sujeito a} & G(x,y) \leq 0 \\ & (S) & y \in R(x) := \arg\min\limits_{y \in Y} f(x,y) \\ & & \text{sujeito a} \quad g(x,y) \leq 0 \end{array} \tag{2.7}$$

Conforme mencionado anterior, se $R(x)$ reduz-se a um único elemento para cada x, ou seja $y = R(x)$, o problema de dois níveis fica bem definido, pois a resposta do seguidor é única, não havendo então distinção entre estas duas abordagens.

As abordagens pessimista e otimista são normalmente utilizadas quando considera-se algum nível de cooperação entre os níveis. Por exemplo, quando o líder assume que pode influenciar a decisão do seguidor no sentido de apoiá-lo, então a abordagem otimista é a mais adequada. Esta abordagem também pode ser aplicada para o caso do seguidor poder participar dos ganhos obtidos pelo líder, obviamente que o seguidor terá total interesse em apoiá-lo [Bialas and Karwan, 1984]. Se a colaboração entre os níveis não for possível, então o líder é forçado a escolher uma abordagem que limita os danos resultantes de uma seleção

desfavorável do seguidor, onde enquadra-se a abordagem pessimista.

Um outro cenário onde o seguidor têm mais de uma opção para responder surge no problema em dois níveis onde o seguidor possui múltiplos objetivos para otimizar (uma breve descrição do problema de otimização multiobjetivo será apresentada na seção 2.4.2). Neste caso, a resposta do seguidor será um conjunto de soluções não-dominadas, que descrevem a melhor resposta para a solução do problema multiobjetivo. Ambas as abordagens otimista e pessimista também podem ser utilizadas neste contexto [Sakawa and Nishizaki, 2009].

2.4 Problemas relacionados

Outros importantes problemas na programação matemática, relacionados com a programação em dois níveis, são discutidas a seguir.

2.4.1 Problema de otimização "minimax"

O problema de otimização "minimax" (ou "maximin"), surge quando as funções objetivo dos dois níveis diferem apenas no sinal. Esta situação corresponde à otimização no pior caso, ou seja, busca-se pelo melhor desempenho no pior caso possível. Desta forma, considerando $F = -f$, o problema (2.1) pode ser simplificadamente reescrito como

$$\min_{x \in X} \max_{y \in Y} F(x, y) \tag{2.8}$$

O problema (2.8) pode ser visto como um jogo antagonista entre dois jogadores, onde o jogador A tenta encontrar a solução x que minimiza uma função objetivo, enquanto que o jogador B busca pela solução y que maximiza a mesma função objetivo. Assim, busca-se pela solução (x^*, y^*) que satisfaça

$$F(x^*, y) \leq F(x^*, y^*) \leq F(x, y^*) \quad \forall x \in X, y \in Y. \tag{2.9}$$

Como exemplo de aplicação, pode-se pensar num jogo "projetista versus

a natureza", num problema de projeto ótimo de estruturas. Neste problema, ilustrado em [Barbosa, 1997], o primeiro jogador –projetista– seleciona o vetor de áreas transversais das barras da estrutura, enquanto que o segundo jogador –natureza– pode aplicar uma carga externa arbitrária à esta estrutura. O objetivo do projetista é minimizar a função *compliance*[5] F, enquanto que a natureza visa maximizar essa função. Assim, a natureza pode escolher um conjunto de cargas que leva às mais severas condições da estrutura, selecionando o pior caso possível das ações externas (maximizar F). Por outro lado, o projetista tenta encontrar a resposta mais eficiente: aquela que minimiza as consequências daquele particular conjunto de ações (minimizar F).

2.4.2 Problema de otimização multiobjetivo

Na otimização multiobjetivo os objetivos do problema são otimizados de forma simultânea. Neste caso, busca-se por soluções de compromisso, ou soluções não-dominadas, que satisfaçam todos os objetivos.

Na programação multiobjetivo o conceito de dominância é utilizado para indicar que não existirá uma única solução ótima, mas sim um conjunto de alternativas que suplantam as demais soluções quando todos os objetivos são considerados. Assim, assumindo um problema onde k objetivos devem ser minimizados, diz-se que a solução x_1 domina uma outra solução x_2, quando

$$F_i(x_1) \leq F_i(x_2) \quad \forall i \in \{1, ..., k\} \quad \text{e} \quad \exists j \in \{1, ..., k\} : F_j(x_1) < F_j(x_2) \qquad (2.10)$$

i.e., a solução x_1 não é pior que x_2 em todos os objetivos e é melhor em pelo menos um deles. Desta forma, todos os possíveis pares de soluções podem ser comparados a fim de encontrar as soluções ditas não-dominadas, que são aquelas que apresentam melhor desempenho com relação à todos os objetivos. Estas soluções formam o conjunto ótimo de Pareto, ou fronteira de Pareto, no espaço dos objetivos.

[5] *Compliance* é a propriedade de um material sofrer deformações quando submetido a aplicação de uma força.

Soluções de um problema de otimização multiobjetivo, construídas a partir de um problema multinível, em geral, não são soluções viáveis para o problema multinível [Dempe, 2002].

Vários autores têm se interessado na relação entre problemas multinível e problemas multiobjetivo. Nos trabalhos [Bard, 1984; Ünlü, 1987] os autores alegam que uma solução ótima para o problema em dois níveis linear é uma solução não-dominada para os problemas de nível superior e inferior. No entanto, ainda não foram encontradas condições que garantam que a solução ótima de um problema de PDN é uma solução ótima de Pareto para o problema bi-objetivo associado. De fato, nos trabalhos [Candler, 1988; Clark and Westerberg, 1988; Wen and Hsu, 1989] podem ser encontrados contra-exemplos mostrando que a solução ótima de um problema em dois níveis e uma solução ótima de Pareto do problema multiobjetivo representam dois conceitos distintos.

Considere como exemplo o problema (2.5) agora formulado como um problema de otimização bi-objetivo da seguinte forma:

$$\begin{array}{ll} \min\limits_{x\geq 0, y\geq 0} & F(x,y) = x - 4y \\ \min\limits_{y\geq 0} & f(y) = y \\ \text{sujeito a} & -x - y \leq -3 \\ & 2x + y \leq 12 \\ & -2x + y \leq 0 \\ & -3x + 2y \geq -4 \end{array} \tag{2.11}$$

A Figura 2.4 apresenta a região viável do problema (2.11) que domina a solução ótima do problema em dois níveis, sendo este conjunto representado pela região em cinza formado pelos pontos A, B e C. Os segmentos entre os pontos (3,6), (1,2) e (2,1) contêm todas as soluções não-dominadas, que formam a fronteira ótima de Pareto no espaço de objetivos.

Note que, as únicas soluções não-dominadas que são viáveis para o problema em dois níveis (2.5) estão contidas no segmento de (1,2) a (2,1), que possuem

os piores valores para a função objetivo do líder. Isso sugere que o ferramental da otimização multiobjetivo não pode ser diretamente aplicado ao problema de otimização multinível.

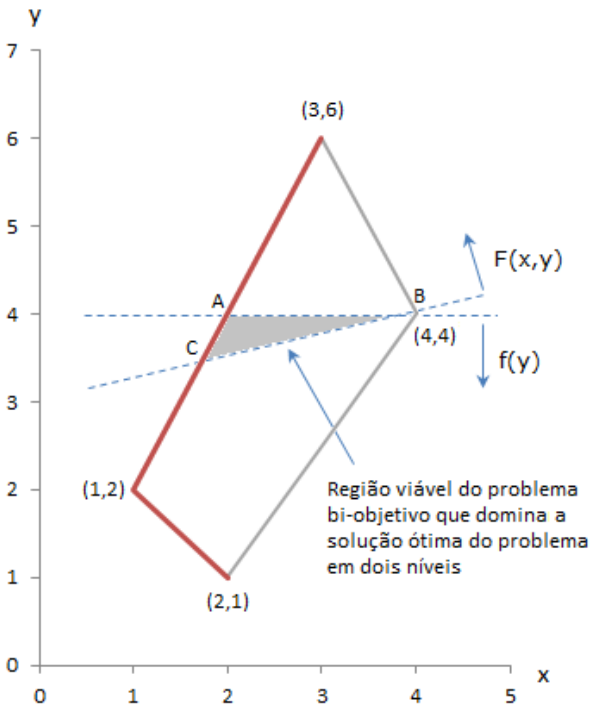


Figura 2.4: Região viável e soluções não-dominadas, no espaço das variáveis de decisão do problema bi-objetivo (2.11).

2.4.3 Problema de equilíbrio de Nash

Na teoria de jogos, o equilíbrio de Nash é um conceito de solução de um jogo não-cooperativo, envolvendo dois ou mais jogadores, onde os jogadores jogam ao mesmo tempo sem qualquer estrutura hierárquica. O equilíbrio de Nash consiste em uma solução onde nenhum jogador pode aumentar seu ganho alterando de forma unilateral sua estratégia.

Uma das mais famosas aplicações deste conceito é usada no jogo conhecido como Dilema do Prisioneiro. O Dilema do Prisioneiro surge quando dois suspeitos A e B são acusados de um mesmo crime e são presos em celas separadas, sem a possibilidade de se comunicarem. A seguinte proposta lhes é feita pelos policiais:

"você pode escolher entre confessar ou negar o crime". A decisão de cada um resulta nas seguintes situações: se ambos negarem, ambos serão presos por um ano; se os dois confessarem –delatarem um ao outro–, ambos serão presos por três anos; se um confessar e o outro negar, aquele que confessou será libertado enquanto que, o que negou, será submetido à uma pena de 10 anos de prisão. Assim, os possíveis resultados são descritos na Tabela 2.1.

Tabela 2.1: Possíveis resultados no Dilema do Prisioneiro.

		Suspeito B
Suspeito A	Confessar	Negar
Confessar	(3,3)	(0,10)
Negar	(10,0)	(1,1)

Desta tabela, assumindo a hipótese de ausência de cooperação prévia entre os suspeitos, a melhor opção para ambos é confessar, sendo esta (Confessar, Confessar) a solução de equilíbrio de Nash. Se analisarmos a questão do ponto de vista de cada jogador, cada um pode raciocinar da seguinte maneira: "Assim como eu, o outro prisioneiro pode negar ou confessar o crime. Se ele confessar, minha melhor opção é confessar também, pois ficarei preso por três anos em vez de dez. Se ele negar, minha melhor opção ainda é confessar, pois ficarei livre em vez de condenado a um ano. A melhor opção nos dois casos é confessar, portanto, eu confessarei". Se forem racionais, os prisioneiros pensarão desta maneira e ficarão presos por três anos.

Não é difícil perceber que existe uma maneira de melhorar a situação de um dos suspeitos sem piorar a situação do outro, sendo o equilíbrio de Nash neste caso uma solução que não é solução ótima de Pareto. Se ambos pudessem cooperar, eles decidiriam por negar o crime (Negar, Negar) pois isso reduziria a pena de ambos em dois anos. No entanto, isso não ocorre porque ambos seguem a racionalidade e estando neste ponto, ambos teriam um incentivo muito grande de confessar o crime: adquirir sua liberdade.

A ideia por trás do conceito de equilíbrio formulado pelo matemático John Forbes Nash Jr. [Nash, 1951] é que não se pode prever o resultado das escolhas de

vários tomadores de decisão se as decisões forem analisadas de forma isolada. Em vez disso, deve-se perguntar o que cada jogador faria, tendo em conta a tomada de decisão dos outros.

Para formalizar este conceito, suponha um caso com dois jogadores. O jogador A escolhe uma estratégia x e o jogador B escolhe simultaneamente uma estratégia y. O par de estratégias (x, y) é dito em *equilíbrio de Nash* se x é a melhor resposta para y e y é a melhor resposta para x. A motivação vem do seguinte fato: se os jogadores escolhem estratégias que são melhores respostas umas às outras, então nenhum jogador tem incentivo para mudar para uma estratégia diferente. Desta forma, busca-se pela solução (x^*, y^*), chamada *Solução de Equilíbrio de Nash*, que apresenta a seguinte característica

$$f_1(x^*, y^*) \leq f_1(x, y^*) \quad \forall x \in X \quad \text{e} \quad f_2(x^*, y^*) \leq f_2(x^*, y) \quad \forall y \in Y. \tag{2.12}$$

Como ilustração da diferença entre a estratégia de Stackelberg e a de Nash, suponha que deseja-se otimizar [Wang and Periaux, 2001]:

$$\begin{aligned} \min_x \quad & f_1(x, y) = (x-1)^2 + (x-y)^2 \\ \min_y \quad & f_2(x, y) = (y-3)^2 + (x-y)^2 \\ & -5 \leq x, y \leq 5 \end{aligned} \tag{2.13}$$

Na estratégia de Nash, os jogadores possuem o mesmo papel, e na de Stackelberg os papéis são definidos de forma hierárquica. A solução analítica das duas estratégias para o problema (2.13) é apresenta na Tabela 2.2.

Tabela 2.2: Comparação entre soluções na estratégia de Stackelberg e na de Nash.

		Stackelberg	
	Nash	f_1 como líder	f_2 como líder
x	1.666	1.4	1.8
y	2.333	2.2	2.6
f_1	**0.888**	**0.8**	**1.28**
f_2	**0.888**	**1.28**	**0.8**

Na estratégia de Stackelberg, para diferentes líderes, têm-se diferentes soluções, sendo estas diferentes da solução de equilíbrio de Nash. Neste caso, o decisor que assume o papel de líder alcança soluções melhores do que aquele que assume o papel o seguidor.

De acordo com Sefrioui e Periaux [Sefrioui and Periaux, 2000], pode ser muito difícil encontrar uma solução de equilíbrio de Nash utilizando uma abordagem clássica. Além disso, provar que uma determinada solução está em equilíbrio de Nash, é uma tarefa geralmente mais fácil do que obter efetivamente tal solução, principalmente se os objetivos relacionados envolvem funções não-diferenciáveis.

Veremos no capítulo seguinte, na seção 3.1.2, o problema em dois níveis envolvendo múltiplos seguidores. Em uma determinada classe destes problemas, o conceito de equilíbrio de Nash será utilizado para a resolução do nível inferior, que envolve dois seguidores. Nestes problemas, dada a variável fixada pelo líder os seguidores respondem, de maneira não-cooperativa, atendendo ao equilíbrio de Nash.

2.5 Métodos de resolução

A programação em dois níveis vem sendo estudada por diversos pesquisadores, tanto no campo dos métodos de otimização clássicos assim como utilizando métodos evolucionistas. Embora existam muitas aplicações em que a modelagem em dois níveis seja adequada, na prática, a resolução do problema na forma original em dois níveis é pouco utilizada, devido principalmente à falta de algoritmos eficientes para tratar as dificuldades inerentes deste tipo de aplicação [Colson et al., 2005a].

Além disso, as dificuldades que surgem na resolução deste tipo de problema, como não-convexidade e não-diferenciabilidade, impedem que métodos eficientes, utilizados para a resolução de problemas clássicos de programação matemática, possam ser diretamente aplicados na resolução dos problemas em dois níveis. Assim, não é de se surpreender que grande parte dos métodos desenvolvidos focam

nos casos considerados "simples", que possuem propriedades bem comportadas, como nos problemas em que as funções objetivo e/ou restrições envolvidas são lineares, quadráticas ou convexas. Em particular, a instância de problemas de PDN mais estudada foi por um longo tempo aquela envolvendo apenas o caso linear [Colson et al., 2007].

Como dito, a versão linear do problema em dois níveis é uma das mais estudadas, pois apresenta uma importante propriedade: a solução ótima do problema de PDN linear ocorre em um dos vértices da região viável. Esta propriedade motivou o desenvolvimento de métodos de otimização baseados em algum tipo de esquema de enumeração de vértices. Assim, com base nesta propriedade, Candler e Townsley [Candler and Townsley, 1982] e Bialas e Karwan [Bialas and Karwan, 1984] propuseram métodos que encontram a solução ótima global de problemas de PDN linear enumerando os pontos extremos da região viável. Dentre estes, o método conhecido como Késimo-Melhor (*Kth-Best Method*), proposto em [Bialas and Karwan, 1984], tem mostrado ser uma valiosa ferramenta de análise com uma gama de aplicações para a versão linear da PDN [Bard, 1998].

Nos casos mais gerais, envolvendo funções lineares e não-lineares, uma das estratégias mais utilizadas é investigar as propriedades da região induzida para obter condições (necessárias e suficientes) para substituir o problema do nível inferior por condições de Karush-Kunh-Tucker (KKT), transformando um problema com dois níveis em um único nível (conforme discutido na seção 2.1.2). Esta estratégia viabiliza o uso de métodos de otimização, clássicos ou evolucionistas, para tratar o problema com um único nível. No entanto, conforme mencionado anteriormente, nem sempre é possível aplicar esta abordagem, e geralmente, o problema resultante não é equivalente ao problema original como, por exemplo, quando o problema do nível inferior é não-convexo [Chinchuluun et al., 2009; Dempe, 2002].

Além disso, esta abordagem não pode ser utilizada nos problemas de otimização onde a forma analítica das funções objetivo e das restrições não

estão disponíveis, como é o caso na chamada otimização "black-box". Neste tipo de otimização a função objetivo, por exemplo, é determinada através de uma simulação ou de medidas experimentais que fornecem uma resposta para o problema, dado um conjunto de valores de entrada.

Além da abordagem de enumeração de vértices, aplicada apenas na PDN linear, outros métodos clássicos de otimização comumente utilizados englobam as técnicas: (i) Branch-and-Bound [Beresnev, 2013; Shi et al., 2006]; (ii) funções de penalização [Zheng et al., 2012; Ishizuka and Aiyoshi, 1992; Aiyoshi and Shimizu, 1984]; (iii) métodos de descida [Takemura, 2007; Dempe, 2002; Savard and Gauvin, 1994]; e (iv) métodos de região de confiança [Liu et al., 2013; Colson et al., 2005b]. No entanto, a maioria dessas abordagens torna-se obsoleta à medida que o problema em dois níveis torna-se mais complexo, sendo necessário recorrer a alguma técnica mais geral.

– **Metaheurísticas** –

No campo dos métodos evolucionistas, diversas metaheurísticas vêm sendo desenvolvidas para lidar com problemas de PDN. Estas técnicas podem ser classificadas de acordo com as seguintes estratégias [Talbi, 2013]:

- Abordagem sequencial aninhada: nesta classe o problema de nível inferior é resolvido para cada solução fixada no nível superior, onde as metaheurísticas envolvidas são executadas de forma aninhada. As metodologias desenvolvidas na presente tese se enquadram nesta categoria.

- Transformação em um único nível: nesta abordagem o problema é reduzido à um problema de um único nível, utilizando por exemplo a abordagem de transformação KKT. Assim, qualquer metaheurística poderia ser utilizada para resolver o problema com um único nível.

- Abordagem multiobjetivo: nesta classe o problema multinível é transformado em um problema de otimização multiobjetivo, permitindo

assim o uso de metaheurísticas multiobjetivo para tratar o problema.

- Abordagem co-evolucionista: esta é considerada a mais geral das metodologias. Este tipo de técnica utiliza várias metaheurísticas (geralmente uma para cada nível) que coevoluem de forma "paralela" trocando informações.

Os algoritmos desenvolvidos em [Angelo et al., 2014; Sinha et al., 2014; Angelo et al., 2013; Kuo and Huang, 2009; Yin, 2000; Mathieu et al., 1994] fazem uso da abordagem sequencial aninhada, onde nos quatro primeiros trabalhos utilizou-se algoritmos evolucionistas nos dois níveis do problema. Já em [Calvete et al., 2011; Baskan and Haldenbilen, 2011; Calvete et al., 2008; Yin, 2000; Mathieu et al., 1994] os autores utilizam um algoritmo evolucionista para a resolução do nível superior e um método exato para a resolução do nível inferior.

A transformação do problema em dois níveis para um único nível foi explorada por diversos autores no campo das metaheurísticas. Uma vez que o problema de PDN é transformado, qualquer metaheurística tradicional pode ser utilizada: algoritmo genético [Li and Wang, 2011; Wang et al., 2008; Li and Wang, 2007], evolução diferencial [Koh, 2011, 2007], e recozimento simulado [Sahin and Ciric, 1998].

A Abordagem multiobjetivo foi explorada em [Li et al., 2010; Fliege and Vicente, 2006], onde os autores propuseram metodologias que transformam o problema em dois níveis em um problema multiobjetivo equivalente.

Outros autores vêm se dedicando a área das metaheurísticas co-evolucionistas como em [Deb et al., 2013; Li and Fang, 2014; Legillon et al., 2012; Koh, 2009; Oduguwa and Roy, 2002], onde, geralmente, cada nível do problema é resolvido por uma metaheurística. Essas técnicas então coevoluem através de troca de informações entre as populações.

No próximo capítulo, o problema em dois níveis com um líder e múltiplos seguidores será apresentado.

Capítulo 3

Problema em Dois Níveis com Múltiplos Seguidores

Uma abordagem importante e cheia de desafios na otimização em dois níveis, também de interesse nesta tese, é a modelagem de aplicações que possuem múltiplos líderes e/ou múltiplos seguidores. A resolução deste tipo de problema não é uma tarefa simples pois modelar a interação de novos líderes e/ou seguidores no processo de otimização torna o problema ainda mais complexo.

Um exemplo deste tipo de aplicação ocorre na área de políticas governamentais para o planejamento do uso de terras [Bard, 1998]. Por exemplo, como líder, o objetivo do governo poderia ser maximizar o uso consciente das terras através da criação de políticas de desenvolvimento agrícola adequadas. Vários grupos envolvendo, por exemplo, agricultores, ambientalistas e grupos indígenas afetarão a decisão do governo na implantação de política para o uso de terras. Cada grupo, como um seguidor, tem seus próprios objetivos para otimizar. Estes seguidores podem compartilhar as mesmas variáveis de decisão, ou podem ter os mesmos objetivos ou restrições. Assim, a decisão do governo (líder) é parcialmente dependente das respostas apresentadas por cada grupo (os seguidores).

Outro exemplo prático, descrito em [Safari et al., 2014], envolve uma aplicação real de uma companhia de gerenciamento de recursos hídricos localizada no Irã. A modelagem proposta apresenta a companhia de distribuição de água como líder e três agentes como seguidores: o setor de agricultura, o setor doméstico

e o setor industrial. O objetivo do líder é maximizar seu lucro com a venda da água para os agentes no nível inferior, que por sua vez desejam maximizar objetivos próprios.

Neste capítulo, abordaremos o problema de programação em dois níveis com um líder e múltiplos seguidores. Neste caso, a decisão do líder é afetada pela reação de múltiplos seguidores, em mesma condição de igualdade, onde para cada possível decisão do líder, cada seguidor terá uma reação (resposta) diferente.

A relação entre múltiplos seguidores pode ser dada de diversas formas: eles podem ou não compartilhar suas variáveis de decisão, ou seja, pode ou não haver troca de informações entre eles; eles podem ter objetivos e restrições individuais, mas podem trabalhar de forma cooperativa; ou eles podem ter objetivos ou restrições em comum. Uma classificação das possíveis relações entre múltiplos seguidores pode ser dada conforme descrito na Tabela 3.1 [Lu et al., 2006].

Tabela 3.1: Classificação da relação entre os seguidores em problemas com múltiplos seguidores.

Tipo de relação	Fator de relação			Classificação
	Var. de decisão	Objetivos	Restrições	
Não-cooperativo	Individual	Individual	Individual	S_1
Cooperativo	Compartilhado	Compartilhado	Compartilhado	S_2
			Individual	S_3
		Individual	Compartilhado	S_4
			Individual	S_5
Parcialmente cooperativo	Parcialmente individual e parcialmente compartilhado	Compartilhado	Compartilhado	S_6
			Individual	S_7
		Individual	Compartilhado	S_8
			Individual	S_9

De acordo com esta classificação, abordaremos os problemas classificados pelos modelos S_1 e S_4 onde, no primeiro, não há cooperação entre os seguidores, assim não há compartilhamento das variáveis de decisão, objetivos e restrições, e no segundo, as variáveis de decisão e as restrições são compartilhadas e os objetivos são individuais.

3.1 Modelo Matemático

Cada situação apresentada na Tabela 3.1 requer a definição de um modelo específico para descrever o problema e uma determinada abordagem que descreva o conceito de solução dos seguidores e, consequentemente, da solução ótima do problema líder. Assim, os modelos S_1 e S_4 podem ser descritos conforme apresentados a seguir.

3.1.1 Seguidores independentes

O modelo S_1, também chamado de problema de *programação em dois níveis com múltiplos seguidores independentes* (PDNMSI) [Calvete and Galé, 2007; Arora and Narang, 2009] pode ser formulado como:

$$\begin{aligned} \min_{x \in X, y_i \in Y_i} \quad & F(x, y_1, y_2, ..., y_m) \\ \text{sujeito a} \quad & G(x, y_1, y_2, ..., y_m) \leq 0 \\ & y_i \in \arg\min_{y_i \in Y_i} f_i(x, y_i) \\ & \text{sujeito a} \quad g_i(x, y_i) \leq 0 \\ & \text{com} \quad i = 1, ..., m \end{aligned} \tag{3.1}$$

onde $x \in X \subset \mathbb{R}^n$ e $y_i \in Y_i \subset \mathbb{R}^{m_i}$, com $i = 1, ..., m$ indicando m diferentes seguidores. Esta formulação apresenta os seguidores de forma independente, ou seja, eles possuem objetivos e restrições independentes e não compartilham nenhum tipo de informação. Desta forma, a função objetivo f_i e o conjunto de restrições g_i do i-ésimo seguidor incluem apenas as variáveis do líder e as suas próprias variáveis de decisão. No caso do líder, tanto a função objetivo quando seu conjunto de restrições contêm suas próprias variáveis e as variáveis de decisão de todos os m seguidores.

Tomando como base as definições relativas a PDN apresentadas em [Bard, 1998] para o caso com um seguidor, as seguintes definições podem ser estabelecidas para o caso com múltiplos seguidores independentes [Calvete and Galé, 2007]:

- Conjunto das restrições do problema de PDNMSI:

  $\Omega := \{(x, y_1, y_2, ..., y_m) : x \in X, y_i \in Y_i, G(x, y_1, y_2, ..., y_m) \leq 0,$

  $g_i(x, y_i) \leq 0, i = 1, ..., m\}$

- Conjunto viável do i-ésimo seguidor, para cada $x \in X$:

  $\Omega_{y_i} := \{y_i \in Y_i : g_i(x, y_i) \leq 0, i = 1, ..., m\}$

- Projeção de Ω no espaço de decisão do líder:

  $P(X) := \{x \in X : \exists y_i \in Y_i, G(x, y_1, y_2, ..., y_m) \leq 0, g_i(x, y_i) \leq 0,$

  $i = 1, ..., m\}$

- Reação racional do i-ésimo seguidor para $x \in P(X)$:

  $R_i(x) := \{y_i \in Y_i : y_i(x) \in \arg\min\{f_i(x, \hat{y}_i) : \hat{y}_i \in \Omega_{y_i}\}, i = 1, ..., m\}$

- Região induzida (ou conjunto viável) da PDNMSI:

  $RI_{PDNMSI} := \{(x, y_1, y_2, ..., y_m) : (x, y_1, y_2, ..., y_m) \in \Omega, y_i \in R_i(x),$

  $i = 1, ..., m\}$

Neste modelo, para cada $x \in X$ fixado pelo líder, o i-ésimo seguidor responde de maneira ótima de acordo com sua função objetivo. Assim como no problema com um único seguidor, o problema de PDNMSI está bem posto se para cada $x \in X$ o i-ésimo seguidor responder com pelo menos uma resposta, e se $R_i(x)$ for um mapeamento do tipo ponto-ponto. Desta forma, a solução ótima do problema (3.1) satisfaz a seguinte relação

$$F(x^*, y_1^*, y_2^*, ..., y_m^*) \leq F(x, y_1, y_2, ..., y_m) \tag{3.2}$$

para qualquer $x, y_i \in \Omega$, com $i = 1, ..., m$, sendo y_i^* a solução ótima do i-ésimo seguidor em resposta a x^*.

3.1.2 Seguidores dependentes

O modelo S_4, chamado aqui de problema de *programação em dois níveis com múltiplos seguidores dependentes* (PDNMSD), pode ser formulado como [Lu et al.,

2006]:

$$\begin{aligned} \min_{x \in X, y_i \in Y_i} \quad & F(x, y_1, y_2, ..., y_m) \\ \text{sujeito a} \quad & G(x, y_1, y_2, ..., y_m) \leq 0 \\ & \min_{y_i \in Y_i} f_i(x, y_1, y_2, ..., y_m) \\ & \text{sujeito a} \quad g(x, y_1, y_2, ..., y_m) \leq 0 \\ & \text{com} \quad i = 1, ..., m \end{aligned} \tag{3.3}$$

onde $x \in X \subset \mathbb{R}^n$ e $y_i \in Y_i \subset \mathbb{R}^{m_i}$, com $i = 1, ..., m$ seguidores. Observe que nesta formulação os seguidores compartilham as variáveis de decisão e as restrições, mas possuem objetivos independentes. A função objetivo f_i do i-ésimo seguidor inclui as variáveis do líder, suas próprias variáveis e as variáveis dos demais seguidores. O conjunto de restrições g é compartilhado entre os seguidores, contendo as variáveis do líder e de todos os seguidores. No caso do líder, tanto a função objetivo quanto seu conjunto de restrições contêm suas próprias variáveis e as variáveis de decisão de todos os seguidores.

Assim, as seguintes definições podem ser descritas para o modelo S_4 [Lu et al., 2007a]:

- Conjunto das restrições do problema de PDNMSD:
  $\Omega := \{(x, y_1, y_2, ..., y_m) : x \in X, y_i \in Y_i, G(x, y_1, y_2, ..., y_m) \leq 0,$
  $g(x, y_1, y_2, ..., y_m) \leq 0, i = 1, ..., m\}$

- Conjunto viável do i-ésimo seguidor para cada $x \in X$:
  $\Omega_{y_i} := \{y_i \in Y_i : g(x, y_1, y_2, ..., y_m) \leq 0, i = 1, ..., m\}$

- Projeção de Ω no espaço de decisão do líder:
  $P(X) := \{x \in X : \exists y_i \in Y_i, G(x, y_1, y_2, ..., y_m) \leq 0,$
  $g(x, y_1, y_2, ..., y_m) \leq 0, i = 1, ..., m\}$

- Reação racional do i-ésimo seguidor para $x \in P(X)$:
  $R_i(x) := \{y_i \in Y_i : y_i(x) \in \arg\min\{f_i(x, \hat{y}_i, y_j, j = 1, ..., m, j \neq i) : \hat{y}_i \in$
  $\Omega_{y_i}\}, i = 1, ..., m\}$

- Região induzida (ou conjunto viável) da PDNMSD:

  $RI_{PDNMSD} := \{(x, y_1, y_2, ..., y_m) : (x, y_1, y_2, ..., y_m) \in \Omega, y_i \in R_i(x),$
  $i = 1, ..., m\}$

Como todos os seguidores no nível inferior compartilham informações, assume-se que eles definem sua estratégia "simultaneamente" para um dado $x \in X$. Assim, a solução ótima do nível inferior, i.e., a solução de equilíbrio de Nash para um dado um x, definida por $(x, y_1^*, y_2^*, ..., y_m^*)$, satisfaz

$$f_i(x, y_1^*, ..., y_{i-1}^*, y_i^*, y_{i+1}^*, ..., y_m^*) \leq f_i(x, y_1^*, ..., y_{i-1}^*, y_i, y_{i+1}^*, ..., y_m^*) \quad (3.4)$$

para qualquer $(y_1^*, ..., y_{i-1}^*, y_i, y_{i+1}^*, ..., y_m^*) \in Y_i$ e $i = 1, ..., m$. Aqui, assume-se que a solução de equilíbrio de Nash é única. Desta forma, conforme definição apresentada na seção 2.4.3, na solução de equilíbrio de Nash, nenhum seguidor pode melhorar seu ganho alterando de forma unilateral sua estratégia.

Finalmente, a solução ótima do problema (3.3) denominada *solução de Stackelberg-Nash* é definida por $(x^*, y_1^*, y_2^*, ..., y_m^*)$ onde $x^* \in \Omega$ e $(y_1^*, y_2^*, ..., y_m^*)$ é a solução de equilíbrio de Nash com relação à x^*. Assim, $(x^*, y_1^*, y_2^*, ..., y_m^*)$ é uma solução de Stackelberg-Nash, se e somente se,

$$F(x^*, y_1^*, y_2^*, ..., y_m^*) \leq F(\overline{x}, \overline{y}_1, \overline{y}_2, ..., \overline{y}_m) \quad (3.5)$$

para qualquer $(\overline{x}, \overline{y}_1, \overline{y}_2, ..., \overline{y}_m) \in \Omega$ onde $(\overline{y}_1, \overline{y}_2, ..., \overline{y}_m)$ é a solução de equilíbrio de Nash em resposta a $\overline{x}$.

3.2 Métodos de resolução

O problema com seguidores independentes pode ser resolvido seguindo a mesma linha de raciocínio da PDN onde, por exemplo, as condições KKT podem ser utilizadas para transformar os problemas do nível inferior em restrições, e em seguida adicioná-las ao problema líder, conforme feito nos trabalhos [Lu et al., 2007a, 2006]. Esta mesma abordagem também foi utilizada em [Wang et al., 2000],

porém considerando o cenário onde os seguidores são dependentes. Contudo é importante relembrar que também no caso com múltiplos seguidores a abordagem via KKT nem sempre pode ser utilizada e que o problema transformado pode não ser equivalente ao problema original.

Outro fato em comum é que a PDNMSI linear possui a mesma propriedade que a PDN linear, ou seja, a solução ótima global do problema ocorre em um vértice da RI_{PDNMSI}. Esta propriedade de fato ocorre pois demonstrou-se em [Calvete and Galé, 2007] que o problema com múltiplos seguidores independentes pode ser reformulado para um problema com um único seguidor, o que possibilita a aplicação de todos os resultados teóricos e algoritmos já conhecidos para o caso linear com um único seguidor. Como consequência, esta propriedade vem sendo explorada em boa parte dos trabalhos envolvendo múltiplos seguidores, uma vez que o caso linear ainda é um dos mais estudados nesta área. Como exemplo, podemos citar os trabalhos [Arora and Narang, 2009; Zhang et al., 2008; Shi et al., 2007] onde o algoritmo Késimo-Melhor foi utilizado, tomando como base esta propriedade.

Já em [Safari et al., 2014; Wang et al., 2000; Liu, 1998], o problema em dois níveis com múltiplos seguidores foi resolvido considerando o modelo de equilíbrio de Nash no nível inferior da hierarquia. Neste caso, para cada solução do líder, os seguidores respondem simultaneamente, seguindo a estratégia de equilíbrio de Nash.

Um ponto interessante a se destacar é que, assim como na PDN, os problemas de PDN com múltiplos seguidores (PDNMS) também são sensíveis quanto à localização das restrições. Alguns estudos foram desenvolvidos para a PDN no sentido de investigar as consequências de se transferir as restrições do nível superior, que envolvem as variáveis do nível inferior, para o problema de nível inferior. Mostrou-se em [Audet et al., 2006; Mersha and Dempe, 2006; Dempe, 2002], que nem sempre é possível utilizar essa abordagem, uma vez que esta mudança poderia ocasionar a transferência da solução ótima global para outra região ou poderia até mesmo tornar o problema sem solução. No entanto, essa abordagem

foi considerada em [Shi et al., 2007, 2005], para o caso com múltiplos seguidores independentes, sendo posteriormente questionada e discutida em [Calvete and Galé, 2007], conforme apresentado a seguir.

3.3 Considerações com relação a localização das restrições

No trabalho desenvolvido em [Shi et al., 2007, 2005], que trata o caso com seguidores independentes, tanto a região induzida do problema –$R_i(x)$– quanto o conjunto viável de cada seguidor –Ω_{y_i}– foram definidos de uma forma diferente da que foi apresentada anteriormente na seção 3.1.1. Os autores definem o conjunto viável do i-ésimo seguidor, para cada $x \in X$, como

$$\Omega_{y_i SLZ} := \{y_i \in Y_i : (x, y_i) \in \Omega, i = 1, ..., m\}$$

e consequentemente,

$$R_{iSLZ}(x) := \{y_i \in Y_i : y_i(x) \in \arg\min\{f_i(x, \hat{y}_i) : \hat{y}_i \in \Omega_{y_i SLZ}\}, i = 1, ..., m\},$$

ou seja, o conjunto viável dos seguidores é construído incluindo também as restrições do nível superior, sendo que estas afetam apenas o processo de decisão do líder, e não dos seguidores. A partir desta definição gerou-se uma contradição com relação ao fato dos seguidores terem sido considerados como independentes, pois as restrições do líder incluem também as restrições de todos os seguidores. Como consequência Shi et al. definem a região induzida do problema como

$$RI_{SLZ} := \{(x, y_i) : (x, y_i) \in \Omega, y_i \in R_{iSLZ}, i = 1, ..., m\}.$$

Tal fato foi investigado em [Calvete and Galé, 2007], onde mostrou-se que, ao se transferir as restrições do nível superior para o nível inferior, o problema transformado não era equivalente ao problema de PDNMSI originalmente formulado. Observemos como exemplo o problema a seguir ilustrado na Figura 3.1 [Calvete and Galé, 2007].

$$
\begin{array}{lll}
\min_{x,y,z} & F(x,y) = x & \\
\text{s.a.} & z \leq 1 & \\
& x \geq 0 & \\
& \min_y \ f_1(x,y) = -y & \min_z \ f_2(x,z) = -z \\
& \text{s.a. } x + y \leq 3 & \text{s.a. } 2x + z \leq 2 \\
& \quad y \geq 0 & \quad z \geq 0
\end{array}
\tag{3.6}
$$

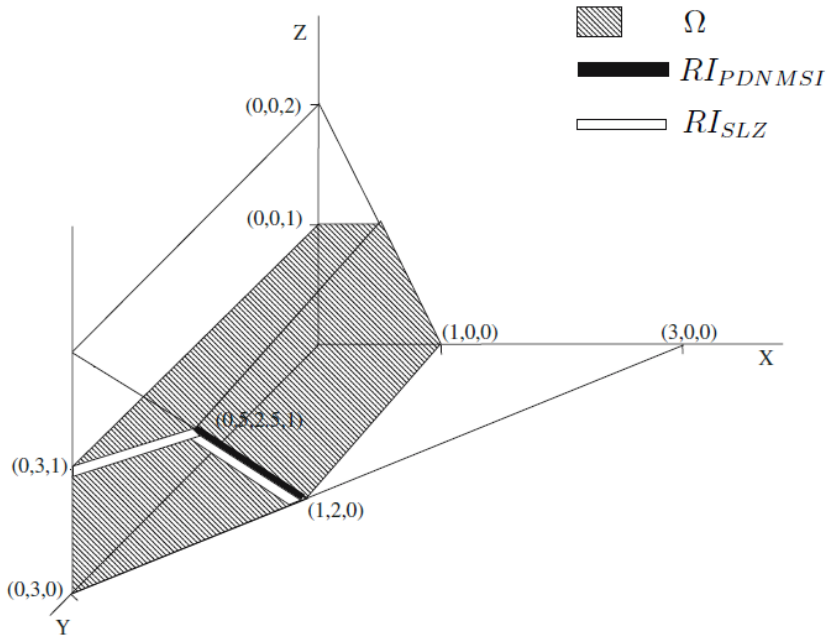


Figura 3.1: Conjunto de restrições Ω e região induzida do problema com múltiplos seguidores.

O conjunto viável Ω deste problema é

$$\Omega = \{(x,y,z) \in \mathcal{R}^3 : z \leq 1, x + y \leq 3, 2x + z \leq 2, x \geq 0, y \geq 0, z \geq 0\}$$

para cada $x \in X$, a reação racional dos seguidores é

$$R_1 = \{y \in \mathcal{R} : y \leq 3 - x, y \geq 0\}, \qquad R_2 = \{z \in \mathcal{R} : z \leq 2 - 2x, z \geq 0\}$$

e a região induzida do problema é

$$RI_{PDNMSI} = \{(x,y,z) \in \mathcal{R}^3 : x \in [0.5, 1], y = 3 - x, z = 2 - 2x\}.$$

A solução ótima deste problema ocorre no ponto $(0.5, 2.5, 1)$, sendo 0.5 o valor ótimo da função objetivo do líder. No entanto, seguindo a formulação apresentada por Shi et al., tem-se que

$$R_{1SLZ} = \{y \in \mathcal{R} : x \geq 0, y \geq 0, z \geq 0, y \leq 3 - x, z \leq 1, 2x + z \leq 2\};$$
$$R_{2SLZ} = \{z \in \mathcal{R} : x \geq 0, y \geq 0, z \geq 0, x + y \leq 3, z \leq 1, z \leq 2 - 2x\};$$
$$RI_{SLZ} = \begin{cases} (x, 3 - x, 1) & \text{se } x \in [0, 0.5], \\ (x, 3 - x, 2 - 2x) & \text{se } x \in [0.5, 1]. \end{cases}$$

Assim, a solução ótima seria $(0, 3, 1)$ e o valor ótimo da função objetivo do líder seria igual a 0. Entretanto, $(0, 3, 1)$ não é um ponto viável para o problema (3.6), uma vez que para $x = 0$, a solução ótima do segundo seguidor é $z = 2$, que está fora da região viável.

3.4 Exemplo de solução para a estratégia de equilíbrio de Nash

Como dito anteriormente, parte dos problemas com múltiplos seguidores que serão abordados neste trabalho caracteriza-se por possuir seguidores que compartilham informações/variáveis de decisão. Seguindo a estratégia de equilíbrio de Nash, cada seguidor busca otimizar sua função levando em conta também a decisão dos demais seguidores. Este ponto de equilíbrio pode ser obtido via um processo de otimização, ou via a descrição da forma analítica da solução.

A seguir, será apresentado um exemplo de um problema de PDNMSD onde a solução analítica do equilíbrio de Nash, para o problema de nível inferior, será numericamente determinada. Problemas deste tipo serão abordados na seção de experimentos 8.2.

3.4.1 Solução analítica

A obtenção da forma analítica da solução de equilíbrio de Nash para um problema sem restrições, inicia-se pela construção do conjunto de reação racional de cada seguidor/jogador, onde cada conjunto deve conter as melhores soluções que

um jogador pode alcançar para diferentes estratégias de seus oponentes. Assim, suponha dois jogadores A e B, sendo R_A e R_B o conjunto de reação racional do jogador A e B, respectivamente.

$$R_A = \{(x^*, y) \in X \times Y \text{ tal que } f_A(x^*, y) \leq f_A(x, y), \forall (x, y)\}$$
$$R_B = \{(x, y^*) \in X \times Y \text{ tal que } f_B(x, y^*) \leq f_B(x, y), \forall (x, y)\}$$

Desta forma, R_A e R_B podem ser construídos pela obtenção de x e y que satisfazem as seguintes equações

$$\begin{cases} R_A = \left\{x | \frac{\partial f_A(x,y)}{\partial x} = 0\right\} \\ R_B = \left\{y | \frac{\partial f_B(x,y)}{\partial y} = 0\right\} \end{cases}$$

O equilíbrio de Nash é então a interseção destes dois conjuntos [Sefrioui and Periaux, 2000]. Tomando como exemplo um caso particular sem restrições contendo dois seguidores que atendem ao equilíbrio de Nash, considere o seguinte problema:

$$\begin{aligned} \min_{x_1,x_2} \quad & F(x_1, x_2, y, z) = -(x_1 + x_2)^2 + \tfrac{1}{2}(y^2 + z^2) \\ \text{s.a.} \quad & \min_{y} f_1(x_1, x_2, y, z) = (y - (x_1 + 2))^2 + (z - (x_2 + 1))^2 \\ & \min_{z} f_2(x_1, x_2, y, z) = (y - (x_1 + 1))^2 + (z - (x_2 + 2))^2 \end{aligned} \tag{3.7}$$

A reação racional do primeiro e do segundo seguidor R_1 e R_2, respectivamente, é definida pela solução das seguintes equações

$$\begin{cases} R_1 = \left\{y | \frac{\partial f_1(x_1,x_2,y,z)}{\partial y} = 0\right\} \\ R_2 = \left\{z | \frac{\partial f_2(x_1,x_2,y,z)}{\partial z} = 0\right\} \end{cases}$$

resultando em

$$\frac{\partial f_1(x_1,x_2,y,z)}{\partial y} = 0 \Leftrightarrow 2(y - (x_1 + 2)) = 0 \Leftrightarrow 2y - 2(x_1 + 2) = 0 \Leftrightarrow y = x_1 + 2.$$
$$\frac{\partial f_2(x_1,x_2,y,z)}{\partial z} = 0 \Leftrightarrow 2(z - (x_2 + 2)) = 0 \Leftrightarrow 2z - 2(x_x + 2) = 0 \Leftrightarrow z = x_2 + 2.$$

Segue-se que a reação racional R_1 é a equação $y = x_1 + 2$ e a reação racional R_2 é a equação $z = x_2 + 2$. A solução de equilíbrio de Nash do nível inferior é portanto obtida por $y^N = x_1 + 2$ e $z^N = x_2 + 2$. Substituindo estes valores na função do líder, tem-se

$$\begin{aligned} F(x_1, x_2, y, z) &= -(x_1 + x_2)^2 + \tfrac{1}{2}(y^2 + z^2) \\ &= -x_1 - x_2 + \tfrac{1}{2}((x_1 + 2)^2 + (x_2 + 2)^2) \\ &= -x_1 - x_2 + \tfrac{1}{2}(x_1^2 + 4x_1 + 4) + \tfrac{1}{2}(x_2^2 + 4x_2 + 4) \end{aligned}$$

Resolvendo então as seguintes equações obtêm-se a solução ótima do problema (3.7):

$$\frac{\partial F(x_1,x_2,y,z)}{\partial x_1} = 0 \Leftrightarrow -1 + \tfrac{1}{2}(2x_1 + 4) = 0 \Leftrightarrow -1 + x_1 + 2 = 0 \Leftrightarrow x_1 = -1.$$
$$\frac{\partial F(x_1,x_2,y,z)}{\partial x_2} = 0 \Leftrightarrow -1 + \tfrac{1}{2}(2x_2 + 4) = 0 \Leftrightarrow -1 + x_2 + 2 = 0 \Leftrightarrow x_2 = -1.$$

Assim, a solução ótima do problema (3.7) ocorre no ponto $(x_1^*, x_2^*, y^N, z^N) = (-1, -1, 1, 1)$ que resulta em $F^* = 3$, $f_1^* = 1$ e $f_2^* = 1$.

O próximo capítulo é dedicado às metaheurísticas utilizadas para o desenho dos métodos de otimização propostos.

Capítulo 4

Metaheurísticas Utilizadas

O problema de otimização em dois níveis pode demandar elevado custo computacional, pois cada solução do nível superior está associada à resposta de um outro problema de otimização. Tendo em vista as diversas dificuldades que surgem em aplicações na otimização em dois níveis, tais como não-convexidade e não-diferenciabilidade, grande número de variáveis e/ou restrições, e não-unicidade de solução ótima no problema seguidor, as metaheurísticas surgem como ferramentas poderosas para viabilizar a resolução de tais problemas [Talbi, 2013].

Metaheurísticas são procedimentos utilizados para resolver problemas de otimização sem garantias de que a solução ótima exata será obtida, mas sim uma *boa solução*[1] para o problema. Estas técnicas são baseadas em operações suficientemente gerais que as permitem serem aplicáveis a uma variedade de problemas de otimização, sendo que algumas delas são dotadas de mecanismos específicos que podem ser mais adequados para determinado tipo de problema.

Os problemas de programação em dois níveis aqui abordados abrangem uma gama de situações que incluem problemas lineares, não-lineares, com ou sem restrições, onde o processo de otimização ocorre no espaço contínuo ou no discreto. Desta maneira, buscou-se por metaheurísticas eficientes que pudessem ser melhor aplicadas aos diferentes tipos de problemas a serem considerados.

Assim, para o tratamento das aplicações envolvendo otimização

[1] Entende-se por *boa solução* aquela que é melhor que a maioria das alternativas disponíveis [Gaspar-Cunha et al., 2013].

combinatória, foi utilizado o método de Otimização por Colônia de Formigas (ACO). Para o tratamento das aplicações envolvendo otimização com variáveis contínuas, utilizou-se o método de Evolução Diferencial (DE). Estas duas técnicas foram utilizadas de forma aninhada, onde cada uma ficou responsável por otimizar um nível do problema. Estas técnicas são conhecidas por serem robustas e eficazes no tratamento de diversos problemas de otimização, em especial os combinatoriais no caso da ACO [Ostfeld, 2011] e os de otimização global no caso da DE [Chakraborty, 2008].

A seguir, as metaheurísticas utilizadas serão descritas separadamente, onde cada método será apresentado e detalhado de forma independente. Os algoritmos em dois níveis desenvolvidos, que utilizam estas técnicas, e a análise dos experimentos computacionais realizados serão descritos posteriormente nos capítulos 6, 7 e 8.

4.1 Otimização por Colônia de Formigas (ACO)

Quando observamos uma única formiga pensamos em sua fragilidade física e em suas limitações de ações e tomadas de decisão. Entretanto, uma colônia de formigas é capaz de organizar-se de forma a realizar tarefas tão complexas que vão além da capacidade individual desses indivíduos. Como exemplo, as formigas tecelãs (*Oecophylla*) são capazes de formar incríveis pontes com seus próprios corpos para aproximar folhas mais distantes e uní-las com a seda produzida por suas larvas de modo a construir seus ninhos. Mais ainda, as formigas cortadeiras (*Atta*) fazem ninhos no solo, em câmaras subterrâneas ligadas por túneis que podem chegar a 8m de profundidade com uma área de 50m^2. Já as formigas-de-correição ou caçadoras (*Eciton*) em sua fase de migração, organizam verdadeiros exércitos envolvendo milhares de indivíduos a fim de coletar presas.

A grande questão envolvendo colônia de formigas é como elas se organizam se não há presença de um líder organizador das tarefas. Traduzindo livremente de [Gordon, 1999]:

> *Nenhum indivíduo tem conhecimento sobre o que deve ser feito para completar qualquer tarefa da colônia. Cada formiga "se vira" à sua maneira através do minúsculo mundo que a cerca. As formigas se juntam, se separaram e "vão a luta". De algum modo, esses pequenos eventos criam padrões que as guiam para o comportamento coordenado das colônias.*

Esta e muitas outras impressionantes capacidades ainda motivam pesquisadores a compreender melhor essa fascinante estrutura organizacional. Imagina-se ser esta estrutura social ou "inteligência coletiva" que as permita realizar tarefas complexas e sofisticadas. A ideia básica é de que um grupo de indivíduos pode fazer mais juntos do que eles poderiam fazer sozinhos. Este comportamento complexo e aparentemente inteligente emerge a partir da sinergia criada por simples interações entre os indivíduos que seguem regras simples.

Uma colônia de formigas é organizada em castas cujas funções variam de acordo com o tamanho do inseto (*e.g.* proteger a colônia, buscar por alimentos e transportar mantimentos). A interação entre os indivíduos ocorre através de um fenômeno denominado por estigmergia[2] , termo introduzido pelo zoólogo francês Pierre-Paul Grassé em 1959 [Grassé, 1959, 1984]. Estigmergia refere-se à noção de que a ação de um determinado agente deixa sinais no meio ambiente, e este sinal poderá ser percebido por outros agentes de forma a incitar ou determinar suas ações subsequentes. Em diversas espécies de formigas esta sinalização (ou comunicação) é feita através da deposição de feromônio[3] no meio ambiente, que é uma substância química de reconhecimento utilizada para sinalização de, por exemplo, alimento, perigo e maturação sexual.

A fim de estudar e observar o comportamento das formigas, Goss e colaboradores [Goss et al., 1989] desenvolveram uma experiência conectando o ninho de uma colônia de formigas a uma fonte de alimento através de uma ponte dupla. Foram realizados experimentos com dois módulos idênticos da ponte dupla variando apenas o comprimento dos braços, onde no primeiro experimento os

[2] Estigmergia: palavra proveniente do grego *stigma* (marca, sinal) e *érgon* (ação, trabalho).

[3] Feromônio: palavra proveniente do grego *phero* (transportar, transmitir) e *hormona*, do grego (excitar).

braços tinham o mesmo comprimento e no segundo os braços tinham comprimentos diferentes.

A Figura 4.1 apresenta um esquema adaptado destes experimentos [Golliat et al., 2013], onde no teste com os caminhos de comprimento diferente, no instante inicial da exploração a primeira formiga saía do ninho escolhendo aleatoriamente um dos braços da ponte e, ao encontrar o alimento, esta retornava ao ninho selecionando uma rota ao acaso, depositando feromônio ao longo do caminho (Figura 4.1a). Chegando ao ninho, as demais formigas eram recrutadas para a exploração de rotas até o alimento. No início do recrutamento as formigas escolhiam aleatoriamente um dos braços da ponte para seguir (Figura 4.1b). Após certo tempo, a maioria das formigas escolhia a menor rota e, eventualmente, algumas delas exploravam o trajeto mais longo (Figura 4.1c).

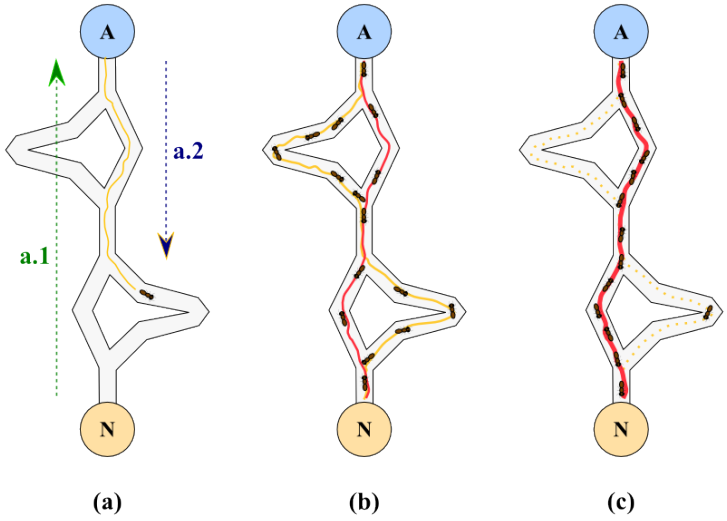


Figura 4.1: Experiência da ponte dupla com braços de tamanhos diferentes.

A explicação deste comportamento, discutida em [Dorigo and Stutzle, 2004], é que, no início do experimento não há nenhum feromônio depositado no meio, e assim, os dois braços da ponte possuem a mesma chance de serem escolhidos pelas formigas. Encontrando o alimento, a formiga retorna ao seu ninho depositando feromônio que, ao longo do tempo, sofre um processo de evaporação natural. Como um dos braços é menor que o outro, as formigas que escolheram o braço de menor comprimento são as primeiras a chegar no alimento e retornar ao ninho.

Neste ponto, quando elas precisam decidir entre o caminho mais curto e o mais longo, a maior concentração de feromônio no caminho mais curto irá influenciar sua decisão em favor deste caminho. Com isso, como no braço de menor comprimento a deposição de feromônio é rapidamente realimentada (*feedback* positivo) por um número cada vez maior de formigas, após algum tempo esta rota torna-se mais atrativa para as formigas subsequentes. O fato de algumas formigas eventualmente percorrerem o caminho mais longo pode ser interpretado como um tipo de "caminho exploratório".

A experiência das duas pontes mostrou claramente que as formigas possuem uma incrível capacidade de otimização: através do uso de regras probabilísticas baseadas em informações locais, elas são capazes de encontrar o menor caminho entre dois pontos em seu ambiente. Assim, baseado neste experimento, Dorigo e colaboradores [Dorigo, 1992; Dorigo and Gambardella, 1997b; Dorigo and Caro, 1999] desenvolveram os primeiros algoritmos de otimização baseados no comportamento das formigas.

4.1.1 Metaheurística ACO

Uma formiga artificial no método de otimização por colônia de formigas é um agente estocástico –formiga artificial– que constrói uma solução candidata através do incremento de componentes de solução à uma solução inicialmente vazia, até que esta seja completada. Assim, a metaheurística ACO pode ser aplicada a qualquer problema combinatório em que uma heurística construtiva possa ser definida.

A representação mais direta do espaço de busca para diversos problemas combinatórios, que podem ser resolvidos pelos algoritmos ACO, pode ser dada através de um grafo completamente conectado $G = (N, A)$ com N representando o conjunto de nós e A o conjunto de arestas conectando estes nós. Cada aresta $a_{ij} \in A$ possui um valor d_{ij} pré-determinado, representando o custo (distância, peso, tempo, etc) entre os nós i e j, com $i, j \in N$.

Nos algoritmos ACO, a construção da solução é feita de forma probabilística,

guiada pela informação heurística (η) que representa informações *a priori* associadas ao problema, e pela trilha artificial de feromônio (τ), que codifica uma memória relacionada ao processo de busca, que é continuamente atualizada pelas formigas artificiais. A informação heurística pode ser defina por $\eta_{ij} = 1/d_{ij}$, ou seja, a visibilidade do movimento da formiga artificial de seguir do nó i para o nó j é inversamente proporcional ao custo entre estes dois nós. Já a trilha de feromônio τ_{ij}, associada a aresta a_{ij}, corresponde à preferência da escolha do movimento para o nó j partindo do nó i. Estes valores são utilizados pela regra de decisão das formigas artificiais no momento de escolher (probabilisticamente) a componente de solução –nó do grafo– a ser incluída na solução em construção.

Depois de cada formiga artificial construir uma solução –caminho no grafo– a trilha de feromônio é atualizada de acordo com a qualidade das soluções candidatas obtidas: a componente de solução associada às boas soluções é reforçada (*feedback* positivo) enquanto que um certo nível de evaporação é aplicado em todas as arestas. Quanto mais formigas artificiais "atravessarem" uma certa aresta, maior será a concentração de feromônio nela.

Um dos algoritmos mais bem sucedidos da metaheurística ACO, e que foi utilizado nesta tese como base para os experimentos computacionais, é o Ant Colony System (ACS) [Dorigo and Gambardella, 1997a,b]. A principal diferença em relação ao seu antecessor, o Ant System, é que ele explora mais fortemente a experiência de busca acumulada pelas formigas artificiais, através de uma ação mais agressiva na escolha das regras, e aplica procedimentos de atualização local na trilha de ferômonio, de modo a aumentar a probabilidade de escolhas de caminhos alternativos para a exploração.

4.1.2 Algoritmo Ant Colony System (ACS)

O pseudo-código do algorimo ACS pode ser apresentado conforme o Algoritmo 1.

Algoritmo 1: Algoritmo ACS.

entrada: it_{max} (número de iterações), A_{max} (número de formigas), α (influência do feromônio), β (influência da informação heurística), γ (taxa de evaporação local) e ρ (taxa de evaporação global).

```
1  IniciarDados();
2  for i ← 1 to it_max do
3      for h ← 1 to A_max do
4          for j ← 1 to n do
5              RegraDeDecisão();        /* n é o número de variáveis */
6              AtualizarFeromônioLocal();
7          AvaliarSolução(a_h);
8          AplicarBuscaLocal(a_h);                          /* opcional */
9      a_best ← MelhorSolução(A_max);
10     AtualizarFeromônioGlobal(a_best);
```

saída : Melhor solução a_{best}

Na inicialização dos dados tem-se tipicamente a leitura da instância do problema e a inicialização das formigas artificiais e das matrizes de feromônio e de informação heurística.

A construção de uma solução no ACS se inicia quando uma formiga artificial h se move de um nó i para outro nó j. A escolha da próxima componente de solução é feita de forma determinística ou probabilística, dada por

$$j = \begin{cases} \arg\max\limits_{j \in \mathcal{N}_i^h} \{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta\}, & \text{se } q \leq q_0 \\ p_{ij}, & \text{caso contrário.} \end{cases} \tag{4.1}$$

com

$$p_{ij} = \begin{cases} \dfrac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^h} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, & \text{se } j \in \mathcal{N}_i^h \\ 0, & \text{caso contrário.} \end{cases} \tag{4.2}$$

onde α e β são os parâmetros que regulam a influência do feromônio τ_{ij} e da informação heurística η_{ij}, respectivamente, e $\mathcal{N}_i^h$ representa o conjunto viável da formiga artificial h.

Assim, a probabilidade de uma formiga artificial se mover de um nó i para outro nó j, depende da variável aleatória q uniformemente distribuída sobre o intervalo $[0, 1]$ e um parâmetro $0 \leq q_0 \leq 1$ definido pelo usuário. Com

probabilidade q_0 seleciona-se o melhor movimento possível, dado o conhecimento obtido refletido nas trilhas de feromônio, e na informação heurística. Neste caso, intensifica-se a exploração no espaço de busca. Enquanto que, com probabilidade $(1 - q_0)$ diversifica-se a exploração no espaço de busca.

A regra de atualização local da trilha de feromônio é aplicada por todas as formigas artificiais após ela "atravessar" uma aresta durante a construção da solução. Esta regra é dada por

$$\tau_{ij} \leftarrow (1 - \varphi)\tau_{ij} + \varphi\tau_0 \tag{4.3}$$

onde $\varphi \in (0, 1)$ é o coeficiente de decaimento de feromônio e τ_0 é o valor inicial da trilha artificial de feromônio. Diminuindo a concentração de feromônio nas arestas, a trilha torna-se menos atraente para outras formigas artificiais seguirem, possibilitando assim um aumento na exploração das arestas que ainda não foram atravessadas.

Após a construção da solução, um procedimento de busca local pode ser aplicado na tentativa de melhorar a solução previamente obtida.

Na atualização global, a evaporação e o depósito de feromônio são feitos somente nas arestas pertencentes à melhor solução encontrada até o momento (*best-so-far* - L_{bs}) ou pela melhor solução encontrada na atual iteração (*iteration-best* - L_{ib}). Assim, ao final de cada iteração, a atualização global do feromônio é realizada da seguinte forma

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij}^{best}, \quad \text{se } (i, j) \text{ pertencer à melhor solução} \tag{4.4}$$

onde $\rho \in (0, 1]$ é a taxa de evaporação do feromônio (coeficiente de redução) e $\Delta\tau_{ij}^{best}$ é a quantidade de feromônio depositada associada à qualidade da melhor solução obtida, que pode ser a L_{bs} ou L_{ib}. Por meio da equação (4.4), quanto melhor a qualidade da solução, maior será a quantidade de feromônio depositada nas arestas pertencentes à solução. Em geral, as arestas que forem mais vezes

utilizadas e que fazem parte de boas soluções, possuirão uma maior concentração de feromônio. Como consequência, estas arestas terão mais chances de serem percorridas em iterações subsequentes do algoritmo.

4.2 Evolução Diferencial (DE)

O método de Evolução Diferencial foi desenvolvido por Kenneth Price e Rainer Storn para ser um método de otimização confiável, versátil e de fácil utilização [Price et al., 2005]. A primeira publicação do DE surgiu em forma de um relatório técnico publicado em 1995 [Storn and Price, 1995]. Desde a sua criação, o DE vem chamando a atenção da comunidade científica em todo o mundo [Chakraborty, 2008]. Segundo Das e Suganthan [Das and Suganthan, 2011]:

> *O método de Evolução Diferencial (DE) é sem dúvida um dos mais poderosos algoritmos de otimização global estocástico da atualidade.*

Esta afirmação pode ser comprovada em decorrência do crescente número de variantes do algoritmo básico que apresentam um considerável aumento de desempenho e que vem ao longo dos anos se destacando em diversas competições na área da computação evolutiva [Das and Suganthan, 2011].

Assim como a maioria dos métodos evolucionistas, o DE é um método populacional que se inicia com um conjunto de soluções candidatas (representadas por vetores), aleatoriamente distribuídas pelo espaço de busca, onde a geração de novas soluções é dada pela perturbação de soluções já existentes.

O DE é considerado como um método simples e eficiente para otimização global. Sua simplicidade deve-se, principalmente, ao reduzido número de parâmetros requerido pelo método. O DE pode ser intuitivamente descrito como um método de manipulação de vetores que representam as soluções candidatas do problema, onde a heurística básica de evolução do método baseia-se nos movimentos gerados no espaço de busca pelo acréscimo de um determinado passo sobre um vetor base. Tal passo é definido pela ponderação da diferença entre outros

vetores pertencentes à população, sendo esta operação (mutação) definida por:

$$v_{novo} = v_{base} + F.(v_1 - v_2) \tag{4.5}$$

onde F é um parâmetro fornecido pelo usuário e que determina a ponderação da diferença, v_{novo} é o novo vetor gerado, aquele utilizado para participar do processo de recombinação, e v_{base}, v_1 e v_2 são vetores selecionados aleatoriamente na população. Uma ilustração do "movimento" gerado pelas soluções candidatas pode ser vista na Figura 4.2.

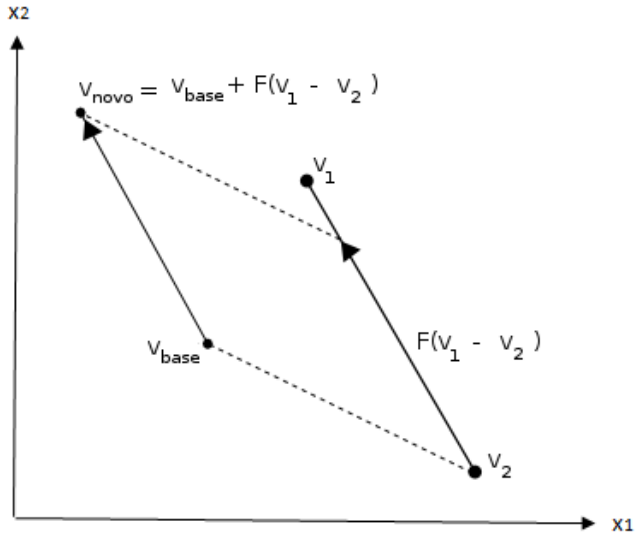


Figura 4.2: Exemplo da operação de mutação no método DE.

Com base neste operador, importantes observações podem ser destacadas [Krempser, 2014]: (i) a direção de busca do algoritmo é definida pela diferença entre os vetores v_1 e v_2, que portanto, guiam o algoritmo para as regiões a serem exploradas; (ii) com o parâmetro F, pode-se controlar a amplitude da busca na direção da diferença aplicada; e (iii) o ponto de partida da aplicação da diferença é definido pelo vetor v_{base}, que determina o ponto do espaço de busca onde uma perturbação será realizada. Assim, além da taxa de recombinação, o número de diferenças aplicadas, a amplitude da busca e a maneira como os indivíduos são selecionados determinam a intensificação ou a diversificação da exploração no espaço de busca.

4.2.1 Descrição do método

O algoritmo básico do método de evolução diferencial inicia-se pela geração aleatória de soluções candidatas ou indivíduos, representados por vetores, onde o conjunto desses indivíduos representa uma população. A evolução dos indivíduos é dada através de sucessivas gerações, onde o DE realiza mutações e recombinações na população a fim de gerar uma nova população.

Os parâmetros utilizados pelo DE são: (F) a amplitude ou ponderação da diferença empregada; (CR) a probabilidade de ocorrência de recombinação; e (pop) o número de indivíduos/vetores da população. Um pseudo-código do DE básico é apresentado no Algoritmo 2.

Algoritmo 2: Algoritmo DE/rand/1/bin

entrada: pop (tamanho da população), gen (núm. de gerações), F (fator de mutação), CR (taxa de recombinação)

```
1  G ← 0;
2  GerarPopulaçãoInicialAleatória(pop);
3  for i ← 1 to pop do
4      Avaliar f(x⃗_{i,G});          /* x⃗_{i,G} é um indivíduo da população */
5  for G ← 1 to gen do
6      for i ← 1 to pop do
7          SelecionarAleatoriamente(r1, r2, r3);      /* r1 ≠ r2 ≠ r3 ≠ i */
8          jRand ←AleatorioInt(1, n);   /* n (número de variáveis) */
9          for j ← 1 to n do
10             if Aleatorio(0,1) < CR or j = jRand then
11                 u_{i,j,G+1} = x_{r1,j,G} + F.(x_{r2,j,G} − x_{r3,j,G});   /* mutação */
12             else
13                 u_{i,j,G+1} = x_{i,j,G};
14         Avaliar f(u⃗_{i,G+1});
15         if f(u⃗_{i,G+1}) ≤ f(x⃗_{i,G}) then
16             x⃗_{i,G+1} = u⃗_{i,G+1};
17         else
18             x⃗_{i,G+1} = x⃗_{i,G};
```

Na geração de um novo indivíduo, o parâmetro CR determina a probabilidade de ocorrência da recombinação. Assim, a mutação é realizada com probabilidade CR, e com probabilidade $(1-CR)$ o valor presente no indivíduo alvo é copiado diretamente para o novo indivíduo. Para garantir a realização da mutação

em ao menos uma componente do novo indivíduo, seleciona-se previamente uma componente do mesmo ($jRand$), a qual sofrerá mutação, independentemente da probabilidade CR, evitando a simples cópia do indivíduo alvo.

Desta forma, a operação de mutação define o passo de busca realizado, e a operação de recombinação define a intercalação de valores gerados pela mutação e valores presentes em um indivíduo alvo. Estas etapas possuem diversas variações, as quais caracterizam as variantes do DE.

4.2.2 Variantes utilizadas

A variante do DE que descreve o Algoritmo 2 é denominada de DE/rand/1/bin, pois o vetor base é selecionado aleatoriamente (*rand*), apenas 1 vetor de diferença é adicionado a este, e o modelo de recombinação segue uma distribuição binomial (*bin*). Assim, as variantes do DE seguem o seguinte padrão [Krempser, 2009]:

DE/*mecanismo-de-seleção*/*número-de-diferenças*/*modelo-de-recombinação*.

O *mecanismo-de-seleção* descreve de que forma os vetores serão selecionados, o *número-de-diferenças* indica quantas diferenças entre indivíduos serão realizadas e o *modelo-de-recombinação* representa como o operador de recombinação será aplicado. A partir desta caracterização as variantes do DE que serão utilizadas neste trabalho, incluindo a variante DE/rand/1/bin, serão apresentadas a seguir.

– DE/best/1/bin –

Este modelo diferencia-se da variante DE/rand/1/bin apenas na maneira de seleção do indivíduo base. Nesta variante, o novo indivíduo a ser gerado utiliza o indivíduo de melhor aptidão (x_{best}) da população como vetor base, e os indivíduos x_{r_1} e x_{r_2} selecionados aleatoriamente, conforme

$$u_{i,G+1} = x_{best,G} + F.(x_{r_1,G} - x_{r_2,G}) \tag{4.6}$$

– **DE/target-to-best/1/bin** –

Esta variante utiliza o melhor indivíduo da população x_{best}, dois vetores distintos selecionados aleatoriamente x_{r_1} e x_{r_2}, e o indivíduo alvo x_i (aquele que será utilizado na comparação após a mutação, chamado vetor corrente, ou vetor alvo), para a geração de um novo indivíduo, conforme

$$u_{i,G+1} = x_{i,G} + F.(x_{best,G} - x_{i,G}) + F.(x_{r_1,G} - x_{r_2,G}) \tag{4.7}$$

– **DE/target-to-rand/1/bin** –

Esta variante utiliza três vetores distintos selecionados aleatoriamente $(x_{r_1}, x_{r_2}, x_{r_3})$ e o vetor alvo x_i, para a geração de um novo indivíduo, conforme

$$u_{i,G+1} = x_{i,G} + F.(x_{r_3,G} - x_{i,G}) + F.(x_{r_1,G} - x_{r_2,G}) \tag{4.8}$$

O capítulo seguinte apresenta todos os problemas-teste utilizados para analisar o desempenho e a aplicabilidade dos métodos de otimização desenvolvidos.

Capítulo 5

Problemas em Dois Níveis Abordados

Os problemas de otimização em dois níveis abordados nos experimentos computacionais abrangem diversos casos que incluem problemas lineares, não-lineares, com restrições e sem restrições, onde o processo de otimização ocorre no espaço contínuo ou no discreto, dependendo da formulação do problema. As aplicações abordadas foram divididas em três classes: Classe 1 - problemas de otimização com variáveis contínuas, lineares, não-lineares, com restrições e sem restrições; Classe 2 - problemas de planejamento de produção e distribuição; e Classe 3 - problemas em dois níveis com múltiplos seguidores. Estes problemas foram utilizados para validar as técnicas desenvolvidas no presente trabalho.

A primeira classe abrange diferentes problemas em dois níveis que foram coletados de diversas referências da literatura. A diversidade destes problemas possibilita validar o desempenho dos métodos propostos em sua robustez e eficácia, uma vez que permite explorar problemas que apresentam características distintas, como por exemplo: número de variáveis de decisão; diferentes tipos de funções (lineares e não-lineares); e problemas com a presença de restrições no nível superior e/ou inferior. Em especial, os problemas denominados SMD (descritos na seção 5.1.2), foram exclusivamente desenvolvidos para induzir dificuldades aos problemas de PDN. Estes podem ser ajustados de forma a escalar o problema e induzir dificuldades em ambos os níveis, de forma independente, ou através da interação entre os níveis.

Já a segunda classe, aborda um relevante problema presente na área de Pesquisa Operacional que é o problema hierárquico de planejamento de produção e distribuição. Neste problema, existem dois tomadores de decisão que controlam, respectivamente, o processo de produção e o de distribuição de produtos. Claramente, diferentes estratégias de otimização estão associadas aos diferentes decisores. A abordagem deste tipo de problema nos permite, além de avaliar o desempenho e a robustez dos métodos propostos, ampliar o leque de aplicações reais utilizando as metaheurísticas citadas.

A classe de problemas que aborda o caso com múltiplos seguidores permite avaliar a generalidade do método de otimização proposto, uma vez que poucas modificações foram necessárias à metodologia desenvolvida que trata dos problemas com um único seguidor. Além disso, em muitos problemas práticos, a presença de múltiplos seguidores é comumente considerada, tornando-se interessante a abordagem deste tipo de aplicação e consequentemente o desenvolvimento de métodos para resolvê-los.

5.1 Classe 1 - Problemas lineares, não-lineares, restritos e irrestritos

Nesta seção, 25 problemas de otimização em dois níveis serão descritos, divididos em duas categorias: 19 problemas teste (Pr.1-Pr.19) e 6 problemas denominados SMD (SMD1-SMD6).

5.1.1 Problemas teste

Os problemas descritos nesta seção foram coletados de diversas referências na literatura, onde o espaço de busca é contínuo, envolvendo 2, 4 ou 8 variáveis de decisão. A maioria destes problemas apresenta restrições no nível superior e/ou inferior, onde estão presentes funções lineares e não-lineares. Abaixo segue a descrição dos 19 problemas teste, indicando sua referência de origem e o melhor valor conhecido na literatura para as funções objetivo do líder (F) e do seguidor (f). O ponto de interrogação "?" indica que o valor não foi fornecido pela referência.

Tabela 5.1: Descrição dos 19 problemas teste.

Prob.	Descrição	Solução de Ref.
Pr.1	$\min_x F(x,y) = x^2 + (y-10)^2$ s.t. $-x + y \leq 0$ $x \in [0, 15]$ $\min_y f(x,y) = (x + 2y - 30)^2$ s.t. $x + y - 20 \leq 0$ $y \in [0, 20]$ [Shimizu and Aiyoshi, 1981]	$F^* = 100$ $f^* = 0$
Pr.2	$\min_x F(x,y) = (x_1 - 30)^2 + (x_2 - 20)^2 - 20y_1 + 20y_2$ s.t. $x_1 + 2x_2 \geq 30$ $x_1 + x_2 \leq 25$ $x_2 \leq 15$ $\min_y f(x,y) = (x_1 - y_1)^2 + (x_2 - y_2)^2$ s.t. $y_1, y_2 \in [0, 20]$ [Shimizu and Aiyoshi, 1981]	$F^* = 225$ $f^* = 100$
Pr.3	$\max_x F(x,y) = 8x_1 + 4x_2 - 4y_1 + 40y_2 + 4y_3$ s.t. $x \geq 0$ $\max_y f(x,y) = -x_1 - 2x_2 - y_1 - y_2 - 2y_3$ s.t $-y_1 + y_2 + y_3 \leq 1$ $2x_1 - y_1 + 2y_2 - 0.5y_3 \leq 1$ $2x_2 + 2y_1 - y_2 - 0.5y_3 \leq 1$ $y \geq 0$ [Candler and Townsley, 1982]	$F^* = 29.2$ $f^* = -3.2$
Pr.4	$\max_x F(x,y) = 2x_1 - x_2 - 0.5y_1$ s.t. $x \geq 0$ $\max_y f(x,y) = -x_1 - x_2 + 4y_1 - y_2$ s.t. $2x_1 - y_1 + y_2 \geq 2.5$ $-x_1 + 3x_2 - y_2 \geq -2$ $-x_1 - x_2 \geq -2$ $y \geq 0$ [Bard and Falk, 1982]	$F^* = 1.75$ $f^* = 0$
Pr.5	$\min_x F(x,y) = 2x_1 + 2x_2 - 3y_1 - 3y_2 - 60$ s.t. $x_1 + x_2 + y_1 - 2y_2 - 40 \leq 0$ $x \in [0, 50]$ $\min_y f(x,y) = (y_1 - x_1 + 20)^2 + (y_2 - x_2 + 20)^2$ s.t. $2y_1 - x_1 + 10 \leq 0$ $2y_2 - x_2 + 10 \leq 0$ $y \in [-10, 20]$ [Aiyoshi and Shimizu, 1984]	$F^* = 0$ $f^* = 200$
Pr.6	$\min_x F(x,y) = (x-5)^2 + (2y+1)^2$ s.t. $x \geq 0$ $\min_y f(x,y) = (y-1)^2 - 1.5xy$ s.t. $3x - y \geq 3$ $-x + 0.5y \geq -4$ $-x - y \geq -7$ $y \geq 0$ [Bard, 1988]	$F^* = 17$ $f^* = ?$
Pr.7	$\min_x F(x,y) = -x_1^2 - 3x_2 - 4y_1 + y_2^2$ s.t. $x \geq 0$ $x_1^2 + 2x_2 \leq 4$ $\min_y f(x,y) = 2x_1^2 + y_1^2 - 5y_2$ s.t. $x_1^2 - 2x_1 + x_2^2 - 2y_1 + y_2 \geq 3$ $x_2 + 3y_1 - 4y_2 \geq 4$ $y \geq 0$ [Bard, 1988]	$F^* = -12.6787$ $f^* = -1.0156$

Prob.	Descrição	Solução da Ref.
Pr.8	$\max_x F(x,y) = x + 3y$ s.t. $x \geq 0$ $\max_y f(x,y) = x - 3y$ s.t. $-x - 2y \leq -10$ $x - 2y \leq 6$ $2x - y \leq 21$ $x + 2y \leq 38$ $-x + 2y \leq 18$ $y \geq 0$ [Anandalingam and White, 1990]	$F^* = 49$ $f^* = -17$
Pr.9	$\min_x F(x,y) = (x-1)^2 + 2y_1^2 - 2x$ s.t. $x \geq 0$ $\min_y f(x,y) = (2y_1 - 4)^2 + (2y_2 - 1)^2 + xy_1$ s.t. $4x + 5y_1 + 4y_2 \leq 12$ $-4x - 5y_1 + 4y_2 \leq -4$ $4x - 4y_1 + 5y_2 \leq 4$ $-4x + 4y_1 + 5y_2 \leq 4$ $y \geq 0$ [Savard and Gauvin, 1994]	$F^* = -1.21$ $f^* = 7.61$
Pr.10	$\min_x F(x,y) = x_1^2 - 2x_1 + x_2^2 - 2x_2 + y_1^2 + y_2^2$ s.t. $x \geq 0$ $\min_y f(x,y) = (y_1 - x_1)^2 + (y_2 - x_2)^2$ s.t. $y \in [0.5, 1.5]$ [Falk and Liu, 1995]	$F^* = -1$ $f^* = 0$
Pr.11	$\min_x F(x,y) = 16x^2 + 9y^2$ s.t. $x \geq 0$ $-4x + y \leq 0$ $\min_y f(x,y) = (x + y - 10)^4$ s.t. $4x + y - 50 \leq 0$ $y \geq 0$ [Shimizu and Lu, 1995]	$F^* = 2250$ $f^* = 197.75$
Pr.12	$\min_x F(x,y) = x - 4y$ s.t. $x \geq 0$ $\min_y f(y) = y$ s.t. $-x - y \leq -3$ $-2x + y \leq 0$ $2x + y \leq 12$ $-3x + 2y \geq -4$ $x, y \geq 0$ [Bard, 1998]	$F^* = -12$ $f^* = 4$
Pr.13	$\min_x F(x,y) = x + y$ s.t. $x \geq 0$ $\min_y f(x,y) = -5x - y$ s.t. $-x - y/2 \leq -2$ $-x/4 + y \leq 2$ $x + y/2 \leq 8$ $x - 2y \leq 4$ $y \geq 0$ [Bard, 1998]	$F^* = 3.111$ $f^* = -6.667$
Pr.14	$\min_x F(x,y) = (x-1)^2 + (y-1)^2$ s.t. $x \geq 0$ $\min_y f(x,y) = 0.5y^2 + 500y - 50xy$ s.t. $y \geq 0$ [Bard, 1998]	$F^* = 1$ $f^* = 0$

Prob.	Descrição	Solução da Ref.
Pr.15	$\max_x F(x,y) = 100x + 1000y_1$ s.t. $x \in [0,1]$ $\max_y f(x,y) = y_1 + y_2$ s.t. $x + y_1 - y_2 \leq 1$ $y_1 + y_2 \leq 1$ $y \in [0,1]$ [Oduguwa and Roy, 2002]	$F^* = 1000$ $f^* = 1$
Pr.16	$\min_x F(x,y) = (x-3)^2 + (y-2)^2$ s.t. $x \in [0,8]$ $\min_y f(x,y) = (y-5)^2$ s.t. $2x - y \geq -1$ $-x + 2y \geq 2$ $-x - 2y \geq -14$ $y \geq 0$ [Rajesh et al., 2003]	$F^* = 5$ $f^* = 4$
Pr.17	$\min_x F(x,y) = (x-3)^2 + (y-2)^2$ s.t. $2x - y \geq -1$ $-x + 2y \geq 2$ $-x - 2y \geq -14$ $x \in [0,8]$ s.t. $\min_y f(x,y) = (y-5)^2$ $y \geq 0$ [Rajesh et al., 2003]	$F^* = 9$ $f^* = 0$
Pr.18	$\max_x F(x,y) = -2x + 11y$ s.t. $x \leq 0$ $\max_y f(x,y) = -x - 3y$ s.t. $x - 2y \leq 4$ $2x - y \leq 24$ $3x + 4y \leq 96$ $x + 4y \leq 126$ $-4x + 5y \leq 65$ $x + 4y \geq 8$ $y \leq 0$ [Rajesh et al., 2003]	$F^* = 85.0909$ $f^* =?$
Pr.19	$\min_x F(x,y) = \dfrac{1 + x_1 - x_2 + 2y_2}{8 - x_1 - 2y_1 + y_2 + 5y_3}$ s.t. $x_i \geq 0, i = 1, 2.$ $\min_y f(x,y) = \dfrac{1 + x_1 + x_2 + 2y_1 - y_2 + y_3}{6 + 2x_1 + y_1 + y_2 - 3y_3}$ s.t. $-y_1 + y_2 + y_3 + y_4 = 1$ $2x_1 - y_1 + 2y_2 - 0.5y_3 + y_5 = 1$ $2x_2 + 2y_1 - y_2 - 0.5y_3 + y_6 = 1$ $y_i \geq 0, i = 1, ..., 6.$ [Calvete and Galé, 2004]	$F^* = 0.081$ $f^* = 0.666$

É importante mencionar que estes problemas foram resolvidos por diversas outras técnicas que podem ser mais eficientes do que aquelas desenvolvidas neste trabalho. No entanto, algumas delas resolvem apenas um tipo de problema, o linear por exemplo, como é o caso encontrado nas referências [Candler and Townsley, 1982; Bard and Falk, 1982; Anandalingam and White, 1990]. Mais ainda, ao contrário do que está sendo proposto aqui, em grande parte dessas referências o

desempenho dos métodos é validado em um número reduzido de problemas testes, em alguns caso em apenas um ou dois problemas, como em [Shimizu and Aiyoshi, 1981; Candler and Townsley, 1982; Anandalingam and White, 1990; Savard and Gauvin, 1994; Falk and Liu, 1995; Shimizu and Lu, 1995; Calvete and Galé, 2004].

A Tabela 5.2 apresenta um resumo das metodologias utilizadas nas referências de origem para a resolução dos 19 problemas teste abordados. Como dito anteriormente, algumas delas possuem a propriedade de resolver apenas um tipo de problema, sendo esta informação também apresentada na tabela.

Tabela 5.2: Resumo das metodologias utilizadas nas referências de origem para a resolução dos 19 problemas teste.

Problema	Metodologia	Tipo de problema
Pr.1 e 2	Penalização (transformação do problema para um nível)	Não-linear
Pr.3	Enumeração de vértices	Linear
Pr.4	Branch-and-Bound (transformação do problema para um nível)	Linear
Pr.5	Penalização (transformação do problema para um nível)	Não-linear
Pr.6 e 7	Branch-and-Bound	Não-linear
Pr.8	Penalização	Linear
Pr.9	Método de descida (transformação do problema para um nível)	Não-linear
Pr.10	Método de descida	Não-linear
Pr.11	Penalização (transformação do problema para um nível)	Não-linear
Pr.12, 13 e 14	Soluções obtidas por inspeção na geometria do problema	-
Pr.15	Algoritmo Genético Coevolutivo	Não-linear
Pr.16,17 e 18	Busca Tabu	Não-linear
Pr.19	Enumeração de vértices (transformação do problema para um nível)	Linear

5.1.2 Problemas SMD

Os problemas SMD[1] (SMD1-SMD6) propostos em [Sinha et al., 2012] foram estrategicamente desenvolvidos para induzir dificuldades na resolução dos problemas de otimização em dois níveis. As funções objetivo são determinadas pela complexidade envolvendo os níveis superior e inferior de forma independente,

[1] A sigla SMD representa o sobrenome dos desenvolvedores dos problemas: Sinha, Malo e Deb.

e pela complexidade envolvendo a interação entre os dois níveis.

Nestes problemas, as funções do nível superior e inferior são dividas em três componentes, cada uma especializada por induzir algum tipo de dificuldade ao problema. Assim, as funções do líder e do seguidor podem ser descritas de forma genérica como:

$$\begin{aligned} \text{(Líder)} \quad & F(x,y) = F1(x_1) + F2(y_1) + F3(x_2,y_2) \\ \text{(Seguidor)} \quad & f(x,y) = f1(x_1,x_2) + f2(y_1) + f3(x_2,y_2) \\ \text{onde} \quad & x = (x_1,x_2) \text{ e } y = (y_1,y_2). \end{aligned} \tag{5.1}$$

Na equação acima, as varáveis de nível superior x e inferior y foram decompostas em dois vetores. Os vetores x_1 e y_1 são utilizados para influenciar a complexidade dos níveis superior e inferior de maneira independente. Já os vetores x_2 e y_2 são responsáveis por induzir dificuldades devido a interação entre os níveis. De forma similar, a decomposição da função do líder e do seguidor tem como finalidade especializar cada termo para um determinado propósito. A Tabela 5.3 sumariza o propósito de cada variável de decisão e as componentes das funções objetivo [Sinha et al., 2012].

Tabela 5.3: Papel das componentes nas funções SMD.

Painel A: Decomposição das variáveis de decisão

Variáveis do nível superior		Variáveis do nível inferior	
Vetor	Propósito	Vetor	Propósito
x_1	dificuldade no nível superior	y_1	dificuldade no nível inferior
x_2	interação com o nível inferior	y_2	interação com o nível superior

Painel B: Decomposição das funções objetivo

Função objetivo do nível superior		Função objetivo do nível inferior	
Componente	Propósito	Componente	Propósito
$F1(x_1)$	dificuldade na convergência	$f1(x_1,x_2)$	dependência funcional
$F2(y_1)$	conflito/cooperação	$f2(y_1)$	dificuldade na convergência
$F3(x_2,y_2)$	dificuldade na interação	$f3(x_2,y_2)$	dificuldade na interação

Os termos $F1(x_1)$ e $f2(y_1)$ são responsáveis por induzir apenas dificuldades na convergência dos níveis superior e inferior, respectivamente. Os termos $F3(x_2,y_2)$ e $f3(x_2,y_2)$ são termos de interação, que induzem dificuldades na

interação entre as variáveis do nível superior e inferior. Os termos $F2(y_1)$, $f2(y_1)$, $F3(x_2, y_2)$ e $f3(x_2, y_2)$ podem ser utilizados para caracterizar uma situação de conflito ou cooperação entre os níveis superior e inferior. A interação cooperativa, indica que a melhora do nível inferior faz com que ocorra também uma melhora no nível superior (e.g. $F2 = f2$). Em uma situação de conflito, a melhora do nível inferior provoca a degradação do nível superior (e.g. $F2 = -f2$). Já em uma interação "mista", ambas as situações de conflito e cooperação ocorrem (e.g. $F2 = f2$ e $F3 = \sum_i (x_2^i)^2 - f3$). Finalmente o termo $f1(x_1, x_2)$ é um termo fixo e não induz dificuldades no nível inferior, mas é utilizado juntamente com o termo $f3(x_2, y_2)$ para criar uma dependência funcional entre a melhor solução do nível inferior e as variáveis do nível superior.

Desta forma, as funções teste SMD1 a SMD6 são resumidamente caracterizadas da seguinte forma:

SMD1. Interação cooperativa; nível inferior convexo com relação as variáveis de nível inferior; e nível superior com região induzida convexa.

SMD2. Interação conflitante; nível inferior convexo com relação as variáveis de nível inferior; e nível superior com região induzida convexa.

SMD3. Interação cooperativa; nível inferior multimodal (função Rastrigin); e nível superior com região induzida convexa.

SMD4. Interação conflitante; nível inferior multimodal (função Rastrigin); e nível superior com região induzida convexa.

SMD5. Interação conflitante; nível inferior multimodal (função de Rosenbrock); e nível superior com região induzida convexa.

SMD6. Interação conflitante; nível inferior com infinitas soluções ótimas globais para cada x; nível superior com região induzida convexa.

Abaixo segue a descrição dos problemas SMD com suas componentes, os intervalos de cada variável de decisão e a descrição do ponto ótimo.

Tabela 5.4: Descrição dos problemas SMD.

Problema	Componentes das funções	Intervalo das variáveis
SMD1	$F1 = \sum_{i=1}^{p}(x_1^i)^2$	
	$F2 = \sum_{i=1}^{q}(y_1^i)^2$	$x_1^i \in [-5, 10], \forall i \in \{1, 2, ..., p\}$
	$F3 = \sum_{i=1}^{r}(x_2^i)^2 + \sum_{i=1}^{r}(x_2^i - \tan y_2^i)^2$	$x_2^i \in [-5, 10], \forall i \in \{1, 2, ..., r\}$
	$f1 = \sum_{i=1}^{p}(x_1^i)^2$	$y_1^i \in [-5, 10], \forall i \in \{1, 2, ..., q\}$
	$f2 = \sum_{i=1}^{q}(y_1^i)^2$	$y_2^i \in (\frac{-\pi}{2}, \frac{\pi}{2}), \forall i \in \{1, 2, ..., r\}$
	$f3 = \sum_{i=1}^{r}(x_2^i - \tan y_2^i)^2$	
Ponto de ótimo: $x = 0$, $y_1^i = 0, \forall i \in \{1, 2, ..., q\}$ e $y_2^i = \tan^{-1} x_2^i, \forall i \in \{1, 2, ..., r\}$.		
SMD2	$F1 = \sum_{i=1}^{p}(x_1^i)^2$	
	$F2 = -\sum_{i=1}^{q}(y_1^i)^2$	$x_1^i \in [-5, 10], \forall i \in \{1, 2, ..., p\}$
	$F3 = \sum_{i=1}^{r}(x_2^i)^2 - \sum_{i=1}^{r}(x_2^i - \log y_2^i)^2$	$x_2^i \in [-5, 1], \forall i \in \{1, 2, ..., r\}$
	$f1 = \sum_{i=1}^{p}(x_1^i)^2$	$y_1^i \in [-5, 10], \forall i \in \{1, 2, ..., q\}$
	$f2 = \sum_{i=1}^{q}(y_1^i)^2$	$y_2^i \in (0, e], \forall i \in \{1, 2, ..., r\}$
	$f3 = \sum_{i=1}^{r}(x_2^i - \log y_2^i)^2$	
Ponto de ótimo: $x = 0$, $y_1^i = 0, \forall i \in \{1, 2, ..., q\}$ e $y_2^i = \log^{-1} x_2^i, \forall i \in \{1, 2, ..., r\}$.		
SMD3	$F1 = \sum_{i=1}^{p}(x_1^i)^2$	
	$F2 = \sum_{i=1}^{q}(y_1^i)^2$	$x_1^i \in [-5, 10], \forall i \in \{1, 2, ..., p\}$
	$F3 = \sum_{i=1}^{r}(x_2^i)^2 + \sum_{i=1}^{r}((x_2^i)^2 - \tan y_2^i)^2$	$x_2^i \in [-5, 10], \forall i \in \{1, 2, ..., r\}$
	$f1 = \sum_{i=1}^{p}(x_1^i)^2$	$y_1^i \in [-5, 10], \forall i \in \{1, 2, ..., q\}$
	$f2 = q + \sum_{i=1}^{q}((y_1^i)^2 - \cos 2\pi y_1^i)$	$y_2^i \in (\frac{-\pi}{2}, \frac{\pi}{2}), \forall i \in \{1, 2, ..., r\}$
	$f3 = \sum_{i=1}^{r}((x_2^i)^2 - \tan y_2^i)^2$	
Ponto de ótimo: $x = 0$, $y_1^i = 0, \forall i \in \{1, 2, ..., q\}$ e $y_2^i = \tan^{-1} x_2^i, \forall i \in \{1, 2, ..., r\}$.		
SMD4	$F1 = \sum_{i=1}^{p}(x_1^i)^2$	
	$F2 = -\sum_{i=1}^{q}(y_1^i)^2$	$x_1^i \in [-5, 10], \forall i \in \{1, 2, ..., p\}$
	$F3 = \sum_{i=1}^{r}(x_2^i)^2 - \sum_{i=1}^{r}(\lvert x_2^i\rvert - \log(1 + y_2^i))^2$	$x_2^i \in [-1, 1], \forall i \in \{1, 2, ..., r\}$
	$f1 = \sum_{i=1}^{p}(x_1^i)^2$	$y_1^i \in [-5, 10], \forall i \in \{1, 2, ..., q\}$
	$f2 = q + \sum_{i=1}^{q}((y_1^i)^2 - \cos 2\pi y_1^i)$	$y_2^i \in [0, e], \forall i \in \{1, 2, ..., r\}$
	$f3 = \sum_{i=1}^{r}(\lvert x_2^i\rvert - \log(1 + y_2^i))^2$	
Ponto de ótimo: $x = 0$, $y_1^i = 0, \forall i \in \{1, 2, ..., q\}$ e $y_2^i = \log^{-1} \lvert x_2^i\rvert - 1, \forall i \in \{1, 2, ..., r\}$.		
SMD5	$F1 = \sum_{i=1}^{p}(x_1^i)^2$	
	$F2 = -\sum_{i=1}^{q-1}((y_1^{i+1} - (y_1^i)^2)^2 + (y_1^i - 1)^2)$	$x_1^i \in [-5, 10], \forall i \in \{1, 2, ..., p\}$
	$F3 = \sum_{i=1}^{r}(x_2^i)^2 - \sum_{i=1}^{r}(\lvert x_2^i\rvert - (y_2^i)^2)^2$	$x_2^i, y_2^i \in [-5, 10], \forall i \in \{1, 2, ..., r\}$
	$f1 = \sum_{i=1}^{p}(x_1^i)^2$	$y_1^i \in [-5, 10], \forall i \in \{1, 2, ..., q\}$
	$f2 = \sum_{i=1}^{q-1}((y_1^{i+1} - (y_1^i)^2)^2 + (y_1^i - 1)^2)$	
	$f3 = \sum_{i=1}^{r}(\lvert x_2^i\rvert - (y_2^i)^2)^2$	
Ponto de ótimo: $x = 0$, $y_1^i = 1, \forall i \in \{1, 2, ..., q\}$ e $y_2^i = \sqrt{\lvert x_2^i\rvert}, \forall i \in \{1, 2, ..., r\}$.		
SMD6	$F1 = \sum_{i=1}^{p}(x_1^i)^2$	
	$F2 = -\sum_{i=1}^{q}(y_1^i)^2 + \sum_{i=q+1}^{q+s}(y_1^i)^2$	$x_1^i \in [-5, 10], \forall i \in \{1, 2, ..., p\}$
	$F3 = \sum_{i=1}^{r}(x_2^i)^2 - \sum_{i=1}^{r}(x_2^i - y_2^i)^2$	$x_2^i, y_2^i \in [-5, 10], \forall i \in \{1, 2, ..., r\}$
	$f1 = \sum_{i=1}^{p}(x_1^i)^2$	$y_1^i \in [-5, 10], \forall i \in \{1, 2, ..., q + s\}$
	$f2 = \sum_{i=1}^{q}(y_1^i)^2 + \sum_{i=q+1}^{q+s-1}(y_1^{i+1} - y_1^i)^2$	
	$f3 = \sum_{i=1}^{r}(x_2^i - y_2^i)^2$	
Ponto de ótimo: $x = 0$, $y_1^i = 0, \forall i \in \{1, 2, ..., q\}$ e $y_2^i = x_2^i, \forall i \in \{1, 2, ..., r\}$.		

Em todos os estes problemas o valor da função objetivo do líder e do seguidor é igual a zero no ponto de ótimo. A dimensão de cada variável de decisão é definida como $dim(x_1) = p$, $dim(x_2) = r$, $dim(y_1) = q$ e $dim(y_2) = r$. Para o problema SMD6, $dim(y_1) = q + s$. Nos experimentos computacionais realizados, foram consideradas 10 variáveis de decisão em cada problema, definindo-se $p = 3$, $q = 2$

e $r = 2$ para os problemas SMD1 até SMD5, e $s = 2$ para o problema SMD6.

5.2 Classe 2 - Problema de planejamento de produção e distribuição

O modelo do problema de planejamento de produção e distribuição (PPD) aqui considerado foi inicialmente formulado em [Calvete et al., 2011]. A modelagem proposta descreve um problema em dois níveis envolvendo dois tomadores de decisão que representam diferentes empresas. A empresa de distribuição, que lidera o modelo hierárquico, busca minimizar os custos de distribuição e aquisição de produtos para atender a um conjunto de revendedores, sendo influenciada pelo comportamento do decisor no nível inferior. No nível inferior, a empresa produtora é responsável por controlar o processo de produção, onde ao receber do líder a demanda de produção, define quais de suas fábricas irão produzir os produtos, buscando minimizar seus custos operacionais. Um esquema com a representação do problema pode ser visto na Figura 5.1, onde tem-se 2 depósitos, 3 fábricas e 11 revendedores.

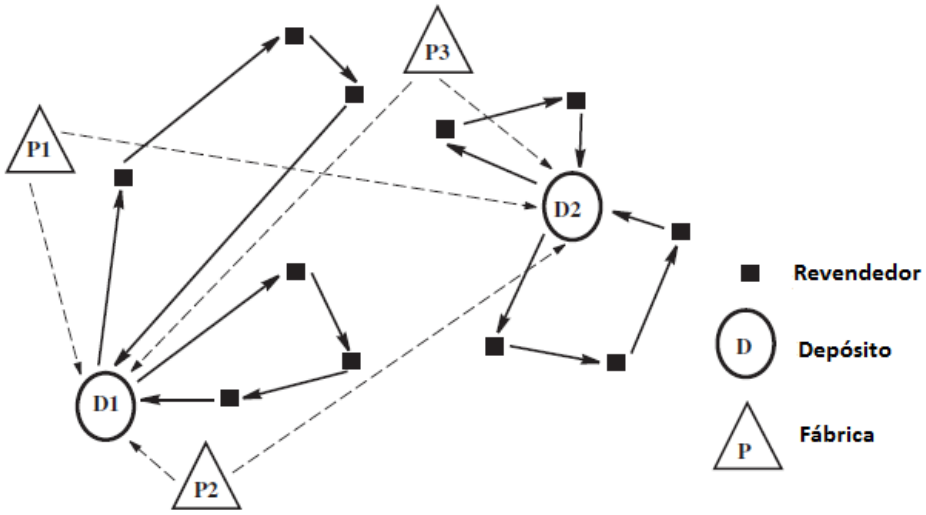


Figura 5.1: Esquema do problema de planejamento de produção e distribuição.

Este problema envolve dois problemas de otimização bem conhecidos: o Problema de Roteamento de Veículos com Múltiplos Depósitos (PRVMD) –no nível superior– e o Problema de Transporte (PT) –no nível inferior–, sendo estes

hierarquicamente relacionados.

Nas próximas seções estes dois problemas serão descritos separadamente a fim de facilitar o entendimento dos mesmos e em seguida a formulação e a interpretação do problema em dois níveis será apresentada.

5.2.1 Problema de roteamento de veículos com múltiplos depósitos

O problema de roteamento de veículos com múltiplos depósitos (PRVMD) é um problema de otimização combinatória onde o objetivo é atender um conjunto de clientes utilizando uma frota de veículos alocados em um ou mais depósitos. Uma frota de veículos homogêneos, alocada em cada depósito, precisa percorrer um conjunto de clientes de modo a atender suas demandas. Cada veículo inicia e termina sua rota no mesmo depósito, visitando cada cliente apenas uma vez. O objetivo do problema é minimizar o custo total da viagem (caminho percorrido por exemplo) atendendo a demanda de todos os clientes.

O PRVMD pode ser representado através de um grafo $G(V, E)$ onde $V = \{v_1, ..., v_n\} \bigcup V_0$ é o conjunto de nós do grafo, composto por clientes e depósitos, com $V_0 = \{v_{n+1}, ..., v_m\}$ sendo o conjunto de depósitos e $E = \{(i, j) : i, j \in V\}$ sendo o conjunto de arcos que conectam todos os nós do grafo, com exceção de conexões entre os depósitos. Para cada cliente i, com $i = 1, ..., n$, é associado uma demanda q_i e um tempo de duração de serviço t_i. Cada veículo s, com $s = 1, ..., S$, tem capacidade máxima de carga Q e tempo máximo de duração da rota T que precisam ser respeitados. Cada rota R_s associada a um veículo s é definida por $R_s = \{d, u_1, ..., u_w, d\}$, com $d \in V_0$ e $u_1, ..., u_w \in V/V_0$. O custo c_{ij}, associado a uma aresta (i, j), representa a distância entre os clientes i e j, ou a distância entre um cliente i e um depósito j (ou vice-versa). A função objetivo do problema é então minimizar a distância total percorrida pelos veículos.

Definindo as variáveis de decisão como

$$x_{ij}^s = \begin{cases} 1, & \text{se a rota do veículo } R_s \text{ usa a aresta } (i, j) \text{ e} \\ 0, & \text{caso contrário.} \end{cases} \tag{5.2}$$

o PRVMD pode ser escrito como

$$\min_{x} \quad \sum_{s=1}^{S}\sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}x_{ij}^{s} \tag{5.3}$$

$$\text{s.t.} \quad \sum_{j=1}^{m} x_{0j}^{s} + \sum_{j=1}^{m} x_{1j}^{s} = 1, \quad 1 \leq s \leq S \tag{5.4}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{m} x_{ij}^{s} t_i \leq T, \qquad 1 \leq s \leq S \tag{5.5}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{m} x_{ij}^{s} q_i \leq Q, \qquad 1 \leq s \leq S \tag{5.6}$$

$$\sum_{i=1}^{m} x_{il}^{s} - \sum_{j=1}^{m} x_{lj}^{s} = 0, \quad 1 \leq s \leq S, 1 \leq l \leq m \tag{5.7}$$

$$x_{ij}^{s} \in \{0,1\}, \qquad (i,j) \in E \tag{5.8}$$

Assim, uma solução viável para este problema é composta por um conjunto de rotas $R_1, \ldots, R_S$, que especifica a ordem em que os clientes serão visitados, com cada rota iniciando e terminando em um mesmo depósito, conforme restrição (5.4). Cada rota R_s é viável se o veículo não exceder o tempo máximo de duração da rota nem a capacidade total de carga, conforme apresentado pelas restrições (5.5) e (5.6), respectivamente, enquanto visita cada cliente apenas uma vez, como especificado pela restrição (5.7).

5.2.2 Problema de transporte

O problema de transporte (PT) consiste em determinar a forma mais econômica para enviar um bem/produto, disponível em quantidades limitadas em certos locais (origens) para outros locais onde é necessário (destinos). As origens representam as fábricas, que têm uma certa quantidade de produtos disponível para atender às demandas dos depósitos, que representam os destinos.

Os custos associados a este problema podem ser representados por uma matriz, chamada matriz de custos, tal que o elemento c_{ij} representa o custo de transportar uma unidade do produto da origem i para o destino j. Assume-se que os produtos são homogêneos, podem ser convenientemente enviados de

todas as origens para todos os destinos e que o custo de transporte é diretamente proporcional ao número de unidades do produto a ser transportado.

O PT pode ser escrito como

$$\min_{y} \quad \sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij}y_{ij} \tag{5.9}$$

$$\text{s.t.} \quad \sum_{j=1}^{M} y_{ij} = P_i \quad i = 1, ..., N \tag{5.10}$$

$$\sum_{i=1}^{N} y_{ij} = D_j \quad j = 1, ..., M \tag{5.11}$$

$$y_{ij} \in \mathbb{N} \tag{5.12}$$

onde N representa o número de fábricas e M o número de depósitos. A restrição (5.10) especifica que a soma dos produtos a serem enviados de uma fábrica é igual a sua disponibilidade total, a restrição (5.11) determina que a soma dos produtos a serem enviados aos destinos precisa ser igual à demanda total dos depósitos, e a restrição (5.12) indica que a variável de decisão y_{ij} assume apenas valores inteiros positivos. Aqui, assume-se que a disponibilidade total de produtos nas fábricas é igual à demanda total dos depósitos, ou seja

$$\sum_{i=1}^{N} P_i = \sum_{i=1}^{M} D_j. \tag{5.13}$$

Portanto, uma solução viável que satisfaça todas as restrições significa que o plano de transporte atende exatamente a todas as demandas de cada depósito e utiliza todos os recursos de cada fábrica.

Embora o PT considerado aqui seja linear e existam métodos exatos disponíveis para a resolução deste problema, o método de evolução diferencial que será utilizado também é capaz de resolver a versão não-linear do problema (alguns exemplos do PT não-linear podem ser encontrados em [Michalewicz, 1996]).

5.2.3 Formulação do problema em dois níveis

Considerando as definições e notações anteriores, a formulação em dois níveis envolvendo os problemas de transporte e roteamento é descrita da seguinte forma

$$
\begin{aligned}
\text{(L)} \quad \min_{x,y} F(x,y) = & \sum_{s\in S}\sum_{(i,j)\in E} c^1_{ij}x^s_{ij} + \sum_{n\in N}\sum_{l\in M} c^3_{nl}y_{nl} \\
\text{s.a} \quad & x^s_{ij} \in \{0,1\}, \quad (i,j)\in E \\
& x^s_{ij} \text{ satisfazer as restrições do PRVMD e} \\
& y_{nl} \text{ resolver (S)} \\
\text{(S)} \quad & \min_{y} f(y) = \sum_{n\in N}\sum_{l\in M} c^2_{nl}y_{nl} \\
& \sum_{l=1}^{M} y_{nl} = P_n, \quad n = 1,...,N \\
& \sum_{n=1}^{N} y_{nl} = D_l(x), \quad l = 1,...,M \\
& y_{nl} \in \mathbb{N}
\end{aligned}
\tag{5.14}
$$

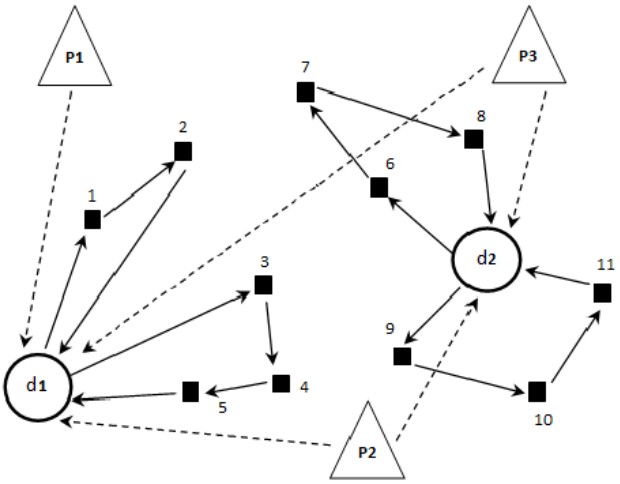
No modelo em dois níveis, os custos da empresa de distribuição, no nível superior, são de dois tipos: (i) o custo de transportar os itens passando pelos revendedores i e j utilizando o veículo s (dado por $c^1_{ij}x^s_{ij}$) e (ii) o custo de adquirir os produtos fabricados na fábrica n para o depósito l (dado por $c^3_{nl}y_{nl}$).

A companhia de produção, no nível inferior, recebe a ordem de serviço da companhia de distribuição informando a demanda necessária para abastecer os depósitos. Com esta informação a empresa produtora irá produzir os produtos em suas fábricas para atender a demanda dos depósitos, buscando minimizar seus custos de produção. O custo de produção dos produtos requeridos pelo depósito l depende da alocação de produção para uma determinada fábrica n (dado por $c^2_{nl}y_{nl}$).

Assim, as rotas associadas à x, que representa uma solução candidata do PRVMD, irá fornecer uma ordem de serviço contendo a demanda total dos depósitos $D(x) = \{D_1(x),...,D_M(x)\}$, onde $D_l(x)$ com $l = 1,...,M$, representa a demanda do depósito l. Essa ordem é passada para a empresa de produção, que

irá determinar o melhor planejamento possível da quantidade de produtos a serem fabricados nas fábricas para suprir as demandas dos depósitos.

Tomando como exemplo o esquema da Figura 5.1, apresenta-se na Figura 5.2 duas possíveis soluções candidatas para o problema, onde tem-se para cada depósito dois veículos alocados.

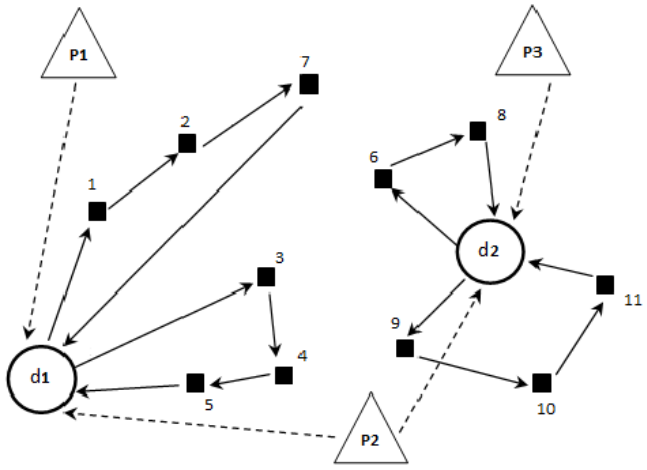


Nível superior

Depósito	Veículo	Rota	Demanda
d_1	s_1	$[d_1, 1, 2, d_1]$	100
d_1	s_2	$[d_1, 3, 4, 5, d_1]$	100
d_2	s_3	$[d_2, 6, 7, 8, d_2]$	200
d_2	s_4	$[d_2, 9, 10, 11, d_2]$	120

Nível inferior

	d_1	d_2	Oferta
P_1	100	50	150
P_2	50	190	240
P_3	50	80	130
Demanda	200	320	



Nível superior

Depósito	Veículo	Rota	Demanda
d_1	s_1	$[d_1, 1, 2, 7, d_1]$	150
d_1	s_2	$[d_1, 3, 4, 5, d_1]$	100
d_2	s_3	$[d_2, 6, 8, d_2]$	150
d_2	s_4	$[d_2, 9, 10, 11, d_2]$	120

Nível inferior

	d_1	d_2	Oferta
P_1	150	0	150
P_2	100	140	240
P_3	0	130	130
Demanda	250	270	

Figura 5.2: Duas soluções candidatas para o problema de planejamento de produção e distribuição.

Assim, cada veículo da companhia de distribuição constrói uma rota de entrega que passa por um conjunto de revendedores. No exemplo de cima da Figura 5.2, o veículo s_1 sai do depósito d_1 atendendo os clientes 1 e 2, em seguida retorna para o depósito d_1. Esta rota requer que o veículo atenda uma demanda de 100 unidades de produto. O veículo s_2 sai do mesmo depósito atendendo os clientes 3, 4 e 5, retornando para o depósito de onde ele partiu. Esta rota requer atender uma demanda de 100 unidades de produto. O mesmo raciocínio pode ser seguido para os veículos s_3 e s_4, com uma demanda de 200 e 120 por veículo, respectivamente.

Conhecendo a rota de cada veículo, é possível determinar a demanda total necessária para atender aquele conjunto de revendedores, e assim determinar a quantidade total de produtos necessários em cada depósito, onde $D = \{200, 320\}$ no exemplo de cima e $D = \{250, 270\}$ no exemplo de baixo. A informação passada para o nível inferior é exatamente este vetor de demandas D.

Estabelecida a demanda dos depósitos, cada fábrica irá produzir os produtos buscando minimizar seus custos operacionais. O vetor de disponibilidade/oferta P, que representa a disponibilidade de produtos em cada fábrica, é fixado em cada problema, e neste exemplo é dado por $P = \{150, 240, 130\}$. A empresa de produção então irá planejar quantos produtos cada uma de suas fábricas irá produzir para atender os depósitos. No exemplo da parte de cima o esquema de produção estabelece que a fábrica P_1 irá produzir 100 unidades de produto para o depósito d_1 e 50 unidades para o depósito d_2, a fábrica P_2 irá produzir 50 e 190 unidades para os depósitos d_1 e d_2, respectivamente e a fábrica P_3 irá produzir 50 unidades para o depósito d_1 e 80 unidades para o depósito d_2. Note que em ambos os exemplos o total de demandas é igual ao total de disponibilidades.

Resolvido o problema da companhia de produção, a empresa de distribuição saberá em quais fábricas adquirir os produtos, permitindo a esta completar o cálculo da sua função de custo.

5.3 Classe 3 - Problemas em dois níveis com múltiplos seguidores

Nesta seção, serão descritos os problemas teste de PDNMS, onde as variáveis de decisão do problema são dividas entre o líder e cada um dos seguidores. No primeiro conjunto abordaremos problemas testes com múltiplos seguidores independentes, e no segundo, problemas com múltiplos seguidores dependentes.

5.3.1 Problemas com seguidores independentes

Esta classe de problemas considera os seguidores como independentes, ou seja, considera-se que não há comunicação entre eles, e que portanto, eles não compartilham informações. A função objetivo e o conjunto de restrições de cada seguidor incluem apenas as variáveis do líder e suas próprias variáveis. Na Tabela 5.5 tem-se a descrição dos cinco problemas analisados com suas respectivas soluções ótimas.

Todos esses problemas foram resolvidos em suas referências de origem por métodos clássicos de otimização, onde os problemas 1, 3 e 4 foram transformados em um problema equivalente com um único nível e os problemas 2 e 5 foram transformados em problemas contendo apenas um seguidor. Ou seja, nestas referências nenhum deles foi resolvido em sua forma original.

5.3.2 Problemas com seguidores dependentes

Nesta classe os seguidores compartilham informações e agem de forma não-cooperativa atendendo ao equilíbrio de Nash. A função objetivo e o conjunto de restrições dos seguidores incluem as variáveis do líder, suas próprias variáveis e as variáveis do outro seguidor. A descrição dos três problemas abordados, com suas respectivas soluções ótimas, é apresentada na Tabela 5.6.

Em [Wang et al., 2000] os problemas 2 e 3 foram transformados em um problema com um único nível, através da abordagem KKT, e posteriormente resolvidos utilizando o método Branch and Bound.

Tabela 5.5: Descrição dos problemas com seguidores independentes.

Prob.	Descrição	Sol.Ref.
Pr.1	$\min_x \; F(x,y,z) = 3x_1 + 8x_2 + 7y + 11z$ s.a. $5x_1 + 2x_2 + y + 6z \leq 40$ $6x_1 - x_2 + 13y \leq 15$ $x_1 + x_2 - 7z \leq 10$ $7y + 4z \leq 20$ $x \geq 0$ $\min_y f_1(x,y) = 2x_1 + x_2 - y$ $\quad$ $\min_z f_2(x,z) = 15x_1 - x_2 + 80z$ s.a. $5x_1 + 7y \leq 15$ $\quad$ s.a. $40x_1 + z \leq 5$ $-4x_2 + 25y \leq 3$ $\quad$ $z \geq 0$ $y \geq 0$ [Lu et al., 2006]	$F^* = 111.69$ $f_1^* = 10.02$ $f_2^* = 11.61$
Pr.2	$\min_x \; F(x,y,z) = x$ s.a. $z \leq 1$ $x \geq 0.5$ $\min_y \; f_1(x,y) = -y$ $\quad$ $\min_z \; f_2(x,z) = -z$ s.a. $x + y \leq 3$ $\quad$ s.a. $2x + z \leq 2$ $y \geq 0$ $\quad$ $z \geq 0$ [Calvete and Galé, 2007]	$F^* = 0.5$ $f_1^* = -2.5$ $f_2^* = -1$
Pr.3	$\min_x \; F(x,y,z) = x - 2y - 4z$ s.a. $-x + 3y \leq 4$ $-x + z \leq 1$ $x \geq 0$ $\min_y f_1(x,y) = x + y$ $\quad$ $\min_z f_2(x,z) = x + z$ s.a. $x - y \leq 0$ $\quad$ s.a. $x + z \leq 4$ $-x - y \leq 0$ $\quad$ $2x - 5z \leq 1$ $y \geq 0$ $\quad$ $2x + z \geq 1$ $z \geq 0$ [Shi et al., 2005]	$F^* = -4.4$ $f_1^* = 4$ $f_2^* = 2.6$
Pr.4	$\min_x \; F(x,y,z) = x - 2y - 4z$ s.a. $x + y + z \leq 4$ $-x + 3y \leq 4$ $-x + z \leq 1$ $x \geq 0$ $\min_y f_1(x,y) = x + y$ $\quad$ $\min_z f_2(x,z) = -2x + z$ s.a. $x - y \leq 0$ $\quad$ s.a. $2x - 5z \leq 1$ $y \geq 0$ $\quad$ $2x + z \geq 1$ $z \geq 0$ [Lu et al., 2007b]	$F^* = -4$ $f_1^* = 0$ $f_2^* = 1$
Pr.5	$\max_x \; F(x,y,z) = \dfrac{3z+7}{x+y+5}$ s.a. $y + 4z \leq 9$ $x \in \{0,1\}$ $\min_y \; f_1(x,y) = \dfrac{3x+2y}{4x+2y+6}$ $\quad$ $\min_z \; f_2(x,z) = \dfrac{2z}{x+2z+7}$ s.a. $x + y \leq 4$ $\quad$ s.a. $x + 3z \leq 10$ $y \in \{0,1\}$ $\quad$ $z \in \{0,1\}$ [Arora and Narang, 2009]	$F^* = 1.666$ $f_1^* = 0.25$ $f_2^* = 0.222$

Tabela 5.6: Descrição dos problemas com seguidores dependentes.

Prob.	Descrição	Sol.Ref.
Pr.1	$\min_{x_1,x_2} \quad F(x_1,x_2,y,z) = -(x_1+x_2)^2 + \frac{1}{2}(y^2+z^2)$ s.a. $\quad \min_y f_1(x_1,x_2,y,z) = (y-(x_1+2))^2 + (z-(x_2+1))^2$ $\quad \min_z f_2(x_1,x_2,y,z) = (y-(x_1+1))^2 + (z-(x_2+2))^2$ *proposto no presente trabalho.	$F^* = 3$ $f_1^* = 1$ $f_2^* = 1$
Pr.2	$\max_x \quad F(x,y,z) = 7x+5y+8z$ s.a. $\quad \max_y f_1(x,y,z) = 3y+z \qquad \max_z f_2(x,y,z) = y-z$ $\quad x+y+z \leq 3, \qquad -x+y \leq 0, \qquad y+z \leq 2,$ $\quad x-y-z \leq 1, \qquad x,y,z \geq 0.$ [Wang et al., 2000]	$F^* = 22$ $f_1^* = 1$ $f_2^* = -1$
Pr.3	$\max_x \quad F(x,y,z) = 7x+5y+8z$ s.a. $\quad \max_y f_1(x,y,z) = -2y^2 - 2yz + 3y$ $\quad \max_z f_2(x,y,z) = -yz - z^2 + 6z$ $\quad x+y+z \leq 3, \qquad -x+y \leq 0, \qquad y+z \leq 2,$ $\quad x-y-z \leq 1, \qquad x,y,z \geq 0.$ [Wang et al., 2000]	$F^* = 22.5$ $f_1^* = 0$ $f_2^* = 6.75$

Os três capítulos seguintes apresentam as metodologias propostas e os experimentos computacionais para a resolução dos problemas que foram descritos ao longo deste capítulo.

Capítulo 6

Metodologias Propostas e Experimentos Computacionais para a Resolução dos Problemas da Classe 1

Esta seção apresenta a primeira etapa dos experimentos computacionais, onde propõe-se o desenvolvimento de dois métodos de otimização para a resolução dos problemas da Classe 1, que foram apresentados na seção 5.1.

A primeira proposta tem como objetivo resolver problemas de PDN utilizando o método de evolução diferencial em cada nível do problema. Assim, um trabalho foi desenvolvido nesta linha resultando na publicação [Angelo et al., 2013]. O algoritmo BLDE proposto e os experimentos computacionais realizados serão apresentados na seção 6.2.

A partir deste trabalho, surgiram novas linhas de investigação na tentativa de melhorar o desempenho do algoritmo BLDE. Assim, este método foi aprimorado e novos experimentos foram realizados, resultando na publicação [Angelo et al., 2014]. O algoritmo desenvolvido, BLDE$_M$, faz uso do metamodelo descrito na seção 6.3.1.2. A descrição deste algoritmo e os experimentos computacionais realizados serão apresentados na seção 6.3.

Ambas as propostas utilizaram o mesmo tratamento de restrições para a resolução dos problemas em dois níveis com restrições, sendo esta abordagem descrita a seguir.

6.1 Tratamento de restrições

Alguns dos problemas de otimização abordados possuem restrições que precisam ser satisfeitas, tanto no problema de nível superior quando no de nível inferior. Para estes problemas, utilizou-se o critério de seleção proposto por Deb em [Deb, 2000] para o tratamento das restrições.

O critério de seleção de Deb define a escolha de quais dos novos indivíduos gerados devem permanecer na população para a próxima geração. Neste critério, o indivíduo gerado é comparado com o indivíduo que o gerou e aquele que não violar nenhuma restrição terá preferência sobre aquele que violar alguma restrição. Assim, indivíduos factíveis/viáveis (que satisfazem as restrições) e infactíveis/inviáveis (que não satisfazem as restrições) poderão permanecer na população, porém os indivíduos factíveis sempre terão preferência quando comparados aos infactíveis. Deste modo, tem-se os seguintes critérios de seleção:

- qualquer solução viável é preferível a uma solução inviável;
- entre duas soluções viáveis, aquela que tiver o melhor valor na função objetivo é preferida.
- entre duas soluções inviáveis, aquela que tiver o menor valor de violação das restrições é preferida.

Estes critérios são considerados rigorosamente no nível superior e inferior para o tratamento das restrições do próprio nível. No entanto, quando há violação de alguma restrição no nível inferior, essa violação deverá impactar de alguma forma a qualidade da solução do nível superior. A maneira adotada para "propagar" tal violação foi realizada através da penalização da função objetivo do nível superior, sendo o valor da violação obtido conforme a formulação:

$$v_j(x,y) = \begin{cases} |g_j(x,y)|, & \text{para uma restrição de igualdade e} \\ \max\{0, g_j(x,y)\}, & \text{para uma restrição de desigualdade.} \end{cases} \tag{6.1}$$

Desta forma, quando houver violação de alguma restrição no problema do nível inferior, penaliza-se a função objetivo do nível superior da seguinte maneira:

$$F(x, y) = F(x, y) + \sum_{j=1}^{r_1} v_j(x, y) \tag{6.2}$$

onde r_1 denota o número de restrições do nível inferior.

6.2 Evolução Diferencial em Dois Níveis (BLDE)

No primeiro algoritmo proposto, BLDE [Angelo et al., 2013], o problema de nível inferior é resolvido para cada solução candidata do nível superior. O método desenvolvido é classificado como aninhado, onde um algoritmo DE, denominado DE-Líder, é utilizado para gerar e evoluir as variáveis do nível superior, e para cada uma dessas variáveis, um outro algoritmo DE, denominado DE-Seguidor, é executado para resolver o problema do nível inferior, através da geração e evolução das variáveis do nível inferior. O pseudo-código da metodologia proposta é apresentado pelos algoritmos 3 (DE-Líder) e 4 (DE-Seguidor).

Nos algoritmos 3 e 4 cada indivíduo i da população é representado por um vetor ($\overrightarrow{x}$) e cada componente j representa uma variável do problema. A descrição de cada etapa do método pode ser apresentada da seguinte forma:

Passo 0: Inicialização. O algoritmo inicia com uma população de indivíduos/vetores de tamanho np, onde cada indivíduo contêm os valores das variáveis x. As variáveis do líder (x) são iniciadas aleatoriamente e, para cada uma delas, as variáveis do seguidor são obtidas executando-se o procedimento de nível inferior (DE-Seguidor), responsável por obter as variáveis $y(x)$ associadas.

Passo 1: Evolução do nível superior. Seguindo a descrição básica do algoritmo DE-Líder, os indivíduos no nível superior são mutados e recombinados utilizando a variante DE/target-to-rand/1/bin (equação (4.8)).

Passo 2: Avaliação dos indivíduos do nível superior. Para avaliar os indivíduos do nível superior, onde a função de aptidão é associada à função de

nível superior juntamente com suas restrições, o procedimento de nível inferior é executado (DE-Seguidor) para cada indivíduo. A solução retornada, que é a melhor solução obtida no procedimento de nível inferior, é utilizada para avaliar o indivíduo do nível superior.

Passo 3: Evolução do nível inferior. Para cada variável x fixada no nível superior, o algoritmo DE-Seguidor é executado para obter as variáveis $y(x)$ associadas. As variáveis do seguidor são iniciadas de forma aleatória e os indivíduos são evoluídos utilizando a variante DE/target-to-rand/1/bin. Cada indivíduo é avaliado de acordo com a função objetivo do nível inferior e suas restrições. Após $genf$ gerações, o algoritmo retorna a melhor solução obtida para o problema de nível inferior.

Passo 4: Finalização. Após gen gerações o algoritmo retorna a melhor solução obtida para o problema de nível superior.

Algoritmo 3: Algoritmo DE-Líder.

Entrada : np (tamanho da população), gen (núm. de gerações), F (fator de mutação), CR (taxa de recombinação)

```
1  G ← 0;
2  CriarPopulacaoInicialAleatoria(np);
3  for i ← 1 to np do
4  |   y(x)_{i,G} = DE-Seguidor(npf, genf, F, CR, x_{i,G});
5  |   Avaliar F(x_{i,G}, y(x)_{i,G});          /* x_{i,G} é um indivíduo da população */
6  for G ← 1 to gen do
7  |   for i ← 1 to np do
8  |   |   SelecionarAleatoriamente(r1, r2, r3);                  /* r1 ≠ r2 ≠ r3 ≠ i */
9  |   |   jRand ←AleatorioInt(1, n);                  /* n variáveis do líder */
10 |   |   for j ← 1 to n do
11 |   |   |   if Aleatorio(0,1) < CR or j = jRand then
12 |   |   |   |   u_{i,j,G+1} = x_{i,j,G} + F.(x_{r3,j,G} − x_{i,j,G}) + F.(x_{r1,j,G} − x_{r2,j,G});
13 |   |   |   else
14 |   |   |   |   u_{i,j,G+1} = x_{i,j,G};
15 |   |   y(x)_{i,G+1} = DE-Seguidor(npf,genf,F,CR,u_{i,G+1});
16 |   |   AnalisarRestricoesLider(u_{i,G+1}, y(x)_{i,G+1});
17 |   |   if F(u_{i,G+1}, y(x)_{i,G+1}) ≤ F(x_{i,G}, y(x)_{i,G}) then
18 |   |   |   x_{i,G+1} = u_{i,G+1};
19 |   |   else
20 |   |   |   x_{i,G+1} = x_{i,G};
```

Retornar: MelhorIndividuoLider

Algoritmo 4: Algoritmo DE-Seguidor.

Entrada : npf (tamanho da população do seguidor), genf (núm. de gerações do DE seguidor), F (fator de mutação), CR (taxa de recombinação), $\overrightarrow{v}$ (variáveis do líder)

```
1  G ← 0;
2  CriarPopulacaoInicialAleatoria(npf);
3  for i ← 1 to npf do
4      Avaliar f(v, x_{i,G});                    /* x_{i,G} é um indivíduo da população */
5  for G ← 1 to genf do
6      for i ← 1 to npf do
7          SelecionarAleatoriamente(r1, r2, r3);                /* r1 ≠ r2 ≠ r3 ≠ i */
8          jRand ←AleatorioInt(1, nf);                /* nf variáveis do seguidor */
9          for j ← 1 to nf do
10             if Aleatorio(0,1) < CR or j = jRand then
11                 u_{i,j,G+1} = x_{i,j,G} + F.(x_{r3,j,G} − x_{i,j,G}) + F.(x_{r1,j,G} − x_{r2,j,G});
12             else
13                 u_{i,j,G+1} = x_{i,j,G};
14         AnalisarRestricoesSeguidor(v, u_{i,G+1});
15         if f(v, u_{i,G+1}) ≤ f(v, x_{i,G}) then
16             x_{i,G+1} = u_{i,G+1};
17         else
18             x_{i,G+1} = x_{i,G};
```

Retornar: MelhorIndividuoSeguidor

6.2.1 Experimentos computacionais

Nesta seção, os problemas teste Pr.1 a Pr.18 (descritos na seção 5.1.1), foram utilizados para avaliar o desempenho e a qualidade das soluções obtidas pelo método BLDE. O método proposto também foi testado no conjunto de problemas SMD1 a SMD6 (descritos na seção 5.1.2), a fim de melhor analisar seu desempenho, uma vez que estes problemas induzem dificuldades em ambos os níveis do problema.

Nestes experimentos realizou-se 30 execuções independentes para cada um dos problemas analisados. Os experimentos foram realizados no cluster do Laboratório Nacional de Computação Científica modelo Sun Blade x6250, com 2 Processadores Intel Xeon E5440 e 16 GB memória.

– Definição do valor dos parâmetros –

O valor dos parâmetros utilizados no algoritmo BLDE, em ambos os métodos/níveis, foram:

- F: a ponderação da diferença é de 0.7.

- CR: a taxa de recombinação é de 0.9.

- np e npf: a população do nível superior e inferior é de 30 indivíduos.

Para a resolução dos 18 problemas teste adotou-se 200 gerações no nível superior ($gen = 200$) e 100 gerações no nível inferior ($genf = 100$), totalizando 6000 e 18×10^6 avaliações de função no nível superior e inferior, respectivamente. Estes valores foram definidos de modo a permitir que o algoritmo encontre soluções de qualidade em todos os problemas analisados. Para os problemas SMD, o número de avaliações de função para os níveis superior e inferior foi de 2400 e 7.2×10^6, respectivamente, com $gen = 80$ e $genf = 100$. O número de avaliações de função do nível superior foi definido de forma a aproximar-se do valor adotado em [Sinha et al., 2012].

A partir de testes preliminares, verificou-se que a variante DE/target-to-rand/1/bin apresentou melhores resultados dentre as variantes descritas na seção 4.2.2. Por esta razão utilizou-se esta variante em ambos os níveis do problema para a realização destes experimentos.

– Resultados e discussões –

A Tabela 6.1 apresenta os resultados obtidos na resolução dos 18 problemas teste, indicando a melhor solução (x^*, y^*), e os correspondentes valores dos objetivos F^* e f^* obtidos no níveis superior e inferior, respectivamente. O ponto de interrogação "?" indica que os valores não foram fornecidos pela referência. Os resultados obtidos pelo BLDE foram comparados com valores apresentados por outros métodos de diferentes fontes da literatura.

Em todos os 18 problemas teste, o BLDE foi capaz de alcançar a solução ótima, ou muito próximo dela, e encontrou uma solução melhor para o problema 4. Neste problema, a solução de referência $(1, 0, 0.5, 1)$ indicada em [Bard and Falk, 1982], apresenta neste ponto duas restrições ativas. No entanto, o BLDE obteve a solução $(2, 0, 1.5, 0)$ que fornece todas as três restrições ativas neste ponto.

Tabela 6.1: Comparação das melhores soluções obtidas pelo BLDE e outros métodos para os problemas 1 a 18.

Prob.	Referência	(x^*, y^*)	F^*	f^*
1	Shimizu and Aiyoshi [1981]	(10, 10)	100	0
	Aiyoshi and Shimizu [1984]	(10, 10)	100	0
	Visweswaran et al. [1996]	(10, 10)	100	0
	Bard [1998]; Koh [2007]	(10, 10)	100	0
	Rajesh et al. [2003]	(10, 10)	100	0
	Koh [2007]	(10, 10)	100	0
	Oduguwa and Roy [2002]	(10.03, 9.969)	100.58	0.001
	Zhu et al. [2006]	(10.01, 9.93)	100.205	0.0169
	BLDE	(10, 10)	100	0
2	Shimizu and Aiyoshi [1981]	(20, 5, 10, 5)	225	100
	Visweswaran et al. [1996]	(20, 5, 10, 5)	225	100
	Zhu et al. [2006]	(20, 5, 10, 5)	225	100
	BLDE	(19.9999, 5.00004, 10, 4.99941)	224.988	99.9987
3**	Candler and Townsley [1982]	(0, 0.9, 0, 0.6, 0.4)	29.2	-3.2
	Bard and Falk [1982]	(0, 0.9, 0, 0.6, 0.4)	29.2	-3.2
	Bard [1998]	(0, 0.9, 0, 0.6, 0.4)	29.2	-3.2
	Guang-Min et al. [2003]	(0, 0.899, 0, 0.06, 0.394)	29.172	-3.186
	Guang-Min et al. [2005]	(0, 0.898, 0, 0.599, 0.399)	29.148	-3.193
	Zhu et al. [2006]	(0, 0.9, 0, 0.6, 0.3999)	29.198	-3.2
	Guang-Min et al. [2007]	(0, 0.8993, 0, 0.5995, 0.3981)	29.170	-3.1805
	BLDE	(0, 0.899998, 2.15e-6, 0.600001, 0.4000001)	29.2	-3,2
4**	Bard and Falk [1982]	(1, 0, 0.5, 1)	1.75	0
	Guang-Min et al. [2005]	(1, 0, 0.5, 1)	1.75	0
	Guang-Min et al. [2007]	(0.9993, 0, 0.4993, 1.0008)	1.748	-0.0028
	BLDE	(2, 0, 1.5, 0)	3.25	4
5	Aiyoshi and Shimizu [1984]	(25, 30, 5, 10)	5	0
	Bard [1998]; Li and Fang [2012]	(25, 30, 5, 10)	5	0
	Li and Fang [2012]	(25, 30, 5, 10)	5	0
	Ishizuka and Aiyoshi [1992]	(0, 0, -10, -10)	0	200
	Visweswaran et al. [1996]	(0, 0, -10, -10)	0	200
	Facchinei et al. [1999]	(25.00125, 30, ?, ?)	4.999	?
	Zhu et al. [2006]	(24.87, 29.98, 5.1, 10.31)	3.47	0.1618
	Koh [2007]	(0, 30, 10, -10)	0	100
	BLDE	(0, 0, -10, -10)	0	200
6	Bard [1988, 1998]	(1.778, 2.333)	42.5	-4.445
	Visweswaran et al. [1996]	(1, 0)	17	1
	Rajesh et al. [2003]	(1, 0)	17	1
	Koh [2007]	(1, 0)	17	1
	BLDE	(1, 0)	17	1
7	Facchinei et al. [1999]	(0, 2, ?, ?)	-12.678	?
	BLDE	(0, 1.99996, 1.87549, 0.906606)	-12.679	-1.0155

**Problema de maximização no nível superior e inferior.

Prob.	Referência	(x^*, y^*)	F^*	f^*
8**	Anandalingam and White [1990]	(16, 11)	49	-17
	Radjef and Anzi [2010]	(16, 11)	49	-17
	Guang-Min et al. [2003]	(15.959, 10.972)	48.875	-16.957
	Guang-Min et al. [2005]	(15.9972, 10.9945)	48.9806	-16.9863
	Guang-Min et al. [2007]	(15.9966, 10.9933)	48.9764	-16.9833
	BLDE	(16, 11)	49	-17
9	Savard and Gauvin [1994]	(1.889, 0.889, 0)	-1.21	7.61
	Bard [1998]	(1.889, 0.889, 0)	-1.21	7.61
	Oduguwa and Roy [2002]	(?, ?, ?)	3.57	2.4
	Zhu et al. [2006]	(1.87, 0.89, 0)	-1.2031	7.5927
	BLDE	(1.88889, 0.888889, 0)	-1.2098	7.61728
10	Falk and Liu [1995]; Koh [2007]	(0.5, 0.5, 0.5, 0.5)	-1	0
	Facchinei et al. [1999]	(0.5, 0.5, 0.5, 0.5)	-1	0
	BLDE	(0.5, 0.5, 0.5, 0.5)	-1	0
11	Shimizu and Lu [1995]	(11.25, 5)	2250	197.75
	Bard [1998]	(11.25, 5)	2250	197.75
	BLDE	(11.25, 5)	2250	197.754
12	Bard [1998]	(4, 4)	-12	4
	Radjef and Anzi [2010]	(4, 4)	-12	4
	BLDE	(4, 4)	-12	4
13	Bard [1998]	(0.889, 2.222)	3.111	-6.667
	Zhu et al. [2006]	(0.889, 2.222)	3.111	-6.667
	Rajesh et al. [2003]	(0.889, 2.222)	3.111	-6.667
	BLDE	(0.888889, 2.22222)	3.111	-6.6666
14	Bard [1998]	(1, 0)	1	0
	Oduguwa and Roy [2002]	(10.04, 0.1429)	82.44	-0.271
	Koh [2009]	(?, ?)	81.33	0.34
	BLDE	(1, 0)	1	0
15**	Oduguwa and Roy [2002]	(?, ?, ?)	1000	1
	Li and Fang [2012]	(1, 0, 1)	1000	1
	BLDE	(0, 0.999657, 0.000342909)	999.657	1
16	Rajesh et al. [2003]	(1, 3)	5	4
	Koh [2007]	(4, 3)	2	4
	BLDE	(1, 3)	5	4
17	Rajesh et al. [2003]	(3, 5)	9	0
	Koh [2007]	(3, 5)	9	0
	BLDE	(3, 5)	9	0
18**	Rajesh et al. [2003]	(17.4545, 10.90909)	85.0909	?
	BLDE	(17.4545, 10.9091)	85.0909	-50.1818

**Problema de maximização no nível superior e inferior.

Em seguida, a Tabela 6.2 sumariza os resultados do BLDE nas 30 execuções realizadas para cada um dos 18 problemas teste, onde NS e NI indicam os valores obtidos da função objetivo para o nível superior e inferior, respectivamente.

Conforme apresentado na Tabela 6.2, o algoritmo apresentou um comportamento estável uma vez que, para a maior parte dos problemas testados,

o método foi capaz de atingir a solução ótima em todas as 30 execuções, indicado pelo desvio padrão igual ou próximo de zero.

Tabela 6.2: Resumo dos resultados obtidos pelo BLDE em 30 execuções para os problemas 1 a 18.

Prob.	Nível	Mínimo	Mediana	Média	Máximo	Desvio Padrão
1	NS	100	100	100	100	0
	NI	0	0	0	0	0
2	NS	224.9880	224.9920	224.9919	224.9940	1.6e-03
	NI	99.9808	99.9976	99.9967	99.9994	3.4e-03
3**	NS	16	29.2	28.7600	29.2002	2.4
	NI	-6.5	-3.2	-3.3100	-3.1999	6.0e-01
4**	NS	3.25	3.6142	3.6247	3.9892	2.6e-01
	NI	-1.9141	0.9842	0.9863	4	2.1
5	NS	-7.38e-08	0	-2.64e-09	0	1.3e-08
	NI	100	200	193.3333	200	2.53e+01
6	NS	16.8612	17	16.995417	17.0023	2.5e-02
	NI	0.9939	1	0.9997993	1	1.0e-03
7	NS	-12.6905	-12.6795	-12.6800	-12.6792	20e-03
	NI	-1.0156	-1.01545	-1.0153	-1.0133	4.1e-04
8**	NS	49	49	49	49	0
	NI	-17	-17	-17	-17	0
9	NS	-1.4074	-1.4074	-1.4074	-1.4074	0
	NI	7.6172	7.6172	7.6172	7.6172	0
10	NS	-1	-1	-1	-1	1.8e-06
	NI	1.09e-12	4.00e-12	1.21e-11	1.44e-10	2.9e-11
11	NS	2250	2250	2250	2250	0
	NI	197.754	197.754	197.754	197.754	0
12	NS	-12	-12	-12	-12	0
	NI	4	4	4	4	0
13	NS	3.1111	3.1111	3.1111	3.1111	0
	NI	-6.6666	-6.6666	-6.6666	-6.6666	0
14	NS	1	1	1	1	0
	NI	0	0	0	0	0
15**	NS	980.1060	993.6590	993.2020	999.6570	4.6
	NI	1	1	1	1	0
16	NS	5	5	5	5	0
	NI	4	4	4	4	0
17	NS	9	9	9	9	0
	NI	0	0	0	0	0
18**	NS	79.8455	85.0909	84.4802	85.0909	1.5
	NI	-50.1818	-50.1818	-49.9680	-48.3459	5.2e-01

**Problema de maximização no nível superior e inferior.

Os experimentos realizados para os 18 problemas sugerem que o algoritmo é robusto, uma vez que foi capaz de resolver diferentes tipos de problemas. Sua robustez também pode ser demonstrada pela comparação de seus resultados com

os obtidos por outros métodos, sendo que alguns destes foram desenvolvidos para tratar apenas uma classe específica de problemas, como por exemplo os problemas lineares.

Para a resolução dos problemas SMD1 a SMD6, a Tabela 6.3 apresenta uma comparação dos resultados obtidos pelo BLDE e o Algoritmo Genético (AG) desenvolvido em [Sinha et al., 2012]. A tabela indica o número médio de avaliações de função (AF) do nível superior (NS) e inferior (NI) assim como a precisão (Prec.) alcançada pelos métodos com relação ao valor da função objetivo.

Tabela 6.3: Comparação do valor médio de avaliações de função e precisão alcançada pelo BLDE e um AG, para os níveis superior e inferior, na resolução dos problemas SMD1 a SMD6.

	AG [Sinha et al., 2012]				BLDE**	
Prob.	AF NS	AF NI	Prec. NS	Prec. NI	Prec. NS	Prec. NI
SMD1	2644	1724241	3.4e-05	1.6e-05	3.491e-06	1.935e-06
SMD2	2404	1568099	5.0e-06	5.0e-06	1.294e-06	6.514e-07
SMD3	2338	1483884	5.9e-05	2.6e-05	4.096e-06	2.917e-06
SMD4	1720	1187981	2.6e-05	2.7e-05	2.296e-05	5.140e-05
SMD5	3010	2093391	4.0e-06	3.0e-06	1.581e-06	1.379e-06
SMD6	3212	2429352	1.45e-04	7.1e-05	3.470e-06	2.067e-06

**Adotou-se 2400 (superior) e 7.2×10^6 (inferior) avaliações de função.

Os resultados descritos na Tabela 6.3 demonstram que o BLDE foi capaz de obter a solução ótima e atingir uma melhor precisão em ambos os níveis em todos os problemas SMD. Porém, o BLDE realizou um número maior de avaliações de função, principalmente no nível inferior, quando comparado com o AG. A estabilidade do BLDE também foi observada para esta classe de problemas, como mostra a Tabela 6.4, pelo valor do desvio padrão próximo de zero em todos os problemas testados.

Tabela 6.4: Resumo dos resultados obtidos pelo BLDE em 30 execuções para os problemas SMD1 a SMD6.

Prob.	Nível	Mínimo	Mediana	Média	Máximo	Desvio Padrão
SMD1	NS	3.73e-07	3.49e-06	3.83e-06	1.23e-05	3.05e-06
	NI	2.46e-07	1.93e-06	2.59e-06	8.00e-06	2.36e-06
SMD2	NS	2.32e-07	1.29e-06	1.68e-06	8.18e-06	1.50e-06
	NI	1.57e-07	6.51e-07	9.43e-07	2.74e-06	7.09e-07
SMD3	NS	1.06e-06	4.09e-06	4.78e-06	9.90e-06	2.56e-06
	NI	4.59e-07	2.91e-06	3.93e-06	1.62e-05	3.62e-06
SMD4	NS	-4.01e-04	-2.29e-05	-4.76e-05	1.24e-06	8.14e-05
	NI	2.4e-07	5.10e-05	1.0e-04	7.76e-04	1.60e-04
SMD5	NS	2.97e-07	1.58e-06	2.93e-06	1.10e-05	2.67e-06
	NI	1.61e-07	1.37e-06	2.06e-06	6.50e-06	1.93e-06
SMD6	NS	3.90e-07	3.47e-06	3.77e-06	1.24e-05	3.08e-06
	NI	2.81e-07	2.06e-06	2.62e-06	8.07e-06	2.39e-06

– Conclusões –

A partir destes experimentos foi possível levantar alguns pontos interessantes e questionamentos acerca da metodologia proposta. No primeiro deles, podemos destacar que diferentes variantes do DE podem ser utilizadas, assim como os valores dos parâmetros podem ser ajustados de diferentes formas para cada um dos níveis do problema. Como nenhum ajuste de parâmetros foi realizado nestes experimentos, possivelmente poderia haver um aumento no desempenho do método se esses valores fossem adaptados aos problemas testados. Este argumento também é válido para o número de avaliações de função realizadas, pois o algoritmo é capaz de obter boas soluções para alguns dos problemas analisados com um orçamento menor no número de avaliações de função. Uma outra maneira de aumentar o desempenho seria através da implementação de um critério de parada mais eficiente, que pudesse interromper o método sem que um número excessivo de avaliações de função fosse realizado.

Apesar do BLDE ter sido capaz de resolver todos os problemas analisados, observou-se um número expressivo de avaliações de função, principalmente no nível inferior, quando comparado com o algoritmo genético proposto em [Sinha et al., 2012], na resolução dos problemas SMD. Assim, na tentativa de reduzir o número

de avaliações em ambos os níveis e consequentemente melhorar o desempenho do BLDE, um novo método será proposto, o BLDE$_M$, com duas novas melhorias: (i) implementação de um critério de parada mais eficiente em ambos os níveis; e (ii) utilização de um modelo de substituição, ou metamodelo, para substituir o processo de otimização do nível inferior. Esta nova proposta é descrita a seguir.

6.3 DE em Dois Níveis Assistido por Metamodelo (BLDE$_M$)

Assim como o BLDE, o método BLDE$_M$, utiliza dois algoritmos de DE aninhados, cada um responsável por resolver um nível do problema. A diferença com relação ao seu antecessor é que o BLDE$_M$ (i) faz uso de um critério de parada mais eficiente e (ii) de um modelo de substituição que determina, de forma aproximada, as variáveis do nível inferior. Para os problemas com restrições, o mesmo tratamento de restrições foi utilizado, conforme descrito na seção 6.1.

Esta nova metodologia, que resultou na publicação [Angelo et al., 2014], tem como principal objetivo melhorar a eficiência do método BLDE, através da redução do número de avaliações de função, principalmente no nível inferior.

Assim como no método anterior, no método BLDE$_M$, cada indivíduo $\overrightarrow{x}_i$ da população é representado por um vetor e cada componente $\overrightarrow{x}_{i,j}$ representa uma variável do problema. O procedimento `SeguidorAproximado()` obtêm os valores de $y(x)$ através do metamodelo adotado, que será apresentado posteriormente na seção 6.3.1.2. Cada etapa do método pode ser descrita da seguinte forma:

Passo 0: Inicialização. O algoritmo inicia com uma população de indivíduos/vetores de tamanho np contendo os valores das variáveis x. Estas variáveis são iniciadas aleatoriamente e, para cada indivíduo do nível superior, as variáveis do nível inferior são obtidas executando-se o procedimento de nível inferior (DE$_M$-Seguidor), responsável por gerar as variáveis $y(x)$. Após a obtenção das variáveis do seguidor, os pares $(x, y(x))$ gerados são incluídos na base de dados $\mathcal{D}$ a ser usada no cálculo das aproximações do metamodelo.

Passo 1: Evolução do nível superior. Seguindo a descrição do DE$_M$-Líder,

os indivíduos do nível superior são mutados e recombinados utilizando uma das seguintes variante: DE/best/1/bin (equação (4.6)) ou DE/target-to-best/1/bin (equação (4.7)).

Passo 2: Avaliação dos indivíduos do nível superior. Para avaliar os indivíduos do nível superior, onde a aptidão é associada à função objetivo e restrições do nível superior, o procedimento do nível inferior precisa ser executado. A solução retornada é utilizada para avaliar o indivíduo do nível superior.

Passo 3: Procedimento do nível inferior. Para cada variável x do nível superior um dos seguintes procedimentos é adotado para a obtenção das variáveis do nível inferior:

Passo 3.1: Modelo evolutivo. O algoritmo DE$_M$-Seguidor é executado, onde os indivíduos do nível inferior são mutados e recombinados utilizando uma das seguintes variante: DE/best/1/bin ou DE/target-to-best/1/bin. Cada indivíduo é avaliado de acordo com sua função de aptidão e restrições. Ao final do procedimento, o algoritmo retorna a melhor solução $y(x)$. Após este processo o par $(x, y(x))$ é incluído na base de dados $\mathcal{D}$ do metamodelo.

Passo 3.2: Modelo de aproximação. As variáveis $y(x)$ são calculadas através da equação (6.3) utilizando os k vizinhos mais próximo de x pertencentes à base de dados $\mathcal{D}$. Em seguida, estas soluções são avaliadas de acordo com a função de aptidão do nível inferior juntamente com suas restrições.

Passo 4: Finalização. Quando o critério de parada do nível superior é satisfeito o algoritmo retorna a melhor solução obtida para o problema do nível superior.

No método BLDE$_M$, o parâmetro β descreve a probabilidade de uso do metamodelo e o parâmetro γ indica o número inicial de gerações em que o metamodelo não é executado, ou seja, o metamodelo só poderá ser utilizado após γ gerações do nível superior.

O pseudo-código da metodologia é apresentado nos algoritmos 5 (DE$_M$-Líder) e 6 (DE$_M$-Seguidor), que descrevem o nível superior e inferior do BLDE$_M$.

Algoritmo 5: Algoritmo DE_M-Líder.

Entrada : np (tamanho da população), F (fator de mutação), CR (taxa de recombinação)

```
1  G ← 0;
2  CriarPopulacaoInicialAleatoria(np);
3  for i ← 1 to np do
4      y(x_{i,G}) = DE_M-Seguidor(npf, F, CR, x_{i,G});
5      Avaliar F(x_{i,G}, y(x_{i,G}));
6      InserirBaseMetamodelo(x_{i,G}, y(x_{i,G}));
7  while (critério de parada não satisfeito) do
8      G ++;
9      for i ← 1 to np do
10         SelecionarAleatoriamente(r1, r2, r3);          /* r1 ≠ r2 ≠ r3 ≠ i */
11         jRand ←AleatorioInteiro(1, n);
12         for j ← 1 to n do
13             if Aleatorio(0, 1) < CR or j = jRand then
14                 u_{i,j,G+1} = Eq. (4.6) ou (4.7);
15             else
16                 u_{i,j,G+1} = x_{i,j,G};
17             if Aleatorio(0, 1) ≤ β and G ≥ γ then
18                 y(x_{i,G+1}) = SeguidorAproximado(npf, u_{i,G+1});
19             else
20                 y(x_{i,G+1}) = DE_M-Seguidor(npf, F, CR, u_{i,G+1});
21                 InserirBaseMetamodelo(u_{i,G+1}, y(x_{i,G+1}));
22             AnalisarRestricoesLider(u_{i,G+1}, y(x_{i,G}));
23             if F(u_{i,G+1}, y(x_{i,G+1})) ≤ F(x_{i,G}, y(x_{i,G})) then
24                 x_{i,G+1} = u_{i,G+1};
25             else
26                 x_{i,G+1} = x_{i,G};
```

Retornar: MelhorIndividuoLíder

Algoritmo 6: Algoritmo DE_M-Seguidor.

Entrada : npf (tamanho da população do seguidor), F (fator de mutação), CR (taxa de recombinação), $\overrightarrow{\text{v}}$ (variáveis do líder)

```
1  G ← 0;
2  CriarPopulacaoInicialAleatoria(npf);
3  for i ← 1 to npf do
4      Avaliar f(v, x_{i,G});          /* x_{i,G} é um indivíduo da população */
5  while (critério de parada não satisfeito) do
6      G ++;
7      for i ← 1 to npf do
8          SelecionarAleatoriamente(r1, r2, r3);          /* r1 ≠ r2 ≠ r3 ≠ i */
9          jRand ←AleatorioInteiro(1, nf);
10         for j ← 1 to nf do
11             if Aleatorio(0, 1) < CR or j = jRand then
12                 u_{i,j,G+1} = Eq. (4.6) ou (4.7);
13             else
14                 u_{i,j,G+1} = x_{i,j,G};
15         AnalisarRestricoesSeguidor (v, u_{i,j,G+1});
16         if f(v, u_{i,G+1}) ≤ f(v, x_{i,G}) then
17             x_{i,G+1} = u_{i,G+1};
18         else
19             x_{i,G+1} = x_{i,G};
```

Retornar: MelhorIndividuoSeguidor

6.3.1 Modelo de substituição (Metamodelo)

Um modelo é a representação de algum fenômeno que está sendo observado. Entende-se que modelos são representações simplificadas da realidade que preservam uma equivalência adequada, ou ainda, protótipos do que se pretende reproduzir ou imitar, e que devem possuir alguma utilidade. Segundo a bem conhecida frase de George E. P. Box:

"Essentially all models are wrong, but some are useful."

Um modelo de substituição, ou metamodelo, é utilizado para simular o comportamento de um modelo, tanto quanto for possível, sendo computacionalmente menos custoso de se avaliar. Assim, os modelos de substituição são comumente utilizados para substituir simulações que demandam alto custo computacional por um modelo de aproximação mais econômico, ou seja, por um modelo simplificado do modelo original de simulação. Desta forma, os metamodelos surgem como uma alternativa para reduzir o custo computacional e melhorar o desempenho dos métodos de otimização.

Conforme anteriormente mencionado no texto, a metodologia proposta implementa dois algoritmos evolucionistas aninhados. Assim, para cada solução x do nível superior, obtido por um algoritmo evolucionista, um outro algoritmo evolucionista será executado para a determinação da variável $y(x)$ correspondente. Como exemplo, considere a metodologia proposta com uma população com n_s indivíduos no nível superior e n_i indivíduos no nível inferior, e tomando m_s gerações no nível superior e m_i gerações no nível inferior, tem-se tipicamente um número total de avaliações de funções do nível inferior na ordem de $n_s \times m_s \times n_i \times m_i$.

Com isso, o estágio de maior demanda computacional é o processamento ótimo do nível inferior para cada solução fixada pelo nível superior. Nos casos em que a avaliação da função objetivo do nível inferior demandar alto custo computacional, a demanda por processamento também irá aumentar significativamente, podendo inviabilizar o uso do método.

Assim, uma das propostas deste trabalho foi utilizar um metamodelo em substituição ao modelo original de simulação do nível inferior[1] na tentativa de diminuir o número de avaliações de função neste nível. O modelo de aproximação foi utilizado para explicitar de forma adequada a relação entrada/saída do modelo original de simulação de modo a guiar a busca em direção a soluções promissoras. Para isso, foi necessário um banco de dados de "amostras" (gerado pelo modelo original de simulação) e uma medida de similaridade (para comparar as novas amostras com as amostras no banco de dados) para o cálculo das aproximações.

6.3.1.1 Método dos Vizinhos Mais Próximos

O modelo de aproximação proposto utiliza como base o método conhecido como $k-$NN (da sigla em inglês para *k-Nearest Neighbors*), onde k indica o número de vizinhos utilizados para a aproximação. Para o cálculo das aproximações no $k-$NN, um conjunto de amostras é requerido e as k amostras mais próximas ao ponto que deseja-se determinar devem ser encontradas. Neste caso, a aproximação consiste em calcular a média ponderada entre soluções previamente avaliadas, sendo esta ponderação dada por uma medida de similaridade. Essa medida de similaridade tem como propósito associar uma maior contribuição às amostras mais próximas.

Considere o problema de se encontrar uma função (aplicação) que atribua para um determinado $\overline{x}$ um valor $\widehat{f}(\overline{x})$. Seja $f(x)$ o valor da função objetivo, avaliado pelo modelo original de simulação, correspondente à solução x, e seja $\mathcal{D} = \{(x_h, f(x_h)); h = 1, \ldots, \mathcal{M}\}$ o conjunto de soluções avaliadas pelo modelo original de simulação, onde $\mathcal{M}$ indica o número de amostras do conjunto $\mathcal{D}$. Tem-se que a aproximação da função objetivo para uma nova solução candidata $\overline{x}$ é dada por

$$\widehat{f}(\overline{x}) = \frac{\sum_{j=1}^{|\mathcal{N}|} s(\overline{x}, x_j)^p f(x_j)}{\sum_{j=1}^{|\mathcal{N}|} s(\overline{x}, x_j)^p} \tag{6.3}$$

onde $|\mathcal{N}|$ denota a cardinalidade do conjunto $\mathcal{N}$ composto pelos k elementos do

[1] Aqui o modelo original de simulação é o método de evolução diferencial do nível inferior.

conjunto $\mathcal{D}$ mais similares (vizinhos mais próximos) à variável $\overline{x}$, com $x_j \in \mathcal{D}$. $s(\overline{x}, x_j)$ é a medida de similaridade entre $\overline{x}$ e x_j definida por

$$s(\overline{x}, x_j) = \frac{1}{d_E(\overline{x}, x_j)}. \tag{6.4}$$

sendo $d_E(\overline{x}, x_j)$ a distância Euclidiana entre $\overline{x}$ e x_j. Neste caso, se $\overline{x} = x_h$ para algum $x_h \in \mathcal{D}$ então $\widehat{f}(\overline{x}) = f(x_h)$.

Tomando como base o modelo de aproximação descrito pela equação (6.3), propõe-se aqui uma modificação no método k-NN, que consiste agora em aproximar todo o vetor/solução $y(x)$ do nível inferior, para cada vetor/solução x do nível superior. Esta aplicação tem como propósito substituir o procedimento computacionalmente intensivo do nível inferior por um modelo de aproximação, baseado no k-NN, para a determinação das variáveis $y(x)$. É importante ressaltar que aproximação não será feita para determinar a aptidão de um determinado indivíduo/vetor, mas sim para determinar o indivíduo resultante no nível inferior. A economia no número de avaliações de função então se dá pela substituição do processo de otimização do nível inferior pelo modelo de aproximação.

O metamodelo proposto, que é baseado no k-NN, é utilizado no algoritmo em dois níveis chamado BLDE$_M$, que será apresentado com detalhes na seção 6.3.

6.3.1.2 Proposta do metamodelo baseado no k-NN

O algoritmo BLDE$_M$ que faz uso do metamodelo utiliza dois algoritmos de DE aninhados, cada um responsável por otimizar um nível do problema. Neste processo, as variáveis x do nível superior, são determinadas por um algoritmo de DE e as variáveis $y(x)$ associadas do nível inferior, podem ser determinadas de duas maneiras: (i) através da execução de um outro algoritmo de DE para determinar os valores de $y(x)$; ou (ii) através da execução do metamodelo que irá aproximar os valores de $y(x)$.

A diferença entre estas duas formas é que, no primeiro caso, os valores de $y(x)$ são obtidos depois da execução total do algoritmo de DE para o nível

inferior, sendo este considerado aqui como o modelo original de simulação, como dito anteriormente. Já na outra forma, utilizando o metamodelo, os valores de $y(x)$ são calculados de maneira aproximada. Assim, como o modelo original de simulação não é executado, há uma economia no número de avaliações de função decorrente do processo de otimização do DE.

A determinação do vetor de resposta $y(\overline{x})$, dado um novo vetor do líder $\overline{x}$, é dada pela média ponderada pela similaridade entre soluções do líder previamente avaliadas. Define-se agora $\mathcal{D} = \{(x_h, y(x_h)); h = 1, \ldots, \mathcal{M}\}$ como o conjunto de amostras que serão utilizadas para o cálculo das aproximação do metamodelo, onde dado x_h do líder, $y(x_h)$ foi obtido via a execução do modelo original de simulação, sendo $\mathcal{M}$ o número de amostras (pares $(x, y(x))$ do conjunto $\mathcal{D}$. A seguinte aproximação para cada variável do vetor $\widehat{y}(\overline{x})$ do seguidor, para um novo vetor $\overline{x}$ do líder, é definida por

$$y_j(\overline{x}) \approx \widehat{y_j}(\overline{x}) = \frac{\sum_{p=1}^{|\mathcal{N}|} s(\overline{x}, x_p)^2 y_j(x_p)}{\sum_{p=1}^{|\mathcal{N}|} s(\overline{x}, x_p)^2} \tag{6.5}$$

onde $j = 1, \ldots, nf$ indica o índice da variável do nível inferior e $|\mathcal{N}|$ denota a cardinalidade do conjunto $\mathcal{N}$ composto pelos k elementos do conjunto $\mathcal{D}$ mais similares ao vetor $\overline{x}$. Os vetores $x_p \in \mathcal{N}$ são os vizinhos mais próximos de $\overline{x}$ e $s(\overline{x}, x_p)$ é a medida de similaridade entre $\overline{x}$ e x_p dada pela equação (6.4). Neste caso, se $\overline{x} = x_h$ para algum $x_h \in \mathcal{D}$ então $\widehat{y_j}(\overline{x}) = y_j(x_h)$.

Para ilustrar este modelo de aproximação, suponha que o conjunto $\mathcal{D}$ contenha $\mathcal{M} = 5$ elementos, com os valores de $(x, y(x))$, obtidos pelo modelo original de simulação, descritos na tabela `Conjunto` $\mathcal{D}$. Suponha agora que deseja-se obter o valor aproximado de $y(\overline{x})$ para $\overline{x} = (2.5, 2)$. Assumindo $k = 3$, tem-se primeiro que encontrar 3 elementos de $\mathcal{D}$ mais próximos de $\overline{x} = (2.5, 2)$. A partir da medida de similaridade $s(.,.)$ temos na tabela `Similaridades` os resultados para cada x_h em $\mathcal{D}$.

Quanto maior o valor de $s(\overline{x}, x_h)$, maior é a similaridade entre $\overline{x}$ e x_h. Assim, serão utilizados para o cálculo da aproximação de $y(\overline{x})$ os valores dos elementos B,

Conjunto $\mathcal{D}$

	x		$y(x)$	
	x_1	x_2	y_1	y_2
A	1	4	2	1
B	2	1	3	2
C	4	2	1	3
D	2	3	3	1
E	3	1	2	2

Similaridades

	$s(\overline{x}, x_h)$
A	0.4
B	**0.894**
C	0.667
D	**0.894**
E	**0.894**

D e E do conjunto $\mathcal{D}$, sendo então o conjunto $\mathcal{N} = \{B, D, E\}$. Pela equação (6.5) para $\overline{x} = (2.5, 2)$ tem-se como resultado da aproximação $\widehat{y}(\overline{x}) = (2.667, 1.66667)$. Abaixo, segue uma ilustração do espaço das variáveis x e $y(x)$, indicando os valores de $\overline{x}$ e $\widehat{y}(\overline{x})$.

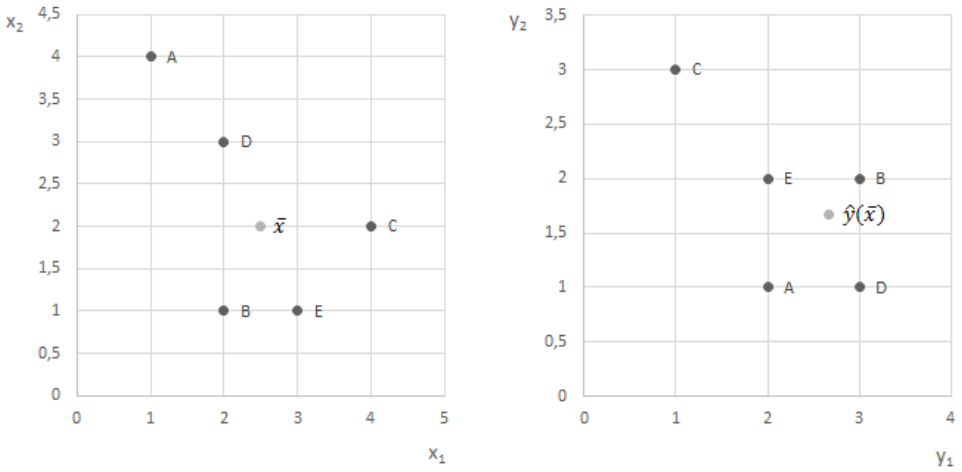


Figura 6.1: Espaço das variáveis do líder (lado esquerdo) e do seguidor (lado direito).

6.3.2 Critério de parada

O critério de parada adotado no BLDE$_M$, proposto em [Sinha et al., 2012], é baseado na variância da população corrente com relação à população inicial. Este critério foi utilizado em ambos os níveis do algoritmo, i.e., no BLDE$_M$-Líder e BLDE$_M$-Seguidor. Assim, para o método do nível superior, quando o valor α_S dado por

$$\alpha_S = \sum_{i=1}^{n} \frac{\sigma^2(x_i^t)}{\sigma^2(x_i^{initial})} \tag{6.6}$$

torna-se menor que α_S^{stop}, o algoritmo termina.

Em (6.6), $x_i^t : i = \{1, ..., n\}$ são as variáveis do nível superior na geração t e $x_i^{initial} : i = \{1, ..., n\}$ são as variáveis da população inicial do nível superior gerada aleatoriamente, onde n é o número de variáveis do nível superior.

Um critério similar foi adotado para o método do nível inferior, onde o algoritmo termina quando o valor α_I descrito a seguir é menor que α_I^{stop}.

$$\alpha_I = \sum_{i=1}^{nf} \frac{\sigma^2(y_i^t)}{\sigma^2(y_i^{initial})} \tag{6.7}$$

Em (6.7), $y_i^t : i = \{1, ..., nf\}$ são as variáveis do nível inferior na geração t e $y_i^{initial} : i = \{1, ..., nf\}$ são as variáveis da população inicial do nível inferior gerada aleatoriamente, onde nf é o número de variáveis do nível inferior.

Assim, os algoritmos finalizam a execução quando os critérios (6.6) e (6.7) são atingidos, para o nível superior e inferior, respectivamente, ou quando o número máximo de gerações é alcançado.

6.3.3 Experimentos computacionais

Para validar o algoritmo, 25 funções teste foram utilizadas: 19 funções lineares e não-lineares, com e sem restrições (descritas na seção 5.1.1); e 6 funções SMD, sem restrições (descritas na seção 5.1.2). O objetivo dos experimentos foi analisar o desempenho do método quando o metamodelo era utilizado em substituição ao processo de otimização do nível inferior. Assim, a metodologia proposta foi testada com diferentes probabilidades β de uso do metamodelo, onde o BLDE$_M$ foi analisado do ponto de vista da qualidade das soluções geradas e do número de avaliações de função economizadas no processo.

Para estes experimentos 30 execuções independentes foram realizadas para cada problema, onde duas variantes do DE foram analisadas: DE/best/1/bin e DE/target-to-best/1/bin. Os experimentos foram realizados no cluster do Laboratório Nacional de Computação Científica modelo Sun Blade x6250, com 2 Processadores Intel Xeon E5440 e 16 GB memória.

– **Definição do valor dos parâmetros** –

Experimentos preliminares foram realizados para encontrar valores razoáveis para os parâmetros do BLDE$_M$. Segue abaixo o valor de cada parâmetro utilizado:

- F: a ponderação da diferença é de 0.8.
- CR: a taxa de recombinação é de 0.9.
- np e npf: a população do nível superior e inferior é de 30 indivíduos.
- α_S^{stop} e α_I^{stop}: o nível de precisão adotado no critério de parada em ambos os níveis é de 10^{-5}.
- β: a probabilidade de uso do metamodelo assume os valores 0 (sem utilização), 0.3, 0.5 e 0.8.
- k: número de vizinhos utilizados para calcular as variáveis do nível inferior via o metamodelo é de 2.
- γ: o número de gerações iniciais em que o metamodelo não é utilizado é igual a 1.

O número máximo de gerações adotado foi igual ao utilizado nos experimentos realizados na seção anterior, ou seja, nos 19 problemas teste, $gen = 200$ no algoritmo do líder e $genf = 100$ no algoritmo do seguidor, e para os problemas SMD, $gen = 80$ e $genf = 100$.

– **Resultados e discussões** –

A primeira etapa dos experimentos consiste em testar o método nos 19 problemas teste. Para este conjunto de problemas, o método BLDE$_M$ com a variante DE/best/1/bin não obteve bom desempenho. Este comportamento foi devido ao processo de busca agressivo desta variante, que realiza a busca apenas em torno do melhor indivíduo da população, e que não foi capaz de tratar a diversidade

dos problemas testados. Por isso, para os problemas 1 a 19, apenas os resultados com a variante DE/target-to-best/1/bin serão reportados.

A Tabela 6.5 apresenta a mediana e a média do valor da função objetivo do nível superior e inferior. A última coluna indica a melhor solução conhecida na literatura. Nestes experimentos, o valor da função objetivo de ambos os níveis é analisado conforme varia a probabilidade de uso do metamodelo β.

Tabela 6.5: Média e mediana do valor da função objetivo para os níveis superior e inferior obtido pelo BLDE$_M$ nos problemas 1 a 19.

		Nível superior		Nível inferior		
Problema	β	Mediana	Média	Mediana	Média	Solução de Ref.
1	0.8	99.75	99.18	0.002961	0.1089	
	0.5	99.99	99.57	0.0001027	0.07055	$F^* = 100$
	0.3	100	99.73	1.008e-05	0.05117	$f^* = 0$
	0.0	100	100	2.212e-06	4.061e-06	
2	0.8	224.9	223	99.13	94.23	
	0.5	225.1	224.5	99.77	99.16	$F^* = 225$
	0.3	225.1	225	99.76	99.36	$f^* = 100$
	0.0	225.1	225.1	99.83	99.79	
3**	0.8	28.43	28.55	-3.154	-5.816	
	0.5	28.76	28.72	-3.156	-3.135	$F^* = 29.2$
	0.3	28.86	28.39	-3.182	-3.276	$f^* = -3.2$
	0.0	28.91	28.84	-3.174	-3.16	
4**	0.8	3.327	3.13	2.461	2.268	
	0.5	3.31	3.386	2.715	2.322	$F^* = 3.25$
	0.3	3.251	3.337	3.587	2.824	$f^* = 4$
	0.0	3.247	3.245	3.945	3.936	
5	0.8	0	-1.661	200	192.2	
	0.5	0	-0.6545	200	198.1	$F^* = 0$
	0.3	0	-0.5524	200	198.7	$f^* = 200$
	0.0	0	-1.409e-06	200	193.3	
6	0.8	16.32	12.43	0.6178	-1.548	
	0.5	17	14.77	0.9964	-0.05756	$F^* = 17$
	0.3	17	15.14	0.9989	0.06603	$f^* = 1$
	0.0	17	17	0.9989	0.9925	
7	0.8	-12.82	-12.8	-0.9483	-0.8823	
	0.5	-12.83	-12.84	-0.933	-0.9066	$F^* = -12.679$
	0.3	-12.83	-12.83	-0.9578	-0.9172	$f^* = -1.015$
	0.0	-12.82	-12.83	-0.9686	-0.951	
8**	0.8	48.84	47.39	-16.89	-15.85	
	0.5	48.96	48.59	-16.98	-16.71	$F^* = 49$
	0.3	48.97	48.96	-16.98	-16.97	$f^* = -17$
	0.0	48.96	48.95	-16.98	-16.97	

**Problema de maximização no nível superior e inferior.

		Nível superior		Nível inferior		
Problema	β	Mediana	Média	Mediana	Média	Solução da Ref.
9	0.8	-1.544	-1.658	7.948	8.555	
	0.5	-1.416	-1.51	7.627	8.031	F^* = -1.407
	0.3	-1.405	-1.422	7.61	7.727	f^* = 7.61
	0.0	-1.407	-1.405	7.616	7.606	
10	0.8	-1.025	-1.024	0.00113	0.01522	
	0.5	-1.024	-1.035	0.001833	0.00852	F^* = -1
	0.3	-1.017	-1.037	0.0006187	0.02157	f^* = 0
	0.0	-1.015	-1.015	0.000304	0.0003688	
11	0.8	2049	1913	85.91	3937	
	0.5	2099	1963	155.6	4077	F^* = 2250
	0.3	2210	2119	156.9	472	f^* = 197.75
	0.0	2248	2248	197.3	192.5	
12	0.8	-12.02	-12.77	4	4.069	
	0.5	-12.02	-12.54	4	4.058	F^* = -12
	0.3	-11.99	-12.4	3.997	4.038	f^* = 4
	0.0	-11.99	-11.98	3.997	3.993	
13	0.8	3.117	3.262	-6.696	-7.312	
	0.5	3.117	3.117	-6.696	-6.699	F^* = 3.111
	0.3	3.113	3.114	-6.686	-6.702	f^* = -6.662
	0.0	3.113	3.117	-6.682	-6.696	
14	0.8	1	0.839	0	175.2	
	0.5	1	0.9553	0	90.97	F^* = 1
	0.3	1	0.9996	0	27.77	f^* = 0
	0.0	1	1	0	0	
15**	0.8	951.3	760.5	1	1	
	0.5	1000	854.4	1	1	F^* = 1000
	0.3	1000	881.1	1	1	f^* = 1
	0.0	1000	885.2	1	1	
16	0.8	4.766	4.581	3.845	4.025	
	0.5	4.805	4.738	3.911	3.904	F^* = 5
	0.3	4.957	4.88	4.168	4.075	f^* = 4
	0.0	4.997	4.997	4.025	4.033	
17	0.8	9	9	3.974e-13	1.804e-11	
	0.5	9	9	7.861e-14	4.893e-12	F^* = 9
	0.3	9	9	5.652e-14	3.538e-13	f^* = 0
	0.0	9	9	1.36e-14	1.813e-13	
18**	0.8	84.78	81.76	-50.07	-49	
	0.5	84.94	83.55	-50.13	-49.64	F^* = 85.09
	0.3	85.01	84.93	-50.15	-50.12	f^* = -50.181
	0.0	85.02	84.98	-50.15	-50.14	
19	0.8	0.1522	0.1617	0.5003	0.5755	
	0.5	0.1851	0.1908	0.4238	0.5248	F^* = 0.081
	0.3	0.1747	0.1833	0.4197	0.5119	f^* = 0.666
	0.0	0.1713	0.1801	0.5467	0.5507	

**Problema de maximização no nível superior e inferior.

A partir da Tabela 6.5, é possível observar que o método proposto, sem utilizar o metamodelo ($\beta = 0$), foi capaz de alcançar, ou de chegar muito próximo, das melhores soluções conhecidas em todos os problemas testados. No entanto, quando a probabilidade de uso do metamodelo aumenta, para alguns problemas, o método parece não ser capaz de alcançar as melhores soluções conhecidas. Pode-se observar que na maioria dos casos, quando $\beta \geq 0.5$, as soluções desviam-se ligeiramente dos resultados esperados.

Esse comportamento indica que, para alguns casos, o metamodelo utilizado não foi capaz de auxiliar a busca em direção ao ótimo global do problema, principalmente quando a probabilidade de uso do metamodelo é alta. Porém, a degradação das soluções é pequena quando comparada à economia no número de avaliações de função do nível inferior, como mostra a Tabela 6.6.

A Tabela 6.6, descreve os resultados referentes ao número de avaliações de função realizadas para os problemas 1 a 19, informando o valor da mediana e da média do número de avaliações de função do nível superior e inferior, assim como a porcentagem de redução (%Redução) no número de avaliações de função realizadas no nível inferior para cada valor de β, com base nos valores da mediana. A porcentagem de redução foi calculada em relação ao número de avaliações de função realizadas sem o uso do metamodelo, i.e, com $\beta = 0$.

Analisando os resultados, tem-se que nos problemas 5, 14 e 17, mesmo com uma alta probabilidade de uso do metamodelo ($\beta = 0.8$), o método foi capaz de encontrar a solução ótima com uma economia no número de avaliações de função no nível inferior de aproximadamente 77%, 57% e 70%, respectivamente. Nos demais casos, com exceção do problema 16, o método alcançou boas soluções e gerou uma economia significativa no número de avaliações de função no nível inferior, sendo que houve um pequeno aumento no número de avaliações de função no nível superior. Em alguns outros casos, como por exemplo nos problemas 7, 11, 15 e 17, houve ganho adicional, com a redução também no número de avaliações de função no nível superior.

Tabela 6.6: Média e mediana do número de avaliações de função para os níveis superior e inferior obtidos pelo BLDE_M nos problemas 1 a 19.

		Avaliações de função do nível superior		Avaliações de função do nível inferior		Nível inferior
Problema	β	Mediana	Média	Mediana	Média	%Redução
1	0.8	840	1677	140200	246200	63.076
	0.5	750	1615	259200	512000	31.736
	0.3	690	1035	269500	433400	29.023
	0.0	631	634.5	379700	379800	-
2	0.8	4425	4039	1470000	1541000	52.396
	0.5	1966	2211	1756000	1940000	43.135
	0.3	1834	2268	2334000	2776000	24.417
	0.0	1818	1818	3088000	3094000	-
3**	0.8	1296	1403	768700	816500	66.535
	0.5	1040	1111	1465000	1430000	36.221
	0.3	1024	1014	1819000	1838000	20.810
	0.0	963.5	974.3	2297000	2391000	-
4**	0.8	1264	1435	1065000	1070000	61.160
	0.5	875	890.7	1646000	1651000	39.971
	0.3	822.5	837.2	2013000	2072000	26.586
	0.0	758.5	751.2	2742000	2715000	-
5	0.8	6040	5832	613400	605100	77.597
	0.5	6036	6037	1471000	1456000	46.275
	0.3	6032	6035	1954000	1973000	28.634
	0.0	6032	6033	2738000	2926000	-
6	0.8	6030	4987	495100	438800	80.643
	0.5	6031	5852	1271000	1245000	49.901
	0.3	6031	6032	1766000	1776000	30.390
	0.0	6030	5844	2537000	2442000	-
7	0.8	1812	2805	507900	649700	88.643
	0.5	2908	3602	1546000	1924000	65.429
	0.3	4602	3792	3242000	2808000	27.504
	0.0	4306	3958	4472000	4122000	-
8**	0.8	900.5	994	122700	130500	61.811
	0.5	783	806.3	223900	219400	30.314
	0.3	734.5	736.7	256300	262900	20.230
	0.0	653.5	661.1	321300	324200	-
9	0.8	774	2453	112800	286600	57.737
	0.5	678	1813	192300	459000	27.951
	0.3	568.5	1304	215900	454700	19.108
	0.0	553	559	266900	272000	-
10	0.8	902	1660	196300	291900	66.496
	0.5	830.5	1289	383300	538300	34.579
	0.3	789	1154	478500	632700	18.331
	0.0	728	739	585900	595400	-

**Problema de maximização no nível superior e inferior.

Problema	β	Avaliações de função do nível superior Mediana	Média	Avaliações de função do nível inferior Mediana	Média	Nível inferior %Redução
11	0.8	6030	5848	556300	541900	81.052
	0.5	6032	6033	1500000	1485000	48.910
	0.3	6032	6016	2089000	2089000	28.849
	0.0	6033	6033	2936000	2945000	-
12	0.8	941.5	2719	279900	372100	55.705
	0.5	670	1892	399000	626200	36.857
	0.3	604	1513	494300	733500	21.776
	0.0	583	581.2	631900	642600	-
13	0.8	690	690.7	86200	88980	65.589
	0.5	604	613.6	152000	158200	39.321
	0.3	573	581.1	189400	188500	24.391
	0.0	561	561.1	250500	250300	-
14	0.8	453	1934	70580	162400	57.172
	0.5	424.5	991.9	105700	205500	35.862
	0.3	439	621.9	129300	175700	21.541
	0.0	420	433	164800	169500	-
15**	0.8	6094	5555	2604000	2518000	88.795
	0.5	6462	5849	9650000	8123000	58.477
	0.3	6719	6063	14780000	12610000	36.403
	0.0	7057	6399	23240000	20090000	-
16	0.8	6030	5681	821800	801800	-70.533
	0.5	6030	6030	1864000	1874000	-286.802
	0.3	6030	5665	2530000	2408000	-425.005
	0.0	615	2119	481900	1339000	-
17	0.8	480	490	65940	61410	70.109
	0.5	480	494	117800	121200	46.600
	0.3	480	478	156300	154800	29.148
	0.0	510	491	220600	213400	-
18**	0.8	1114	1588	148400	191400	59.298
	0.5	874.5	1242	226200	332700	37.050
	0.3	785	1012	277500	351900	23.889
	0.0	741.5	747.4	364600	369600	-
19	0.8	5388	4464	5318000	5404000	8.421
	0.5	1635	2291	4829000	6748000	16.842
	0.3	1461	2171	5336000	8668000	8.111
	0.0	1116	1391	5807000	7449000	-

**Problema de maximização no nível superior e inferior.

No problema 16, houve um comportamento inesperado com relação ao número de avaliações de função do nível inferior: ocorreu um aumento neste número e não uma redução, como esperado. Houve também um aumento desproporcional no número de avaliações de função do nível superior quando comparado aos demais problemas analisados. Tal comportamento é devido ao fato do metamodelo ter convergido para um ótimo global correspondente ao metamodelo, o que causou

um "retardo" na convergência do método para o ótimo global do problema. Como consequência, houve um aumento no número de gerações, e assim no número de avaliações de função em ambos os níveis.

A segunda etapa dos experimentos analisa o desempenho do método BLDE$_M$ na resolução dos problemas SMD. A Tabela 6.7 apresenta a mediana e a média do valor da função objetivo do nível superior e inferior onde "target" e "best" indicam os resultados obtidos pelas variantes DE/target-to-best/1/bin e DE/best/1/bin, respectivamente. Já a Tabela 6.8, apresenta a mediana e a média do número de avaliações de função do nível superior e inferior, assim como a porcentagem de redução (%Redução) no número de avaliações de função no nível inferior para cada valor de β, com base no valor da mediana. Lembrando que a porcentagem de redução foi calculada em relação ao número de avaliações de função realizadas sem o uso do metamodelo, i.e, com $\beta = 0$.

Para os problemas SMD, ambas as variantes foram capazes de resolver todos os problemas quando o metamodelo não foi utilizado ($\beta = 0$). Podemos destacar também os resultados obtidos para os problemas SMD1 e SMD3 onde, mesmo com uma alta probabilidade de uso do metamodelo ($\beta = 0.8$) o algoritmo foi capaz de encontrar o ótimo global com uma economia de aproximadamente 74% em ambos os problemas nas duas variantes. Para os demais problemas SMD, assim como ocorreu nos problemas 1 à 19, quando a probabilidade de uso do metamodelo é alta ($\beta \geq 0.5$), as soluções desviaram-se ligeiramente dos resultados esperados. Este fato pode ter ocorrido pois, diferentemente dos problemas SMD1 e SMD3, os demais problemas são caraterizados por apresentarem uma situação de conflito entre os níveis, o que pode ter dificultado o auxílio do metamodelo na busca por melhores soluções. No entanto, ainda assim, houve redução no número de avaliações de função principalmente no nível inferior.

Em relação às duas variantes, observa-se que ambas geraram soluções similares. Porém, a variante DE/best/1/bin realizou menos avaliações de função nos problemas SMD, principalmente no nível inferior, demonstrando sua

superioridade no quesito desempenho.

Tabela 6.7: Média e mediana do valor da função objetivo para os níveis superior e inferior obtido pelo BLDE$_M$ nos problemas SMD1 a SMD6.

			Nível superior		Nível inferior	
Problema	β	Variante	Mediana	Média	Mediana	Média
SMD1	0.8	target	5.018e-05	5.096e-05	3.396e-05	3.382e-05
	0.5	target	4.157e-05	4.456e-05	2.207e-05	2.861e-05
	0.3	target	3.754e-05	4.068e-05	2.09e-05	2.472e-05
	0.0	target	4.209e-05	4.34e-05	2.229e-05	2.259e-05
	0.8	best	4.849e-05	7.222e-05	2.685e-05	4.241e-05
	0.5	best	4.506e-05	4.638e-05	2.699e-05	3.093e-05
	0.3	best	3.439e-05	3.507e-05	1.83e-05	2.291e-05
	0.0	best	3.786e-05	3.993e-05	1.833e-05	2.172e-05
SMD2	0.8	target	-0.7954	-2.235	3.522	7.213
	0.5	target	-0.08149	-0.3615	0.3892	2.136
	0.3	target	-2.872e-05	-0.1987	0.0002816	1.029
	0.0	target	9.218e-06	1.094e-05	7.948e-06	8.786e-06
	0.8	best	-1.854	-2.494	4.703	8.028
	0.5	best	-0.0008475	-0.4245	0.007381	3.17
	0.3	best	3.01e-06	-0.1375	3.595e-05	1.485
	0.0	best	1.175e-05	1.159e-05	7.196e-06	7.062e-06
SMD3	0.8	target	3.214e-05	3.543e-05	1.412e-05	2e-05
	0.5	target	3.411e-05	3.554e-05	1.954e-05	2.282e-05
	0.3	target	3.885e-05	3.992e-05	1.905e-05	2.354e-05
	0.0	target	3.017e-05	3.244e-05	1.418e-05	1.828e-05
	0.8	best	3.725e-05	3.769e-05	1.863e-05	2.419e-05
	0.5	best	3.547e-05	3.857e-05	2.037e-05	2.402e-05
	0.3	best	3.816e-05	3.898e-05	1.84e-05	2.037e-05
	0.0	best	3.713e-05	4.048e-05	1.501e-05	2.061e-05
SMD4	0.8	target	-0.2956	-0.3005	0.6799	0.619
	0.5	target	-0.00557	-0.07087	0.01513	0.1951
	0.3	target	-2.977e-06	-0.01263	1.443e-05	0.0724
	0.0	target	4.6e-07	3.118e-07	3.152e-06	3.932e-06
	0.8	best	-0.03767	-0.1241	0.3506	0.4932
	0.5	best	-3.005e-05	-0.03306	0.0002353	0.1174
	0.3	best	9.793e-07	-0.003058	3.518e-06	0.03883
	0.0	best	1.459e-06	1.583e-06	1.048e-06	1.161e-06
SMD5	0.8	target	-0.1303	-1.29	1.791	11.18
	0.5	target	5.663e-06	-0.1738	0.001259	4.929
	0.3	target	8.34e-06	-0.001052	5.225e-05	0.08306
	0.0	target	3.254e-05	3.52e-05	1.595e-05	1.993e-05
	0.8	best	-0.1989	-1.29	1.473	4.267
	0.5	best	4.457e-06	-0.292	0.0002627	1.966
	0.3	best	1.323e-05	-0.02034	3.015e-05	0.1522
	0.0	best	3.444e-05	3.54e-05	2.031e-05	1.972e-05
SMD6	0.8	target	-2.161	-4.687	7.027	17.94
	0.5	target	-0.007443	-1.548	0.04469	8.684
	0.3	target	4.215e-06	-0.5741	9.236e-05	3.094
	0.0	target	2.898e-05	2.819e-05	4.767e-05	5.152e-05
	0.8	best	-1.098	-3.824	3.395	12.62
	0.5	best	-0.0007801	-0.4176	0.006854	1.689
	0.3	best	6.286e-06	-0.04317	0.0002171	0.2733
	0.0	best	3.171e-05	3.662e-05	2.107e-05	2.456e-05

Tabela 6.8: Média e mediana do número de avaliações de função para os níveis superior e inferior obtidos pelo BLDE$_M$ nos problemas SMD1 a SMD6.

			Avaliações de função do nível superior		Avaliações de função do nível inferior		Nível inferior
Problema	β	Variante	Mediana	Média	Mediana	Média	%Redução
SMD1	0.8	target	2940	2954	1751000	1751000	76.967
	0.5	target	2940	2920	4066000	4052000	46.514
	0.3	target	2880	2882	5336000	5292000	29.808
	0.0	target	2850	2868	7602000	7611000	-
	0.8	best	1800	1814	474000	461600	74.084
	0.5	best	1755	1770	1021000	1006000	44.177
	0.3	best	1740	1767	1391000	1371000	23.948
	0.0	best	1710	1737	1829000	1854000	-
SMD2	0.8	target	6030	5925	3509000	3417000	53.455
	0.5	target	6030	5136	7342000	6521000	2.613
	0.3	target	3165	4317	6154000	7424000	18.371
	0.0	target	2850	2839	7539000	7518000	-
	0.8	best	6031	6031	1271000	1249000	23.664
	0.5	best	6031	4968	2637000	2323000	-58.378
	0.3	best	2070	3399	1484000	2206000	10.871
	0.0	best	1770	1790	1665000	1703000	-
SMD3	0.8	target	3075	3094	2232000	2245000	74.286
	0.5	target	2925	2922	4570000	4538000	47.350
	0.3	target	2895	2918	6264000	6245000	27.834
	0.0	target	2910	2872	8680000	8653000	-
	0.8	best	1860	1874	448800	477700	73.459
	0.5	best	1800	1797	1005000	984600	40.568
	0.3	best	1770	1759	1254000	1247000	25.843
	0.0	best	1710	1736	1691000	1706000	-
SMD4	0.8	target	6030	5946	3220000	3317000	61.616
	0.5	target	6030	5597	6913000	6772000	17.594
	0.3	target	3345	4132	6414000	7224000	23.543
	0.0	target	3180	3170	8389000	8340000	-
	0.8	best	6032	6032	1054000	1028000	29.214
	0.5	best	6030	5130	1923000	1815000	-29.147
	0.3	best	2250	3346	1322000	1699000	11.216
	0.0	best	1951	1963	1489000	1503000	-
SMD5	0.8	target	6030	6030	7772000	7655000	55.410
	0.5	target	6030	5898	17800000	18110000	-2.123
	0.3	target	4605	4805	19520000	20600000	-11.991
	0.0	target	2895	2884	17430000	17370000	-
	0.8	best	6031	5803	1286000	1258000	30.033
	0.5	best	5986	4780	2417000	2365000	-31.502
	0.3	best	2626	3286	2143000	2273000	-16.594
	0.0	best	1740	1727	1838000	1859000	-
SMD6	0.8	target	6030	6030	2911000	2931000	55.894
	0.5	target	6030	4905	5878000	5427000	10.939
	0.3	target	3030	3693	4898000	5771000	25.788
	0.0	target	2880	2878	6600000	6601000	-
	0.8	best	6031	6031	1222000	1203000	17.655
	0.5	best	6031	4893	1955000	1958000	-31.739
	0.3	best	1980	3045	1259000	1721000	15.162
	0.0	best	1680	1692	1484000	1508000	-

– Conclusões –

Através destes experimentos foi possível observar que a variante DE/target-to-best/1/bin gerou melhores resultados para os problemas 1 a 19 e a variante DE/best/1/bin foi mais eficiente na resolução dos problemas SMD.

O BLDE$_M$ encontrou soluções de qualidade em todos os problemas quando a probabilidade de uso do metamodelo foi de até 50%, ocasionando uma redução significativa no número de avaliações de função do nível inferior. Assim, o metamodelo e o novo critério de parada foram capazes de reduzir o número de avaliações de função, principalmente no nível inferior, em todos os problemas.

Para alguns caso, com probabilidade de uso do metamodelo acima de 50% gerou-se soluções de baixa qualidade. Uma possível causa seria a simplicidade do metamodelo adotado que, com alta probabilidade de uso, não foi capaz de fornecer boas aproximações para os problemas testados.

Dos experimentos, é importante ressaltar que uso do metamodelo permitiu a redução no número de avaliações de função sem comprometer a qualidade das soluções obtidas. No entanto, observou-se que seu uso deve ser balanceado para alcançar uma relação de compromisso entre soluções de boa qualidade e redução no número de avaliações de função.

6.4 Considerações finais

Neste capítulo, dois algoritmos foram implementados para a resolução de diversos problemas de otimização em dois níveis. As metodologias propostas fizeram uso de dois métodos de evolução diferencial, cada um responsável por otimizar um nível do problema.

O primeiro algoritmo desenvolvido, BLDE, apresentou robustez e flexibilidade, sendo capaz de encontrar a solução ótima, ou muito próxima dela, em todos os problemas, e apresentou uma melhor solução para o problema 4.

O segundo algoritmo, BLDE$_M$, melhorou o desempenho do método anterior através da utilização de um metamodelo no nível inferior e de um critério de parada

mais eficiente em ambos os níveis. Com até 50% de uso do metamodelo, obteve-se em média[2] uma redução no número de avaliações de função no nível inferior de 40.3%, no entanto, essa redução causou uma pequena degradação na qualidade das soluções geradas de aproximadamente 3.95%. Já com 80% de probabilidade de uso do metamodelo obteve-se 64.57% de redução no número de avaliações de função no nível inferior, sendo que a degradação da qualidade das soluções geradas foi de 4.67%.

Destes resultados podemos concluir que o uso do metamodelo possibilitou melhorar o desempenho do método pois foi capaz de reduzir de forma significativa o número de avaliações de função, principalmente no nível inferior.

O próximo capítulo apresenta as metodologias desenvolvidas para tratar o problema de planejamento de produção e distribuição, onde dois novos métodos de otimização serão propostos.

[2] Quando comparado com a não utilização do metamodelo ($\beta = 0$).

Capítulo 7

Metodologias Propostas e Experimentos Computacionais para a Resolução dos Problemas da Classe 2

Esta seção apresenta a segunda etapa dos experimentos computacionais onde dois métodos de otimização serão apresentados para a resolução dos problemas da Classe 2, que foram descritos na seção 5.2. Estes métodos utilizam o algoritmo ACS para resolver o problema de nível superior e o algoritmo DE para resolver o problema de nível inferior.

Os estudos desenvolvidos em torno da primeira metodologia, denominada Bilevel ACO+DE, deram origem ao trabalho [Angelo and Barbosa, 2013], onde o método proposto foi analisado e comparado com um outro método presente na literatura [Calvete et al., 2011]. A descrição deste algoritmo e os experimentos computacionais realizados serão descritos na seção 7.4.

A segunda metodologia, denominada Bilevel ACO+DE$^+$, vem aprimorar o método desenvolvido anteriormente, onde propõe-se uma nova modelagem para o algoritmo e um critério de parada mais eficiente. A descrição deste método e os experimentos computacionais serão descritos na seção 7.5. Os resultados obtidos e as análises realizadas deram origem ao trabalho [Angelo and Barbosa, –] que encontra-se em processo de revisão. Nesta etapa, buscou-se estudar as implicações do uso de um método aproximativo no nível inferior, investigando como soluções

aproximadas neste nível podem afetar o processo de busca no nível superior e a solução final do problema em dois níveis.

Ambos os algoritmos desenvolvidos foram analisados utilizando o mesmo conjunto de instâncias, descritas a seguir.

7.1 Descrição das instâncias do problema em dois níveis

Os experimentos foram realizados em um conjunto de instâncias do problema de planejamento de produção e distribuição (PPPD). Estas instâncias foram adaptadas de instâncias do problema de roteamento de veículos com múltiplos depósitos (PRVMD), disponíveis em `http://neo.lcc.uma.es/vrp/`.

As instâncias do PRVMD fornecem apenas informações relativas ao problema de roteamento, como por exemplo, a localização dos depósitos e dos revendedores, a demanda e o tempo de entrega dos revendedores, a máxima duração das rotas e a capacidade máxima de cada veículo. Desta forma, os dados referentes ao problema de transporte (PT) tiveram que ser incluídos nestas instâncias, como por exemplo, a localização das fábricas, os custos associados à produção em uma fábrica para um certo depósito e a disponibilidade total de produtos em cada fábrica. A Tabela 7.1 resume a descrição das instâncias utilizadas nos experimentos computacionais.

Tabela 7.1: Descrição das instâncias do PPPD em dois níveis.

Problema	Depósitos	Veículos	Fábricas	Revendedores	T	Q
BiPR01	4	1	4	48	500	200
BiPR02	4	2	4	96	480	195
BiPR03	4	3	4	144	460	190
BiPR04	4	4	4	192	440	185
BiPR05	4	5	4	240	420	180
BiPR06	4	6	4	288	400	175
BiPR07	6	1	6	72	500	200
BiPR08	6	2	6	144	475	190
BiPR09	6	3	6	216	450	180
BiPR10	6	4	6	288	425	170

T é o tempo máximo de duração de uma rota e Q é a capacidade do veículo.

As 10 instâncias analisadas variam de 48 a 288 revendedores com 4 ou 6 depósitos e fábricas. O posicionamento das fábricas foi feito de forma aleatória

num domínio definido por $[-200, 200] \times [-200, 200]$. A disponibilidade de produtos em cada fábrica foi selecionada aleatoriamente no intervalo $[0, TD]$ uniformemente distribuída, onde TD é a demanda total dos revendedores, sendo que a soma das disponibilidades é igual à demanda total de produtos. O custo c^1_{ij}, que representa a distância entre as localidades, é obtido calculando-se a distância Euclidiana entre as localizações i e j. O custo c^3_{ij}, que representa o custo de aquisição dos produtos na fábrica i pelo depósito j, é gerado aleatoriamente e uniformemente distribuído no intervalo $[0.5, 1]$. Finalmente, o custo c^2_{ij}, que representa o custo de produção da fábrica i para o depósito j, é composto pela soma de dois termos. O primeiro termo é gerado aleatoriamente e uniformemente distribuído no intervalo $[2, 5]$ e o segundo é proporcional à distância entre a fábrica i e o depósito j. A descrição dos dados adicionados em cada uma das 10 instâncias do PRVMD encontra-se no Apêndice A.

Desta forma, as instâncias do problema de planejamento de produção e distribuição foram construídas para que o problema de otimização formulado na equação (5.14) seja resolvido.

7.2 Otimização do nível superior – ACO para o PRVMD

O método de resolução do problema do nível superior utilizou como base o algoritmo ACS, descrito na seção 4.1.1, para resolver o PRVMD, que foi descrito na seção 5.2.1.

– Construção da solução –

Cada formiga/solução no ACO contém um conjunto de veículos que irá construir uma solução completa para o PRVMD, i.e., diversos ciclos Hamiltonianos (sub-rotas) passando por cada revendedor uma única vez, partindo e retornando de um mesmo depósito. No algoritmo, cada formiga apresenta basicamente as seguintes estruturas: (i) uma matriz, que armazena em cada linha uma sub-rota gerada por um determinado veículo; (ii) um vetor, contendo os nós (revendedores)

que ainda não foram visitados, tal que seus valores são configurados para 1 se um nó já foi visitado, e 0 se o nó ainda não foi visitado; e (iii) uma variável contendo o valor da solução, i.e., o valor da função objetivo.

Durante a construção da solução, a escolha dos nós a serem visitados é feita de forma determinística ou probabilística, conforme as equações (4.1) e (4.2), respectivamente. Antes de cada passo, a formiga verifica se a escolha do próximo nó viola alguma restrição, ou seja, verifica-se se a capacidade do veículo foi violada ou se o tempo total de serviço foi excedido. Se alguma dessas restrições for violada, a formiga tenta selecionar um outro nó, verificando se nenhum destas restrição foi violada. Se não for possível selecionar um novo nó sem violar alguma restrição, a formiga então retorna para o depósito de onde ela partiu.

Desta forma, um conjunto de rotas são geradas para cada formiga, tendo ao final uma solução completa que visita todos os revendedores uma única vez. Note que, o tratamento das restrições no ACO é intrinsecamente formulado na estrutura da formiga e no processo de construção da solução, não havendo assim necessidade de um procedimento adicional para o tratamento das restrições.

– Procedimento de busca local 2-opt –

Após a construção da solução, um procedimento de busca local é aplicado, na sub-rota de cada veículo, na tentativa de melhorar as soluções obtidas. Nesta etapa, utilizou-se o método conhecido como 2-opt [Hoos and Stutzle, 2004]. De forma resumida, o método 2-opt realiza possíveis trocas entre pares de arcos, refazendo as conexões quando houver uma melhoria na rota. Assim, o procedimento testa todas as possíveis trocas entre os arcos até que nenhuma melhoria possa ser obtida. Uma ilustração deste procedimento é apresentado na Figura 7.1, adaptada de [Hoos and Stutzle, 2004], indicando duas possíveis trocas em um ciclo Hamiltoniano, que começa e termina em um depósito d.

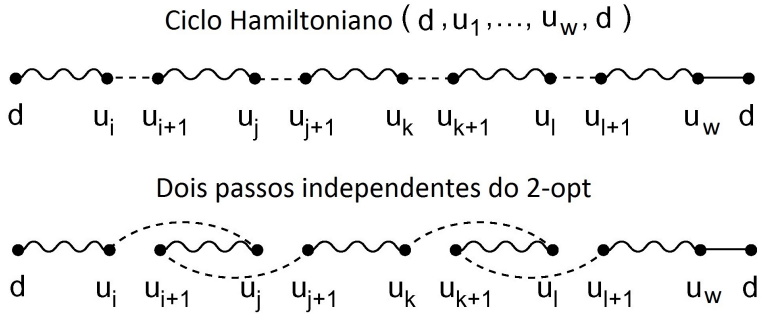


Figura 7.1: Exemplo da busca local 2-opt.

Após este procedimento, a atualização local das trilhas de feromônio é realizada nos caminhos construídos pelas formigas, conforme equação (4.3).

– Atualização global de feromônio –

A atualização global do feromônio é realizada após a execução do procedimento do nível inferior, quer irá retornar as variáveis do nível inferior associadas à solução gerada pelo ACO. Assim, após o cálculo do valor da função objetivo do líder, a atualização global da matriz de feromônio é realizada, conforme equação (4.4).

7.3 Otimização do nível inferior – DE para o PT

O método de resolução do problema do nível inferior utilizou como base o algoritmo DE, descrito na seção 4.2.1, para resolver o PT, que foi descrito na seção 5.2.2.

O método de DE desenvolvido foi construído especificamente para o problema de transporte, onde foram incorporadas informações relativas ao problema para a modelagem das soluções candidatas e dos operadores de manipulação das soluções. Assim, três etapas do algoritmo base descrito na seção 4.2.1 foram adaptadas: (i) a população inicial gerada satisfaz todas as restrições do problema; (ii) apenas o operador de mutação é utilizado; (iii) como a operação de mutação pode tornar a solução gerada infactível, pequenos ajustes são aplicados para "corrigir" a solução, de forma a torná-la factível.

– **Inicialização da população** –

No início do algoritmo, a população inicial é gerada de forma que todas as restrições sejam satisfeitas, conforme proposto em [Michalewicz, 1996]. O Algorimo 7 apresenta este procedimento.

Algoritmo 7: Procedimento de inicialização

Entrada : Vetores $demanda[N]$ e $oferta[M]$
Rotule todos os números de 1 à $N \times M$ como não selecionados;
repeat
  Selecione aleatoriamente um número q ainda não selecionado entre 1 e $N \times M$;
  Rotule q como selecionado;
  (linha) $i = \lfloor (q/N) \rfloor$;
  (coluna) $j = q \ mod \ N$;
  $val = min(demanda[i], oferta[j])$;
  (solução inicial) $u_{ij} = val$;
  $demanda[i] = demanda[i] - val$;
  $oferta[j] = oferta[j] - val$;
until (todos os números forem selecionados);
Retornar: Solução u que satisfaz todas as restrições

Com este procedimento de inicialização, a operação de mutação é capaz de manter as restrições lineares (5.10) e (5.11) para qualquer valor de ponderação da diferença $F \in (0, 1]$. No entanto, quando valores reais são utilizados para F, a restrição de integralidade (5.12) é violada. Desta forma, adotando-se $F = 1$, todas as variáveis de decisão mantêm-se inteiras após a mutação.

– **Ajustes nas soluções candidatas** –

Para cada variável de decisão, limites superior e inferior são definidos, conforme descritos pelas restrições (5.10), (5.11) e (5.12). Assim, após a operação de mutação, quando uma componente u_{ij} de uma nova solução candidata é gerada fora destes limites, a seguinte operação de projeção é realizada

$$\text{se } u_{ij} > P_i \text{ ou } u_{ij} > D_j \text{ ou } u_{ij} < 0 \text{ então } u_{ij} = 0. \tag{7.1}$$

Como consequência desta projeção, a solução candidata torna-se infactível

uma vez que pode ocorrer uma das seguintes situações

$$\sum_{j=1}^{M} u_{ij} < P_i \quad \text{para } i = 1, ..., N \quad \text{e/ou} \tag{7.2}$$

$$\sum_{i=1}^{N} u_{ij} < D_j \quad \text{for } j = 1, ..., M. \tag{7.3}$$

Assim, é necessário completar a solução com os valores faltantes, a fim de satisfazer todas as restrições. Considere um exemplo onde utiliza-se a variante DE/rand/1/bin e sejam x_{r_1}, x_{r_2} e x_{r_3} três soluções candidatas selecionadas aleatoriamente na população

$$x_{r_1} = \begin{pmatrix} 10 & 0 \\ 0 & 6 \\ 0 & 12 \end{pmatrix}, x_{r_2} = \begin{pmatrix} 1 & 9 \\ 6 & 0 \\ 12 & 0 \end{pmatrix}, x_{r_3} = \begin{pmatrix} 0 & 10 \\ 0 & 6 \\ 5 & 7 \end{pmatrix}$$

Deste exemplo tem-se $P = (10, 6, 12)$ (somatório em linha) e $D_{r_1} = (10, 18)$, $D_{r_2} = (19, 9)$ e $D_{r_3} = (5, 23)$ (somatório em coluna). Realizando a operação de mutação $x_{novo} = x_{r_1} + F.(x_{r_2} - x_{r_3})$ para a obtenção de uma nova solução, temos

$$x_{novo} = \begin{pmatrix} 10 & 0 \\ 0 & 6 \\ 0 & 12 \end{pmatrix} + F. \begin{pmatrix} 1-0 & 9-10 \\ 6-0 & 0-6 \\ 12-5 & 0-7 \end{pmatrix} = \begin{pmatrix} 11 & -1 \\ 6 & 0 \\ 7 & 5 \end{pmatrix}$$

A solução resultante, que gera $D_{novo} = (24, 4)$, contêm valores que estão fora dos limites estabelecidos pelas restrições. Aplicando-se então a operação de projeção (7.1) obtemos

$$x_{novo} = \begin{pmatrix} 0 & 0 \\ 6 & 0 \\ 7 & 0 \end{pmatrix}$$

Esta operação torna a solução candidata infactível, de acordo com as situações (7.2) e (7.3). Assim, um procedimento de correção é aplicado em x_{novo}

gerando a seguinte solução

$$x_{novo} = \begin{pmatrix} 10 & 0 \\ 6 & 0 \\ 8 & 4 \end{pmatrix}$$

onde tal solução satisfaz todas as restrições.

7.4 ACO+DE em Dois Níveis (Bilevel ACO+DE)

Como dito anteriormente, o método Bilevel ACO+DE utiliza como base o algoritmo ACS para a resolução do problema líder (PRVMD), e o algoritmo DE com a variante DE/rand/1, sem operador de recombinação, para a resolução do problema seguidor (PT). O pseudo-código do método proposto é apresentado pelos Algoritmos 8 e 9, que seguem as descrições apresentadas nas seções 7.2 e 7.3.

O detalhamento de cada etapa do método é dado da seguinte forma:

Passo 0: Iniciar dados. O Algoritmo 8 inicia a partir da leitura da instância do PPPD, onde são fornecidas informações sobre quantidade e capacidade de cada veículo, número de fábricas e revendedores, custos dos níveis superior e inferior envolvidos, entre outros. Nesta etapa, o valor dos parâmetros utilizados pelos algoritmos ACO e DE são inicializados, assim como as matrizes de feromônio e de informação heurística.

Passo 1: Gerar solução inicial. Através da execução de uma heurística gulosa uma solução inicial a_{ini} é gerada para o PRVMD. Associado a esta solução, tem-se o vetor de demandas $D(x_{ini})$ que será enviado para o algoritmo DE-Seguidor, responsável pela obtenção da solução do PT associado. Após a obtenção do $y(x_{ini})$ associado, a solução a_{ini} é avaliada de acordo com a função objetivo do nível superior, representada por F_{ini}.

Passo 2: Iniciar matriz de feromônio. Após a construção da solução inicial, as trilhas de feromônio τ_{ij} são inicializadas com o valor τ_0, associado ao valor da solução inicial F_{ini}.

Passo 3: Construir soluções para o problema de dois níveis. A cada iteração, uma única solução será construída para o problema em dois níveis, onde o ACO é utilizado para resolver o problema líder (Algoritmo 8) e o DE para resolver o problema seguidor (Algoritmo 9).

Passo 3.1: Resolução do problema líder. Nesta etapa, uma solução completa e viável para o PRVMD será construída por cada formiga a_h seguindo a regra de decisão pseudo-aleatória do algoritmo ACS (equações 4.1 e 4.2). Após a construção da solução, o procedimento de busca local 2-opt [Hoos and Stutzle, 2004] é aplicado, a fim de melhorar a qualidade da solução obtida. Em seguida é realizado o procedimento de atualização local de feromônio (equação 4.3). Depois de todas as formigas terem construído uma solução para o PRVMD, seleciona-se a melhor solução gerada na atual iteração x_{ib}, considerando apenas a primeira parcela da função objetivo do líder, descrita pela equação (5.14)[1] . Apenas para esta solução o algoritmo DE-Seguidor com variante DE/rand/1, sem operador de recombinação, será executado.

Passo 3.2: Resolução do problema seguidor.

O vetor de demandas $D(x_{ib})$ associado a solução x_{ib} é enviado para o nível inferior, responsável por resolver o PT associado. Após $genf$ gerações, a melhor solução obtida $y(x_{ib})$ em relação a função objetivo do seguidor é retornada para o líder.

Passo 4: Aplicar atualização global de feromônio. Após a construção da solução para o problema em dois níveis, a próxima etapa seleciona a melhor solução até então obtida a_{bf}, considerando a função objetivo do líder. A atualização global do feromônio (equação (4.4)) é realizada apenas na rota construída pela formiga/solução a_{bf}.

Passo 5: Finalização. O algoritmo termina depois de it_{max} iterações, onde a melhor solução a_{bf} obtida para o problema líder é retornada.

[1] A primeira parcela da equação (5.14) é descrita por $\sum_{s \in S} \sum_{(i,j) \in E} c_{ij}^1 x_{ij}^s$.

Algoritmo 8: Bilevel ACO+DE

Entrada : it_{max} (número de iterações), A_{max} (número de formigas), α (influência do feromônio), β (influência da informação heurística), γ (taxa de evaporação local) e ρ (taxa de evaporação global).

// Iniciar dados
Ler instâncias do problema em dois níveis;
Alocar matriz de feromônio e de informação heurística;

// Gerar solução inicial
Resolver o PRVMD, utilizando uma heurística gulosa, para obter a solução a_{ini};
$y(x_{ini})$ = DE-Seg. (npf, genf, F, x_{ini}); // x_{ini}: vetor de demandas associado à a_{ini}
F_{ini} = `AvaliarSoluçãoLider`$(a_{ini}, y(x_{ini}))$;

// Iniciar matriz de feromônio
Calcular valor inicial do feromônio (τ_0), de acordo com F_{ini};
Iniciar trilhas de feromônio com o valor τ_0;

// Construir soluções para o problema de dois níveis
for $i \leftarrow 1$ **to** it_{max} **do**
  for $h \leftarrow 1$ **to** A_{max} **do**
    // Resolução do problema líder
    Resolver o PRVMD, utilizando ACO, para obter a solução a_h;
    Aplicar procedimento de busca local em a_h;
    Atualização local das trilhas de feromônio com relação à a_h;
    `AvaliarSoluçãoPRVMD`(a_h);
  Selecionar a melhor solução gerada (a_{ib}) de acordo com o PRVMD;

  // Resolução do problema seguidor
  $y(x_{ib})$ = DE-Seg.(npf, genf, F, x_{ib}); // x_{ib}: vetor de demandas associado à a_{ib}
  F = `AvaliarSoluçãoLider`$(a_{ib}, y(x_{ib}))$;

  // Aplicar atualização global de feromônio
  Selecione a melhor formiga (a_{bf}), de acordo com a função objetivo do líder (F);
  Aplicar atualização global de feromônio, associada à solução a_{bf};

Retornar: `MelhorSoluçãoLíder`

Algoritmo 9: DE-Seguidor.

Entrada : npf (tamanho da população), genf (número de gerações), F (fator de mutação), $\overrightarrow{v}$ (variáveis do líder)

G $\leftarrow 0$;
`CriarPopulacaoInicial(npf,` $\overrightarrow{v}$`);` /* Michalewicz [1996] */
for $i \leftarrow 1$ **to** npf **do**
  Avaliar $f(\overrightarrow{v}, \overrightarrow{x}_{i,G})$; /* $\overrightarrow{x}_{i,G}$ é um indivíduo da população */
for G $\leftarrow 1$ **to** genf **do**
  for $i \leftarrow 1$ **to** npf **do**
    `SelecionarAleatoriamente`(r_1, r_2, r_3); /* $r_1 \neq r_2 \neq r_3 \neq i$ */
    for $j \leftarrow 1$ **to** nf **do**
      $u_{i,j,G+1} = x_{r_3,j,G} + F.(x_{r_1,j,G} - x_{r_2,j,G})$;
  `AplicarOperadorProjecao(`$\overrightarrow{u}_{i,G+1}$`)`;
  `AjustarSolucao(`$\overrightarrow{u}_{i,G+1}$`)`;
  Avaliar $f(\overrightarrow{v}, \overrightarrow{u}_{i,G+1})$;
  if $f(\overrightarrow{v}, \overrightarrow{u}_{i,G+1}) \leq f(\overrightarrow{v}, \overrightarrow{x}_{i,G})$ **then**
    $\overrightarrow{x}_{i,G+1} = \overrightarrow{u}_{i,G+1}$;
  else
    $\overrightarrow{x}_{i,G+1} = \overrightarrow{x}_{i,G}$;

Retornar: `MelhorSoluçãoSeguidor`

7.4.1 Experimentos computacionais

O algoritmo Bilevel ACO+DE foi testado em 10 instâncias do PPPD. Os resultados obtidos foram analisados com relação aos resultados do algoritmo BACS, desenvolvido em [Calvete et al., 2011], onde um algoritmo ACO foi utilizado para a resolução do problema líder e um método exato foi utilizado para a resolução do problema seguidor.

Infelizmente, a comparação direta com os resultados obtidos pelo BACS não pôde ser realizada, uma vez que termos aleatórios foram utilizados para a geração das instâncias do problema em dois níveis, conforme descrito na seção 7.1. Como consequência, a instância utilizada em [Calvete et al., 2011] não é exatamente a mesma que foi utilizada para estes experimentos.

Neste experimento, 10 execuções independentes foram realizadas para cada uma das 10 instâncias do problema. Os experimentos foram realizados no cluster do Laboratório Nacional de Computação Científica modelo Sun Blade x6250, com 2 Processadores Intel Xeon E5440 e 16 GB memória.

– Definição do valor dos parâmetros –

Os parâmetros do ACO foram configurados de acordo com os valores estabelecidos em [Calvete et al., 2011]. Nenhum ajuste de parâmetros foi realizado para o algoritmo DE. Os parâmetros utilizados foram configurados da seguinte forma:

- Para o algoritmo de nível superior (ACO):
  - α e β: a importância relativa das trilhas de feromônio e da informação heurística são 1 e 2, respectivamente.
  - ρ e φ: as taxas de evaporação global e local do feromônio são ambas definidas com 0.1.
  - q_0: o parâmetro que regula entre a exploração e a intensificação da busca é definido com 0.9.

- τ_0: o valor inicial da matriz de feromônio é dado por $1/R.F_{inicial}$, onde R é o número de revendedores.
- A_{max}: o número total de formigas é igual a 50.
- it_{max}: o número máximo de iterações é igual a 4000.

• Para o algoritmo de nível inferior (DE):

- npf: o número total de indivíduos na população é igual a 30.
- F: o fator de escala que controla a magnitude da perturbação nos indivíduos é igual a 1.
- $genf$: o número máximo de gerações é igual 2000.

Nenhum teste preliminar foi realizado para avaliar o desempenho das diferentes variantes do DE na resolução do nível inferior. A variante DE/rand/1, sem operador de recombinação, foi utilizada simplesmente por ser a primeira e mais simples das variantes do DE apresentadas na seção 4.2.2.

– **Resultados e discussões** –

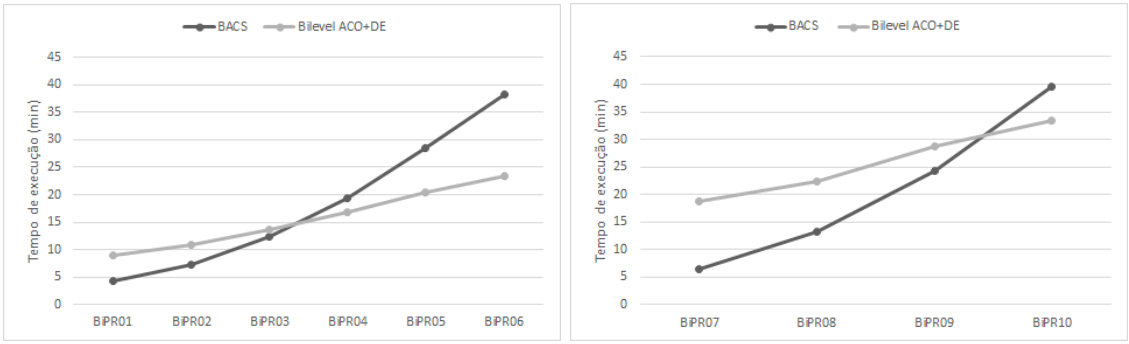
A Tabela 7.2 apresenta os resultados comparativos para as 10 instâncias do PPPD, onde F^{bs}, $\overline{F}$ e F^{ws} indicam o melhor, a média e o pior valor da função objetivo do líder, e $\%F$ representa a diferença em porcentagem entre o valor médio $\overline{F}$ e o melhor valor obtido F^{bs}, i.e., $\%F = ((\overline{F} - F^{bs})/F^{bs}) \times 100$.

Os resultados apresentados indicam que o algoritmo proposto foi capaz de gerar soluções competitivas quando comparado com algoritmo BACS proposto em [Calvete et al., 2011]. Além disso, o algoritmo Bilevel ACO+DE apresentou-se estável uma vez que a diferença entre os valores médios e as melhores soluções obtidas foi menor que 6%, no pior caso, e menor que 4% nos demais problemas testados. O crescimento no número de depósitos, fábricas e revendedores não afetou a estabilidade do método, uma vez que a qualidade das soluções manteve-se estável, conforme as instâncias foram crescendo.

Tabela 7.2: Comparação dos resultados obtidos pelo Bilevel ACO+DE com os obtidos pelo BACS.

		BACS [Calvete et al., 2011]			Bilevel ACO+DE			
Prob.	R, D, F	F^{bs}	$\overline{F}$	$\%F$	F^{bs}	$\overline{F}$	F^{ws}	$\%F$
BiPR01	48, 4, 4	1439.97	1475.39	2.46	1337.67	1360.08	1377.19	1.67
BiPR02	96, 4, 4	2236.25	2282.11	2.05	2291.71	2342.58	2389.33	2.22
BiPR03	144, 4, 4	3336.38	3479.40	4.29	3305.71	3408.11	3456.66	3.10
BiPR04	192, 4, 4	4750.10	4836.10	1.81	4295.86	4404.46	4517.50	2.53
BiPR05	240, 4, 4	5178.45	5371.51	3.73	5316.79	5367.00	5456.38	0.94
BiPR06	288, 4, 4	6037.67	6147.06	1.81	6206.87	6349.11	6447.15	2.29
BiPR07	72, 6, 6	1815.77	1883.48	3.73	1844.02	1948.87	2017.48	5.69
BiPR08	144, 6, 6	3449.53	3558.58	3.16	3417.32	3554.87	3663.03	4.02
BiPR09	216, 6, 6	4686.25	4798.19	2.39	4929.42	4986.80	5029.99	1.16
BiPR10	288, 6, 6	7338.37	7459.10	1.65	6680.31	6790.70	6911.25	1.65

Nota-se também a capacidade de escalabilidade do método proposto quando comparado com o BACS. A Figura 7.2 apresenta a média do tempo de execução (em minutos) dos dois algoritmos para as diferentes instâncias do PPPD.



(a) Instâncias com 4 fábricas e 4 depósitos. (b) Instâncias com 6 fábricas e 6 depósitos.

Figura 7.2: Média do tempo de execução dos algoritmos BACS e Bilevel ACO+DE nos problemas BiPR01-BiPR10.

Conforme a instância do problema cresce, em número de revendedores, o tempo de execução do Bilevel ACO+DE tem um crescimento mais suave do que o BACS, indicando melhor escalabilidade. Em média, nas instâncias com 4 depósitos e 4 fábricas, a diferença do tempo de execução de uma instância maior com relação à uma instância menor é de aproximadamente 35% no BACS e de 16% no Bilevel ACO+DE. Já para as instâncias com 6 depósitos e 6 fábricas, tem-se para o mesmo indicador 44% no BACS e 17% no Bilevel ACO+DE aproximadamente.

– Conclusões –

A partir destes experimentos, foi possível observar que o método proposto é capaz de gerar boas soluções para os problemas testados, além de ter potencial para ser aplicado em instâncias maiores do PPPD.

Antes da realização destes experimentos, testes preliminares foram realizados com o objetivo de analisar o comportamento do método. Nestes testes, observou-se um elevado tempo computacional quando o DE foi executado para cada solução do nível superior. Por esta razão, no método Bilevel ACO+DE apresentado na seção 7.4, optou-se por executar o DE apenas para a melhor solução obtida para o PRVMD, ou seja, optou-se por executar o DE uma vez à cada iteração para uma única solução candidata do líder.

Como a metodologia proposta trabalha com um conjunto de soluções no nível superior, é esperado que para cada uma delas uma solução do nível inferior possa ser associada. Desta forma, na tentativa de viabilizar a execução do DE para cada solução candidata do nível superior, um novo método foi desenvolvido com algumas modificações com relação ao seu antecessor.

7.5 Aperfeiçoamento do Bilevel ACO+DE (Bilevel ACO+DE$^+$)

Esta seção apresenta o algoritmo Bilevel ACO+DE$^+$, onde utilizou-se um critério de parada mais eficiente no nível inferior, a fim de viabilizar a execução do DE para cada solução do nível superior. No nível superior o mesmo algoritmo ACO é utilizado, enquanto que no nível inferior o algoritmo de DE com variante DE/target-to-rand/1, sem operador de recombinação, é aplicado. O pseudo-código do método é apresentado pelos algoritmos 10 e 9 onde, neste último, apenas o critério de parada foi modificado. A descrição do algoritmo é dada por:

Passo 0: Iniciar dados. O algoritmo inicia efetuando a leitura da instância do PPPD e em seguida o valor dos parâmetros utilizados pelos algoritmos ACO e DE são iniciados, assim como a matriz de feromônio e de informação heurística.

Passo 1: Gerar solução inicial. Através da execução de uma heurística gulosa [Hoos and Stutzle, 2004], uma solução inicial é gerada para o PRVMD. O vetor de demanda $D(x_{ini})$ associado à esta solução é enviado para o método DE, responsável pela resolução do PT associado.

Passo 2: Iniciar matriz de feromônio. Após a construção da solução inicial, as trilhas de feromônio τ_{ij} são inicializadas com o valor τ_0, associado ao valor da função objetivo do líder da solução inicial (F_{ini}).

Passo 3: Construir soluções para o problema de dois níveis. A cada iteração, A_{max} soluções são construídas para o problema em dois níveis, onde o ACO é utilizado para resolver o problema líder e o DE para resolver o problema seguidor.

Passo 3.1: Resolução do problema líder. Cada formiga a_h resolve o PRVMD seguindo a regra de decisão do algoritmo ACS (equações 4.1 e 4.2). Nesta etapa, cada formiga artificial constrói uma rota completa para o problema. Após a construção da solução, o procedimento de busca local 2-opt [Hoos and Stutzle, 2004] é aplicado a fim de melhorar a qualidade das soluções obtidas. Em seguida é realizado o procedimento de atualização local de feromônio (equação 4.3) nos "caminhos" percorridos pelas formigas.

Passo 3.2: Resolução do problema seguidor. Cada solução a_h do nível superior possui um vetor de demandas $D(x_h)$ que será enviado para o problema seguidor. Assim, o DE com variante DE/target-to-rand/1, sem operador de recombinação, é executado para a resolução do PT associado. Após o critério de parada do nível inferior ser satisfeito, o algoritmo retorna para o líder a melhor solução obtida.

Passo 4: Aplicar atualização global de feromônio. A atualização global do feromônio (equação 4.4) é realizada apenas na rota construída pela melhor formiga até então obtida (a_{bf}), de acordo com o valor da função objetivo do nível superior.

Passo 5: Finalização. O algoritmo termina após it_{max} iterações, quando então a melhor solução obtida para o problema líder é retornada.

Algoritmo 10: Bilevel ACO+DE

Entrada : it_{max} (número de iterações), A_{max} (número de formigas), α (influência do feromônio), β (influência da informação heurística), γ (taxa de evaporação local) e ρ (taxa de evaporação global).

// Iniciar dados
1 Ler instâncias do problema em dois níveis;
2 Alocar matriz de feromônio e de informação heurística;

// Gerar solução inicial
3 Resolver o PRVMD, utilizando uma heurística gulosa, para obter a solução a_{ini};
4 $y(x_{ini})$ = DE-Seg. (npf, genf, F, x_{ini}); `//` x_{ini}: `vetor de demandas associado à` a_{ini}
5 F_{ini} = `AvaliarSoluçãoLider(`$a_{ini}, y(x_{ini})$`);`

// Iniciar matriz de feromônio
6 Calcular valor inicial do feromônio (τ_0), de acordo com F_{ini};
7 Iniciar trilhas de feromônio com o valor τ_0;

// Construir soluções para o problema de dois níveis
8 **for** $i \leftarrow 1$ **to** it_{max} **do**
9 **for** $h \leftarrow 1$ **to** A_{max} **do**
  // Resolução do problema líder
10 Resolver o PRVMD, utilizando ACO, para obter a solução a_h;
11 Aplicar procedimento de busca local em a_h;
12 Atualização local das trilhas de feromônio com relação à a_h;
  // Resolução do problema seguidor
13 $y(x_h)$ = DE-Seg.(npf,genf,F,x_h); `//` x_h: `vetor de demandas associado à` a_h
14 F = `AvaliarSoluçãoLider(`$a_h, y(x_h)$`);`
  // Aplicar atualização global de feromônio
15 Selecione a melhor formiga (a_{bf}), de acordo com a função objetivo do líder (F);
16 Aplicar atualização global de feromônio, associada à solução a_{bf};

Retornar: `MelhorSoluçãoLíder`

7.5.1 Critério de parada

Neste método, o critério de parada do nível superior e inferior foram configurados de maneira diferente. Para o nível superior, o algoritmo ACO termina quando um número máximo de iterações t_{max} é alcançado. Para o nível inferior, duas situações de parada são estabelecidas: o algoritmo termina após gen_{max} iterações; ou até que nenhuma solução melhor seja encontrada após um certo número de gerações *gen*. O parâmetro *gen* varia de acordo com o número de iterações do algoritmo do nível superior, onde o valor *gen* aumenta conforme o número de iterações do nível superior aumenta.

Este procedimento tem como propósito fazer com que o número de gerações do método do nível inferior varie ao longo do processo iterativo do nível superior. Assim, no início da execução um número reduzido de gerações é aplicado, na

tentativa de reduzir, no início do processo, o número de avaliações de função do nível inferior, mesmo que isso possa comprometer a qualidade das soluções. Por outro lado, ao final da execução, e ao longo dela, um número maior de gerações no nível inferior é realizado, na tentativa de garantir a obtenção de soluções de melhor qualidade ao final do processo.

Na primeira iteração do algoritmo ACO, os parâmetros l_{max} e l_{min} são fixados, representando o limite máximo e mínimo de gerações, respectivamente, em que o algoritmo DE continua a execução até que nenhuma solução melhor seja encontrada. Em seguida, definiu-se um intervalo $\mu = (l_{max} - l_{min})/it_{max}$. A cada iteração do ACO, uma variável auxiliar é calculada como: $aux^{(t+1)} = aux^{(t)} + \mu$, com $aux^{(0)} = 0$. Assim, o valor gen é dado por

$$gen = l_{min} + \lfloor aux \rfloor \tag{7.4}$$

Este procedimento faz com que o valor de gen varie de l_{min} a l_{max} conforme o algoritmo ACO "avança", i.e., conforme o crescimento do número de iterações.

7.5.2 Experimentos computacionais

Os experimentos computacionais realizados tiveram como principal objetivo analisar as implicações do uso de um método aproximado no nível inferior –neste caso o DE– investigando como soluções aproximadas neste nível podem afetar o processo de busca do método do nível superior –neste caso o ACO– e a solução final do problema de PDN. O estudo realizado considerou diferentes aspectos da metodologia proposta:

- se o DE é capaz de gerar soluções ótimas e quantas vezes isso ocorre;
- quando o DE não for capaz de gerar a solução ótima, o quão distante a solução gerada está da ótima;
- número de avaliações de função realizadas do nível inferior;

- como as soluções do DE afetam o valor das soluções do nível superior; e
- de que forma a solução final é afetada quando o DE é utilizado para resolver o nível inferior.

Para verificar se o DE é capaz de gerar soluções ótimas na resolução do problema de transporte, comparou-se os resultados obtidos pelo DE com o método exato conhecido como Stepping Stone (SSM). A descrição do método SSM, pode ser encontrada na página `http://orms.pef.czu.cz/text/transProblem.html`. O método Northwest Corner, também disponível nesta página, foi utilizado para gerar a solução inicial do SSM.

O algoritmo Bilevel ACO+DE$^+$ foi testado em 10 instâncias do PPPD, onde 10 execuções independentes foram realizadas para cada uma destas instâncias. Estes experimentos foram realizados no Cluster Sun HPC do Laboratório Nacional de Computação Científica, com a seguinte especificação: modelo Sun Blade x6250, com 2 Processadores Intel Xeon E5440 Quad Core e 16 GB memória PC2-5300 DDR2.

– **Definição do valor dos parâmetros** –

Experimentos preliminares foram efetuados para encontrar valores razoáveis para a configuração dos parâmetros. O valor de cada parâmetro utilizado foi configurado da seguinte forma:

- Para o algoritmo do nível superior (ACO):
  - α e β: a importância relativa das trilhas de feromônio e da informação heurística são 1 e 2, respectivamente.
  - ρ e φ: as taxas de evaporação global e local do feromônio são ambas definidas com 0.1.
  - q_0: o parâmetro que regula entre a exploração e a intensificação da busca é definido com 0.9.

- τ_0: o valor inicial da matriz de feromônio é dado por $1/R.F_{inicial}$.

- A_{max}: o número total de formigas é igual a 40.

- it_{max}: o número máximo de iterações é igual a 4000.

• Para o algoritmo do nível inferior (DE):

- npf: o número total de indivíduos na população é igual a 40.

- F: o fator de escala que controla a magnitude da perturbação nos indivíduos é igual a 1.

- $genf$: o número máximo de gerações é igual 1000.

- gen, l_{max} e l_{min}: o valor de gen indica o número de gerações que o DE continua a execução até que uma melhor solução seja obtida. Este parâmetro varia de $l_{min} = 10$ à $l_{max} = 100$ conforme o número de iterações do algoritmo ACO, no nível superior, cresce. O valor de gen é calculado conforme a equação (7.4).

Testes preliminares indicaram que a variante DE/target-to-rand/1, sem operador de recombinação, gerou melhores resultados na resolução do problema de transporte quando comparado com as demais variantes do DE descritas na seção 4.2.2. Por esta razão, esta variante do DE foi utilizada no nível inferior para estes experimentos.

– Resultados e discussões –

As Figuras 7.3-7.12 exibem a representação da melhor solução obtida em cada iteração do nível superior em uma dada execução do algoritmo Bilevel ACO+DE$^+$. Estes gráficos apresentam informações sobre como as soluções obtidas pelo DE no nível inferior afetam as soluções geradas pelo ACO no nível superior. Os gráficos na parte inferior das figuras referem-se ao problema seguidor, e apresentam a diferença em porcentagem entre as soluções obtidas pelo DE versus a solução ótima obtida pelo SSM, indicando a "distância" entre a solução aproximada e a exata. Os gráficos

na parte superior das figuras apresentam informações sobre como o problema líder foi afetado pelas soluções geradas pelo DE no nível inferior. Quando o DE não é capaz de obter a solução ótima, o problema líder, que é de minimização, pode ser afetado de duas maneiras distintas: ele pode ser afetado de forma *positiva*, onde o valor da função objetivo diminui, melhorando a qualidade da solução –indicado pelos valores negativos no gráfico–; ou ele pode ser afetado de forma *negativa*, quando o valor da função objetivo aumenta, piorando a qualidade da solução –indicado pelos valores positivos no gráfico–. Obviamente que, quando o DE encontra a solução ótima, a função do líder não é afetada, indicado pela ausência da barra vertical nos gráficos.

A partir dos resultados apresentados nas Figuras 7.3-7.12 é possível observar que para os problemas BiPR01 à BiPR06, o DE foi capaz de obter soluções ótimas em boa parte das iterações. No entanto, para os problemas BiPR07 à BiPR10 o DE apresentou dificuldades para encontrar soluções ótimas, provavelmente devido ao aumento da complexidade do nível inferior nestes problemas –aumento no número de fábricas e depósitos–. Estas observações podem ser verificadas pelos valores apresentados na Tabela 7.3, que fornece a porcentagem de soluções ótimas obtidas pelo DE nas 10 execuções do algoritmo Bilevel ACO+DE$^+$ em cada um dos problemas.

Tabela 7.3: Porcentagem de soluções ótimas encontradas pelo DE em 10 execuções.

Problema	Melhor	Média	Mediana	Pior
BiPR01	96.50	95.35	95.30	94.55
BiPR02	97.08	96.03	95.95	95.13
BiPR03	82.75	81.40	81.31	80.35
BiPR04	99.98	99.88	99.86	99.70
BiPR05	83.03	81.39	81.45	80.08
BiPR06	60.05	58.93	58.94	57.88
BiPR07	11.45	8.53	7.96	6.55
BiPR08	3.10	2.52	2.56	1.73
BiPR09	1.93	1.59	1.55	1.28
BiPR10	2.10	1.79	1.78	1.53

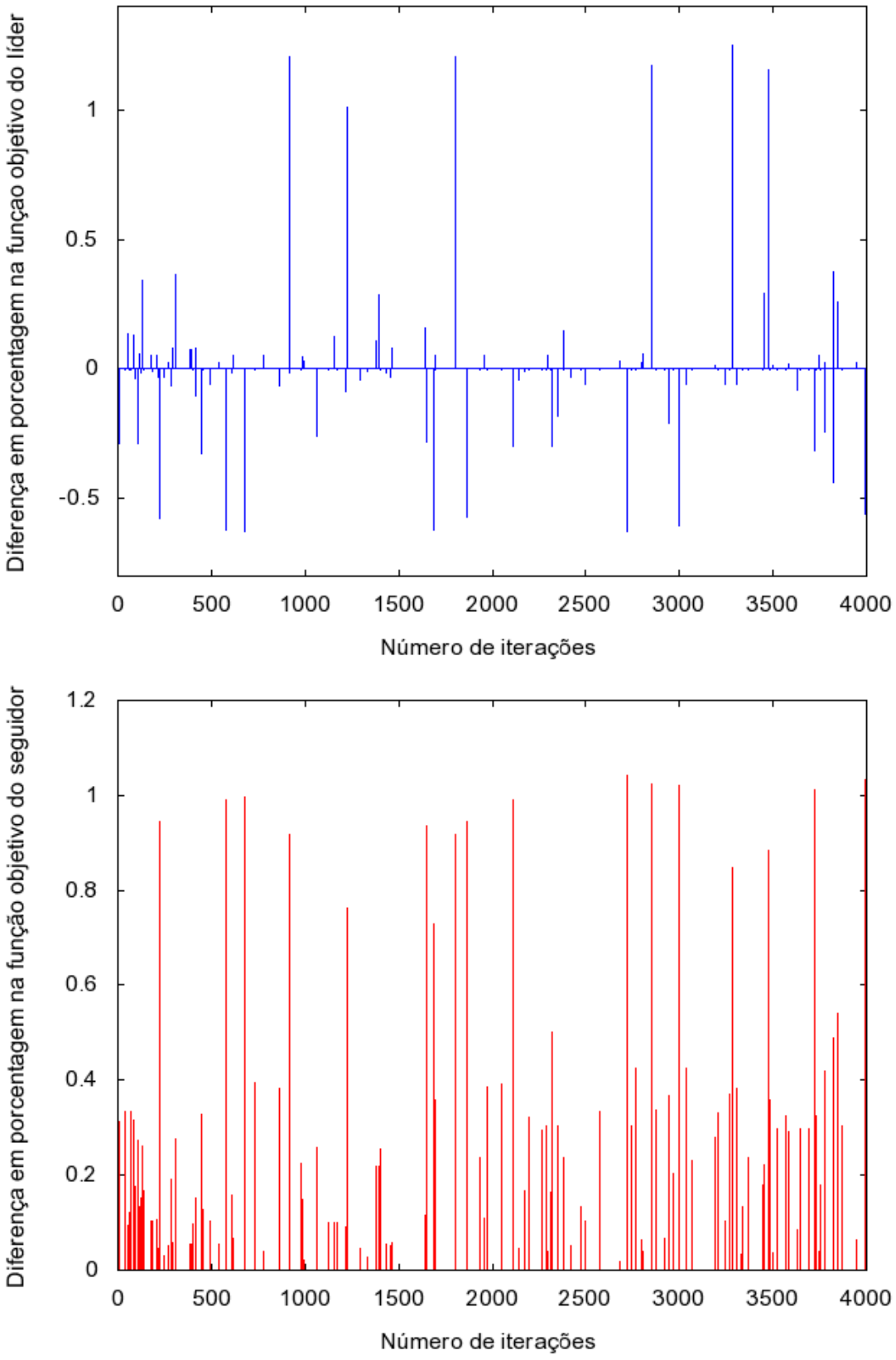


Figura 7.3: Resultados de uma execução da instância BiPR01.

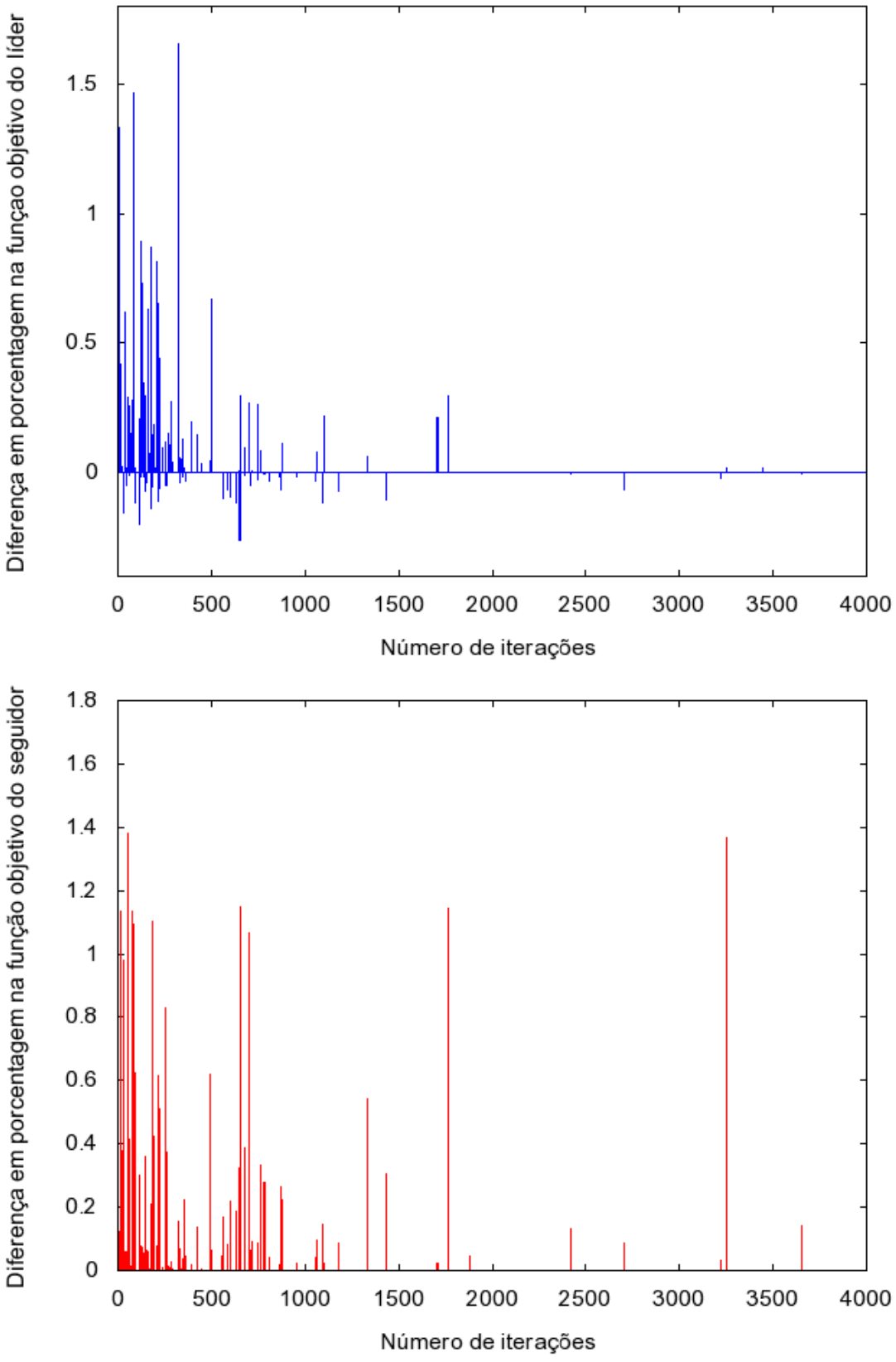


Figura 7.4: Resultados de uma execução da instância BiPR02.

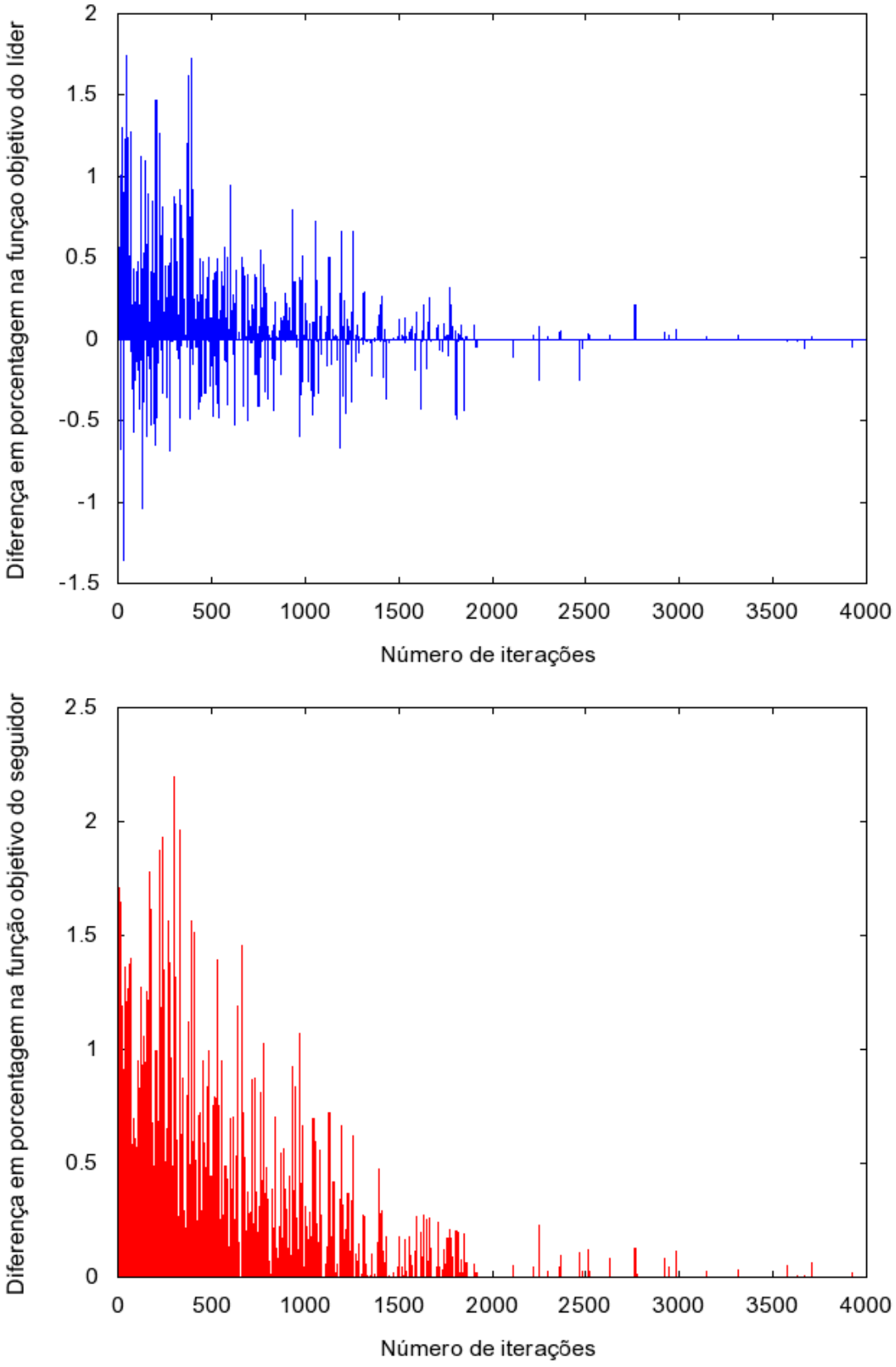


Figura 7.5: Resultados de uma execução da instância BiPR03.

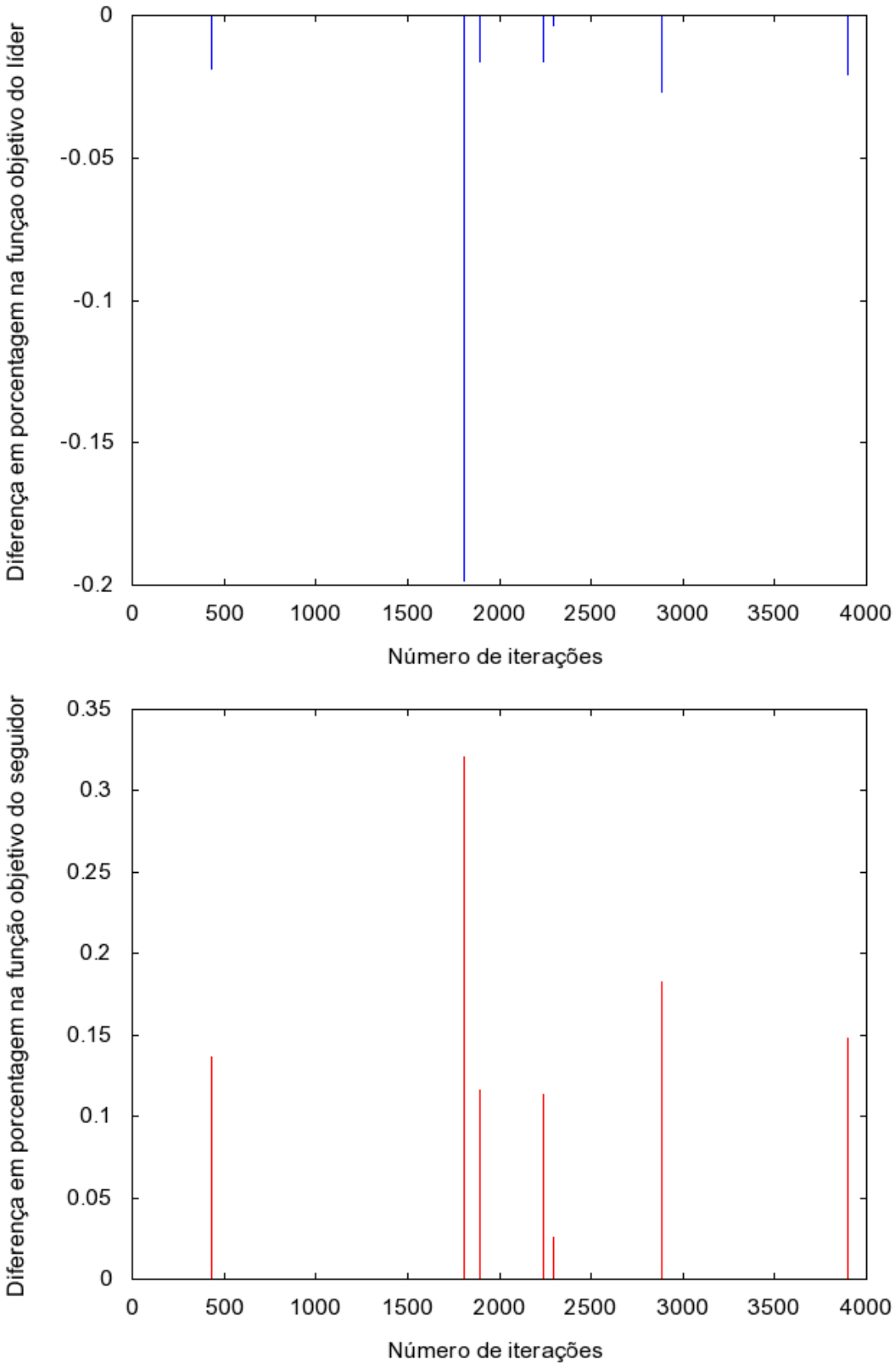


Figura 7.6: Resultados de uma execução da instância BiPR04.

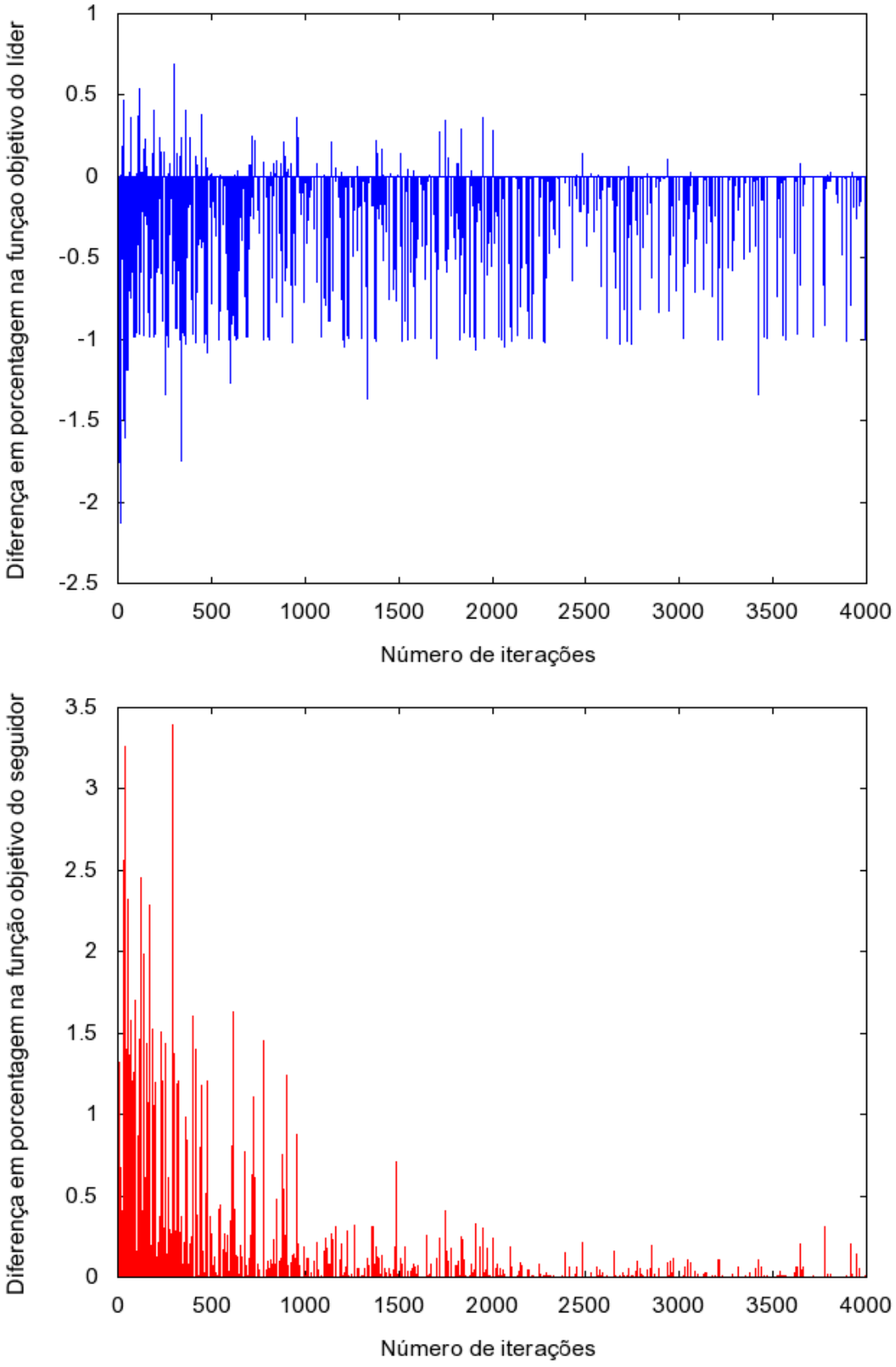


Figura 7.7: Resultados de uma execução da instância BiPR05.

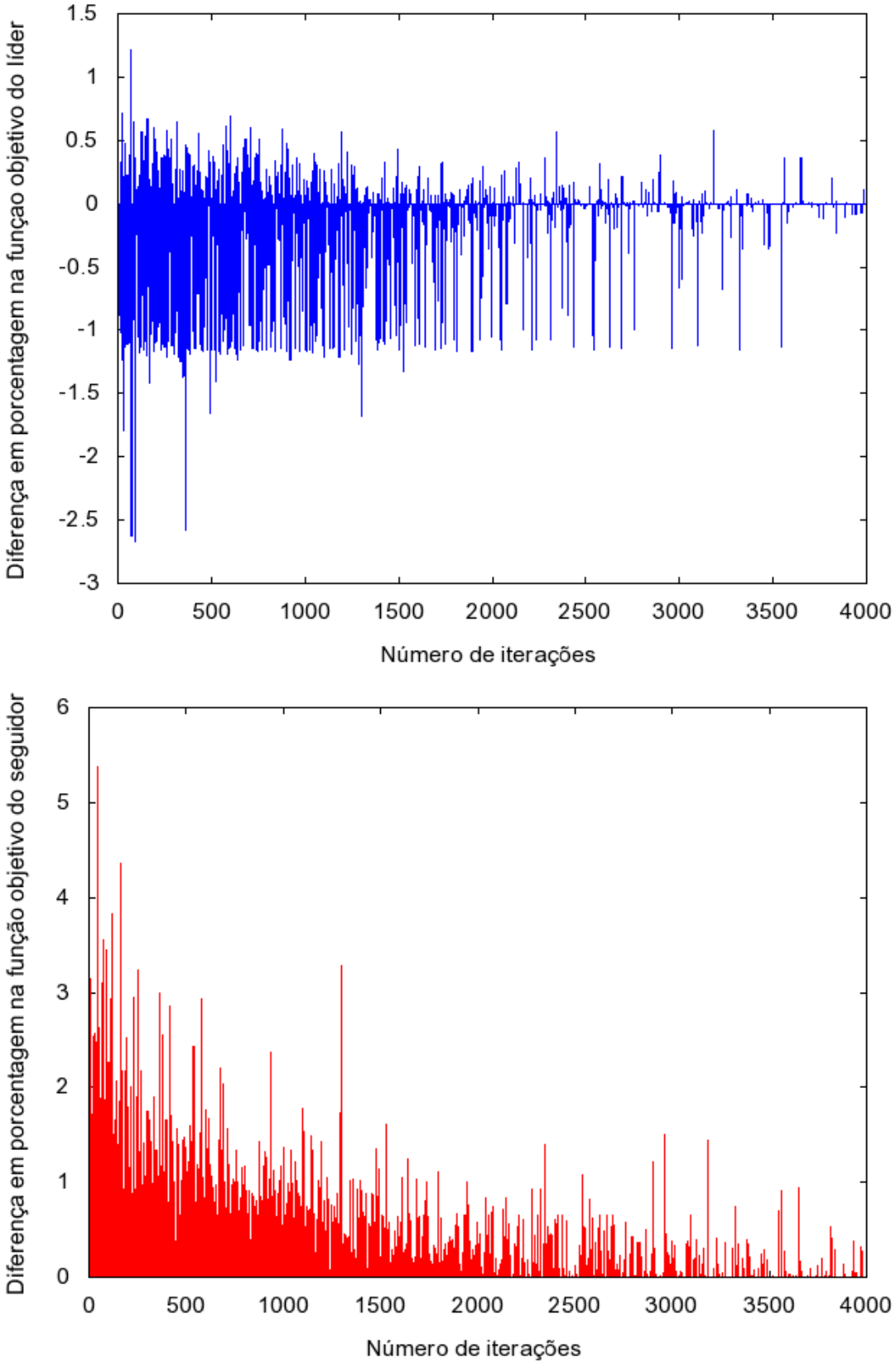


Figura 7.8: Resultados de uma execução da instância BiPR06.

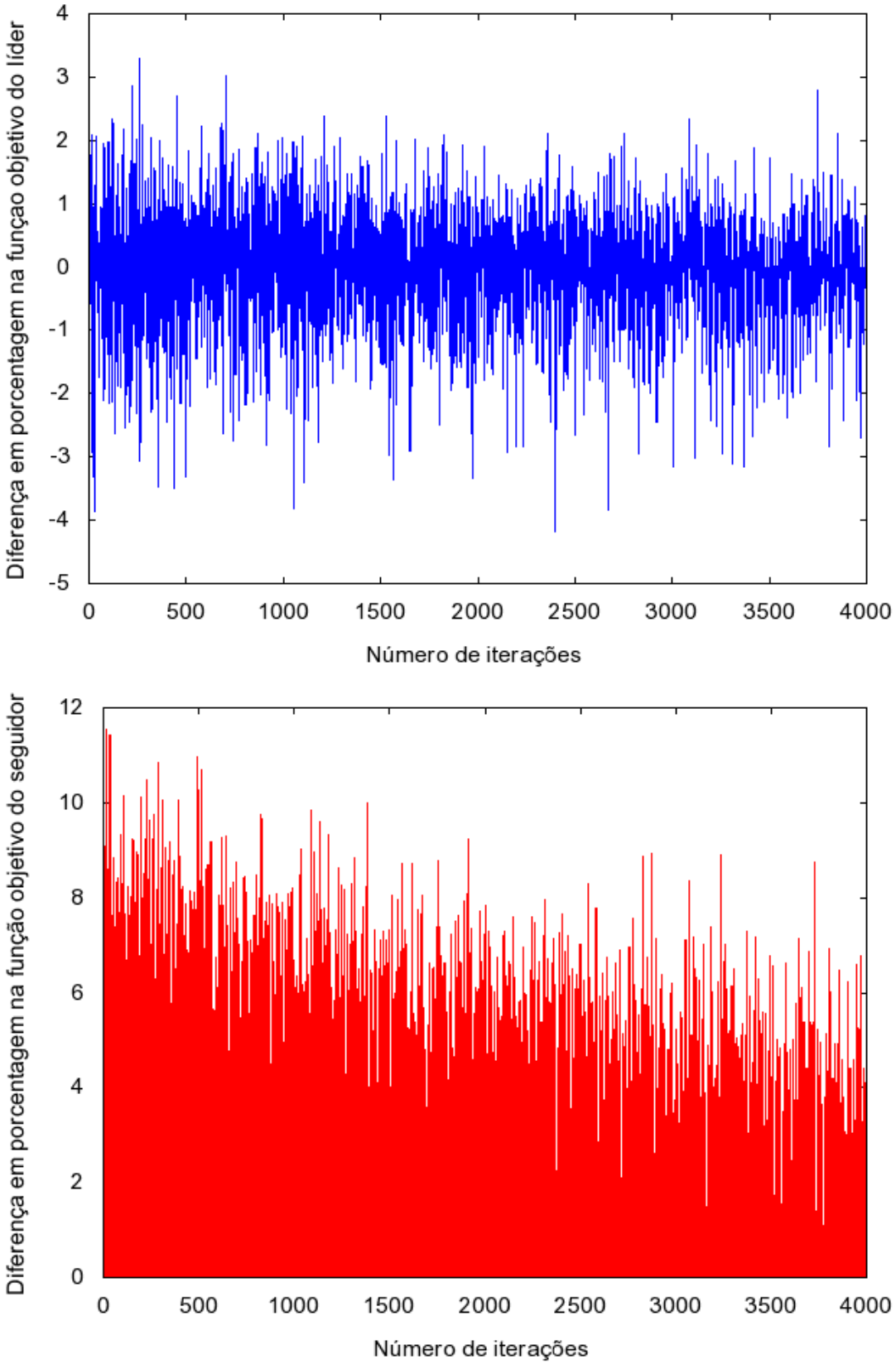


Figura 7.9: Resultados de uma execução da instância BiPR07.

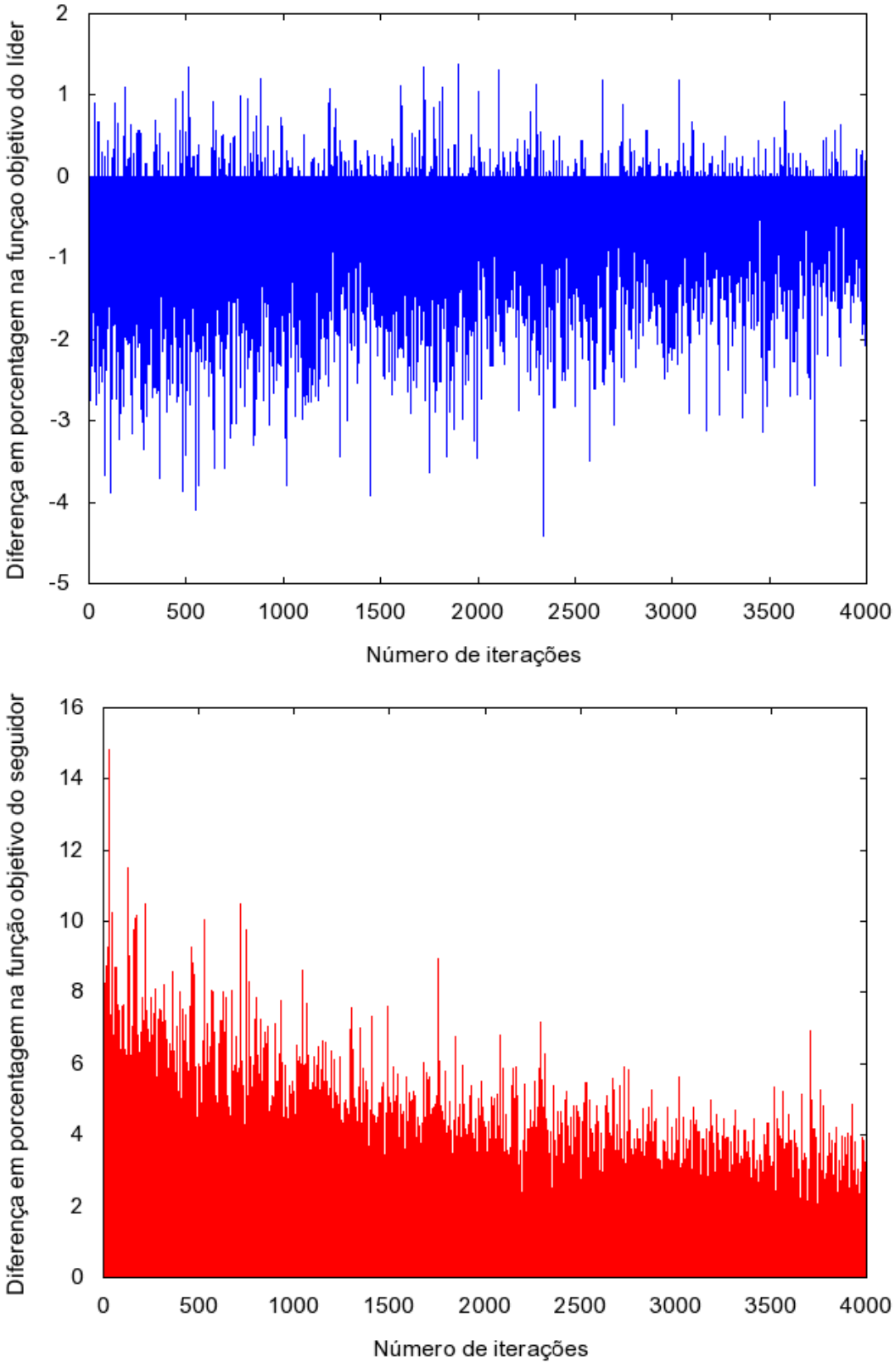


Figura 7.10: Resultados de uma execução da instância BiPR08.

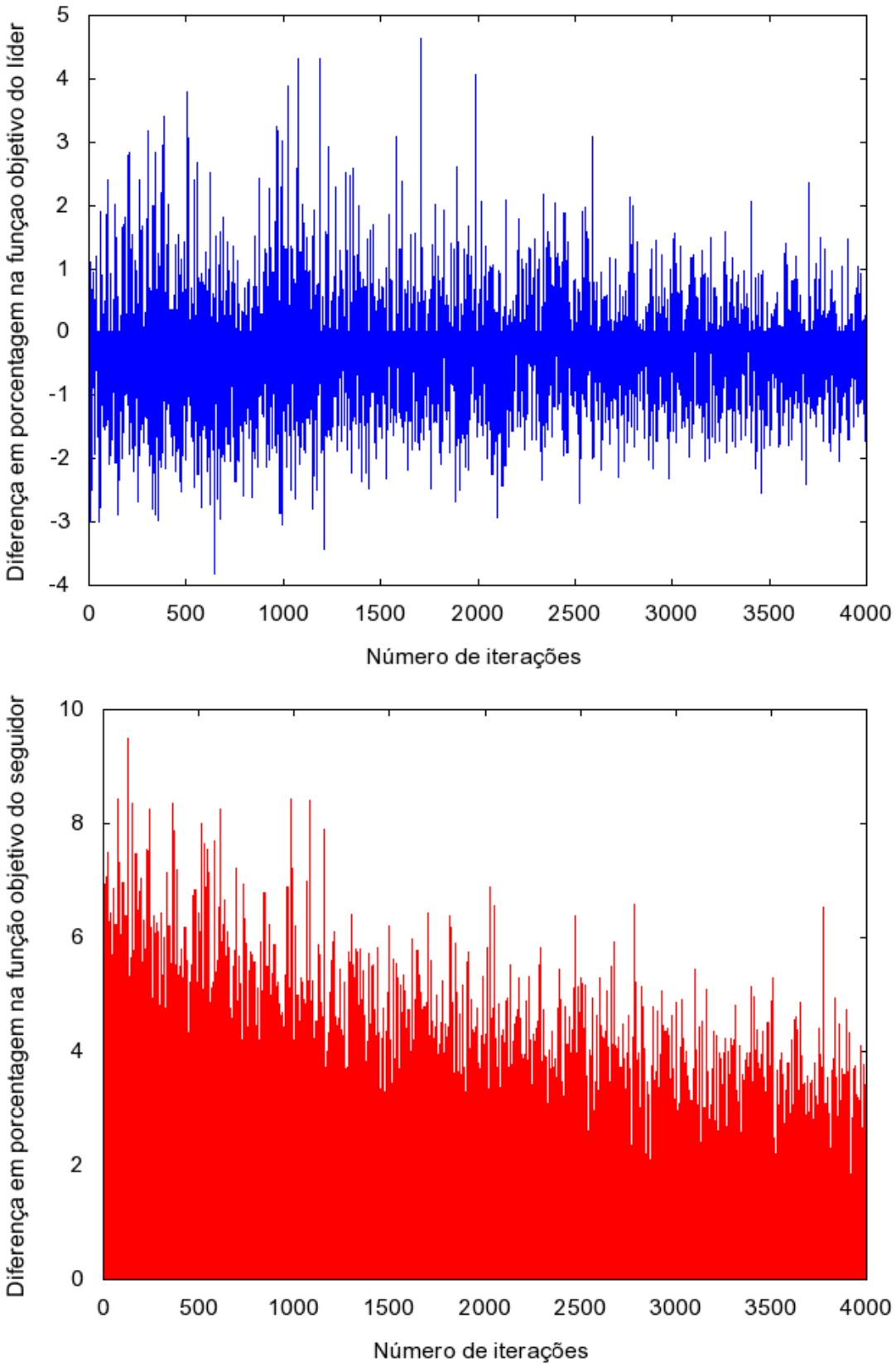


Figura 7.11: Resultados de uma execução da instância BiPR09.

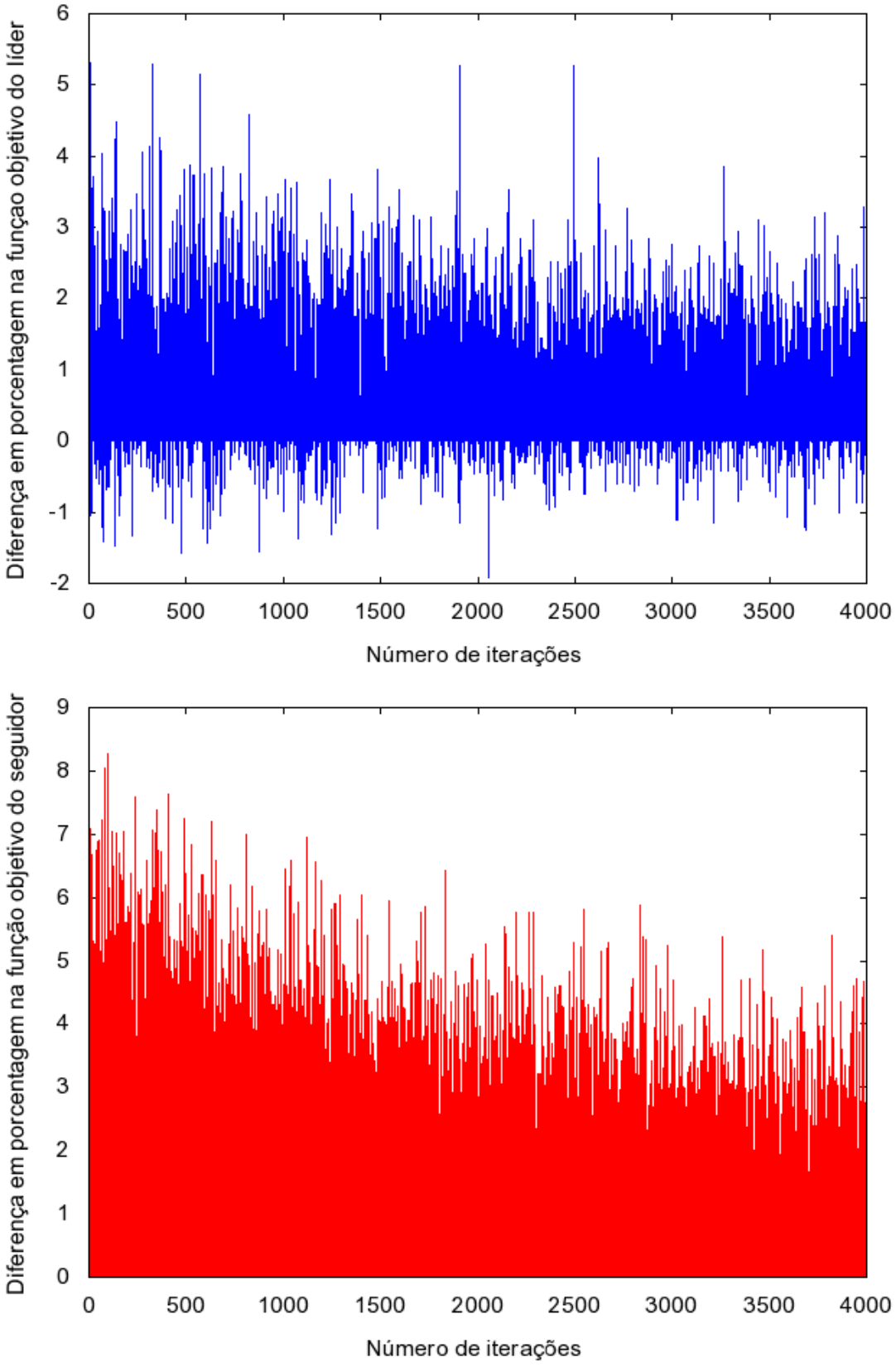


Figura 7.12: Resultados de uma execução da instância BiPR10.

Em todos os problemas analisados, com exceção do BiPR01 e BiPR04, é possível observar que conforme o número de iterações cresce, o DE vem de forma crescente gerando soluções cada vez melhores, próximas das soluções ótimas, conforme indicado pelos gráficos na parte inferior das Figuras 7.3-7.12. Por exemplo, nas primeiras iterações do problema BiPR10, as soluções obtidas pelo DE têm uma diferença percentual com relação à solução ótima de aproximadamente 8%. Ao longo da execução este percentual vai diminuindo alcançando uma diferença percentual em torno de 5% no final da execução. Este comportamento já era esperado, devido ao critério de parada implementado.

Conforme apresentado na Tabela 7.3, o DE obteve mais de 95% de soluções ótimas do nível inferior nos problemas BiPR01, BiPR02 e BiPR04; mais de 80% para os problemas BiPR03 e BiPR05; em torno de 60% para o problema BiPR06; e abaixo de 12% nos demais problemas.

A média do número de avaliações de função do nível inferior (AF.NI.) é apresentada na Figura 7.13, e a média do tempo computacional[2] em 10 execuções do Bilevel ACO+DE$^+$ é apresentada na Figura 7.14.

A Tabela 7.4 apresenta o melhor e o pior valor obtido pelo Bilevel ACO+DE$^+$ em 10 execuções para cada problema, assim como a diferença em porcentagem com relação ao nível inferior (NI) e superior (NS) da respectiva solução. Nesta tabela, a coluna "% no NI" indica a diferença percentual entre a solução obtida pelo DE e a solução obtida pelo SSM no nível inferior, i.e., $(f_{DE} - f_{SSM}/f_{SSM}) \times 100$. A coluna "% no NS" indica a diferença percentual da função objetivo do nível superior quando utiliza-se o DE no nível inferior, em relação a quando utiliza-se o SSM no nível inferior, i.e., $(F_{c/DE} - F_{c/SSM}/F_{c/SSM}) \times 100$. Desta forma, o valor percentual nulo indica que o DE obteve a solução ótima do problema seguidor –igual à gerada pelo SSM–.

[2] O tempo total de execução considerou a execução do SSM para cada solução do líder a cada iteração, para que fosse possível, ao longo da execução, comparar com os resultados obtidos pelo DE.

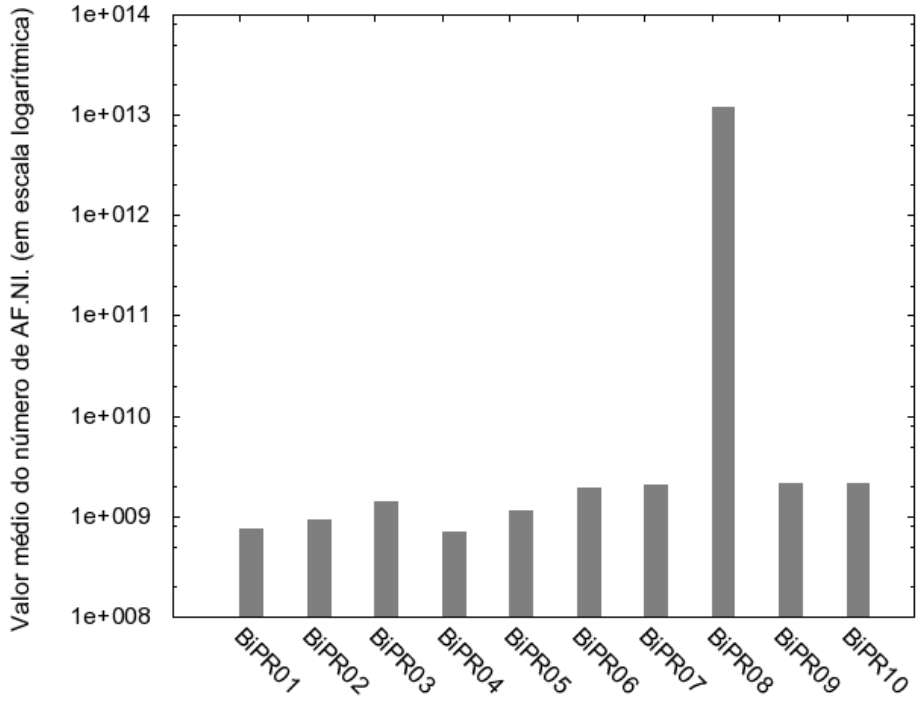


Figura 7.13: Média do número de avaliações de função (AF) do nível inferior (NI) em 10 execuções do método Bilevel ACO+DE$^+$.

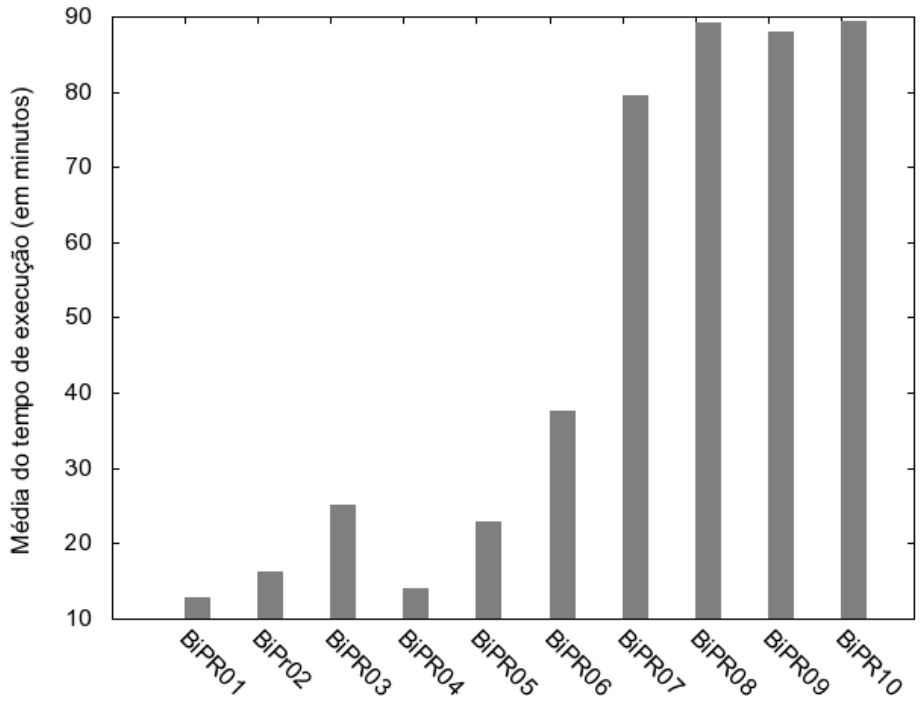


Figura 7.14: Média do tempo computacional em 10 execuções requerido pelo Bilevel ACO+DE$^+$ para a resolução dos PPPD.

Tabela 7.4: Melhor e pior valor de solução encontrados pelo Bilevel ACO+DE$^+$ em 10 execuções e a diferença em porcentagem com relação ao nível inferior (NI) e superior (NS).

Prob.	Melhor solução	% no NI	% no NS	Pior solução	% no NI	% no NS
BiPR01	1321.319	0.36	-0.01	1405.311	0.18	-0.19
BiPR02	2352.270	0	0	2440.190	0	0
BiPR03	3214.667	0	0	3428.235	0.20	-0.44
BiPR04	4325.061	0	0	4546.747	0	0
BiPR05	5209.667	0	0	5485.823	0	0
BiPR06	6452.972	0	0	6664.004	0.66	-1.17
BiPR07	1817.649	5.95	-3.88	1975.490	7.37	-3.20
BiPR08	3429.619	3.41	-1.32	3656.493	3.45	-2.18
BiPR09	4757.578	4.64	-2.03	5010.308	2.51	-1.26
BiPR10	6869.727	1.41	-0.03	7038.966	3.00	0.63

De acordo com a Tabela 7.4, para os problemas BiPR02 a BiPR06, o método DE foi capaz de obter a solução ótima no nível inferior para a melhor solução do nível superior. Já para os problemas BiPR01 e BiPR07 a BiPR10, o DE não obteve a solução ótima no nível inferior para a melhor solução gerada no nível superior. Como consequência, para esses casos onde não obteve-se o valor ótimo no nível inferior, tem-se que a solução final gerada é teoricamente inviável, uma vez que, por definição, o nível inferior precisa responder de maneira ótima. Seguindo o mesmo raciocínio, nas piores soluções encontradas pelo Bilevel ACO+DE$^+$, tem-se que apenas as soluções geradas para os problemas BiPR02, BiPR04 e BiPR05 são viáveis pela definição.

Analisando mais detalhadamente os resultados do problema BiPR10, apresentados na Tabela 7.4, observa-se que, para a melhor solução gerada, o DE não obteve a solução ótima do nível inferior e ficou 1.41% "distante" desta. Como consequência, o nível superior foi afetado, de forma positiva, onde houve uma melhora de 0.03% na função objetivo do líder. Em contrapartida, na pior solução gerada, o DE também não obteve a solução ótima do nível inferior, ficando 3% "distante" desta, e no entanto houve uma piora no valor da solução do líder de aproximadamente 0.6%.

A partir desta análise, é possível observar em todos os problemas que nas soluções finais geradas, mesmo não obtendo-se o ótimo no nível inferior, o nível

superior não foi significativamente afetado, pois gerou-se uma melhora enganosa na função objetivo do nível superior de menos de 4% em todos os problemas analisados. Deste modo, quando o método do nível inferior não for capaz de gerar soluções ótimas, é desejável que a solução do nível superior seja o menos possível afetada, para não induzir a busca do nível superior em direção a soluções enganosas.

Na última análise, realizou-se uma comparação entre o Bilevel ACO+DE$^+$ e este mesmo método utilizando agora o SSM para a resolução do nível inferior, chamado aqui de Bilevel ACO+SSM. Assim, no Bilevel ACO+SSM, utilizou-se o mesmo algoritmo ACO para o nível superior, e o algoritmo SSM para a resolução do nível inferior. A Tabela 7.5 apresenta uma comparação entre os resultados obtidos por estes dois métodos. Os valores em negrito indicam a melhor solução obtida entre os métodos, e os valores com um asterístico "*" no Bilevel ACO+DE$^+$, indicam que a solução obtida possui o valor ótimo no nível inferior.

Tabela 7.5: Melhor, mediana e pior valor de solução obtido em 10 execuções pelo Bilevel ACO+SSM e o Bilevel ACO+DE$^+$.

	Bilevel ACO+SSM			Bilevel ACO+DE$^+$		
Prob.	Melhor	Mediana	Pior	Melhor	Mediana	Pior
BiPR01	1322.165	1360.077	1375.794	**1321.319**	1348.076	1405.311
BiPR02	**2287.229**	2354.739	2454.139	2352.270	2382.723	2440.190
BiPR03	**3188.317**	3295.261	3394.899	3214.667	3307.264	3428.235
BiPR04	4410.933	4497.421	4575.884	**4325.061***	4483.829	4546.747
BiPR05	5259.594	5373.023	5398.114	**5209.667***	5360.079	5485.823
BiPR06	**6386.031**	6561.669	6708.208	6452.972	6543.379	6664.004
BiPR07	1862.238	1915.022	1977.011	**1817.649**	1903.399	1975.490
BiPR08	3431.155	3622.574	3681.858	**3429.619**	3609.838	3656.493
BiPR09	4826.094	4961.650	5010.948	**4757.578**	4916.606	5010.308
BiPR10	**6675.745**	6862.310	7026.912	6869.727	6959.326	7038.966

*Possui o valor ótimo no nível inferior.

Pela Tabela 7.5 observa-se que o Bilevel ACO+DE$^+$ obteve melhores resultados que o Bilevel ACO+SSM nos problemas BiPR01, BiPR04, BiPR05, BiPR07, BiPR08 e BiPR09. Dentre estes, apenas as soluções obtidas para os problemas BiPR04 e BiPR05 contêm a solução ótima do nível inferior. Podemos destacar aqui que, mesmo as soluções do DE não sendo garantidamente ótimas, o Bilevel ACO+DE$^+$ encontrou melhores soluções em dois problemas (BiPR04 e

BiPR05) com o valor ótimo no nível inferior.

Assim, para garantir a obtenção de soluções de qualidade, é necessário que o algoritmo do nível inferior seja executado por tempo suficiente de modo a obter soluções ótimas, ou próximas destas, para não induzir a busca do nível superior em direção a soluções enganosas. Em contrapartida, na tentativa de se obter soluções de alta qualidade, faz-se necessário um grande número de avaliações de função no nível inferior, como mostrou a Figura 7.13, o que pode comprometer o custo computacional. No entanto, se o algoritmo aproximado utilizado no nível inferior é executado por tempo insuficiente, na tentativa de diminuir o custo computacional, pode-se gerar soluções muito distantes da solução ótima, o que pode comprometer a direção de busca do otimizador do nível superior.

– Conclusões –

Destes experimentos conclui-se que se um método aproximado for utilizado no nível inferior é recomendável que este algoritmo seja executado por tempo suficiente de modo a garantir a obtenção de soluções viáveis ou "quase" viáveis, para não influenciar a busca do nível superior em direção a soluções enganosas. No entanto, isso pode aumentar substancialmente o esforço computacional do nível inferior. Neste caso, é recomendável definir cuidadosamente o critério de parada do nível inferior na tentativa de alcançar uma relação de compromisso eficiente entre soluções de qualidade e tempo computacional aceitável.

É importante mencionar também que em testes preliminares o Bilevel ACO+DE$^+$ foi capaz de obter os mesmo resultados utilizando menos iterações do que as que foram reportadas anteriormente nestes testes. De fato, para alguns problemas, com 2000 ou 3000 iterações no nível superior, o algoritmo obteve os mesmos resultados finais.

7.6 Considerações finais

Neste capítulo, dois algoritmos foram desenvolvidos para a resolução do problema de planejamento de produção e distribuição, onde instâncias variando de 48 a 288 revendedores com 4 e 6 depósitos e fábricas foram utilizadas para a análise dos algoritmos propostos.

A primeira parte dos experimentos teve como objetivo analisar o desempenho do método proposto versus o algoritmo BACS proposto em [Calvete et al., 2011]. Em ambos os algoritmos o método de otimização por colônia de formigas foi utilizado para a resolução do problema líder, onde o algoritmo BACS utilizou um método exato no nível inferior e o Bilevel ACO+DE aqui proposto utilizou um método evolucionista no nível inferior, o método de evolução diferencial. A partir dos experimentos verificou-se que o Bilevel ACO+DE gerou resultados competitivos na linha dos problemas testados.

Posteriormente, uma segunda etapa de experimentos foi realizada, onde além de terem sido propostas melhorias no algoritmo Bilevel ACO+DE, novos estudos foram realizados com relação ao novo algoritmo desenvolvido, o Bilevel ACO+DE^{+}. Nesta fase, o objetivo dos estudos foi analisar as implicações do uso de um método aproximado para a resolução do problema do nível inferior, onde buscou-se investigar qual o impacto de se gerar soluções aproximadas neste nível para o processo de otimização do nível superior.

O próximo capítulo apresenta a última etapa de experimentos, onde dois novos métodos serão propostos para a resolução dos problemas em dois níveis contendo múltiplos seguidores.

Capítulo 8

Metodologias Propostas e Experimentos Computacionais para a Resolução dos Problemas da Classe 3

A terceira e última etapa dos experimentos apresenta os métodos de resolução propostos para tratar os problemas da classe 3, descritos na seção 5.3. Estes problemas são caracterizados por possuírem múltiplos seguidores no nível inferior.

Os algoritmos desenvolvidos tomaram como base o método BLDE, apresentado na seção 6.2, que foi adaptado para permitir tratar problemas com múltiplos seguidores. Assim, dois novos algoritmos foram desenvolvidos.

O primeiro método foi desenvolvido para resolver os problemas com múltiplos seguidores independentes, denominado BLDE$_{MSI}$, onde dois algoritmos DE foram utilizados para resolver de forma independente o problema de cada seguidor. O segundo método, desenvolvido para resolver os problemas com múltiplos seguidores dependentes, nomeado de BLDE$_{MSD}$, distingue-se do BLDE$_{MSI}$ na maneira como a solução dos seguidores é obtida. Este segundo método busca pela solução de equilíbrio de Nash (conforme definição apresentada na seção 2.4.3) entre os seguidores para cada variável fixada pelo líder.

8.1 BLDE para Seguidores Independentes (BLDE$_{MSI}$)

Assim como o BLDE, o BLDE$_{MSI}$ é um método aninhado que resolve os problemas do nível inferior para cada solução do nível superior. Desta forma, para cada conjunto de variáveis do nível superior, os (dois) problemas seguidores são resolvidos de forma sequencial e a melhor solução gerada para cada um deles é retornada para o processo de otimização do nível superior. O pseudo-código da metodologia proposta é apresentado pelo Algoritmo 11 (DE-Líder$_{MSI}$).

Algoritmo 11: Algoritmo DE-Líder$_{MSI}$.

Entrada : np (tamanho da população), gen (núm. de gerações), F (fator de mutação), CR (taxa de recombinação)

```
1  G ← 0;
2  CriarPopulacaoInicialAleatoria(np);
3  for i ← 1 to np do
4  |   y⃗ = DE-Seguidor(npf1, genf1, F, CR, x⃗_{i,G});
5  |   z⃗ = DE-Seguidor(npf2, genf2, F, CR, x⃗_{i,G});
6  |   Avaliar F(x⃗_{i,G}, y⃗, z⃗);          /* x⃗_{i,G} é um indivíduo da população */
7  for G ← 1 to gen do
8  |   for i ← 1 to np do
9  |   |   SelecionarAleatoriamente(r1, r2, r3);          /* r1 ≠ r2 ≠ r3 ≠ i */
10 |   |   jRand ←AleatorioInteiro(1, n);      /* n é o número de variáveis do
   |   |   problema líder */
11 |   |   for j ← 1 to n do
12 |   |   |   if Aleatorio(0,1) < CR or j = jRand then
13 |   |   |   |   u_{i,j,G+1} = x_{i,j,G} + F.(x_{r3,j,G} − x_{i,j,G}) + F.(x_{r1,j,G} − x_{r2,j,G});
14 |   |   |   else
15 |   |   |   |   u_{i,j,G+1} = x_{i,j,G};
16 |   |   y⃗ = DE-Seguidor(npf1, genf1, F, CR, u⃗_{i,G+1});
17 |   |   z⃗ = DE-Seguidor(npf2, genf2, F, CR, u⃗_{i,G+1});
18 |   |   AnalisarRestricoesLider (u⃗_{i,G+1}, y⃗, z⃗);
19 |   |   if f1(u⃗_{i,G+1}, y⃗, z⃗) ≤ f1(x⃗_{i,G}, y⃗, z⃗) then
20 |   |   |   x⃗_{i,G+1} = u⃗_{i,G+1};
21 |   |   else
22 |   |   |   x⃗_{i,G+1} = x⃗_{i,G};
```

Retornar: MelhorIndividuoLider

Para cada solução do líder, o Algoritmo 4 (DE-Seguidor) é executado, considerando as variáveis, restrições e função objetivo de cada um dos diferentes seguidores. A descrição de cada etapa do BLDE$_{MSI}$ é apresentada a seguir.

Passo 0: Inicialização. O algoritmo inicia com uma população de vetores de tamanho np contendo os valores das variáveis x do líder. Estas variáveis são

iniciadas aleatoriamente e, para cada uma destas, as variáveis dos seguidores são obtidas executando-se o procedimento de nível inferior (DE-Seguidor) para cada seguidor. Este procedimento é responsável por obter as variáveis y e z do primeiro e do segundo seguidor, respectivamente.

Passo 1: Evolução do nível superior. Seguindo a descrição do algoritmo 11 (DE-Líder$_{MSI}$), os indivíduos do nível superior sofrem mutações e recombinações através da variante DE/target-to-rand/1/bin.

Passo 2: Avaliação dos indivíduos do nível superior. Para avaliar os indivíduos do nível superior, o procedimento do nível inferior (DE-Seguidor) é executado para cada um dos seguidores. As soluções retornadas, que representam a melhor solução obtida para cada problema seguidor, são utilizadas para calcular a função objetivo e restrições dos indivíduos do nível superior.

Passo 3: Evolução do nível inferior. Para cada variável x fixada no nível superior, o algoritmo DE-Seguidor é executado duas vezes, uma para cada problema seguidor. As variáveis do processo seguidor são iniciadas aleatoriamente e os indivíduos são evoluídos utilizando a variante DE/target-to-rand/1/bin, até que o critério de parada seja satisfeito. Cada indivíduo é avaliado de acordo com a função objetivo de cada seguidor, juntamente com suas restrições. Ao final do processo, o algoritmo retorna a melhor solução obtida.

Passo 4: Finalização. Até que o critério de parada seja satisfeito o algoritmo retorna a melhor solução gerada para o problema do nível superior.

O tratamento das restrições foi realizado de forma similar ao apresentado na seção 6.1, onde considerou-se o critério de seleção proposto por Deb em [Deb, 2000]. Assim, o cálculo da função objetivo do líder dependerá agora da violação de suas próprias restrições e da violação das restrições dos dois seguidores.

8.1.1 Experimentos computacionais

Dois experimentos foram realizados para analisar o desempenho do método BLDE$_{MSI}$. No primeiro, utilizou-se um orçamento fixo no número de avaliações de

função em ambos os níveis, e no segundo, aplicou-se o critério de parada descrito na seção 6.3.2, que toma como base a variância da população corrente com relação à população inicial. O algoritmo foi testado nos problemas contendo múltiplos seguidores independentes.

Para estes experimentos foram realizadas 30 execuções independentes para cada um dos problemas. Os experimentos foram realizados em um notebook com processador Intel Core i7 2.4GHz e 16 GB de memória.

– **Definição do valor dos parâmetros** –

Os parâmetros foram configurados como no algoritmo BLDE:

- F: a ponderação da diferença igual a 0.7.
- CR: a taxa de recombinação igual a 0.9.
- np e npf: a população de nível superior e inferior igual a 30 indivíduos.

Para os experimentos utilizando orçamento fixo no número de avaliações de função adotou-se 1500 e 4500000 de avaliações de função para os níveis superior e inferior, respectivamente, onde $gen = 50$ gerações para o nível superior e $genf = 100$ gerações para ambos os seguidores no nível inferior. No segundo experimento, o número máximo de gerações foi o mesmo adotado no experimento anterior e os valores de α_S^{stop} e α_I^{stop}, que indicam o nível de precisão do critério de parada do nível superior e inferior, respectivamente, foram de 1×10^{-6}. Este valor foi ajustado após testes preliminares.

A partir de testes preliminares, verificou-se que a variante DE/target-to-rand/1/bin apresentou melhores resultados dentre as variantes do DE descritas na seção 4.2.2. Por esta razão, esta variante foi utilizada para a realização destes experimentos.

– **Resultados e discussões** –

A Tabela 8.1 sumariza os resultados do BLDE$_{MSI}$, com orçamento fixo no número de avaliações de função, em 30 execuções realizadas para cada problema teste, onde NS, NI_1 e NI_2 indicam os valores da função objetivo do líder e do primeiro e segundo seguidor, respectivamente. A última coluna apresenta o tempo médio de execução em segundos.

A partir da Tabela 8.1, observa-se que o método proposto encontrou a melhor solução descrita na literatura, para os problemas 2 a 5, onde: (i) para os problemas 2 e 5 o método obteve a solução ótima em todas as execuções; (ii) para o problema 3 o método obteve a solução ótima ou muito próxima dela em todas as execuções; e (iii) para o problema 4, em duas das 30 execuções, o método não obteve a solução ótima, ficando preso no ponto de mínimo local $(x, y, z) = (1.75, 1.75, 0.5)$ que fornece $F = -3.75$, $f_1 = 3.5$ e $f_1 = -3$.

Tabela 8.1: Resumo dos resultados obtidos pelo BLDE$_{MSI}$ em 30 execuções, com orçamento fixo de avaliações de função.

Prob.	Nível	Mínimo	Mediana	Média	Máximo	Desvio Padrão	Solução de Ref.	Tempo de exec. (s)
1	NS	0.84	0.84	0.84	0.84	0	$F^* = 111.69$	2.52
	NI_1	-0.12	-0.12	-0.12	-0.12	0	$f_1^* = 10.02$	
	NI_2	9.3e-11	9.4e-11	9.4e-11	9.5e-11	0	$f_2^* = 11.61$	
2	NS	0.5	0.5	0.5	0.5	0	$F^* = 0.5$	2.60
	NI_1	-2.5	-2.5	-2.5	-2.5	0	$f_1^* = -2.5$	
	NI_2	-1	-1	-1	-1	0	$f_2^* = -1$	
3	NS	-4.4	-4.3999	4.3993	-4.3841	0.0028	$F^* = -4.4$	2.53
	NI_1	3.9878	3.9999	3.9995	4	0.0022	$f_1^* = 4$	
	NI_2	2.5914	2.5999	2.5996	2.6	0.0015	$f_2^* = 2.6$	
4	NS	-4	-3.996	-3.9715	-3.7499	0.0657	$F^* = -4$	2.51
	NI_1	0	0.0010	0.2366	3.4999	0.8870	$f_1^* = 0$	
	NI_2	-2.999	0.9978	3.4999	1	1.0131	$f_2^* = 1$	
5**	NS	1.6667	1.6667	1.6667	1.6667	0	$F^* = 1.666$	2.52
	NI_1	0.25	0.25	0.25	0.25	0	$f_1^* = 0.25$	
	NI_2	0.2222	0.2222	0.2222	0.2222	0	$f_2^* = 0.222$	

**Problema de maximização no nível superior e inferior.

A solução obtida para o problema 1 é melhor que a melhor solução descrita na literatura. Para este problema, é importante mencionar que em [Lu et al.,

2006], referência da qual este exemplo foi retirado, utilizou-se a abordagem KKT para transformar o problema em um único nível. Nesse problema transformado, duas restrições de desigualdade do problema 1 tornam-se ativas, e o método clássico de otimização utilizado obtêm então o ponto ótimo $(x_1^*, x_2^*, y^*, z^*) = (0.12, 11.79, 2.01, 0.27)$, fornecendo $F^* = 111.69$, $f_1^* = 10.02$ e $f_2^* = 11.61$. No entanto, no problema originalmente formulado, tais restrições não estão ativas e o por isso o BLDE$_{MSI}$ obteve uma solução diferente, melhor que a solução da literatura. O que era esperado uma vez que o problema original é menos restrito que o problema transformado em [Lu et al., 2006]. De fato, a melhor solução obtida pelo BLDE$_{MSI}$ foi $(x_1^*, x_2^*, y^*, z^*) = (0, 0, 0.12, 0)$, fornecendo então $F^* = 0.84$, $f_1^* = -0.12$ e $f_2^* = 0$. Assim, tem-se no problema 1 um exemplo onde o problema transformado, utilizando a abordagem KKT, não é equivalente ao problema original.

Na segunda parte dos experimentos, utilizou-se no método BLDE$_{MSI}$ um critério de parada baseado na variância da população, conforme descrição na seção 6.3.2. O objetivo foi verificar a possibilidade de se reduzir o número de avaliações de função em ambos os níveis sem comprometer a qualidade das soluções geradas.

A Tabela 8.2 sumariza os resultados do BLDE$_{MSI}$, com critério de parada baseado na variância da população, em 30 execuções realizadas para cada um dos problemas. A última coluna apresenta o tempo médio de execução em segundos. Em seguida, a Tabela 8.3 apresenta o número de avaliações de função do nível superior e inferior, assim como a porcentagem de redução (%Redução) no número de avaliações de função realizadas em ambos os níveis, tomando como base o valor da mediana.

Pelas Tabelas 8.2 e 8.3 observa-se que o método proposto foi capaz de obter as soluções ótimas, ou muito próxima delas, em todos os problemas analisados. Além disso, o método apresentou-se estável uma vez que o valor do desvio padrão dos resultados obtidos foi próximo, ou igual a zero, em todos os problemas. Observa-se também que houve uma economia significativa no número de avaliações de

Tabela 8.2: Resumo dos resultados obtidos pelo BLDE$_{MSI}$ em 30 execuções, com critério de parada baseado na variância da população.

Prob.	Nível	Mínimo	Mediana	Média	Máximo	Desvio Padrão	Solução de Ref.	Tempo de exec. (s)
	NS	0.3974	0.6316	0.6416	0.8149	0.1192	$F^* = 0.84$	
1	NI_1	-0.1158	-0.0793	-0.0744	0.0015	0.0297	$f_1^* = -0.12$	0.17
	NI_2	-0.0001	0.0584	0.1012	0.5480	0.1476	$f_2^* = 0$	
	NS	0.5	0.5	0.5	0.5	0	$F^* = 0.5$	
2	NI_1	-2.5	-2.4998	-2.4995	-2.4962	0.0007	$f_1^* = -2.5$	0.11
	NI_2	-1	-0.9995	-0.9993	-0.9972	0.0006	$f_2^* = -1$	
	NS	-4.4684	-4.4263	-4.4297	-4.4021	0.0155	$F^* = -4.4$	
3	NI_1	3.9834	3.9932	3.9927	3.9974	0.0036	$f_1^* = 4$	0.58
	NI_2	2.5968	2.6032	2.6035	2.6112	0.0037	$f_2^* = 2.6$	
	NS	-4.0131	-4.0032	-3.9866	-3.7824	0.0498	$F^* = -4$	
4	NI_1	0.0024	0.0078	0.0121	0.6997	0.0136	$f_1^* = 0$	0.68
	NI_2	0.8692	0.9936	0.9846	0.9994	0.0276	$f_2^* = 1$	
	NS	1.6667	1.6667	1.6667	1.6667	0	$F^* = 1.666$	
5**	NI_1	0.25	0.25	0.25	0.25	0	$f_1^* = 0.25$	0.09
	NI_2	0.2222	0.2222	0.2222	0.2222	0	$f_2^* = 0.222$	

**Problema de maximização no nível superior e inferior.

Tabela 8.3: Número de avaliações de função realizadas pelo BLDE$_{MSI}$, com critério de parada baseado na variância da população.

		Avaliação de função				
Prob.	Nível	Mínimo	Mediana	Média	Máximo	%Redução
	NS	496	607.5	638.4	1553	59.5
1	NI_1	232710	284415	297584	669390	93.67
	NI_2	183270	219915	230639	547230	95.11
	NS	356	418	418.1667	419	72.13
2	NI_1	161940	189345	185928	216600	95.79
	NI_2	143190	164145	163675	191880	96.35
	NS	610	1510.5	1422.667	1520	0
3	NI_1	438180	1089495	1024834	110844	75.78
	NI_2	385320	935010	881235	940200	79.22
	NS	1505	1512	1511.86	1522	0
4	NI_1	1005420	1064005	1056200	1098840	76.35
	NI_2	1128720	1132215	1132372	1134930	74.83
	NS	240	360	348	480	76
5**	NI_1	96090	138105	134997	181260	96.93
	NI_2	96450	138270	134990	182910	96.92

**Problema de maximização no nível superior e inferior.

função em ambos os níveis em todos os problemas analisados, com exceção do nível superior dos problemas 3 e 4, onde não houve redução no número de avaliações de função. De fato, houve uma redução no número de avaliações de função no nível inferior de mais de 90% nos problemas 1, 2 e 5 e de mais de 70% nos problemas 3 e 4. Quanto ao nível superior, houve uma redução no número de avaliações de função de aproximadamente 60%, 70% e 76% nos problemas 1, 2 e 5, respectivamente.

– Conclusões –

Destes experimentos podemos concluir que a metodologia proposta resolveu com sucesso todos os problemas analisados, onde o método com o critério de parada baseado na variância da população mostrou-se mais eficiente do que aquele que adotou um orçamento fixo no número de avaliações de função em ambos os níveis. Além disso, nenhum ajuste nos parâmetros do DE foi realizado, o que poderia reduzir ainda mais o número de avaliações de função realizadas.

A próxima seção apresenta uma nova proposta para resolver a classe de problemas com múltiplos seguidores dependentes.

8.2 BLDE para Seguidores Dependentes (BLDE$_{MSD}$)

O método BLDE$_{MSD}$ também tomou como base o algoritmo BLDE, onde o método de DE é utilizado para resolver cada nível do problema. A diferença deste novo método encontra-se na maneira de se obter a solução do nível inferior onde, nos problemas com múltiplos seguidores dependentes, busca-se por uma solução que atenda ao equilíbrio de Nash no nível inferior do problema. Para obter tal solução, tomou-se como base um esquema proposto em [Sefrioui and Periaux, 2000] para a "construção" da solução de equilíbrio de Nash. Este esquema e o novo algoritmo proposto serão apresentados a seguir.

8.2.1 Esquema para a obtenção do equilíbrio de Nash

No esquema proposto em [Sefrioui and Periaux, 2000], os autores idealizaram um método para "unir" um Algoritmo Genético (AG) à estratégia de equilíbrio de Nash, de modo que o AG "construísse" a solução de equilíbrio, conforme Figura 8.1. Nesta referência, uma solução para o problema é representada por $s = XY$, onde X corresponde ao objetivo do primeiro jogador e Y ao objetivo do segundo jogador. Atribui-se a tarefa de otimização de X ao jogador A e a tarefa de otimização de Y ao jogador B. Assim, seguindo a estratégia de Nash, o jogador A otimiza s com relação ao seu objetivo através da manipulação de X, enquanto Y é fixada pelo jogador B. Similarmente, o jogador B otimiza s com relação ao seu objetivo através da manipulação de Y, enquanto X é fixada pelo jogador A. Desta forma, duas populações foram utilizadas no processo de otimização, uma para cada jogador.

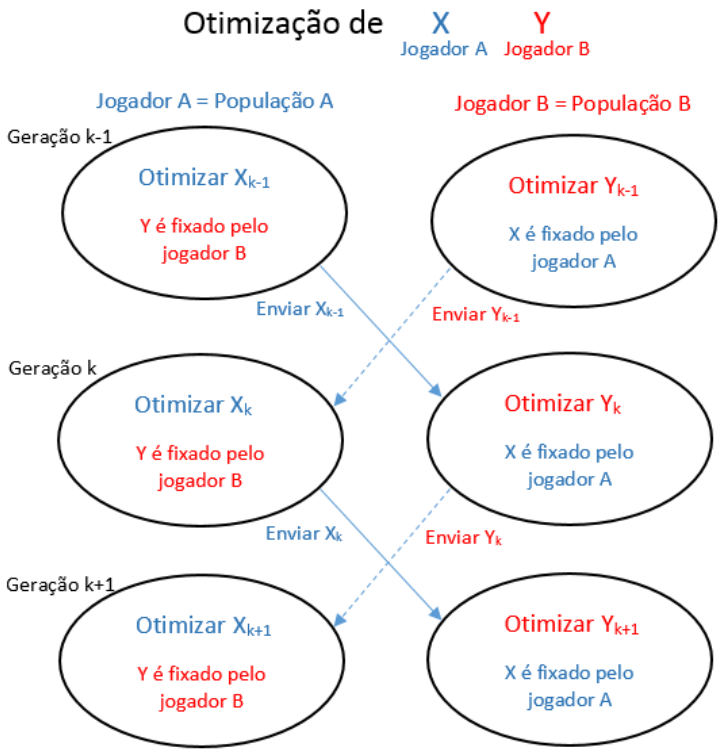


Figura 8.1: Esquema para a obtenção do equilíbrio de Nash proposto em [Sefrioui and Periaux, 2000].

Assim, no esquema da Figura 8.1, seja X_{k-1} o melhor valor encontrado por A na iteração $k-1$, e Y_{k-1} o melhor valor encontrado por B na iteração $k-1$. Na

geração k o jogador A otimiza X_k utilizando Y_{k-1} a fim de avaliar s (neste caso $s = X_k Y_{k-1}$). Ao mesmo tempo, o jogador B otimiza Y_k utilizando X_{k-1} a fim de avaliar s (neste caso $s = X_{k-1} Y_k$). Depois do processo de otimização, o jogador A envia seu melhor valor X_k para o jogador B, que será utilizado na iteração $k+1$. De forma similar, o jogador B envia seu melhor valor Y_k para o jogador A, que irá utiliza-lo na iteração $k+1$. O equilíbrio de Nash é alcançado quando não for mais possível para nenhum dos jogadores melhorar seu respectivo objetivo seguindo este processo.

Tomando como base o esquema da Figura 8.1 para a obtenção da solução de equilíbrio de Nash, elaborou-se um modelo mais econômico do processo de otimização, conforme apresentado na Figura 8.2.

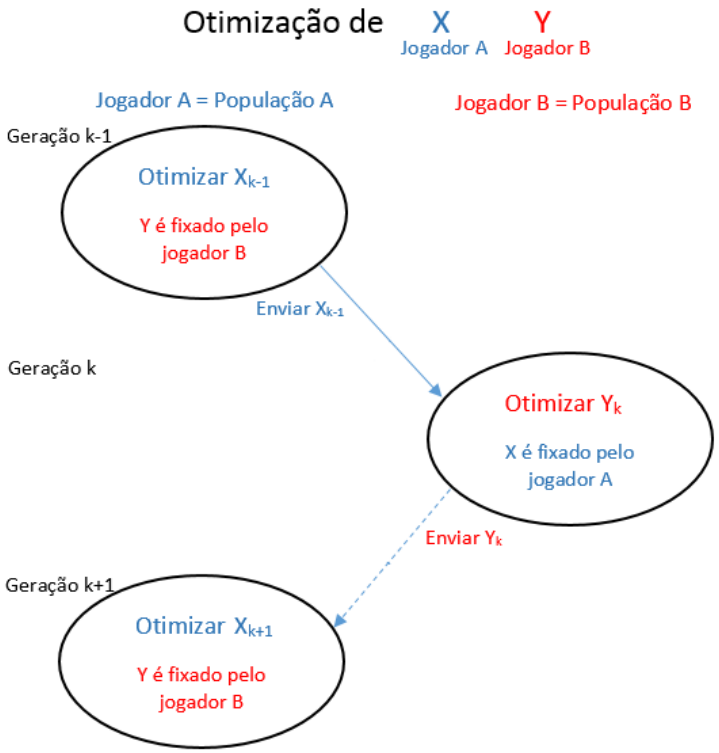


Figura 8.2: Novo esquema para a obtenção do equilíbrio de Nash.

Neste novo esquema proposto, elimina-se uma etapa de processamento a cada geração, ou seja, a cada geração apenas um jogador realiza o processo de otimização. Como consequência, em uma máquina sequencial, este procedimento torna-se menos custoso.

Desta forma, o esquema proposto na Figura 8.2 será utilizado na resolução dos problemas do nível inferior, sendo este esquema implementado para a obtenção da solução de equilíbrio de Nash. Os algoritmos 12 e 13 apresentam a metodologia desenvolvida. Em seguida tem-se a descrição de cada etapa do BLDE$_{MSD}$.

Passo 0: Inicialização. O algoritmo inicia com uma população de vetores de tamanho np contendo os valores das variáveis x do líder. Estas variáveis são iniciadas aleatoriamente e, para cada uma, as variáveis dos seguidores (y e z) são obtidas de forma aleatória dentro dos limites estabelecidos destas variáveis.

Passo 1: Evolução do nível superior. Seguindo a descrição do algoritmo 12, os indivíduos do nível superior evoluídos através da variante DE/rand/1/bin.

Passo 2: Avaliação dos indivíduos do nível superior. Para avaliar os indivíduos do nível superior, o procedimento do nível inferior (DE-Seguidor$_{MSD}$) é executado seguindo o esquema ilustrado na Figura 8.2 onde, a cada ciclo, as variáveis y e z do primeiro e do segundo seguidor, respectivamente, são otimizadas considerando como fixas as variáveis do líder e as variáveis do outro seguidor, correspondente à melhor solução obtida por este na etapa anterior. Assim, após max_ciclos serem realizados, as soluções retornadas, que estão em equilíbrio de Nash, são utilizadas para calcular a função objetivo e restrições da população do nível superior.

Passo 3: Evolução do nível inferior. Cada seguidor otimiza suas variáveis considerando como fixas a variável x do líder e a melhor solução ($\overline{y}$ ou $\overline{z}$) obtida pelo outro seguidor. As variáveis do processo seguidor são iniciadas aleatoriamente e os indivíduos são evoluídos utilizando a variante DE/rand/1/bin, até que o critério de parada seja satisfeito. Cada indivíduo é avaliado de acordo com sua função objetivo, juntamente com suas restrições. Ao final do processo, para um dado x e $\overline{z}$ (resp. x e $\overline{y}$), o algoritmo retorna a melhor solução obtida $\overline{y}$ (resp. $\overline{z}$).

Passo 4: Finalização. Até que o critério de parada seja satisfeito, o Algoritmo 12 retorna a melhor solução gerada para o problema do nível superior.

Algoritmo 12: Algoritmo DE-Líder$_{MSD}$.

Entrada : np (tamanho da população), gen (núm. de gerações), F (fator de mutação), CR (taxa de recombinação)

```
1  G ← 0;
2  CriarPopulacaoInicialAleatoria(np);
3  for i ← 1 to np do
4  |   y⃗ = VetorAleatorio(Y);
5  |   z⃗ = VetorAleatorio(Z);
6  |   Avaliar F(x⃗_{i,G}, y⃗, z⃗);          /* x⃗_{i,G} é um indivíduo da população */
7  for G ← 1 to gen do
8  |   for i ← 1 to np do
9  |   |   SelecionarAleatoriamente(r1, r2, r3);          /* r1 ≠ r2 ≠ r3 ≠ i */
10 |   |   jRand ←AleatorioInteiro(1, n);                /* n variáveis líder */
11 |   |   for j ← 1 to n do
12 |   |   |   if Aleatorio(0,1) < CR or j = jRand then
13 |   |   |   |   u_{i,j,G+1} = x_{r1,j,G} + F.(x_{r2,j,G} − x_{r3,j,G});
14 |   |   |   else u_{i,j,G+1} = x_{i,j,G} ;
15 |   |   z⃗ = VetorAleatorio(Z);
16 |   |   for c ← 1 to max_ciclos do
17 |   |   |   y⃗ = DE-Seguidor(npf1, genf1, F, CR, u⃗_{i,G+1}, z⃗);
18 |   |   |   z⃗ = DE-Seguidor(npf2, genf2, F, CR, u⃗_{i,G+1}, y⃗);
19 |   |   AnalisarRestricoesLider (u⃗_{i,G+1}, y⃗, z⃗);
20 |   |   if F(u⃗_{i,G+1}, y⃗, z⃗) ≤ F(x⃗_{i,G}, y⃗, z⃗) then
21 |   |   |   x⃗_{i,G+1} = u⃗_{i,G+1};
22 |   |   else
23 |   |   |   x⃗_{i,G+1} = x⃗_{i,G};
```

Retornar: MelhorIndividuoLider

Algoritmo 13: Algoritmo DE-Seguidor$_{MSD}$.

Entrada : npf (tamanho da população do seguidor), genf (núm. de gerações do DE seguidor), F (fator de mutação), CR (taxa de recombinação), $\vec{\mathsf{v}}$ (variáveis do líder), $\vec{w}$ (variáveis do outro seguidor)

```
1  G ← 0;
2  CriarPopulacaoInicialAleatoria(npf);
3  for i ← 1 to npf do
4  |   Avaliar f(v⃗, x⃗_{i,G}, w⃗);          /* x⃗_{i,G} é um indivíduo da população */
5  for G ← 1 to genf do
6  |   for i ← 1 to npf do
7  |   |   SelecionarAleatoriamente(r1, r2, r3);          /* r1 ≠ r2 ≠ r3 ≠ i */
8  |   |   jRand ←AleatorioInteiro(1, nf);        /* nf variáveis do seguidor */
9  |   |   for j ← 1 to nf do
10 |   |   |   if Aleatorio(0,1) < CR or j = jRand then
11 |   |   |   |   u_{i,j,G+1} = x_{r3,j,G} + F.(x_{r1,j,G} − x_{r2,j,G});
12 |   |   |   else
13 |   |   |   |   u_{i,j,G+1} = x_{i,j,G};
14 |   |   AnalisarRestricoesSeguidor (v⃗, u_{i,j,G+1}, w⃗);
15 |   |   if f(v⃗, u⃗_{i,G+1}, w⃗) ≤ f(v⃗, x⃗_{i,G}, w⃗) then
16 |   |   |   x⃗_{i,G+1} = u⃗_{i,G+1};
17 |   |   else
18 |   |   |   x⃗_{i,G+1} = x⃗_{i,G};
```

Retornar: MelhorIndividuoSeguidor

Para os problemas com restrições, utilizou-se a mesma abordagem estabelecida no método BLDE$_{MSI}$, descrito na seção 8.1.

8.2.2 Experimentos computacionais

Esta seção apresenta os resultados dos experimentos realizados para analisar o desempenho do método BLDE$_{MSD}$. Este algoritmo foi testado em problemas contendo múltiplos seguidores dependentes, que foram apresentados na seção 5.3.2. Foram realizadas 30 execuções independentes para cada um dos problemas, onde os experimentos foram executados em um notebook com processador Intel Core i7 2.4GHz e 16 GB de memória.

A análise dos resultados segue a mesma abordagem realizada para o método BLDE$_{MSI}$ onde, no primeiro teste, analisou-se o método utilizando um orçamento fixo no número de avaliações de função, em ambos os níveis, e no segundo, adotou-se um critério de parada baseado na variância da população.

– Definição do valor dos parâmetros –

O valor dos parâmetros foi configurado como no algoritmo BLDE$_{MSI}$:

- F: a ponderação da diferença igual a 0.7.
- CR: a taxa de recombinação igual a 0.9.
- np e npf: a população de nível superior e inferior igual a 30 indivíduos.

No primeiro experimento, utilizou-se o mesmo orçamento adotado pelo método BLDE$_{MSI}$ para o número de avaliações de função do nível superior e inferior. Assim, adotou-se 1500 e 4500000 de avaliações de função para o nível superior e inferior, respectivamente. Nestes testes analisou-se o algoritmo com diferentes valores para max_ciclos e gerações $genf$ no nível inferior, onde cada configuração destes parâmetros totalizam ao final da execução 4500000 de avaliações de função do nível inferior.

No segundo experimento, os valores de α_S^{stop} e α_I^{stop}, que indicam o nível de precisão do critério de parada do nível superior e inferior, respectivamente, foi de 10^{-5}. Este valor foi ajustado após testes preliminares. Nestes experimentos, apenas a variante DE/rand/1/bin foi utilizada para a análise de desempenho do algoritmo.

– Resultados e discussões –

As tabelas 8.4 e 8.5 sumarizam os resultados obtidos pelo BLDE$_{MSD}$, em 30 execuções, utilizando um orçamento fixo no número de avaliações de função. As linhas NS, NI$_1$ e NI$_2$ indicam os valores da função objetivo do líder e do primeiro e segundo seguidor, respectivamente, e a última coluna apresenta a média do tempo de execução em segundos. Para cada problema adotou-se três configurações diferentes para os parâmetros max_ciclos e $genf$, indicados nas tabelas por $max\_ciclos \times (genf_1 + genf_2)$, onde $genf_1 = genf_2 = genf$.

Tabela 8.4: Resumo dos resultados obtidos pelo BLDE$_{MSD}$ para o problema 1, com orçamento fixo no número de avaliações de função.

Prob.	Nível	Mínimo	Mediana	Média	Máximo	Desvio Padrão	Solução de Ref.	Tempo de exec. (s)
				$2 \times (50+50)$				
	NS	3	3	3	3	0		
	NI$_1$	1	1	1	1	0		2.37
	NI$_2$	1	1	1	1	0		
				$5 \times (20+20)$				
1	NS	2.9996	2.9997	2.9997	2.9998	6.19e-5	$F^* = 3$	
	NI$_1$	0.9992	0.9995	0.9996	1	0.0002	$f_1^* = 1$	2.51
	NI$_2$	0.9992	0.9997	0.9996	1	0.0002	$f_2^* = 1$	
				$10 \times (10+10)$				
	NS	2.9588	2.9856	2.9831	2.9896	0.0065		
	NI$_1$	0.9518	0.9831	0.9831	1.0041	0.0165		2.79
	NI$_2$	0.9259	0.9784	0.9805	1.0038	0.0199		

Os resultados da Tabela 8.4 indicam que o método proposto foi capaz de obter a solução ótima do problema 1 em todas as 30 execuções, indicando estabilidade no método. Observa-se também que os melhores resultados foram obtidos quando o algoritmo foi configurado com $max\_ciclos = 2$ e $genf = 50$, confirmado pelo

Tabela 8.5: Resumo dos resultados obtidos pelo BLDE_{MSD} para os problema 2 e 3, com orçamento fixo no número de avaliações de função.

Prob.	Nível	Mínimo	Mediana	Média	Máximo	Desvio Padrão	%Suc.	Solução de Ref.	Tempo de exec. (s)
			$2 \times (50+50)$						
	NS	0	21.9992	19.7923	22	6.7102			
	NI_1	0	0.9999	0.8994	1	0.3049	90		2.43
	NI_2	-1	-0.9999	-0.8994	0	0.3049			
			$5 \times (20+20)$						
2**	NS	0	22.0001	19.0646	22.0005	7.6056		$F^* = 22$	
	NI_1	0	1.0001	0.8667	1.0011	0.3457	86.66	$f_1^* = 1$	2.55
	NI_2	-1.0011	-1.0001	-0.8667	0	0.3457		$f_2^* = -1$	
			$10 \times (10+10)$						
	NS	0	22	20.5361	22.0318	5.5823			
	NI_1	0	1.0105	0.9480	1.0595	0.2581	93.33		2.74
	NI_2	-1.0575	-1.0105	-0.9474	0	0.2578			
			$2 \times (50+50)$						
	NS	16	23	22.2999	23	2.1358			
	NI_1	0	0	0	0	0	90		2.40
	NI_2	7.9998	8	7.9999	8	3.16e-5			
			$5 \times (20+20)$						
3**	NS	16	22.9997	22.0664	22.9999	2.4201		$F^* = 22.5$	
	NI_1	0	0	0	0	0	86.66	$f_1^* = 0$	2.57
	NI_2	7.9994	7.9998	7.9997	8	0.0001		$f_2^* = 6.75$	
			$10 \times (10+10)$						
	NS	16	22.9940	22.5267	22.9973	1.7741			
	NI_1	0	0	0	0	0	93.33		2.81
	NI_2	7.9802	7.9947	7.9939	7.9999	0.0047			

**Problema de maximização no nível superior e inferior.

desvio padrão igual a zero nos dois níveis do problema nesta configuração.

A Tabela 8.5 apresenta os resultados obtidos para os problemas 2 e 3, onde o método foi capaz de obter a solução ótima, ou próxima dela, na maior parte das execuções, conforme indicado na coluna "%Suc.", que apresenta a taxa de sucesso, dentre as 30 execuções, em que o algoritmo alcançou a solução ótima do problema. Para estes problemas a estabilidade do método também pôde ser constatada pois, no pior caso, a taxa de sucesso na obtenção da solução ótima foi de aproximadamente 86%.

É importante ressaltar que, para o problema 3, o método proposto foi capaz de obter uma solução melhor do que a solução de referência dada em [Wang et al., 2000]. Analisando este problema, na solução ótima $(1.5, 0, 1.5)$ que fornece $F^* =$

22.5, $f_1^* = 0$ e $f_2^* = 6.75$, indicada pela referência [Wang et al., 2000], apenas uma restrição está ativa no ponto ótimo, enquanto que o método BLDE$_{MSD}$ obteve uma solução com duas restrições ativas no ótimo, sendo este ponto igual a $(1, 0, 2)$, que fornece os valores $F^* = 23$, $f_1^* = 0$ e $f_2^* = 8$.

Para estes problemas, analisando os valores do desvio padrão, os melhores resultados foram obtidos quando o algoritmo foi configurado com $max\_ciclos = 10$ e $genf = 10$.

Nota-se também que, em todos os experimentos, a diferença entre os resultados obtidos com relação as diferentes configurações são pequenas comparando-se os valores dos desvios padrão. Este fato indica que qualquer uma das configurações analisadas habilita o método obter soluções de qualidade.

Na segunda parte dos experimentos, analisou-se o método com um critério de parada baseado na variância da população, a fim de reduzir o número de avaliações de função, sem prejudicar a qualidade das solução. Os resultados destes experimentos são apresentados nas tabelas 8.6 e 8.7 onde, na primeira, são exibidos os resultados relativos ao valor das funções objetivo e na segunda, os valores relativos ao número de avaliações de função realizadas e economizadas.

Pela Tabela 8.6 é interessante notar agora que, para os problemas 1 e 3, as diferentes configurações nos parâmetros max_ciclos e $genf$, não alteraram de maneira significativa os resultados finais obtidos. Tal comportamento pode ser observado pela similaridade dos valores dos desvios padrão nestes dois problemas. Já para o problema 2, a configuração com $max\_ciclos = 5$ e $genf = 20$, apresentou claramente melhores resultados.

Na Tabela 8.7, verifica-se que no problema 1 não houve redução no número de avaliações de funções no nível superior em nenhuma das configurações. Houve apenas redução significativa no nível inferior quando $max\_ciclos = 2$ e $genf = 50$. Ou seja, para obter soluções de boa qualidade neste problema, o algoritmo necessitou de mais de 10, ou 20, gerações no nível inferior. Com isso, não houve redução significativa no número de avaliações de função neste nível quando $genf$ foi

configurado com valor igual a 10 ou 20, pois o método não conseguiu interromper o processo de otimização antes de alcançar estes valores limitantes.

Ainda na Tabela 8.7, nos problemas 2 e 3, houve redução no número de avaliações de função em ambos os níveis, em todos as configurações. A partir destes resultados tem-se que, para qualquer dos valores adotados para max_ciclos e $genf$, o algoritmo obteve redução no número de avaliações de função. No entanto, de acordo com o valor da mediana apresentado na Tabela 8.6 para o problema 2, as soluções obtidas não ficaram tão próximas do valor ótimo, do que quando o orçamento fixo no número de avaliações de função foi adotado. Para este caso, a redução da precisão do critério de parada poderia gerar soluções de melhor qualidade, porém, em contrapartida, poderia não haver uma redução tão significativa no número de avaliações de função.

– Conclusões –

Destes experimentos, conclui-se que o método BLDE$_{MSD}$ foi capaz de resolver com sucesso todos os problemas, sendo que a versão do método com critério de parada baseado na variância da população mostrou-se mais eficiente do que quando adotou-se um orçamento fixo no número de avaliações de função. Mais uma vez, um ajuste mais preciso no valor dos parâmetros utilizados poderia reduzir ainda mais o número de avaliações de função realizadas e possivelmente melhorar a qualidade das soluções geradas.

8.3 Considerações finais

Nesta última etapa de desenvolvimento, dois algoritmos foram propostos para resolver o problema em dois níveis com múltiplos seguidores. O primeiro, o BLDE$_{MSI}$, tratou dos problemas com seguidores independentes, e o segundo, o BLDE$_{MSD}$, tratou os problemas com seguidores dependentes.

Ambos os métodos mostraram-se eficazes na resolução dos problemas testados e a versão dos algoritmos com critério de parada baseado na variância

da população apresentou-se mais eficiente.

Tabela 8.6: Resumo dos resultados obtidos pelo BLDE$_{MSD}$, com critério de parada baseado na variância da população.

Prob.	Nível	Mínimo	Mediana	Média	Máximo	Desvio Padrão	Solução de Ref.	Tempo de exec. (s)
			$2 \times (50 + 50)$					
	NS	2.9973	2.9983	2.9983	2.9989	0.0003		
	NI_1	0.9957	0.9985	0.9983	1.0002	0.0012		0.77
	NI_2	0.9951	0.9982	0.9980	0.9999	0.0014		
			$5 \times (20 + 20)$					
1	NS	2.9974	2.9983	2.9982	2.9989	0.0003	$F^* = 3$	
	NI_1	0.9957	0.9982	0.9980	1.0002	0.0013	$f_1^* = 1$	1.97
	NI_2	0.9953	0.9982	0.9982	1.0002	0.0013	$f_2^* = 1$	
			$10 \times (10 + 10)$					
	NS	2.9723	2.9820	2.9818	2.9884	0.0038		
	NI_1	0.9238	0.9814	0.9789	1.0050	0.0193		2.76
	NI_2	0.9455	0.9832	0.9816	1.0022	0.0166		
			$2 \times (50 + 50)$					
	NS	0	20.5479	18.7253	21.9437	6.1187		
	NI_1	0	0.9451	0.9986	3.0309	0.5197		0.31
	NI_2	-0.9979	-0.8864	-06899	1.0103	0.4713		
			$5 \times (20 + 20)$					
2**	NS	0	21.3712	19.9628	21.9286	5.2655	$F^* = 22$	
	NI_1	0	0.9660	0.9172	1.2434	0.2602	$f_1^* = 1$	0.51
	NI_2	-0.9995	-0.96	-0.8753	0	0.2495	$f_2^* = -1$	
			$10 \times (10 + 10)$					
	NS	0	21.719	18.1433	22.0047	8.0677		
	NI_1	0	0.9853	0.8428	1.0573	0.3496		0.81
	NI_2	-1.0454	-0.9853	-0.8188	0.1620	0.3888		
			$2 \times (50 + 50)$					
	NS	15.9996	22.9832	22.2207	26.2178	2.5588		
	NI_1	0	0	0.0874	1.1156	0.2760		0.35
	NI_2	5.6519	7.9846	7.8324	7.9999	0.5225		
			$5 \times (20 + 20)$					
3**	NS	15.9992	22.9858	22.1872	25.8691	2.4147	$F^* = 23$	
	NI_1	0	0	0.0373	1.1202	0.2045	$f_1^* = 0$	0.51
	NI_2	6.3784	7.9898	7.9324	7.9999	0.2937	$f_2^* = 8$	
			$10 \times (10 + 10)$					
	NS	15.9896	22.9729	22.6811	27.0581	2.5496		
	NI_1	0	0	0.0699	1.0481	0.2389		1.04
	NI_2	5.2113	7.96992	7.7378	7.9998	0.7008		

**Problema de maximização no nível superior e inferior.

Tabela 8.7: Número de avaliações de função realizadas pelo BLDE_{MSD}, com critério de parada baseado na variância da população.

Prob.	Nível	Avaliação de função				
		Mínimo	Mediana	Média	Máximo	%Redução
1				$2 \times (50 + 50)$		
	NS	1500	1500	1500.067	15001	0
	NI_1	1250400	1275165	1275642	1293504	71.66
	NI_2	1253040	1279515	1278207	1309560	71.56
				$5 \times (20 + 20)$		
	NS	1500	1500	1500.167	1502	0
	NI_1	3117660	3176535	3178803	3233400	29.41
	NI_2	3099510	3181980	3181682	3257610	29.28
				$10 \times (10 + 10)$		
	NS	1500	1500	1500.02	1502	0
	NI_1	4324800	4386300	4386175	4448340	2.52
	NI_2	4335510	4381995	4382556	4431810	2.62
2**				$2 \times (50 + 50)$		
	NS	120	482	466.9333	724	67.86
	NI_1	190950	572430	539910	756900	87.27
	NI_2	176070	397065	392951	659280	91.17
				$5 \times (20 + 20)$		
	NS	210	466.5	485.9	810	68.9
	NI_1	386820	873525	896987	1396650	80.58
	NI_2	382920	650340	664598	1011390	85.84
				$10 \times (10 + 10)$		
	NS	180	510.5	473.9333	752	65.96
	NI_1	435660	1343055	1241021	1979760	70.15
	NI_2	431490	1116540	1047047	1623390	75.18
3**				$2 \times (50 + 50)$		
	NS	166	520	522.6	1523	65.33
	NI_1	217650	500310	506502	1133910	88.88
	NI_2	235320	557880	574934	1416750	87.60
				$5 \times (20 + 20)$		
	NS	189	464.5	465.9	1511	69.03
	NI_1	324240	659220	690155	2016150	85.35
	NI_2	421710	967275	986611	3249420	78.505
				$10 \times (10 + 10)$		
	NS	207	557	595	1546	62.86
	NI_1	418470	1233240	1294695	3321510	72.59
	NI_2	514650	1542900	1650983	4365780	65.71

**Problema de maximização no nível superior e inferior.

Capítulo 9

Conclusões e Pesquisas Futuras

O principal objetivo deste trabalho foi o desenho e a implementação computacional de algoritmos robustos e eficientes para resolver problemas de otimização em dois níveis. Como foi visto ao longo do texto, os problemas de PDN constituem uma importante classe de problemas de otimização com uma variedade de aplicações reais em diversos domínios de conhecimento. Contudo, estes problemas apresentam uma série de dificuldades, que constituem um desafio ao desenho de métodos eficientes.

Os métodos computacionais desenvolvidos nesta tese, para a resolução independente dos problemas de nível superior e inferior, fizeram uso de duas metaheurísticas: (i) a otimização por colônia de formigas e (ii) o método de evolução diferencial. Deve-se ter em mente entretanto que os procedimentos propostos são gerais e admitem a substituição destas metaheurísticas por outras eventualmente mais adequadas ao problema em estudo tratado.

Além destes algoritmos, desenvolveu-se também um metamodelo, para a substituição do procedimento computacionalmente intensivo do nível inferior, na tentativa de melhorar a eficiência de um dos métodos propostos.

O desempenho das metodologias propostas foi analisado utilizando diversos problemas em dois níveis divididos em três classes: 1. problemas lineares e não-lineares, com ou sem restrições; 2. uma aplicação na área de pesquisa operacional envolvendo o problema de planejamento de produção e distribuição; e 3. problemas

em dois níveis contendo múltiplos seguidores, onde abordou-se os casos com múltiplos seguidores independentes e dependentes. Comparações com os valores ótimos dos problemas das classes 1 e 3 demonstraram que os métodos propostos são robustos e eficazes na resolução dos problemas abordados. Enquanto que, para os problemas da classe 2, mostrou-se que as metodologias propostas produzem resultados competitivos.

Sendo assim, a partir das metodologias desenvolvidas foi possível resolver com sucesso diversos problemas de PDN, visto que foram obtidos resultados competitivos na linha dos problemas abordados. Vale ressaltar que comparações com outros métodos computacionais ainda são difíceis de serem realizadas, devido à falta de códigos disponíveis na literatura para comparação.

9.1 Contribuições

A tese contribui para a divulgação da relevância e dos principais desafios enfrentados na resolução de problemas de programação em dois níveis, onde foram realizados estudos sobre os conceitos, abordagens e técnicas de resolução envolvendo problemas desta natureza. Além disso, o importante e desafiante problema com múltiplos seguidores também foi apresentado e estudado.

As metodologias propostas para os diferentes tipos de problemas de PDN vêm aumentar o número ainda reduzido de métodos computacionais para tratar este tipo de problema em sua forma original, uma vez que, como foi mencionado anteriormente, grande parte dos trabalhos buscam resolver o problema transformado em um único nível.

Mesmo que os métodos propostos neste trabalho não apresentem eficiência comprovada, a robustez e a flexibilidade deles demonstram sua eficácia e relevância científica no tratamento deste tipo de problema.

Além disso, os estudos e experimentos realizados no tratamento dos problemas da classe 2 deram uma importante contribuição para o entendimento sobre como um método aproximado no nível inferior pode afetar o processo de

otimização do nível superior e consequentemente a solução final do problema. A partir deste experimento foi possível concluir e sugerir pontos importantes sobre a aplicação adequada das metaheurísticas neste tipo de problema.

No mais, ainda não se tem um consenso de qual abordagem, dentre aquelas apresentadas na seção 2.5, é a melhor do ponto de vista prático e computacional para o tratamento dos problemas de PDN. Porém, do presente trabalho, podemos concluir que os métodos aninhados constituem uma abordagem promissora que, no entanto, demanda alto custo computacional. As próprias metaheurísticas aplicadas também são conhecidas por esta característica –efeito colateral da pouca exigência de conhecimento sobre o domínio do problema e de sua natureza populacional–. Desta forma, considerações acerca desta característica precisam ser analisadas e bem trabalhadas.

Por fim, como também é de amplo conhecimento, as metaheurísticas constituem um ferramental poderoso no tratamento de inúmeros problemas complexos de otimização, devido principalmente a sua flexibilidade e robustez, sendo estes seus fortes atrativos e o motivo pela qual destacam-se das técnicas usuais de programação matemática.

9.2 Pesquisas futuras

Nos últimos anos, o desenho de métodos evolucionistas para a resolução de problemas de otimização multinível, vem se tornando uma crescente área de pesquisa.

Não é de se surpreender que grande parte da comunidade científica foca seus esforços na resolução de problemas em dois níveis com características bem-comportadas, como os problemas lineares, quadráticos ou convexos, devido às diversas dificuldades presentes em problemas desta natureza, mesmos nestes casos ditos bem-comportados. Além das dificuldades presentes na resolução deste tipo de problema, avaliar o desempenho dos métodos também constitui um desafio para os pesquisadores da área.

Dito isso, ficam claros os desafios a serem enfrentados e as motivações para se continuar investindo nesta área de trabalho que ainda carece de pesquisas tanto na parte teórica quanto no desenho de métodos computacionais, visando ampliar o leque de aplicações e o desenvolvimento de técnicas eficientes para viabilizar o tratamento de novos problemas.

Assim, o foco de interesse em pesquisas futuras vem no sentido de ampliar as áreas de aplicação, tratando problemas em dois ou mais níveis com múltiplos objetivos por exemplo, e no desenvolvimento de técnicas mais eficientes que explorem o paralelismo inerente das metaheurísticas aqui utilizadas.

Referências Bibliográficas

E. Aiyoshi and K. Shimizu. A solution method for the static constrained Stackelberg problem via penalty method. **Automatic Control, IEEE Transactions on**, 29(12):1111–1114, 1984. ISSN 0018-9286. doi: 10.1109/TAC.1984.1103455.

G. Anandalingam and D. White. A solution method for the linear static Stackelberg problem using penalty functions. **Automatic Control, IEEE Transactions on**, 35(10):1170–1173, 1990. ISSN 0018-9286. doi: 10.1109/9.58565.

J. S. Angelo and H. J. C. Barbosa. A study on the use of heuristics to solve a bilevel programming problem. **International Transactions in Operational Research (ITOR)**, –. Artigo em revisão.

J. S. Angelo and H. J. C. Barbosa. Otimização por colônia de formigas e evolução diferencial na programação em dois níveis para o problema de transporte-roteamento. In **XLV SBPO - Simpósio Brasileiro de Pesquisa Operacional**, 2013. Resumo.

J. S. Angelo, E. Krempser, and H. J. C. Barbosa. Differential evolution for bilevel programming. In **Evolutionary Computation (CEC), 2013 IEEE Congress on**, pages 470–477, 2013. doi: 10.1109/CEC.2013.6557606.

J. S. Angelo, E. Krempser, and H. J. C. Barbosa. Differential evolution assisted by a surrogate model for bilevel programming problems. In **Evolutionary Computation (CEC), 2014 IEEE Congress on**, pages 1784–1791, 2014. doi: 10.1109/CEC.2014.6900529.

S. R. Arora and R. Narang. 0-1 bilevel fractional programming problem with independent followers. **International Journal of Optimization: Theory, Methods and Applications**, 1(2):225–238, 2009.

C. Audet, J. Haddad, and G. Savard. A note on the definition of a linear bilevel programming solution. **Applied Mathematics and Computation**, 181(1): 351–355, 2006. doi: 10.1016/j.amc.2006.01.043.

H. J. C. Barbosa. A coevolutionary genetic algorithm for a game approach to structural optimization. In T. Bäck, editor, **ICGA**, pages 545–552. Morgan Kaufmann, 1997.

J. F. Bard. Optimality conditions for the bilevel programming problems. **Naval Research Logistic**, 31:13–26, 1984.

J. F. Bard. Convex two-level optimization. **Mathematical Programming**, 40: 15–27, 1988. ISSN 0025-5610. doi: 10.1007/BF01580720.

J. F. Bard. Some properties of the bilevel programming problem. **Journal of Optimization Theory and Applications**, 68(2):371–378, 1991. doi: 10.1007/BF00941574.

J. F. Bard. **Practical Bilevel Optimization**. Kluwer Academic Publisher, 1998. ISBN 0-7923-5458-3.

J. F. Bard and J. E. Falk. An explicit solution to the multi-level programming problem. **Computers & Operations Research**, 9(1):77–100, 1982. ISSN 0305-0548. doi: 10.1016/0305-0548(82)90007-7.

O. Baskan and S. Haldenbilen. **Ant Colony Optimization - Methods and Applications**, chapter Ant Colony Optimization Approach for Optimizing Traffic Signal Timings, pages 205–220. Intech, 2011. doi: 10.5772/13665.

O. Ben-Ayed and C. E. Blair. Computational difficulties of bilevel linear programming. **Operations Researches**, 38(3):556–560, 1990.

V. Beresnev. Branch-and-bound algorithm for a competitive facility location problem. **Computers & Operations Research**, (40):2062–2070, 2013.

W. F. Bialas and M. H. Karwan. Two-Level linear programming. **Management Science**, 30(8):1004–1020, 1984.

C. Blair. The computational complexity of multi-level linear programs. **Annals of Operations Research**, 34(1):13–19, 1992.

J. Bracken and J. T. McGill. Mathematical programs with optimization problems in the constraints. **Operations Research**, 21:37–44, 1973.

J. Bracken and J. T. McGill. Defense applications of mathematical programs with optimization problems in the constraints. **Operations Research**, 22:1086–1096, 1974.

H. Calvete and C. Galé. Linear bilevel multi-follower programming with independent followers. **Journal of Global Optimization**, 39(3):409–417, 2007. ISSN 0925-5001. doi: 10.1007/s10898-007-9144-2.

H. I. Calvete and C. Galé. Solving linear fractional bilevel programs. **Operations Research Letters**, 32:143–151, 2004.

H. I. Calvete, C. Galé, and P. M. Mateo. A new approach for solving linear bilevel problems using genetic algorithms. **European Journal of Operational Research**, 188(1):14–28, 2008.

H. I. Calvete, C. Galé, and M.-J. Oliveros. Bilevel model for production-distribution planning solved by using ant colony optimization. **Computers & Operations Research**, 38(1):320–327, 2011. ISSN 0305-0548. doi: 10.1016/j.cor.2010.05.007.

W. Candler. A linear bilevel programming algorithm: a comment. **Computers and Operations Research**, 15(3):297–298, 1988.

W. Candler and R. Townsley. A linear two-level programming problem. **Computers & Operations Research**, 9(1):59–76, 1982. ISSN 0305-0548. doi: 10.1016/0305-0548(82)90006-5.

U. K. Chakraborty, editor. **Advances in Differential Evolution**, volume 143 of **Studies in Computational Intelligence**. Springer, 2008. ISBN 978-3-540-68830-3.

A. Chinchuluun, P. Pardalos, and H.-X. Huang. Multilevel (hierarchical) optimization: Complexity issues, optimality conditions, algorithms. In D. Y. Gao and H. D. Sherali, editors, **Advances in Applied Mathematics and Global Optimization**, volume 17, pages 197–221. Springer US, 2009. ISBN 978-0-387-75713-1. doi: 10.1007/978-0-387-75714-8-6.

P. A. Clark and A. W. Westerberg. A note on the optimality conditions for the bilvel programming problem. **Naval Research Logistic Quarterly**, 35:413–421, 1988.

B. Colson, P. Marcotte, and G. Savard. Bilevel programming: A survey. **4OR**, 3 (2):87–107, 2005a. ISSN 1619-4500. doi: 10.1007/s10288-005-0071-0.

B. Colson, P. Marcotte, and G. Savard. A trust-region method for nonlinear programming: algorithm and computational experience. **Computational Optimization and Applications**, 30(3):211–227, 2005b. ISSN 0926-6003. doi: 10.1007/s10589-005-4612-4.

B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. **Annals of Operations Research**, 153:235–256, 2007. doi: 10.1007/s10479-007-0176-2.

N. Costantino, M. Dotoli, M. Falagario, M. P. Fanti, A. M. Mangini, F. Sciancalepore, and W. Ukovich. A hierarchical optimization technique for the strategic design of distribution networks. **Computers & Industrial Engineering**, 66(4):849–864, 2013.

S. Das and P. N. Suganthan. Differential evolution: A survey of the state-of-the-art. **Evolutionary Computation, IEEE Transactions on**, 15(1):4–31, 2011. ISSN 1089-778X. doi: 10.1109/TEVC.2010.2059031.

K. Deb. An efficient constraint handling method for genetic algorithms. **Computer Methods in Applied Mechanics and Engineering**, 186(2): 311–338, 2000. ISSN 0045-7825. doi: 10.1016/S0045-7825(99)00389-8.

K. Deb and A. Sinha. Solving bilevel multi-objective optimization problems using evolutionary algorithms. In **Proc. of the 5th International Conference on Evolutionary Multi-Criterion Optimization**, EMO '09, pages 110–124, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-01019-4.

K. Deb, S. Gupta, J. Dutta, and B. Ranjan. Solving dual problems using a coevolutionary optimization algorithm. **Journal of Global Optimization**, 57(3):891–933, 2013.

S. Dempe. **Foundations of Bilivel Programming**. Kluwer Academic Publisher, 2002. ISBN 0-7923-5458-3.

S. Dempe. Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. **Optimization**, 52(3):333–359, 2003a. doi: 10.1080/0233193031000149894.

S. Dempe. **Bilevel Programming: A Survey**. Preprint. TU Bergakademie, 2003b.

M. Dorigo. **Optimization, Learning and Natural Algorithms**. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, 1992. in Italian.

M. Dorigo and G. D. Caro. Ant Colony Optimization: A new meta-heuristic. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, **IEEE Congress on Evolutionary Computation**, pages 1470–1477. IEEE Press, Piscataway, NJ, 1999.

M. Dorigo and L. M. Gambardella. Ant colonies for the traveling salesman problem. **BioSystems**, 43(2):73–81, 1997a.

M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. **IEEE Transactions on Evolutionary Computation**, 1(1):53–66, 1997b.

M. Dorigo and T. Stutzle. **Ant Colony Optimization**. The MIT Press, 2004. ISBN 0-262-04219-3.

F. Facchinei, H. Jiang, and L. Qi. A smoothing method for mathematical programs with equilibrium constraints. **Mathematical Programming**, 85:107–134, 1999. ISSN 0025-5610. doi: 10.1007/s10107990015a.

J. E. Falk and J. Liu. On bilevel programming, part i: General nonlinear cases. **Mathematical Programming**, 70:47–72, 1995. ISSN 0025-5610. doi: 10.1007/BF01585928.

D. C. R. Fernandes. **Estratégia Ótima de Oferta em Mercados Competitivos de Energia**. PhD thesis, Universidade Federal do Rio de Janeiro (UFRJ), 2005.

J. Fliege and L. N. Vicente. Multicriteria approach to bilevel optimization. **Journal of Optimization Theory and Applications**, 131(2):209–225, 2006. ISSN 0022-3239. doi: 10.1007/s10957-006-9136-2.

A. Friedlander and F. A. M. Gomes. Solution of a truss topology bilevel programming problem by means of an inexact restoration method. **Computational & Applied Mathematics**, 30:109–125, 2011.

A. Gaspar-Cunha, R. Takahashi, and C. H. Antunes, editors. **Manual de Computação Evolutiva e Metaheurística**. Editora UFMG and Imprensa da Universidade de Coimbra, 2013.

E. Ghotbi and A. K. Dhingra. A bilevel game theoretic approach to optimum design of flywheels. **Engineering Optimization**, 44(11):1337–1350, 2012.

P. V. Z. C. Golliat, J. S. Angelo, and H. J. C. Barbosa. **Manual de Computação Evolutiva e Metaheurística**, chapter Colônia de Formigas, pages 87–105. Editora UFMG and Imprensa da Universidade de Coimbra, 2013.

D. Gordon. **Ants at work: how an insect society is organized**. The Free Press, 1999.

S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the argentine ant. **Naturvissenschaften**, 76:579–581, 1989.

P. P. Grassé. La reconstruction du nid et les coordinations inter-individuelles chez *bellicosi-termes natalensis* et *cubitermes sp.*. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. **Insectes Sociaux**, 6(1):41–80, 1959.

P. P. Grassé. **Termitologia, Tome II: Fondation des Socétés. Construction**. PAaris: Masson, 1984.

W. Guang-Min, W. Zhong-Ping, and W. Xian-Jia. Solving method for a class of bilevel linear programming based on genetic algorithms. Technical report, Wuhan University, 2003.

W. Guang-Min, W. Zhong-Ping, W. Xian-Jia, and C. Ya-Lin. Genetic algorithms for solving linear bilevel programming. In **Proc. of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies**, PDCAT '05, pages 920–924. IEEE Computer Society, 2005.

W. Guang-Min, W. Xian-Jia, W. Zhong-Ping, and J. Shi-Hui. An adaptive genetic algorithm for solving bilevel linear programming problem. **Applied Mathematics and Mechanics**, 28:1605–1612, 2007. ISSN 0253-4827. doi: 10.1007/s10483-007-1207-1.

P. Hansen, B. Jaumard, and G. Savard. New branch-and-bound rules for linear bilevel programming. **SIAM Journal on Scientific and Statistical Computing**, 13(5):1194–1217, 1992. doi: 10.1137/0913069.

P. T. Harker and J.-S. Pang. Existence of optimal solutions to mathematical programs with equilibrium constraints. **Operations Research Letters**, 7(2): 61–64, 1988.

H. Hoos and T. Stutzle. **Stochastic Local Search: Foundations & Applications**. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004. ISBN 1558608729.

S. Huijun, G. Ziyou, and W. Jianjun. A bi-level programming model and solution algorithm for the location of logistics distribution centers. **Applied Mathematical Modelling**, (32):610–616, 2008.

Y. Ishizuka and E. Aiyoshi. Double penalty method for bilevel optimization problems. **Annals of Operations Research**, 34:73–88, 1992. ISSN 0254-5330. doi: 10.1007/BF02098173.

A. Koh. Solving transportation bi-level programs with differential evolution. In **IEEE Congress on Evolutionary Computation, 2007 (CEC 2007)**, pages 2243–2250, 2007. doi: 10.1109/CEC.2007.4424750.

A. Koh. A coevolutionary particle swarm algorithm for bi-level variational inequalities: Applications to competition in highway transportation networks. In R. Chiong and S. Dhakal, editors, **Natural Intelligence for Scheduling, Planning and Packing Problems**, volume 250 of **Studies in Computational Intelligence**, pages 195–217. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-04038-2. doi: 10.1007/978-3-642-04039-9-8.

A. Koh. Differential evolution based bi-level programming algorithm for computing normalized nash equilibrium. In A. Gaspar-Cunha, R. Takahashi, G. Schaefer, and L. Costa, editors, **Soft Computing in Industrial Applications**,

volume 96 of **Advances in Intelligent and Soft Computing**, pages 97–106. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-20504-0. doi: 10.1007/978-3-642-20505-7-8.

E. Krempser. Evolução diferencial para problemas de otimização restrita. Master's thesis, Laboratório Nacional de Computação Científica, 2009.

E. Krempser. **Uso de Metamodelos na Evolução Diferencial para Problemas Envolvendo Simulações de Alto Custo Computacional**. PhD thesis, Laboratório Nacional de Computação Científica, 2014.

R. Kuo and C. Huang. Application of particle swarm optimization algorithm for solving bi-level linear programming problem. **Computers & Mathematics with Applications**, 58(4):678–685, 2009. ISSN 0898-1221.

M. Labbé and A. Violin. Bilevel programming and price setting problems. **4OR**, 11(1):1–30, 2013.

F. Legillon, A. Liefooghe, and E.-G. Talbi. CoBRA: A cooperative coevolutionary algorithm for bi-level optimization. In **IEEE Congress on Evolutionary Computation**, pages 1–8, Brisbane, Australie, 2012. doi: 10.1109/CEC.2012.6256620.

H. Li and L. Fang. An evolutionary algorithm for solving bilevel programming problems using duality conditions. **Mathematical Problems in Engineering**, 2012, 2012. doi: 10.1155/2012/471952.

H. Li and L. Fang. Co-evolutionary algorithm: An efficient approach for bilevel programming problems. **Engineering Optimization**, 46(3):361–376, 2014. doi: 10.1080/0305215X.2013.772601.

H. Li and Y. Wang. A genetic algorithm for solving a special class of nonlinear bilevel programming problems. In Y. Shi, G. van Albada, J. Dongarra, and P. Sloot, editors, **Computational Science - ICCS 2007**, volume 4490 of

Lecture Notes in Computer Science, pages 1159–1162. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-72589-3.

H. Li and Y. Wang. An evolutionary algorithm with local search for convex quadratic bilevel programming problems. **Applied Mathematics & Information Sciences**, 5(2):139–146, 2011.

M. Li, D. Lin, and S. Wang. Solving a type of biobjective bilevel programming problem using nsga-ii. **Computers & Mathematics with Applications**, 59 (2):706 – 715, 2010. ISSN 0898-1221. doi: 10.1016/j.camwa.2009.10.022.

B. Liu. Stackelberg-nash equilibrium for multilevel programming with multiple followers using genetic algorithms. **Computers & Mathematics with Applications**, 36(7):79 – 89, 1998.

G. Liu, S. Xu, and J. Han. A trust region algorithm for solving bilevel programming problems. **Acta Mathematicae Applicatae Sinica**, 29(3):491–498, 2013.

J. Lu, C. Shi, and G. Zhang. On bilevel multi-follower decision making: General framework and solutions. **Information Sciences**, 176(11):1607 – 1627, 2006.

J. Lu, C. Shi, G. Zhang, and T. Dillon. Model and extended Kuhn-Tucker approach for bilevel multi-follower decision making in a referential-uncooperative situation. **Journal of Global Optimization**, 38(4):597–608, 2007a.

J. Lu, C. Shi, G. Zhang, and D. Ruan. An extended branch and bound algorithm for bilevel multi-follower decision making in a referential-uncooperative situation. **International Journal of Information Technology & Decision Making**, 06(02):371–388, 2007b.

R. Lucchetti, F. Mignanego, and G. Pieri. Existence theorems of equilibrium points in stackelberg with constraints. **Optimization**, 18(6):857–866, 1987.

C. M. Macal and A. P. Hurter. Dependence of bilevel mathematical programs on

irrelevant constraints. **Computers & Operations Research**, 24(12):1129 – 1140, 1997.

R. Mathieu, L. Pittard, and G. Anandalingam. Genetic algorithm based approach to bi-level linear programming. **RAIRO - Operations Research - Recherche Operationnelle**, 28(1):1–21, 1994.

X. Meng, B. Cui, L. Jia, Y. Qin, and J. Xu. Networked timetable stability improvement based on a bilevel optimization programming model. **Mathematical Problems in Engineering**, 2014(Article ID 290937):10 pages, 2014.

A. G. Mersha and S. Dempe. Linear bilevel programming with upper level constraints depending on the lower level solution. **Applied Mathematics and Computation**, 180(1):247 – 254, 2006.

Z. Michalewicz. **Genetic Algorithms + Data Structures = Evolution Programs**. Springer-Verlag, 1996.

J. Nash. Non-Cooperative Games. **Annals of Mathematics**, 54(2):286–295, 1951.

V. Oduguwa and R. Roy. Bi-level optimisation using genetic algorithm. In **Proc. of the 2002 IEEE International Conference on Artificial Intelligence Systems (ICAIS'02)**, pages 322–327, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1733-1.

A. Ostfeld, editor. **Ant Colony Optimization: Methods and Applications**. InTech, 2011. ISBN 978-953-307-157-2.

W. Pimentel and M. Fampa. A genetic algorithm to the strategic pricing problem in competitive electricity markets. In **Simpósio Brasileiro de Pesquisa Operacional, SBPO**, pages 3684–3692, 2012.

K. V. Price, R. M. Storn, and J. A. Lampinen. **Differential Evolution: A Practical Approach to Global Optimization**. Springer, 2005. ISBN 978-3-540-31306-9.

M. S. Radjef and A. Anzi. Solving linear bilevel programming by dc algorithm. In **Colloque sur l' Optimisation et les Systèmes d'Information (COSI'2010)**, pages 434–445, 2010.

J. Rajesh, K. Gupta, H. S. Kusumakar, V. K. Jayaraman, and B. D. Kulkarni. A tabu search based approach for solving a class of bilevel programming problems in chemical engineering. **Journal of Heuristics**, 9:307–319, 2003. ISSN 1381-1231. doi: 10.1023/A:1025699819419.

N. Safari, M. Zarghami, and F. Szidarovszky. Nash bargaining and leader-follower models in water allocation: Application to the zarrinehrud river basin, iran. **Applied Mathematical Modelling**, 38:1959–1968, 2014.

K. H. Sahin and A. R. Ciric. A dual temperature simulated annealing approach for solving bilevel programming problems. **Computers & Chemical Engineering**, 23(1):11–25, 1998. ISSN 0098-1354. doi: 10.1016/S0098-1354(98)00267-1.

M. Sakawa and I. Nishizaki. **Cooperative and Noncooperative Multi-Level Programming**. Springer, 2009.

G. Savard and J. Gauvin. The steepest descent direction for the nonlinear bilevel programming problem. **Operations Research Letters**, 15(5):265–272, 1994. ISSN 0167-6377. doi: 10.1016/0167-6377(94)90086-8.

M. Sefrioui and J. Periaux. Nash genetic algorithms: examples and applications. In **Proc. of the 2000 Congress on Evolutionary Computation**, volume 1, pages 509–516, 2000. doi: 10.1109/CEC.2000.870339.

C. Shi, G. Zhang, and J. Lu. The kth-best approach for linear bilevel multi-follower programming. **Journal of Global Optimization**, 33(4):563–578, 2005.

C. Shi, J. Lu, G. Zhang, and H. Zhou. An extended branch and bound algorithm for linear bilevel programming. **Applied Mathematics and Computation**, 180(2):529–537, 2006. ISSN 0096-3003. doi: 10.1016/j.amc.2005.12.039.

C. Shi, H. Zhou, J. Lu, G. Zhang, and Z. Zhang. The kth-best approach for linear bilevel multifollower programming with partial shared variables among followers. **Applied Mathematics and Computation**, 188:1686–1698, 2007.

K. Shimizu and E. Aiyoshi. A new computational method for stackelberg and min-max problems by use of a penalty method. **IEEE Transactions on Automatic Control**, 26(2):460–466, 1981. ISSN 0018-9286. doi: 10.1109/TAC.1981.1102607.

K. Shimizu and M. Lu. A global optimization method for the Stackelberg problem with convex functions via problem transformation and concave programming. **IEEE Transactions on Systems, Man and Cybernetics**, 25(12):1635–1640, 1995. ISSN 0018-9472. doi: 10.1109/21.478449.

K. Shimizu, Y. Ishizuka, and J. F. Bard. **Nondifferentiable and Two-Level Mathematical Programming**. Kluwer Academic Publishers, 1997. ISBN 978-1-4615-6305-1.

A. Sinha, P. Malo, and K. Deb. Unconstrained scalable test problems for single-objective bilevel optimization. In **2012 IEEE Congress on Evolutionary Computation (CEC)**, pages 1–8, 2012. doi: 10.1109/CEC.2012.6256557.

A. Sinha, P. Malo, A. Frantsev, and K. Deb. Multi-objective Stackelberg game between a regulating authority and a mining company: A case study in environmental economics. In **IEEE Congress on Evolutionary Computation, CEC**, pages 478–485, 2013.

A. Sinha, P. Malo, A. Frantsev, and K. Deb. Finding optimal strategies in a multi-period multi-leader-follower Stackelberg game using an evolutionary algorithm. **Computers & Operations Research**, 41:374–385, 2014.

H. V. Stackelberg. **Marktform und Gleichgewicht.** Springer-Verlag, Berlin, 1934. Tradução para inglês: The Theory of the Market Economy, Oxford University Press, 1952.

R. M. Storn and K. V. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, ICSI, 1995.

D. G. Takemura. Determinação de estratégia de oferta no mercado de energia através de otimização em dois níveis. Master's thesis, Universidade Federal de Santa Catarina, 2007.

E.-G. Talbi, editor. **Metaheuristics for Bilevel Optimization**, volume 482 of **Studies in Computational Intelligence**. Springer-Verlag, 2013. ISBN 978-3-642-37838-6.

G. Ünlü. A linear bilevel programming algorithm based on bicriteria programming. **Computers and Operations Research**, 14:173–179, 1987.

L. F. C. N. Vicente. **Programação de Dois Níveis**. PhD thesis, Universidade de Coimbra, 1992.

V. Visweswaran, C. Floudas, M. Ierapetritou, and E. Pistikopoulos. A decomposition-based global optimization approach for solving bilevel linear and quadratic programs. In C. Floudas and P. Pardalos, editors, **State of the Art in Global Optimization**, volume 7 of **Nonconvex Optimization and Its Applications**, pages 139–162. Springer US, 1996. ISBN 978-1-4613-3439-2. doi: 10.1007/978-1-4613-3437-8-10.

G. Wang, Z. Wan, X. Wang, and Y. Lv. Genetic algorithm based on simplex method for solving linear-quadratic bilevel programming problem. **Computers & Mathematics with Applications**, 56(10):2550–2555, 2008.

J. F. Wang and J. Periaux. Multi-point optimization using gas and nash/stackelberg games for high lift multi-airfoil design in aerodynamics. In

Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, volume 1, pages 552–559 vol. 1, 2001. doi: 10.1109/CEC.2001.934440.

Q. Wang, F. Yang, S. Wang, and L. Yi-Hsin. Bilevel programs with multiple followers. **Journal of Systems Science and Complexity**, 13(3):265, 2000.

W. Wei, Y. Liang, F. Liu, S. Mei, and F. Tian. Taxing strategies for carbon emissions: A bilevel optimization approach. **Energies**, 7(4):2228–2245, 2014.

U. P. Wen and S. T. Hsu. A note on a linear bilevel programming algorithm based on bicriteria programming. **Computers and Operations Research**, 16(1): 79–83, 1989.

Y. Yin. Genetic-algorithms-based approach for bilevel programming models. **Jounal of Transportation Engineering**, 126(2):115–120, 2000.

G. Zhang, C. Shi, and J. Lu. An extended kth-best approach for referential-uncooperative bilevel multi-follower decision making. **International Journal of Computational Intelligence Systems**, 1(3):205–214, 2008.

Y. Zheng, Z. Wan, K. Sun, and T. Zhang. An exact penalty method for weak linear bilevel programming problem. **Journal of Applied Mathematics and Computing**, pages 1–9, 2012. ISSN 1598-5865. doi: 10.1007/s12190-012-0620-6.

X. Zhu, Q. Yu, and X. Wang. A hybrid differential evolution algorithm for solving nonlinear bilevel programming with linear constraints. In **Cognitive Informatics. ICCI 2006, 5th IEEE International Conference on**, volume 1, pages 126–131, 2006. doi: 10.1109/COGINF.2006.365687.

Apêndice A

Dados adicionados às instâncias do PRVMD

Instância Pr. 01

Posicionamento das Fábricas		
53	-199.499	25.4341
54	-122.678	123.496
55	34.0037	-8.05078
56	-59.8834	158.385

Matriz de custo c^2			
4.46852	4.41567	3.05949	4.95474
4.30736	3.5406	3.57004	2.59723
2.81138	3.75141	2.44194	2.98878
5.34348	3.88935	2.84834	2.01401

Matriz de custo c^3			
0.504456	0.68894	0.765831	0.785592
0.800882	0.803583	0.583117	0.831523
0.725394	0.676061	0.52852	0.803842
0.891659	0.901303	0.759941	0.650975

Disponibilidade por Fábrica			
167	46	20	424

Instância Pr. 02

Posicionamento das Fábricas		
101	-199.451	156.652
102	95.3948	17.2308
103	159.923	39.8755
104	-21.894	122.654

Matriz de custo c^2			
2.9801	4.10871	3.08081	4.89204
3.59014	2.05548	5.11026	4.12935
5.30158	4.94674	2.60958	5.21139
3.32187	5.25145	4.29181	2.65618

Matriz de custo c^3			
0.727287	0.857677	0.590976	0.536988
0.953368	0.614628	0.62891	0.654042
0.937971	0.568499	0.799066	0.783639
0.923032	0.61005	0.829814	0.78872

Disponibilidade por Fábrica			
424	16	94	686

Instância Pr. 03

Posicionamento das Fábricas		
149	-199.414	-112.156
150	-86.5444	-89.0225
151	-114.17	87.814
152	16.0955	86.9106

Matriz de custo c^2			
4.49187	3.75845	3.15649	5.07959
2.82962	3.57036	4.95826	2.7861
4.84605	4.44986	2.77721	4.22845
4.55049	3.73818	2.52973	3.29844

Matriz de custo c^3			
0.950117	0.526414	0.916135	0.788385
0.605838	0.925687	0.674703	0.976577
0.650533	0.960952	0.569597	0.763421
0.954405	0.818812	0.899686	0.926466

Disponibilidade por Fábrica			
225	697	424	442

Instância Pr. 04

Posicionamento das Fábricas		
197	-199.377	19.0619
198	131.541	-195.288
199	11.7374	135.752
200	54.0971	51.1795

Matriz de custo c^2			
3.00345	3.02287	2.75828	5.02051
4.68387	2.08515	3.32509	4.68226
3.91664	2.47179	2.94476	3.54815
2.53249	5.46433	4.07047	3.9407

Matriz de custo c^3			
0.672933	0.695166	0.741264	0.539781
0.758324	0.736732	0.720496	0.799097
0.863109	0.853389	0.840144	0.743217
0.985763	0.527558	0.969558	0.564196

Disponibilidade por Fábrica			
217	33	63	2164

Instância Pr. 05

Posicionamento das Fábricas		
245	-199.341	150.267
246	-50.3983	98.4588
247	137.657	183.679
248	92.0866	15.4485

Matriz de custo c^2			
4.51521	5.34163	2.90481	4.98893
3.59228	3.60002	2.25543	2.61287
3.53187	4.05733	3.1124	2.54764
3.54199	3.22492	5.29083	4.58287

Matriz de custo c^3			
0.895764	0.863918	0.566408	0.791192
0.91081	0.547777	0.766289	0.621601
0.575671	0.745827	0.610675	0.723014
0.517121	0.73632	0.539415	0.701941

Disponibilidade por Fábrica			
261	1286	130	1674

Instância Pr. 06

Posicionamento das Fábricas		
293	-199.292	-118.54
294	167.675	-7.79443
295	-136.448	-168.395
296	130.088	-20.2826

Matriz de custo c^2			
3.02689	4.74631	3.03305	4.93666
2.5867	2.1149	4.91065	4.36168
3.12891	3.36757	3.28004	4.80537
4.53088	4.80371	3.76933	2.22504

Matriz de custo c^3			
0.61858	0.532655	0.891552	0.542573
0.56328	0.858837	0.812082	0.944136
0.788263	0.638279	0.881222	0.702795
0.548494	0.945097	0.609287	0.839686

Disponibilidade por Fábrica			
702	1267	437	1265

Instância Pr. 07

Posicionamento das Fábricas		
79	-199.255	12.6774
80	-14.2644	-114.06
81	-10.5289	-120.457
82	168.078	-56.0259
83	138.475	48.3474
84	-141.185	146.8

Matriz de custo c^2					
4.29273	4.47776	3.65954	3.25423	2.27289	4.42568
4.29565	3.42021	3.03897	3.36158	5.19706	3.7904
4.60549	4.01111	3.30018	4.09008	3.7412	3.43225
4.83246	4.12058	2.33112	2.1843	3.36533	3.72695
2.74633	3.51491	3.52155	5.3194	4.17234	4.82371
3.78149	3.53762	5.38256	3.31321	5.33789	4.85205

Matriz de custo c^3					
0.823908	0.671163	0.53029	0.760704	0.864544	0.762368
0.907407	0.807001	0.803034	0.883541	0.726997	0.819407
0.790811	0.529954	0.904828	0.595828	0.548463	0.946272
0.737495	0.904843	0.545122	0.651112	0.640919	0.832942
0.768502	0.689505	0.705985	0.680029	0.851344	0.815668
0.962264	0.730659	0.570147	0.559725	0.990417	0.979446

Disponibilidade por Fábrica					
266	105	85	231	60	201

Instância Pr. 08

Posicionamento das Fábricas		
151	-199.219	143.883
152	-196.203	179.687
153	115.378	-72.5181
154	-193.933	-91.757
155	-59.9689	-26.7159
156	-142.991	147.337

Matriz de custo c^2					
3.31959	2.43775	2.88688	4.60314	5.4495	3.33852
3.9084	2.57396	3.90338	4.45626	3.88547	3.90129
3.13802	4.84283	2.25095	2.72269	4.50855	5.21195
4.86866	2.733	3.73081	4.53902	5.27951	3.2863
3.65574	5.08128	3.79323	3.43668	4.09644	3.585
4.29125	4.14583	4.02108	4.5364	2.81704	2.70965

Matriz de custo c^3					
0.639363	0.515549	0.803705	0.733848	0.675558	0.556124
0.841014	0.825465	0.748024	0.922697	0.611484	0.537309
0.641285	0.860576	0.83253	0.60712	0.72248	0.777413
0.504074	0.61623	0.75367	0.618702	0.844615	0.994552
0.939512	0.537309	0.845012	0.670995	0.538392	0.872509
0.893948	0.510071	0.615253	0.704245	0.677557	0.925275

Disponibilidade por Fábrica					
247	130	568	154	286	621

Instância Pr. 09

Posicionamento das Fábricas		
223	-199.17	-124.912
224	21.8696	73.4214
225	-158.715	-24.5918
226	-155.943	-127.488
227	141.588	-101.767
228	-144.798	147.874

Matriz de custo c^2					
2.34645	4.37038	4.45708	2.98339	4.65727	5.11395
4.4937	4.72781	5.31856	3.41121	4.5058	4.34686
4.01331	3.22533	4.20191	4.10983	3.15511	4.55296
4.93627	5.20579	4.88494	3.89364	3.96715	3.39399
3.59651	2.57938	4.94419	4.3276	4.02054	4.86869
4.66359	5.08864	3.22101	3.30775	2.81867	3.56734

Matriz de custo c^3					
0.954848	0.859951	0.577105	0.706992	0.986587	0.849879
0.774621	0.843944	0.693014	0.961852	0.995987	0.755242
0.991775	0.691183	0.760216	0.618412	0.896496	0.608554
0.770669	0.827631	0.962203	0.586291	0.548296	0.656163
0.610508	0.885128	0.984039	0.661962	0.725455	0.929365
0.825617	0.789499	0.66036	0.848766	0.864711	0.871105

Disponibilidade por Fábrica					
310	261	626	742	102	695

Instância Pr. 10

Posicionamento das Fábricas		
295	-199.133	6.29292
296	-160.057	-32.8318
297	-32.7952	23.3467
298	-117.942	-163.231
299	-56.856	-176.818
300	-146.605	148.412

Matriz de custo c^2					
4.37339	2.40517	3.61679	4.95889	4.02268	3.55368
4.18135	3.88147	2.9209	4.68114	3.4622	4.86559
2.47821	3.79507	3.15268	2.7161	3.29048	3.2383
5.59914	3.99347	3.25829	3.24827	5.13399	3.38049
3.69491	4.41383	4.58388	4.69755	3.94455	3.93659
4.70033	3.1047	4.76535	4.95809	3.60461	4.42503

Matriz de custo c^3					
0.770302	0.704337	0.850536	0.680136	0.797617	0.643635
0.708243	0.862407	0.638005	0.500977	0.880474	0.973159
0.84225	0.521775	0.687918	0.629688	0.570498	0.939711
0.537248	0.539018	0.670721	0.55388	0.751991	0.817774
0.781518	0.732933	0.623051	0.652928	0.912519	0.986221
0.757286	0.568911	0.705466	0.993286	0.551851	0.816919

Disponibilidade por Fábrica					
388	1346	95	852	363	806