ALGORITMO *PROGRESSIVE HEDGING* APLICADO AO PROBLEMA DE GESTÃO DE ATIVOS E PASSIVOS DE FUNDOS DE PENSÃO

EUGÊNIO SILVA REZENDE

ALGORITMO *PROGRESSIVE HEDGING* APLICADO AO PROBLEMA DE GESTÃO DE ATIVOS E PASSIVOS DE FUNDOS DE PENSÃO

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do Centro Federal de Educação Tecnológica de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Modelagem Matemática e Computacional.

Orientadora: Elisangela Martins Sá

Belo Horizonte

Dezembro de 2021

Silva Rezende, Eugênio

D1234p Algoritmo *Progressive Hedging* aplicado ao problema de gestão de ativos e passivos de fundos de pensão / Eugênio Silva Rezende. — Belo Horizonte, 2021

xviii, 82 f. : il. ; 29cm

Dissertação (mestrado) — Centro Federal de Educação Tecnológica de Minas Gerais

Orientador: Elisangela Martins Sá

1. Modelagem Matemática e Computacional — Teses. 2. Otimização — Teses. I. Orientador. II. Título.

CDU 519.6*82.10

Agradecimentos

Gostaria de agradecer à minha família, meus amigos, minha orientadora e todas outras pessoas que me acompanharam nessa jornada de encontros e desencontros que foi o desenvolvimento deste trabalho.

"Nós poderíamos ser muito melhores se não quiséssemos ser tão bons."
(Sigmund Freud)

Resumo

Esta dissertação propõe um algoritmo eficiente para resolver um modelo de programação estocástica de gestão de ativos e passivos de fundos de pensão. Para tanto, foi implementada a versão original do algoritmo *Progressive Hedging* (PH) e outras três diferentes versões desse algoritmo. Essas três versões incluem algumas das diversas melhorias propostas na literatura. As melhorias incluídas ao algoritmo original foram: agrupamento de cenários, fixação de variáveis, atualização do parâmetro de penalidade e paralelização. Para definir a versão mais eficiente, os algoritmos implementados foram usados para resolver um conjunto de instâncias geradas. Os resultados obtidos permitiram concluir que inclusão de melhorias ao PH original afetaram significativamente o desempenho desse algoritmo. Além disso, a melhor versão implementada e o *solver* do CPLEX foram aplicados para resolver um conjunto especial de instâncias. Os resultados mostraram que a melhor versão do PH conseguiu resolver instâncias que não puderam ser resolvidas pelo CPLEX.

Palavras-chave: Algoritmo *Progressive Hedging*, Gestão de Ativos e Passivos, Programação Estocástica, Agrupamento de cenários, Paralelização.

Abstract

This master dissertation proposes an efficient algorithm for solving a stochastic programming model for the Asset and Liability Managment problem of a pension fund. To achieve this goal, the *Progressive Hedging* algorithm was implemented along with three modified versions of it. The three modified versions of the PH algorithm were implemented by combining several improvement strategies proposed in the literature. The improvement strategies used were the following: scenario bundling, variable fixing, parallelization and penalty parameter update. In order to determine the most efficient version, the algorithms were used to solve a set of randomly generated instances of the pension fund's ALM problem. Results show that the inclusion of improvement strategies to the original algorithm signficantly enhance its performance. Furthermore, we compared the efficiency of the most efficient version of *Progressive Hedging* implemented to the CPLEX *solver* for solving large instances. The results obtained show that the most efficient version of the PH was able to solve a set of instances that could not be solved by the CPLEX solver.

Keywords: Progressive Hedging algorithm, Asset and Liability Managment, Stochastic Programming, Parallelization, Scenario bundling.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A gestão de ativos e passivos é uma tarefa importante para diversos tipos de instituições. Em muitos casos, como nas instituições financeiras (por exemplo, bancos, fundos de pensão e companhias de seguro), há riscos envolvendo os ativos e passivos sob gestão [Zenios & Ziemba, 2007]. Para gerir os ativos e passivos considerando esses riscos, podem ser utilizados métodos de gestão de ativos e passivos (do inglês, *Asset Liability Management* - ALM) [Zenios & Ziemba, 2007]. A aplicação de técnicas de ALM tem como objetivo equilibrar o valor dos ativos com o valor dos passivos, considerando-se os riscos envolvidos. Dentre as técnicas utilizadas em ALM, está a aplicação de modelos de programação estocástica.

A programação estocástica é um dos ramos da otimização que lida com problemas que envolvem incertezas [Birge & Louveaux, 2011]. Nos modelos de programação estocástica, as incertezas são representadas por variáveis aleatórias e os valores das variáveis aleatórias constituem cenários [Birge & Louveaux, 2011; Cornuejols & Tütüncü, 2006; Kouwenberg & Zenios, 2008]. Na versão determinística equivalente desses modelos, valores dos parâmetros representados por variáveis aleatórias são incorporados no modelo de otimização como realizações particulares destas. Por meio dessa abordagem, a otimização pode ser feita considerando-se diversos cenários possíveis do problema. Isso permite que cenários extremos sejam considerados no processo de otimização [Zenios & Ziemba, 2007].

No caso de modelos de programação estocástica aplicados em ALM, os valores dos ativos e dos passivos podem ser representados por diferentes cenários para exprimir as incertezas envolvidas [Kouwenberg & Zenios, 2008]. Bradley & Crane [1972] foram pioneiros na aplicação de programação estocástica a um problema de ALM. Posteriormente, outros trabalhos foram desenvolvidos, evidenciando a aplicação da programação estocástica a problemas de ALM [Consigli & Dempster, 1998; Carino et al., 1994; Bo-

ender, 1997].

Em muitos problemas de ALM, o planejamento financeiro envolvendo ativos e passivos deve ser feito para horizontes de tempo, que são, geralmente, divididos em momentos (referidos na literatura de programação estocástica como estágios) nos quais decisões relativas aos ativos e passivos são tomadas (por exemplo, compra ou venda de ativos, contratação de empréstimo) [Consigli & Dempster, 1998; de Oliveira et al., 2017; Zenios & Ziemba, 2007]. O planejamento da gestão de ativos e passivos de um fundo de pensão geralmente é feito considerando-se longos horizontes de tempo. Em determinados momentos desse horizonte de tempo, o gestor do fundo deve tomar decisões que podem acarretar na alteração do balanço entre os valores dos ativos e dos passivos. Uma aplicação da programação estocástica com divisão do horizonte de tempo em estágios em problemas de ALM é apresentada em de Oliveira et al. [2017], em que um modelo para gestão de um fundo de pensão foi proposto e o horizonte de tempo é dividido em 10 estágios.

Para problemas que envolvem longos horizontes de incertezas, os modelos de programação estocástica incorporam variáveis de decisão ligadas a cenários que abrangem todos os estágios nos quais o horizonte de tempo é dividido. Nesse contexto, os cenários são geralmente reunidos numa árvore de cenários (também chamada de árvore de eventos) [Birge & Louveaux, 2011; Zenios & Ziemba, 2007]. A estrutura de árvore exprime justamente o processo de ramificação ao longo do tempo dos possíveis valores que podem ser tomados pelos parâmetros do problema considerados incertos. Na árvore de cenários, os nós associados a cada estágio caracterizam esses valores e sua distribuição de probabilidade [Birge & Louveaux, 2011].

Para resolver problemas de programação estocástica, existem diversos métodos disponíveis [Birge & Louveaux, 2011]. De forma geral, são utilizados métodos de programação linear para resolver esses problemas, pois frequentemente problemas de programação estocástica não-lineares são linearizados com o intuito de se beneficiar de métodos robustos da programação linear [Pichler & Tomasgard, 2016]. No entanto, existem métodos específicos para problemas de programação estocástica não-lineares [Pichler & Tomasgard, 2016].

Para resolver problemas lineares de programação estocástica, diversos métodos de resolução podem ser empregados, incluindo heurísticas e resolvedores de propósito geral. No entanto, ao se examinar a literatura referente à resolução de problemas com vários estágios, dois métodos se destacam pela frequência em que são utilizados: os métodos *Nested-Benders* e *Progressive Hedging* [Birge & Louveaux, 2011; Garcia-Gonzalo et al., 2020; Cornuejols & Tütüncü, 2006; Peng et al., 2019]. Esses métodos são geralmente referidos como métodos de decomposição, pela forma como resolvem o

problema.

Para resolver um problema linear de programação estocástica, o método *Nested-Benders* decompõe a árvore de cenários em vários subproblemas e um problema mestre. Os subproblemas criados pelo *Nested-Benders* resultam de um processo de divisão sucessiva da árvore de cenários em estágios [Birge & Louveaux, 2011].

O método *Progressive Hedging*, diferentemente do método *Nested-Benders*, decompõe a árvore de cenários em cenários individuais. O *Progressive Hedging* é um algoritmo iterativo, que basicamente funciona em duas etapas. Na primeira etapa, cada cenário é associado a um subproblema, que é resolvido de forma independente [Garcia-Gonzalo et al., 2020] de todos os outros. Na segunda etapa, as soluções referentes a todos os cenários são agregadas para satisfazer um conjunto de restrições do problema original, chamadas restrições de *não antecipatividade*. Os desvios das soluções de cada cenário em relação a soluções agregadas são penalizados em cada subproblema.

Neste trabalho, são implementadas e aplicadas diferentes versões do *Progressive Hedging* a um modelo de programação estocástica de um problema de ALM de um fundo de pensão. O método *Progressive Hedging* foi escolhido ao invés do método Nested-Benders porque sua implementação é mais simples. Os resultados da aplicação das diferentes versões do *Progressive Hedging* implementadas neste projeto são comparados. Por fim, a melhor versão do *Progressive Hedging* é comparada com o uso do resolvedor CPLEX.

1.1 Justificativa

Em geral, dado um modelo de programação estocástica, quanto maior for o número de cenários da árvore associada ao modelo, maior será o tempo computacional necessário para a sua resolução [de Oliveira et al., 2017; Consigli & Dempster, 1998; Peng et al., 2019]. Além disso, se torna maior também a quantidade de memória necessária para carregar modelos associados a árvores com muitos cenários. Dessa forma, a resolução de modelos de programação estocástica encontra limites no tamanho das árvores de cenários utilizadas [de Oliveira et al., 2017].

Para lidar com essa dificuldade, algumas abordagens são utilizadas. Uma dessas abordagens consiste em obter árvores de cenários mais compactas, para que o problema seja resolvido de forma mais rápida e necessite de menos memória. Algoritmos de *clusterização*, por exemplo, podem ser aplicados a árvores com muitos cenários, com o objetivo de se obter árvores com quantidades reduzidas desses [Beraldi & Bruni, 2014].

Outra forma de se evitar a utilização de árvores com muitos cenários foi apresentada por de Oliveira et al. [2017], em que foram resolvidos modelos referentes a várias árvores binárias com 10 períodos. Para obter a solução desejada, de Oliveira et al. [2017] calcularam a média sobre as soluções obtidas de todas as árvores.

Uma outra abordagem para tentar contornar essa dificuldade é a utilização de algoritmos mais eficientes para resolver o problema de otimização. Em muitos casos, a utilização de resolvedores de propósito geral, como o CPLEX ou Gurobi, pode se mostrar ineficiente tanto em termos de tempo computacional para resolver o problema quanto em termos de memória consumida [Garcia-Gonzalo et al., 2020; Ryan et al., 2013]. Nesses casos, algoritmos de decomposição, como o *Progressive Hedging*, podem ser alternativas eficientes e com bons resultados em termos de qualidade de solução [Garcia-Gonzalo et al., 2020; Ryan et al., 2013].

1.2 Objetivo geral e específicos

O objetivo principal deste trabalho é propor uma versão do *Progressive Hedging* que consiga resolver de forma eficiente o problema de ALM de um fundo de pensão nos casos em que são utilizadas instâncias com muitos cenários.

Os objetivos específicos são:

- Implementar a versão original do *Progressive Hedging*, como proposta por Rockafellar & Wets [1991], e outras versões, que combinem o algoritmo original com propostas de melhorias já descritas na literatura;
- Comparar as diferentes versões do *Progressive Hedging* implementadas;
- Comparar a versão mais eficiente do *Progressive Hedging* ao resolvedor CPLEX.

1.3 Organização do trabalho

Essa dissertação foi dividida em 7 capítulos. A seguir, encontra-se uma breve descrição desses capítulos.

No Capítulo 2, são introduzidos os conceitos de programação estocástica necessários para a compreensão deste trabalho. Ao final do capítulo, é apresentado o algoritmo *Progressive Hedging*.

No Capítulo 3, faz-se uma revisão da literatura. São abordados os trabalhos que aplicam ALM a fundos de pensão e trabalhos que propõem melhorias ao *Progressive Hedging*.

No Capítulo 4, descreve-se o problema de ALM de um fundo de pensão. Além disso, é introduzido um modelo de programação estocástica para esse problema.

No Capítulo 5, é abordada a metodologia para criação das instâncias utilizadas neste trabalho. Além disso, são apresentadas as diferentes versões do *Progressive Hedging* que foram implementadas.

No Capítulo 6, são apresentados os resultados da comparação entre as diferentes versões do *Progressive Hedging*.

No Capítulo 7, são apresentadas as conclusões e sugestões de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo, é apresentada uma revisão teórica sobre Programação Estocástica. A Seção 2.1 inicia-se com uma introdução à programação estocástica. Posteriormente, são apresentados os problemas de dois e multiestágio. Em seguida, são abordados os conceitos de problema determinístico equivalente, árvores de cenários e restrições de não antecipatividade. Ao final dessa seção, é apresentado o método *Progressive Hedging*.

2.1 Programação estocástica: Introdução

A programação estocástica é uma das áreas da otimização que lida com parâmetros incertos. Problemas de diversas áreas se utilizam da modelagem por programação estocástica. Alguns exemplos são: planejamento financeiro, agricultura, saúde, energia, entre outros. Na área da energia, por exemplo, Wallace & Fleten [2003] apresentam diferentes modelos de programação estocástica aplicados a essa área. Dong et al. [2014] apresentam um modelo para utilização de recursos hídricos na agricultura. Birge & Louveaux [2011] exploram a modelagem de diferentes problemas: planejamento financeiro, roteamento de veículos, expansão de capacidade, entre outros.

Os problemas modelados por programação estocástica geralmente envolvem tomada de decisões sob incertezas em relação ao futuro [Birge & Louveaux, 2011]. Nesses problemas, decisões são tomadas "agora"e aguarda-se um momento futuro, independente das incertezas que persistem sobre os parâmetros do problema. Por essa característica, as decisões são chamadas de não antecipatórias.

Há na literatura uma clássica divisão dos problemas de programação estocástica de acordo com o número de estágios: problemas de dois estágios com recurso e os problemas multiestágio [Shapiro et al., 2014],[Birge & Louveaux, 2011]. Nos problemas

de dois estágios com recurso, tomam-se decisões no primeiro estágio e se espera o momento, em que alguns do parâmetros incertos sejam conhecidos, para que um segundo conjunto de decisões seja tomado. Nos caso dos problemas multiestágio, as decisões tomadas em vários estágios são intercaladas por momentos de incerteza.

Para um aprofundamento na teoria de programação estocástica, indica-se a leitura de Birge & Louveaux [2011], Shapiro et al. [2014], e King & Wallace [2012], nas quais são explorados algoritmos de resolução, aspectos detalhados da modelagem, entre outros temas.

2.2 Problemas de dois estágios com recurso

Os problemas de programação estocástica em dois estágios com recurso são caracterizados por dois tipos de decisão [Birge & Louveaux, 2011]:

- Decisões de primeiro estágio: são decisões tomadas no primeiro estágio, antes que o resultado de um evento aleatório seja revelado;
- Decisões de segundo estágio (decisões de recurso): são decisões tomadas no segundo estágio, após se saber o resultado do evento aleatório.

De acordo com Cornuejols & Tütüncü [2006], a formulação genérica de um problema de programação estocástica de dois estágios com recurso pode ser dada por

$$\min_{x_0 \in X_0} \quad f(x_0) + \mathbb{E}[Q(x_1, \omega_1)], \tag{2.1}$$

no qual, para o primeiro estágio, x_0 é um vetor de variáveis de decisão, f é a função objetivo e X_0 é o conjunto de decisões factíveis. O argumento ω_1 que aparece em Q é uma variável aleatória. O termo $Q(x_1, \omega_1)$, chamado termo de recurso, é definido por

$$Q(x_1, \omega_1) := \min_{x_1 \in X_1} \quad g(x_1, \omega_1), \tag{2.2}$$

no qual, para o segundo estágio, $x_1(\omega_1)$ são as variáveis de decisão (também chamadas de decisões de recurso), X_1 é o conjunto de decisões factíveis nesse estágio e g é a função objetivo.

No primeiro estágio, os parâmetros do problema que definem o conjunto X_0 são conhecidos. Portanto, nesse estágio não há incertezas sobre as decisões. No entanto, existem incertezas em relação à região viável X_1, pois os parâmetros do segundo estágio são considerados incertos nesse momento. Entre o primeiro e segundo estágio, a realização da variável aleatória ω_1 permite que os parâmetros do segundo estágio sejam

determinados. Esses parâmetros e o vetor de decisões x_0 determinam o conjunto das decisões factíveis X_1 e as incertezas em relação às decisões do segundo estágio acabam.

O processo de tomada de decisão do problema (2.1) pode ser ilustrado da seguinte forma:

$$\text{decisão } \ x_0 \rightarrow \text{resultado de } \ \omega_1 \rightarrow \text{decisão de recurso } \ x_1$$

em que x_0 e x_1 são as variáveis de decisão que representam, respectivamente, as decisões do primeiro e segundo estágio.

2.3 Problemas multiestágio

Os problemas de programação estocástica multiestágios podem ser entendidos como uma extensão dos problemas de dois estágios com recurso. Eles são caracterizados por decisões de primeiro estágio e decisões sequenciais de recurso. Em um problema de $T+1$ estágios, as decisões de primeiro estágio são seguidas por T decisões de recurso, que ocorrem sequencialmente. Entre cada par de estágios consecutivos, o resultado de um evento aleatório é revelado.

Segundo Cornuejols & Tütüncü [2006], a formulação genérica de um problema multiestágio de $T+1$ estágios é dada por

$$\min_{x_0 \in X_0} f_0(x_0) + \mathbb{E}\left[\min_{x_1 \in X_1} f_1(x_1, \omega_1) + ... + \mathbb{E}\left[\min_{x_T \in X_T} f_T(x_T, \omega_T)\right]...\right]. \tag{2.3}$$

no qual representa-se por x_t as variáveis de decisão, f_t a função objetivo e X_t o conjunto de decisões factíveis, para os estágios $t = 0, ..., T$. O argumento ω_t é a variável aleatória cujo resultado é revelado entre os estágios $t-1$ e t, para $t = 1, ..., T$.

No primeiro estágio, somente os parâmetros desse estágio são conhecidos. Portanto, nesse momento se conhece somente a região viável X_0, e os parâmetros dos estágios $i = 1, ..., T$ são incertos. O conhecimento de X_0 permite escolher o vetor de decisões do primeiro estágio.

Entre o primeiro e o segundo estágio, o valor da variável aleatória ω_1 é revelado. A partir do valor de ω_1, os parâmetros do segundo estágio são determinados. Com esses parâmetros e o vetor de decisões do primeiro estágio, o conjunto X_1 fica determinado e pode-se escolher o vetor de decisões do segundo estágio.

De forma geral, no i-ésimo estágio, somente estão disponíveis as informações sobre os parâmetros e decisões dos estágios $0, 1, ..., i-1$. A realização da variável aleatória ω_i determina os parâmetros do estágio i. Com a informação das decisões dos estágio anteriores, determina-se X_i e as decisões do estágio i podem ser tomadas.

O processo de tomada decisão do problema (2.3) pode ser ilustrado da seguinte forma:

$$\text{decisão } x_0 \rightarrow \text{resultado de } \omega_1 \rightarrow \text{decisão } x_1 \rightarrow \text{resultado de } \omega_2 \rightarrow ... \rightarrow \text{decisão } x_T$$

em que x_0, ..., x_T são as variáveis de decisão que representam as decisões tomadas no primeiro até o T-ésimo estágio.

2.4 Problema determinístico equivalente

Considere o seguinte problema de programação estocástica linear de dois estágios com recurso, abordado por Kouwenberg & Zenios [2008]:

$$\begin{aligned} \min \quad & f(x_0) + \mathbb{E}[Q(x_0, \omega_1)] \\ \text{sujeito a:} \quad & Ax_0 = b \\ & x_0 \in X_0 \end{aligned} \tag{2.4}$$

O termo de recurso $Q(x_0, \omega)$ é dado por

$$\begin{aligned} Q(x_0, \omega_1) := \min \quad & q(x_1, \omega_1) \\ \text{sujeito a:} \quad & W(\omega_1)x_1 = h(\omega_1) - T(\omega_1)x_0 \\ & x_1(\omega_1) \in X_1 \end{aligned} \tag{2.5}$$

No Problema 2.4, os parâmetros b e A são conhecidos já no primeiro estágio. Por sua vez, os parâmetros $T(\omega_1)$, $W(\omega_1)$ e $h(\omega_1)$ são incertos. No modelo do problema, ω_1 é uma variável aleatória, cujo suporte [1] é o conjunto Ω_1. Além disso, os valores da realização de ω_1 determinam o vetor de parâmetros ($W(\omega_1)$, $T(\omega_1)$, $h(\omega_1)$).

Suponha que Ω_1, que representa eventos aleatórios que ocorrem no primeiro estágio, seja um conjunto finito, tal que $\Omega_1 = \{\omega_1^1, \omega_1^2, ..., \omega_1^K\}$. Suponha também que cada ω_1^i determine um único vetor $(W(\omega_1^i), T(\omega_1^i), h(\omega_1^i))$ de parâmetros, com $i = 1, 2, ..., K$. Seja x_1^i o vetor que representa as variáveis de decisão do segundo estágio quando ocorre

[1] O conjunto S é o suporte de uma variável aleatória X quando todos os valores possíveis de X estão em S.

ω_1^i. Nesse caso, para cada ω_1^i, tem-se um problema do tipo (2.5), isto é, temos

$$\begin{aligned} \min \quad & q(x_1, \omega_1^i) \\ \text{sujeito a:} \quad & W(\omega_1^i) = h(\omega_1^i) - T(\omega_1^i)x_0 \\ & x_1^i \in X_1 \end{aligned} \tag{2.6}$$

Seja p^i a probabilidade de ocorrência do vetor de parâmetros $(W(\omega_1^i), T(\omega_1^i), h(\omega_1^i))$. De acordo com Kouwenberg & Zenios [2008], a partir do problema (2.6) e das probabilidades $p^1, p^2, ..., p^K$, pode-se reformular o problema (2.4) como

$$\begin{aligned} \min \quad & f(x_0) + \sum_{i=1}^{K} p^i q(x_1^i, \omega_1^i) \\ \text{sujeito a:} \quad & Ax_0 = b \\ & W(\omega_1^i)x_1^i = h(\omega_1^i) - T(\omega_1^i)x_0 \quad i = 1, ..., K \\ & x_0 \in X_0, x_1^i \in X_1 \qquad\qquad\quad i = 1, ..., K \end{aligned} \tag{2.7}$$

O problema (2.7) é a versão determinística de (2.4). Por essa razão, a expressão (2.7) é chamada de problema determinístico equivalente.

A versão multiestágio do problema (2.4) é uma extensão na qual consideramos $T + 1$ estágios (Kouwenberg & Zenios [2008]). O modelo do problema multiestágio é dado por

$$\begin{aligned} \min \quad & f(x_0) + \mathbb{E}[\min \quad q(x_1, \omega_1) + ... + \mathbb{E}[\min \quad q(x_T, \omega_T)]...] \\ \text{sujeito a :} \quad & Ax_0 = b, \\ & W(\omega_1)x_1 = h(\omega_1) - T(\omega_1)x_0 \\ & \vdots \qquad\quad \vdots \qquad\quad \vdots \\ & W(\omega_T)x_T = h(\omega_T) - T(\omega_T)x_{T-1} \\ & x_i \in X_i \qquad i = 0, ...T. \end{aligned} \tag{2.8}$$

Seja $(\omega_1, ..., \omega_T)$ o vetor que representa os eventos aleatórios que ocorrem nos estágios 1 até o T e $\Omega_1 \times \Omega_2 \times ... \times \Omega_T$ o seu suporte, sendo Ω_t o suporte de ω_t para $t = 1, ..., T$. Se $\Omega_1 \times \Omega_2 \times ... \times \Omega_T$ é finito, por argumento análogo ao utilizado para derivar (2.7), a formulação da versão determinística do problema multiestágio pode ser dada por

$$
\begin{aligned}
\min \quad & f(x_0) + \sum_{i=1}^{K^1} p^i_{K^1} q(x_1, \omega_1) + ... + \sum_{i=1}^{K^T} p^i_{K^T} q(x_T, \omega_T) \\
\text{sujeito a :} \quad & Ax_0 = b, \\
& W(\omega^i_1)x_1 = h(\omega^i_1) - T(\omega^i_1)x_0 \qquad i = 1, ..., K^1 \\
& \vdots \qquad\qquad\qquad \vdots \\
& W(\omega^1_T)x_T = h(\omega^1_T) - T(\omega^1_T)x_{T-1} \quad i = 1, ..., K^T \\
& x_i \in X_i \qquad i = 0, ..., T.
\end{aligned} \tag{2.9}
$$

no qual K^j é a cardinalidade de ω_j e $p^i_{K^j}$ é a probabilidade do evento ω^i_j.

2.5 Árvore de cenários

Para representar os diferentes valores de parâmetros em um problema determinístico equivalente, pode ser utilizada uma estrutura do tipo árvore. A seguir, é abordado como essas estruturas são aplicadas para representar os parâmetros de um problema determinístico equivalente.

Considere um problema de programação estocástica de $T + 1$ estágios. Seja ω_t a variável aleatória cujo resultado é revelado entre os estágios $t - 1$ e t; e Ω_t seu suporte, para $t = 0, ..., T$. Considera-se Ω_0 como um conjunto de um único elemento. Logo, o suporte do vetor $(\omega_0, \omega_1, ..., \omega_T)$ é $\Omega_0 \times \Omega_1 \times ... \times \Omega_T$. Um *cenário* é definido como um elemento do suporte $\Omega_0 \times \Omega_1 \times \Omega_2 \times ... \times \Omega_T$ (Shapiro et al. [2014]). Um cenário, portanto, é uma sequência particular de revelações dos eventos aleatórios que acontecem do estágio 0 ao T. No caso em que $\Omega_0 \times \Omega_1 \times ... \times \Omega_T$ é finito, pode-se representar todos os possíveis cenários numa estrutura de árvore, que é chamada de árvore de cenários (Shapiro et al. [2014]).

A árvore de cenários é dividida em $T+1$ níveis. No primeiro nível, existe somente o nó raiz, que corresponde aos parâmetros do primeiro estágio. Do nó raiz, ramificam-se n_1 nós do segundo nível, em que n_1 corresponde a todos os possíveis resultados de ω_1. Os nós do segundo nível representam todos os possíveis parâmetros do segundo estágio. De cada um dos nós do segundo nível, ramificam-se n_2 nós do terceiro nível, em que n_2 corresponde a todos os possíveis resultados de ω_2. O processo de ramificação continua dessa forma, até que se atinja o $(T + 1)$-ésimo nível da árvore. A Figura 2.1 ilustra a estrutura de uma árvore de três níveis.

Na árvore da Figura 2.1, o nó s_0 representa o nó raiz, os nós s_1, s_2 e s_3 representam os nós do segundo estágio e os nós restantes são do terceiro estágio. No nó raiz, embora

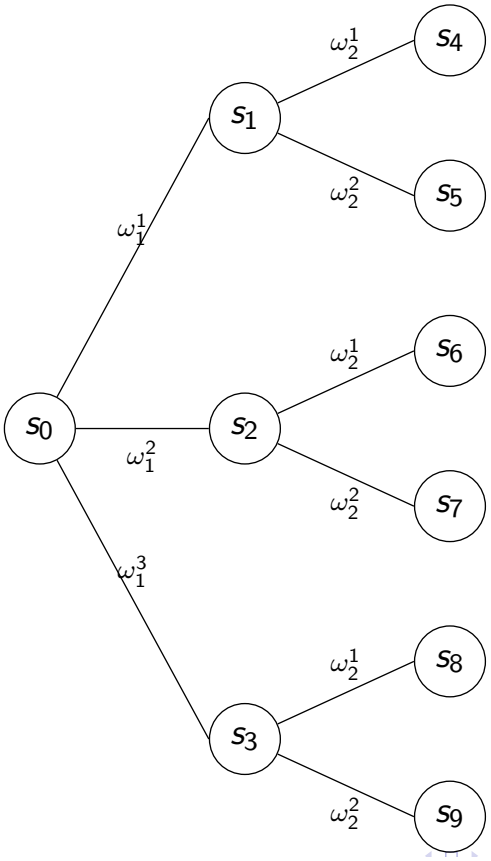

Figura 2.1: Árvore de cenários com três estágios

não mostrado na figura, sabe-se o valor de ω_0, variável que pode tomar somente um único valor. Partindo do nó raiz, ω_1 pode tomar três valores diferentes. Partindo dos nós do segundo nível, ω_2 pode assumir dois valores distintos. Portanto, a árvore possui seis cenários: $(\omega_0, \omega_1^1, \omega_2^1)$, $(\omega_0, \omega_1^1, \omega_2^2)$, $(\omega_0, \omega_1^2, \omega_1^2)$, $(\omega_0, \omega_1^2, \omega_2^2)$, $(\omega_0, \omega_1^3, \omega_1^2)$ e $(\omega_0, \omega_1^3, \omega_2^2)$.

Na construção da árvore, a cada nó gerado é atribuída uma probabilidade. A atribuição de probabilidade acontece da seguinte forma: seja s_t um nó do t-ésimo nível e ω_t a variável aleatória que representa o evento que ocorre entre os estágios t e $t-1$ e cujo suporte é o conjunto $\Omega_t = \{\omega_t^1, ..., \omega_t^M\}$, em que M é a cardinalidade de Ω_t. Se s_t é o nó associado ao resultado $\omega_t^i \in \Omega_t$, a probabilidade de s_t, P_{s_t}, é dada por

$$P_{s_t} = P_{s_{t-1}} P(\omega_t^i), \tag{2.10}$$

em que $P_{s_{t-1}}$ é a probabilidade do nó do qual se origina s_t e $P(\omega_t^i)$ é a probabilidade de ω_t^i. Ao nó raiz é atribuída probabilidade 1. Na Figura 2.2, apresenta-se uma ilustração de uma árvore de três níveis, com probabilidades associadas ao seus nós.

Na árvore da Figura 2.2, há para ω_1 somente dois valores possíveis, ω_1^1 e ω_1^2. As

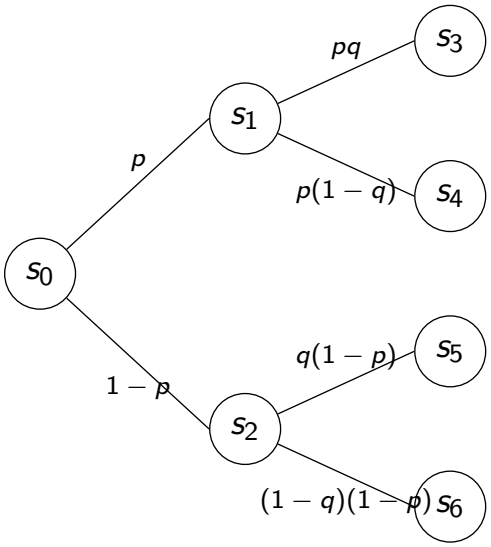

Figura 2.2: Árvore de cenários com três estágios e probabilidades associadas a cada ramificação possível.

probabilidades associadas a ω_1 são $P(\omega_1^1) = p$ e $P(\omega_1^2) = 1 - p$. Para o parâmetro ω_2 há também somente dois valores possíveis, ω_2^1 e ω_2^2. A probabilidades dos eventos de ω_2 são $P(\omega_2^1) = q$ e $P(\omega_2^2) = 1 - q$. Na figura, as probabilidades de cada nó da árvore aparecem nos arcos associados a aqueles.

O número de cenários de uma árvore cresce exponencialmente à medida que se considera mais estágios. Esse fator impõe limites no uso destas estruturas. De fato, em muitos problemas reais, pode ser inviável (ou até mesmo impossível, no caso de distribuições contínuas) a geração de árvores que contenham todos os cenários possíveis. Nesses casos, uma forma de lidar com essa dificuldade é gerar árvores com cenários que aproximem a distribuição de probabilidade de $(\omega_1, ..., \omega_T)$. Na literatura, pode-se encontrar métodos desenvolvidos para esse fim. Xu et al. [2012] abordam um algoritmo de clusterização, que permite obter árvores compactas a partir de muitos cenários gerados de forma independente. Høyland & Wallace [2001] apresentam um método que possibilita a obtenção de árvores com quantidade de cenários reduzidos. As árvores obtidas preservam os momentos observados da distribuição de $(\omega_1, ..., \omega_T)$.

2.6 Restrições de não antecipatividade

Em uma árvore de cenários, cada nó representa um conjunto de parâmetros incertos do problema de programação estocástica que foram revelados após o fim de um período de incertezas. Cada nó também representa um momento em que decisões

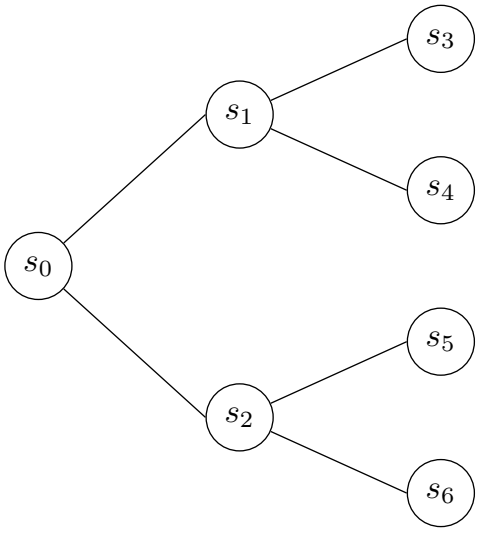

Figura 2.3: Árvore de cenários de um problema de três estágios

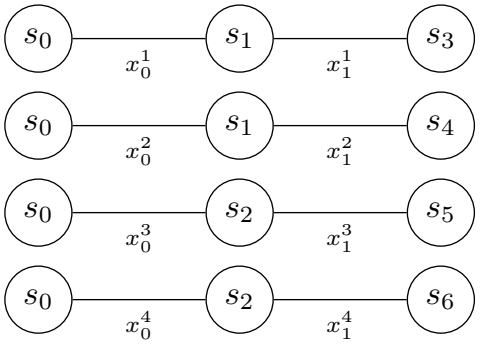

Figura 2.4: Decomposição em cenários e atribuição de variáveis de mesmo estágio para cada cenário.

devem ser tomadas com base nas informações disponíveis, que correspondem aos parâmetros do nó em questão e dos nós que o antecedem. Além disso, as decisões não podem se basear na antecipação de valores que os parâmetros incertos possam assumir no futuro. Na literatura da área, essas características recebem o nome de *não antecipatividade*.

Certas formulações da forma determinística equivalente decompõe a árvore de cenários em cenários individuais. À cada estágio de um cenário são associadas variáveis que representam as decisões tomadas tão logo os parâmetros do estágio sejam revelados. As Figuras 2.3 e 2.4 ilustram essa ideia.

A árvore da Figura 2.3 é decomposta em cenários, resultando nos cenários individuais mostrados na Figura 2.4. Nessa figura, x_i^j designa a variável associada ao estágio i e cenário j, para $i = 0, 1$ e $j = 1, 2, 3, 4$. A partir da Figura 2.4, pode-se observar que as variáveis relacionadas a cada estágio são copiadas para cada um dos cenários. As

formulações desse tipo são referidas como formulações baseadas em cenários.

Nas formulações baseadas em cenários, a não antecipatividade das decisões deve ser incluída na forma de restrições. Dessa forma, variáveis que representam uma ligação entre dois nós específicos da árvore de cenários devem ter o mesmo valor. Na Figura 2.4, x_0^1 e x_0^2, por exemplo, ligam os mesmos nós (S_0 e S_1). A não antecipatividade então determina que x_0^1 e x_0^2 devem tomar o mesmo valor. Tomando novamente como exemplo a árvore da Figura 2.3, se a formulação do problema representado por essa árvore é baseada em cenários, deve-se incluir explicitamente na formulação as restrições

$$\begin{aligned} x_0^1 &= x_0^2 \\ x_0^2 &= x_0^3 \\ x_0^3 &= x_0^4 \end{aligned} \tag{2.11}$$

Essas restrições são chamadas de restrições de não antecipatividade.

2.7 *Progressive Hedging*

Existem várias formas de se resolver um problema de programação estocástica na sua forma determinística equivalente. Uma delas é a aplicação direta de resolvedores de propósito geral à formulação completa do problema, envolvendo todas as variáveis e restrições. Uma outra forma de se resolver envolve a aplicação de algoritmos que decompõem a formulação em subproblemas de mais fácil resolução. O algoritmo *Nested-Benders* [Birge & Louveaux, 2011] divide o problema de programação estocástica em estágios, resolvendo recursivamente subproblemas do nó folha ao nó raiz da árvore de cenários. Outro algoritmo também aplicado a problemas de programação estocástica é o *Progressive Hedging* (PH), proposto originalmente por Rockafellar & Wets [1991].

O algoritmo PH decompõe a formulação original em problemas menores, mais fáceis de resolver. No entanto, o tipo de decomposição é diferente em relação ao *Nested-Benders*: no PH, o problema original é decomposto em subproblemas, em que cada subproblema é formado pelos parâmetros de um único cenário. Os subproblemas associados a cada um dos cenários são resolvidos de forma independente. As restrições de não antecipatividade, que ligam as variáveis de cenários diferentes na formulação original, são incluídas na função objetivo dos subproblemas por meio de uma relaxação lagrangiana. As soluções dos subproblemas que violam essas restrições penalizam o valor da função objetivo.

O PH é um algoritmo iterativo. Para problemas lineares, a cada iteração do algoritmo, as soluções dos subproblemas violam cada vez menos as restrições de não antecipatividade. Além disso, sabe-se que, para essa classe de problemas, as soluções dos subproblemas convergem de forma a satisfazerem as restrições de não antecipatividade. Uma prova da convergência desse algoritmo é dada por Rockafellar & Wets [1991]. Para problemas inteiros mistos, no entanto, o PH é utilizado como heurística, sem garantia que as soluções convirjam para satisfazer essas restrições. A seguir, apresenta-se o passo a passo do PH aplicado a um problema de programação estocástica, seguindo a exposição de Zehtabian & Bastin [2016].

Considere o problema de programação estocástica de $T+1$ estágios, cuja formulação é

$$\min_{x \in X(\omega)} \quad \mathbb{E}[f(x, \omega)] \tag{2.12}$$

Os cenários do problema (2.12) são gerados pelo vetor aleatório $\omega = (\omega_0, ..., \omega_T)$, cujo suporte Ω tem cardinalidade K. Representa-se por $\bar{\omega} = (\bar{\omega}_0, ..., \bar{\omega}_T)$ uma realização particular da variável aleatória ω, ou seja, $\bar{\omega}$ é um cenário. Para cada $\bar{\omega}$, define-se por $x^{\bar{\omega}} = (x_0^{\bar{\omega}}, x_1^{\bar{\omega}}, ..., x_T^{\bar{\omega}})$ o vetor de variáveis de decisão do estágio 0 ao T e $\bar{x}_t^{\bar{\omega}} = (x_0^{\bar{\omega}}, x_1^{\bar{\omega}}, ..., x_t^{\bar{\omega}})$ o vetor de variáveis de decisão do estágio 0 ao t, para algum $t < T$. A partir das definições feitas, a forma determinística equivalente do problema (2.12) pode ser dada por

$$\min \quad \sum_{\bar{\omega} \in \Omega} p_{\bar{\omega}} f(x^{\bar{\omega}}, \bar{\omega}) \tag{2.13}$$

$$\text{sujeito a} \quad x^{\bar{\omega}} \in X(\bar{\omega}), \quad \bar{\omega} \in \Omega \tag{2.14}$$

$$x_t^{\bar{\omega}} \quad \text{é não antecipativo}, \quad t = 0, ..., T-1, \quad \bar{\omega} \in \Omega. \tag{2.15}$$

em que $p_{\bar{\omega}}$ é a probabilidade do cenário $\bar{\omega}$ e $X(\bar{\omega})$ é a região viável para esse cenário.

Para aplicar o PH ao problema (2.13)-(2.15), é necessário que as restrições de não antecipatividade sejam reformuladas, como proposto por Rockafellar & Wets [1991]. Isto pode ser feito da seguinte forma. Seja:

$$S_t(\bar{\omega}) = \{\bar{\omega}' | \bar{\omega}_r' = \bar{\omega}_r, r = 0, ..., t\}$$

o conjunto dos cenários que compartilham os mesmos valores de parâmetros do estágio 0 ao estágio t com o cenário $\bar{\omega}$. Segundo Rockafellar & Wets [1991], as restrições de não antecipatividade (2.15) podem ser formuladas como

$$x_t^{\bar{\omega}} = \hat{x}_t^{\bar{\omega}}, \quad t = 0, 1, ..., T-1, \quad \bar{\omega} \in \Omega, \tag{2.16}$$

em que

$$\hat{x}_t^{\bar{\omega}} = \frac{\sum_{\bar{\omega}' \in S_t(\bar{\omega})} p_{\bar{\omega}'} x_t^{\bar{\omega}'}}{\sum_{\bar{\omega}' \in S_t(\bar{\omega})} p_{\bar{\omega}'}}. \tag{2.17}$$

Tomando como exemplo o problema representado pela árvore da Figura 2.3 e as equações (2.16) e (2.17), as restrições de não antecipatividade podem ser reescritas como

$$\hat{x}_0^j = \frac{p_1 x_0^1 + p_2 x_0^2 + p_3 x_0^3 + p_4 x_0^4}{p_1 + p_2 + p_3 + p_4}, \quad j \in \{1, 2, 3, 4\} \tag{2.18}$$

em que p_1, p_2, p_3, p_4 são respectivamente as probabilidades dos cenários 1, 2, 3 e 4.

Uma versão padrão do algoritmo PH é ilustrada no Algoritmo 1. No algoritmo apresentado, o passo 1 corresponde à fase de inicialização deste. Nesse passo, para cada subproblema, são calculadas as restrições de não antecipatividade e também são definidos os vetores de multiplicadores lagrangianos. É também nesse passo que se escolhe o valor de ρ_0, o fator de penalidade inicial.

No passo 2 do algoritmo, resolvem-se os subproblemas associados aos cenários. No passo 3, as soluções de não antecipatividade são calculadas para cada cenário e estágio t do problema. No passo 4, para cada cenário, os valores dos multiplicadores lagrangianos são atualizados. Por fim, no passo 5, o critério de parada do algoritmo é verificado.

Com relação a detalhes do algoritmo, existem formas diferentes de se escolher a solução inicial $x^{\bar{\omega},0} = (x_1^{\bar{\omega},0}, ..., x_T^{\bar{\omega},0})$. De acordo com Zehtabian & Bastin [2016], uma estratégia comumente usada é resolver o problema

$$\min_{x^{\bar{\omega}} \in X(\bar{\omega})} f(x^{\bar{\omega}}, \bar{\omega}) \tag{2.19}$$

ou seja, resolve-se cada subproblema sem a adição das restrições de não antecipatividade à função objetivo.

O PH é um algoritmo que apresenta diferentes versões quanto ao critério de parada e escolha do fator de penalidade. Existem diferentes critérios de parada propostos. Rockafellar & Wets [1991] propuseram o seguinte critério de parada:

$$\sqrt{\sum_{\bar{\omega} \in \Omega} p_{\bar{\omega}} \left\| x^{\bar{\omega},k+1} - \hat{x}^{\bar{\omega},k} \right\|_2^2} < \epsilon, \tag{2.24}$$

em que ϵ é um número real arbitrário escolhido *a priori*.

Além disso, a escolha do valor inicial do fator de penalidade ρ e sua atualização são

Algoritmo 1: *Progressive Hedging*

1: Inicialização

(i) Defina inicialmente $k = 0$ e compute o vetor $\hat{x}^{\bar{\omega},k} = (\hat{x}_0^{\bar{\omega},k}, ..., \hat{x}_{T-1}^{\bar{\omega},k})$ para cada cenário $\bar{\omega} \in \Omega$.

(ii) Inicialize para cada $\bar{\omega}$ o vetor de multiplicadores lagrangianos $\lambda^{\bar{\omega},k} = (\lambda_0^{\bar{\omega},k}, ..., \lambda_{T-1}^{\bar{\omega},k}) = (0_0, ..., 0_{T-1})$.

(iii) Escolha o fator de penalidade $\rho^k > 0$.

2: Para cada $\bar{\omega} \in \Omega$, resolva o subproblema

$$\min f(x^{\bar{\omega}}, \bar{\omega}) + \sum_{t=0}^{T-1} \left(\lambda_t^{\bar{\omega},k}(x_t^{\bar{\omega}} - \hat{x}_t^{\bar{\omega},k}) + \frac{\rho^k}{2} \left\| x_t^{\bar{\omega}} - \hat{x}_t^{\bar{\omega},k} \right\|^2 \right) \tag{2.20}$$

$$\text{sujeito a} \quad x^{\bar{\omega}} \in X(\bar{\omega}) \tag{2.21}$$

e a solução será $x^{\bar{\omega},k+1} = (x_0^{\bar{\omega},k+1}, ..., x_T^{\bar{\omega},k+1})$.

3: Para cada $\bar{\omega} \in \Omega$, $t = 0, ..., T-1$, compute

$$\hat{x}_t^{\bar{\omega},k+1} = \frac{\sum_{\bar{\omega}' \in S_t(\bar{\omega})} p_{\bar{\omega}'} x_t^{\bar{\omega}',k+1}}{\sum_{\bar{\omega}' \in S_t(\bar{\omega})} p_{\bar{\omega}'}}. \tag{2.22}$$

4: Atualize ρ^{k+1} e compute para cada $\bar{\omega} \in \Omega$, $t = 0, ..., T-1$,

$$\lambda_t^{\bar{\omega},k+1} = \lambda_t^{\bar{\omega},k} + \rho^k \left(x_t^{\bar{\omega},k+1} - \hat{x}_t^{\bar{\omega},k+1}\right). \tag{2.23}$$

5: Pare o algoritmo se foi atingida a convergência, de acordo com o critério de parada. Do contrário, volte ao passo 2 e incremente k.

fatores importantes para a convergência do algoritmo. Existem na literatura diferentes estratégias de atualização de ρ [Zehtabian & Bastin, 2016]. Muitas dessas estratégias dependem do problema ao qual o algoritmo é aplicado [Zehtabian & Bastin, 2016]. Para certos problemas, algumas heurísticas de atualização de ρ se mostraram bem sucedidas [Zehtabian & Bastin, 2016].

Capítulo 3

Trabalhos Relacionados

Neste capítulo são apresentados trabalhos considerados relevantes e relacionados ao trabalho desenvolvido nesta dissertação. Na seção 3.1, são abordados os trabalhos que tratam do problema de ALM aplicado a fundos de pensão. Na seção 3.2, são abordados os trabalhos que propuseram modificações ao PH.

3.1 ALM e fundos de pensão

A seguir, são apresentados trabalhos que envolvem o problema de ALM aplicado a fundos de pensão.

Gondzio & Kouwenberg [2001] apresentam um método de geração de cenários e propuseram o algoritmo *L-shaped* paralelo para resolver um modelo de ALM de um fundo de pensão, em um ambiente de computação de alto desempenho. Os autores geraram árvores com ramificação constante para 6 estágios. O número mínimo de cenários ao qual o modelo foi aplicado foi de 4.096 (ramificação de 4 nós por estágio) e o máximo, 4.826.809 (ramificação de 13 nós por estágio). Os autores mostraram, ao final do trabalho, a eficiência do *L-shaped* paralelo utilizando números variados de processadores para resolver os subproblemas obtidos na decomposição.

Valladão & Veiga [2008] propuseram um modelo de ALM para um fundo de pensão e um método para calcular o equilíbrio de risco. Lauria & Consigli [2017] também propuseram um modelo de ALM para um fundo de pensão. O modelo proposto por Lauria & Consigli [2017] é adaptado para o caso em que o valor da aposentadoria do contribuinte é definido a priori (*defined benefit plan*).

Kouwenberg & Zenios [2008] desenvolveram e testaram três métodos de geração de cenários para o problema de ALM de um fundo de pensão: *Random sampling*, *Adjusted Random Sampling* e *Tree Fitting*. Além disso, os autores propuseram um modelo de

ALM para um fundo de pensão. Para comparar os métodos de geração de cenários, os autores resolveram o modelo aplicado a uma árvore de 5.760 cenários gerada usando cada um dos métodos. Ao comparar os resultados obtidos para os métodos, os autores mostraram que *Adjusted Random Sampling* e *Tree Fitting* são superiores ao *Random sampling*.

Um modelo de ALM para um fundo de pensão, baseado em programação estocástica, foi apresentado por de Oliveira et al. [2017]. Eles utilizaram este modelo para determinar o planejamento das alocações de recursos. Para determinar essas alocações, os autores primeiramente resolveram o modelo proposto para um número determinado de árvores binárias com 10 estágios. Assim, obtiveram para cada árvore as alocações ótimas considerando os cenários desta. Posteriormente, calculou-se o valor médio das alocações, considerando todas as árvores. Os autores justificaram a utilização dessa estratégia, de resolver o problema para várias árvores e depois obter-se os valores médios, pela dificuldade de resolver o modelo para árvores grandes (árvores com mais nós por estágio e também com mais estágios).

Em de Oliveira et al. [2018], são comparados três métodos de geração de árvore de cenários resolvendo um modelo de ALM e analisando a estabilidade do valor da função objetivo para as árvores geradas por cada um desses métodos. Os métodos comparados foram: *Monte Carlo sampling*, *Moment Matching* e *Resampled Average Approximation*. Nesse trabalho, os autores resolveram o modelo aplicado a árvores que são divididas em três grupos: *small* (árvores de 2.187 cenários), *medium* (árvores de 6.561 cenários) e *large* (árvores de 11.664 cenários). As árvores de cada um desses grupos têm diferentes topologias, ou seja, diferentes formas de ramificação. As árvores do grupo *small*, por exemplo, têm 1 nó no estágio 0, 27 nós no estágio 1, 9 nós no estágio 2 e 9 nós do estágio (perfazendo, portanto, os 2.187 cenários). Ao final do estudo, os autores mostraram que os métodos *Moment Matching* e *Resampled Average Approximation* são superiores ao *Monte Carlo sampling* em termos de estabilidade de valor da função objetivo.

Um *framework* que permite a obtenção de árvores de cenários e modelos de ALM que incluem as restrições de não antecipatividade de forma implícita, foi apresentado por de Oliveira & Filomena [2020]. Usando o *framework* desenvolvido, os autores foram capazes de resolver o modelo aplicado a uma árvore de 162.000 cenários (maior quantidade de cenários testada nesse trabalho) em 821,79 segundos.

Todos os trabalhos apresentados nessa seção envolvem algum problema de ALM aplicado a fundos de pensão, embora o enfoque de cada trabalho seja diferente. Os trabalhos de Lauria & Consigli [2017] e Valladão & Veiga [2008] focaram principalmente nos aspectos da modelagem do problema. Kouwenberg & Zenios [2008] e de Oliveira

et al. [2018] abordam principalmente a temática da geração de cenários. Por sua vez, Gondzio & Kouwenberg [2001] e de Oliveira & Filomena [2020] se concentraram em formas de se resolver instâncias com muitos cenários.

Este trabalho aborda o problema de otimização proposto por de Oliveira et al. [2017]. Embora de Oliveira et al. [2017] abordem a dificuldade de se resolver árvores com muitos cenários, esse não é enfoque principal que desenvolveram em seu trabalho. Nesta dissertação, o trabalho desenvolvido tem objetivos semelhantes aos de Gondzio & Kouwenberg [2001] e de Oliveira & Filomena [2020]. Dentre esses dois trabalhos, esta dissertação tem afinidade maior com Gondzio & Kouwenberg [2001], pois ambos os trabalhos têm o objetivo de desenvolver métodos de decomposição que sejam capazes de resolver instâncias com muitos cenários. No entanto, Gondzio & Kouwenberg [2001] aborda um problema de ALM diferente do problema abordado neste trabalho.

3.2 Propostas de melhoria do PH

Desde a publicação da forma original do PH [Rockafellar & Wets, 1991], várias modificações foram desenvolvidas para esse algoritmo. Neste capítulo são abordados alguns dos trabalhos que apresentam essas modificações.

3.2.1 Parâmetro de penalidade

O parâmetro de penalidade tem grande importância na convergência do PH [Zehtabian & Bastin, 2016]. Rockafellar & Wets [1991], quando apresentaram pela primeira vez esse algoritmo, mantiveram o valor de ρ fixo durante todas as iterações. Com o passar do tempo, surgiram na literatura várias estratégias para a escolha do valor inicial e atualização de ρ. A seguir, são apresentadas algumas dessas estratégias.

Watson et al. [2007] introduziram dois métodos para computar o valor de ρ: CP (*cost-proportional*) e SEP (*selecting per-element*). O método CP funciona da seguinte forma: seja $(x_1, ..., x_N)$ o vetor de variáveis de decisão, às quais estão associadas restrições de não antecipatividade. Seja também $(c_1, ..., c_N)$ o vetor de coeficientes da função objetivo. No método CP, $\rho(i)$ é a penalidade da i-ésima variável de decisão e é determinado por

$$\rho(i) = \theta c_i \tag{3.1}$$

em que $\theta > 0$. No método SEP, $\rho(i)$ é dado por

$$\rho(i) = \frac{c_i}{(x_i^{max} - x_i^{min} + 1)}, \tag{3.2}$$

em que

$$x_i^{max} = \max_{\bar{\omega} \in \Omega} x_i^{\bar{\omega}}$$

e

$$x_i^{min} = \min_{\bar{\omega} \in \Omega} x_i^{\bar{\omega}}.$$

O PH de Watson et al. [2007] utilizando os métodos CP e SEP apresentou melhor convergência quando comparado ao algoritmo que utiliza o mesmo valor de ρ fixado *a priori* para todas as variáveis.

Algumas formas de atualização do parâmetro ρ são baseadas em duas métricas. A primeira é:

$$\Delta_P^k = \sum_{\bar{\omega} \in S(\omega)} p_{\bar{\omega}} ||\hat{x}^{\bar{\omega},k} - \hat{x}^{\bar{\omega},k-1}||_2, \tag{3.3}$$

em que $\hat{x}^{\bar{\omega},k}$ representa o vetor de restrições de não antecipatividade associado ao cenário $\bar{\omega}$ e iteração k do PH. A métrica representada pela equação 3.3 é o valor esperado da diferença entre as soluções de não antecipatividade entre duas iterações consecutivas do algoritmo. Essa métrica designa, portanto, a convergência das variáveis primais na iteração k em relação à iteração $k-1$. A segunda métrica é

$$\Delta_D^k = \sum_{\bar{\omega} \in S(\omega)} p_{\bar{\omega}} ||x^{\bar{\omega},k} - \hat{x}^{\bar{\omega},k}||_2, \tag{3.4}$$

em que $x^{\bar{\omega},k}$ representa o vetor de decisões associado ao cenário $\bar{\omega}$ e iteração k do PH. A métrica representada pela equação 3.4 fornece a magnitude das mudanças nas variáveis duais, isto é, nos multiplicadores lagrangianos [Zehtabian & Bastin, 2016]. O valor Δ_D^k representa a convergência das variáveis duais na iteração k.

Gul [2010] utilizou formas específicas das equações (3.3) e (3.4) como métricas para seu método de atualização de ρ. Basicamente, no método proposto por Gul [2010], o valor de ρ é atualizado da seguinte forma:

$$\rho^{k+1} = \begin{cases} \delta\rho^k, & \text{para } \Delta_D^k - \Delta_D^{k-1} > 0, \Delta_P^k - \Delta_P^{k-1} < 0 \\ \frac{1}{\delta}\rho^k, & \text{para } \Delta_P^k - \Delta_P^{k-1} > 0, \Delta_D^k - \Delta_D^{k-1} < 0 \\ \rho^k, & \text{Caso contrário} \end{cases}$$

em que $\delta > 1$. O método não foi comparado a outros já conhecidos. No entanto, Gul [2010] mostrou, através de seus experimentos, a influência do valor inicial ρ^0 na qualidade da soluções obtidas pelo PH.

Zehtabian & Bastin [2016] compilam as principais estratégias de atualização de ρ

e propõem uma estratégia de atualização adaptativa que se baseia nas equações (3.3) e (3.4). A partir dessas métricas, o algoritmo de atualização que propuseram pode diminuir ou aumentar o valor de ρ.

Para alguns problemas testados, Zehtabian & Bastin [2016] foram capazes de mostrar que a estratégia proposta apresentou melhorias, quando comparada a outras estratégias já conhecidas. No entanto, os autores salientam que o algoritmo desenvolvido é uma heurística e que maior número de pesquisas seria necessário para aprofundar a teoria do método e testá-lo em outros problemas.

3.2.2 Agrupamento de cenários

O agrupamento de cenários (*scenario bundling*) é uma estratégia utilizada para melhorar a convergência do PH [Majidi-Qadikolai & Baldick, 2017], [Ryan et al., 2013]. Os cenários são divididos em grupos e cada grupo de cenários tem sua própria formulação, na qual estão incluídas as restrições de não antecipatividade que unem os cenários do grupo. Em vez de resolver um subproblema para cada cenário, o PH resolve subproblemas associados a cada grupo de cenários. As restrições de não antecipatividade relaxadas são as que unem os grupos de cenários entre si. Na divisão dos cenários em grupos, é importante considerar o número de cenários em cada grupo [Majidi-Qadikolai & Baldick, 2017], [Ryan et al., 2013]. O número de cenários deve ser determinado de forma que os subproblemas não sejam muito pequenos e nem muito grandes (subproblemas muito grandes podem consumir muito tempo para serem resolvidos) [Ryan et al., 2013], [Majidi-Qadikolai & Baldick, 2017].

Ryan et al. [2013] utilizaram uma versão do PH com agrupamento de cenários para resolver um problema de programação estocástica. Em seu trabalho, Ryan et al. [2013] obtiveram soluções com *gap* de otimalidade menor quando comparado à versão do algoritmo sem agrupamento de cenários.

Majidi-Qadikolai & Baldick [2017] propuseram um algoritmo para agrupar cenários e um método híbrido que utiliza o PH e a decomposição de Benders para resolver um problema de programação estocástica. O método de agrupamento proposto envolve três etapas: classificação, *clusterização* e agrupamento dos cenários do problema. Em seguida, cada grupo de cenários é associado a um subproblema do PH. No método híbrido proposto, a decomposição de Benders é utilizada para resolver cada subproblema.

Majidi-Qadikolai & Baldick [2017] mostraram que o agrupamento de cenários reduz o tempo gasto pelo PH, além de fornecer soluções com menor *gap* de otimalidade, quando comparado à versão desse algoritmo sem agrupamento de cenários.

3.2.3 Fixação de variáveis

A estratégia de fixar variáveis é uma heurística que pode reduzir o tempo gasto na resolução dos subproblemas do PH [Watson & Woodruff, 2011], [Garcia-Gonzalo et al., 2020]. A ideia que motivou o desenvolvimento dessa heurística se baseia na observação de que, em alguns problemas, certas variáveis de decisão se fixam em determinados valores nas iterações iniciais do PH [Watson & Woodruff, 2011].

Basicamente, a heurística de fixação de variáveis funciona da seguinte forma: seja μ um inteiro positivo e $x_i^{\bar{\omega},k}$ a i-ésima variável de decisão associada ao cenário $\bar{\omega}$ na iteração k. Para cada $\bar{\omega} \in \Omega$ e cada i, se $x_i^{\bar{\omega},\bar{k}} = z$ nas iterações $\bar{k} = k, k+1, ..., k+\mu$, com $k \geq 0$, fixa-se $x_i^{\bar{\omega},k} = z$ em todas as iterações subsequentes.

Existe um *trade-off* na utilização da heurística de fixação. Se o valor de μ escolhido é pequeno, a solução produzida pelo PH ao final da execução do algoritmo pode estar mais distante do ótimo. Por outro lado, valores maiores de μ podem levar a soluções mais próximas do ótimo, mas um maior tempo de execução pode ser gasto até que as variáveis sejam fixadas [Garcia-Gonzalo et al., 2020].

Garcia-Gonzalo et al. [2020] utilizaram o PH com a heurística de fixação para resolver um problema inteiro misto. Foram fixadas variáveis de decisão binárias. Além disso, assim que um determinado percentual de variáveis binárias eram fixadas, o problema determinístico equivalente era resolvido com essas variáveis binárias fixadas. Em um grupo de instâncias, mais de 90% das variáveis foram fixadas. Nas instâncias consideradas mais difíceis, devido à má convergência do algoritmo, os autores estabeleceram como critério de parada do algoritmo o percentual de 80% de variáveis binárias fixadas. Na maior parte das instâncias analisadas, os autores foram capazes de mostrar a superioridade do PH em relação ao *solver* do CPLEX, tanto em termos de tempo de execução quanto em relação *gap* de otimalidade das soluções.

3.2.4 Paralelização

O PH é um algoritmo paralelizável pela independência de cada um dos subproblemas resultantes da decomposição. O algoritmo pode ser paralelizado da seguinte forma: em cada iteração do PH, os subproblemas são distribuídos para vários processadores e resolvidos. Quando todos os subproblemas são resolvidos, eles são novamente reunidos para o cômputo das equações (2.22) e (2.23). A paralelização permite que o problema seja resolvido de forma mais rápida, comparado ao algoritmo puramente sequencial. Ryan et al. [2013] compararam o tempo de execução do PH paralelo ao sequencial. O tempo necessário para resolver a instância do problema estudado pelos autores foi de 840 minutos utilizando o algoritmo sequencial. Já o algoritmo paralelo resolveu

a mesma instância em 15 minutos. Em razão dessa característica, vários trabalhos implementam uma versão paralelizada do PH e a aplicam na resolução de problemas de programação estocástica [Perboli et al., 2017],Garcia-Gonzalo et al. [2020], [Ryan et al., 2013], [Peng et al., 2019].

Capítulo 4

Modelo de ALM para um fundo de pensão

Neste capítulo é apresentado o modelo de programação estocástica de ALM aplicado a um fundo de pensão. Na seção 4.1, apresenta-se uma breve descrição do problema de ALM no contexto de um fundo de pensão. Na seção 4.2, é introduzida uma notação para o modelo. Por fim, na seção 4.3 são apresentadas e discutidas as restrições e a função objetivo. Ao final da seção, o modelo completo é apresentado.

4.1 Descrição do Problema

Este trabalho aborda o problema de programação estocástica para ALM aplicado a fundos de pensão brasileiros, baseado em de Oliveira et al. [2017]. Neste problema, um fundo de pensão deve planejar seus investimentos para um horizonte de tempo dividido em T períodos, de forma que seus recursos financeiros sejam capazes de cobrir seus passivos. Os recursos financeiros do fundo são compostos pela contribuição de associados e sua carteira de ativos, que representam os investimentos. Os passivos podem ser, por exemplo, as aposentadorias pagas aos associados.

O fundo deve planejar seus investimentos, de forma que, para $t = 0, 1, ..., T$, os recursos do fundo sejam capazes de cobrir os passivos. A legislação brasileira especifica que o *funding ratio*, que é a razão entre os recursos do fundo (somatório do valor presente das contribuições futuras e valor do portfólio de investimentos) e o valor presente dos passivos, não deve ser menor que 1 por mais de dois anos consecutivos [de Oliveira et al., 2017].

Para compor a carteira de ativos, o fundo pode comprar e vender diferentes tipos de ativos nos instantes $t = 0, 1..., T$, em que $t = 0$ é o instante inicial em que o

investimento é feito. Os ativos podem ser ações, índices, títulos públicos, entre outros. Em $t = 0$, o fundo dispõe de uma quantia inicial para comprar ativos e os preços desses são desconhecidos para os instantes $t = 1, 2, ..., T$. Além disso, a legislação brasileira determina um limite máximo na participação de cada tipo de ativo na carteira [de Oliveira et al., 2017]. Para que o fundo alcance o objetivo de cobrir seus passivos, ele deve decidir no instante $t = 0$ como alocar seus recursos nos diferentes ativos, para $t = 0, 1, ..., T$, de forma a maximizar o valor do portfólio no instante $t = T$.

4.2 Notação

A Tabela 4.1 apresenta a notação do modelo utilizado neste trabalho. A notação é a mesma apresentada por de Oliveira et al. [2017]. Esta tabela apresenta alguns parâmetros que não foram introduzidos na Seção 4.1 e que serão melhor detalhados na Seção 4.3, quando forem apresentadas as restrições e a função objetivo.

Tabela 4.1: Notação

Índices:	
i	índice do ativo, $i \in \{1, ..., N\}$, em que N é o número de ativos
t	índice do estágio, $t \in \{0, 1, ..., T\}$, em que T é o número de estágios
s	índice do cenário, $s \in S$, em que S é o conjunto de estágios
Parâmetros determinísticos:	
Q	Recursos financeiros do fundo em $t = 0$
K	valor do *funding ratio* definido por lei [de Oliveira et al., 2017]
L_t	valor presente dos passivos futuros em $t = t + 1, ..., T$
l_t	valor do passivo a ser pago no estágio t
F_t	valor presente das contribuições em $t = t + 1, ..., T$
f_t	valor da contribuição no estágio t
M	valor máximo de *underfunding* permitido [de Oliveira et al., 2017]
ρ	fator de desconto
π_i	peso máximo do ativo i no portfólio
p_{ts}	probabilidade do cenário s no estágio t
P_{its}	preço do ativo i no estágio t e cenário s
P_{i0}	preço do ativo i no estágio $t = 0$
Variáveis de decisão:	
X_{its}	quantidade do ativo i em posse, no estágio t e cenário s
B_{its}	quantidade do ativo i comprado, no estágio t e cenário s
V_{its}	quantidade do ativo i vendido, no estágio t e cenário s
C_{ts}	variável binária que tem valor 1 se há *underfunding* no estágio t e cenário s. Caso contrário, seu valor é 0.

4.3 Modelo do problema

O problema descrito no início deste capítulo é modelado como um problema de programação estocástica multiestágio. Neste trabalho, utiliza-se o modelo apresentado por de Oliveira et al. [2017], que representa um problema inteiro-misto de otimização linear estocástica.

Na utilização do modelo de de Oliveira et al. [2017], optou-se excluir as restrições probabilísticas. Essas restrições foram excluídas porque elas acoplam os cenários em um mesmo período, não permitindo desta forma a decomposição do modelo original em subproblemas independentes. Além disso, o cumprimento dessas restrições não é necessário do ponto de vista legal [de Oliveira et al., 2017].

No modelo utilizado, existem quatro variáveis de decisão. As variáveis contínuas X_{its}, B_{its} e V_{its} representam, respectivamente, as quantidades em posse, compradas e vendidas dos ativo i, no t-ésimo estágio e no cenário s. A variável binária C_{ts} indica a ocorrência de *underfunding* no t-ésimo estágio e cenário s.

Nesta seção, as restrições do modelo são apresentadas em grupos, seguindo a nomenclatura que aparece na literatura [Valladão & Veiga, 2008], [Consigli & Dempster, 1998]. Ao final desta seção, apresenta-se a função objetivo e o modelo completo do problema.

- **Restrições de inventário**: As restrições de inventário registram as operações compra e venda de cada tipo de ativo, de forma a manter uma relação temporal coerente de suas quantidades no portfólio. As restrições de inventário do modelo são:

$$X_{its} = X_{i(t-1)s} + B_{its} - V_{its} \quad i = 1, ..., N, t = 1, ..., T, s \in S. \tag{4.1}$$

  As restrições (4.1) indicam que a quantidade de cada ativo i em posse no estágio t é igual à quantidade do ativo i em posse no estágio $t-1$, somada à diferença entre as quantidades comprada e vendida desse mesmo ativo, no estágio t. Em $t = 0$, a quantidade comprada é igual à quantidade em posse, ou seja, $X_{i0s} = B_{i0s}$, para todo $i = 1, ..., N$, $s \in S$.

- **Restrições regulatórias**: As restrições regulatórias representam imposições técnicas que instituições reguladoras exercem sobre os fundos de pensão [de Oliveira et al., 2017]. Essas restrições procuram limitar as atividades de investimentos dos fundos, com o objetivo de garantir que esses consigam honrar suas

obrigações. As restrições regulatórias do modelo são:

$$X_{its}P_{its} \leq \pi_i \sum_{j=1}^{N} X_{jts}P_{jts}, \qquad i = 1, ..., N, t = 1, ..., T, s \in S \tag{4.2}$$

As restrições (4.2) impõem limites na participação de cada de ativo no portfólio em um dado estágio e cenário. Segundo de Oliveira et al. [2017], a legislação brasileira permite, por exemplo, um máximo de 100% de alocação em ativos de renda fixa e 70% em *equities*.

- **Restrições de balanceamento**: As restrições de balanceamento contabilizam as operações de entrada e saídas de recursos. As entradas de recursos podem derivar de vendas de ativos, contribuições de associados, entre outros. As saídas podem se originar, por exemplo, de pagamentos de benefícios e compras de ativos. As restrições de balanceamento do modelo são:

$$Q = \sum_{i=1}^{N} P_{i0}X_{i0} \tag{4.3}$$

$$\sum_{i=1}^{N} P_{its}V_{its} - \sum_{i=1}^{N} P_{its}B_{its} + f_t = l_t \qquad t = 1, ..., T, s \in S. \tag{4.4}$$

$$K(L_t - F_t) - \sum_{i=1}^{N} P_{its}X_{its} \leq MC_{ts} \qquad t = 1, ..., T, s \in S \tag{4.5}$$

$$\sum_{j=0}^{2} C_{t+j,s} \leq 2 \qquad t = 1, ..., T-2, s \in S \tag{4.6}$$

A restrição (4.3) representa a alocação inicial de recursos, considerando que, em $t = 0$, toda a quantidade Q de recursos é usada para construir o portfólio com N ativos e os preços de todos os ativos são conhecidos. As restrições (4.4) indicam que, no estágio t, os recursos provenientes da compras e vendas de ativos somados às contribuições dos participantes do fundo devem ser iguais aos passivos.

Antes de se descrever as restrições (4.5), é necessário definir o conceito de *funding ratio*, que motiva essas restrições. O *funding ratio* é dado por

$$\beta_{ts} = \frac{F_t + \sum_{i=1}^{N} P_{its} X_{its}}{L_t} \tag{4.7}$$

em que a parcela $\sum_{i=1}^{N} P_{its} X_{its}$ é o valor do portfólio. F_t é o valor presente das contribuições futuras e L_t é o valor presente dos passivos futuros. De acordo com de Oliveira et al. [2017], os parâmetros F_t e L_t são dados por

$$F_t = \sum_{k=t}^{T} \frac{f_t}{(1+\rho)^{k-t}}, \quad L_t = \sum_{k=t}^{T} \frac{l_t}{(1+\rho)^{k-t}}, \tag{4.8}$$

em que ρ é o valor de desconto aplicado.

As restrições (4.5) permitem que o fundo tenha um nível máximo de *underfunding* em certos períodos. As restrições (4.6) determinam que o *underfunding* não ocorra por mais de dois anos consecutivos. De acordo com Valladão & Veiga [2008], o *underfunding* é uma quantidade negativa de recursos no horizonte de tempo do problema.

- **Restrições não antecipatórias:** As restrições não antecipatórias forçam soluções de cenários diferentes a compartilharem o mesmo valor nos estágios em que as soluções têm uma história comum. A história comum se refere aos parâmetros aleatórios, revelados de forma sucessiva nos estágios. As restrições de não antecipatividade são dadas por

$$X_{its} = X_{its'}, \quad s' \in \mathcal{H}^t(s), i = 1, ..., N, t = 0, 1, ..., T-1, s \in S \tag{4.9}$$

em que $\mathcal{H}^t(s) = \{s' \in S | \omega_r^s = \omega_r^{s'}, r = 1, ..., t\}$ é o conjunto de cenários que compartilham a mesma história de parâmetros com o cenário s.

- **Função objetivo**: A função objetivo do modelo é dada por

$$\max \quad \sum_{s \in S} \sum_{i=1}^{N} p_{Ts} P_{iTs} X_{iTs}. \tag{4.10}$$

A função objetivo procura maximizar o valor esperado do portfólio no último estágio.

O modelo completo é dado por

$$\max \quad \sum_{s\in S}\sum_{i=1}^{N} p_{Ts}P_{iTs}X_{iTs} \tag{4.11}$$

$$\text{sujeito a: } Q = \sum_{i=1}^{N} P_{i0}X_{i0} \tag{4.12}$$

$$X_{its} = X_{i(t-1)s} + B_{its} - V_{its} \qquad i = 1,...,N, t = 1,...,T, s \in S \tag{4.13}$$

$$\sum_{i=1}^{N} P_{its}V_{its} - \sum_{i=1}^{N} P_{its}B_{its} + f_t = l_t \quad t = 1,...,T, i = 1,...,N, s \in S \tag{4.14}$$

$$X_{its}P_{its} \leq \pi \sum_{i=1}^{N} X_{its}P_{its} \qquad t = 1,...,T, s \in S \tag{4.15}$$

$$K(L_t - F_t) - \sum_{i=1}^{N} P_{its}X_{its} \leq MC_{ts} \quad t = 1,...,T, s \in S \tag{4.16}$$

$$\sum_{j=0}^{2} C_{t+j,s} \leq 2, \qquad t = 1,...,T-2, s \in S \tag{4.17}$$

$$X_{its} = X_{its'} \qquad i = 1,...,N, t = 1,...,T-1, s \in S, s' \in \mathcal{H}^t(s) \tag{4.18}$$

$$X_{its}, B_{its}, V_{its} \geq 0 \qquad i = 1,...,N, t = 1,...,T, s \in S \tag{4.19}$$

$$C_{ts} \in \{0,1\} \qquad t = 1,...,T, s \in S \tag{4.20}$$

Capítulo 5

Metodologia

Neste capítulo, são detalhados os procedimentos utilizados neste trabalho. A Seção 5.1 trata dos parâmetros das instâncias utilizadas neste trabalho e da forma como essas foram geradas. A Seção 5.2 trata dos algoritmos utilizados para resolver o modelo usado neste trabalho.

5.1 Geração das instâncias

Neste trabalho, as instâncias foram geradas de forma aleatória, pois não foi possível a obtenção das instâncias utilizadas por de Oliveira et al. [2017]. As instâncias geradas para este trabalho incluem parâmetros determinísticos, fixados para todas as instâncias, e parâmetros aleatórios. Os parâmetros inclusos nas instâncias são:

i) Número de ativos (N);

ii) Número de estágios (T);

iii) Número de nós que se ramificam por estágio;

iv) Contribuições dos assegurados;

v) Valor do *funding ratio* (K);

vi) Valor máximo de *underfunding* permitido (M);

vii) Recursos financeiros do fundo no instante inicial (Q);

viii) Peso máximo permitido de cada tipo de ativo no portifólio (π);

ix) Cenários de preços dos ativos;

x) Passivos do fundo (*liabilities*).

As instâncias geradas foram divididas em três grupos, designados por A, B e C. No Capítulo 6, os motivos para essa divisão são apresentados.

A seguir, é detalhado o processo de como os cenários de preços e os passivos do fundo foram gerados. As instâncias são geradas usando a linguagem de programação Python.

5.1.1 Cenários de preços dos ativos

Cada instância possui cenários de preços de dois tipos de ativos, um de renda fixa e outro de renda variável. Os cenários de preço de cada ativo são obtidos da árvore de cenários gerada para esse ativo. A geração da árvore de cenários é feita da seguinte forma: ao nó raiz de uma árvore, que corresponde ao estágio 0, associa-se um preço P_0 inicial para o ativo, que é igual para todas as instâncias. Para gerar o preço de um nó do estágio 1, multiplica-se P_0 pelo retorno anual, que é gerado aleatoriamente. De forma geral, para se obter os preços associados aos nós do estágio $t+1$, multiplica-se o preço de cada nó do estágio t por um valor de retorno anual gerado de forma aleatória.

O retorno anual é obtido da seguinte forma: seja $S^i = \{R_0^i, R_1^i, ..., R_N^i\}$ a série histórica mensal de retornos do i-ésimo ativo, em que R_t^i é o retorno do ativo i no tempo t. O método *bootstrap*, utilizado neste trabalho, atribui a cada retorno individual da série histórica uma probabilidade $\frac{1}{N+1}$. Posteriormente, o *bootstrap* seleciona aleatoriamente um valor de retorno. Para computar o retorno anual R_a^i do ativo i, utiliza-se a fórmula

$$R_a^i = (1 + R_{1'}^i)(1 + R_{2'}^i)...(1 + R_{12'}^i) - 1, \tag{5.1}$$

em que $R_{1'}^i, R_{2'}^i, ..., R_{12'}^i$ são retornos selecionados aleatoriamente da série S^i.

Para gerar os cenários de preços dos ativos, foram utilizadas séries temporais de dois índices: IMAB5, que é um índice de ativos de renda fixa, e IBOVESPA, que é um índice de ativos de renda variável. Foram obtidas as séries temporais diárias do IMBA5 e IBOVESPA abrangendo o período de 28/06/2010 até 25/06/2020. A série temporal do IMBA5 foi obtida do "Sistema Gerenciador de Séries Temporais"do Banco Central do Brasil, que pode ser acessado pela *url* https://www3.bcb.gov.br/sgspub/localizarseries/localizarSeries.do?method=prepararTelaLocalizarSeries. Já a série temporal do IBOVESPA foi obtida do site *Yahoo Finance*, cujo acesso é possível pela *url* https://finance.yahoo.

com/. A partir de cada série diária foi obtida a série de retornos mensais de cada um dos índices.

5.1.2 Passivos do fundo

Para cada estágio de uma determinada instância, é gerado aleatoriamente um número que corresponde ao valor do passivo. O valor do passivo é um número gerado pela distribuição uniforme $U[300, 700]$. Portanto, se uma determinada instância tem 10 estágios, o seu passivo consiste em 10 números gerados pela distribuição $U[300, 700]$.

5.2 Métodos de resolução

Neste trabalho, primeiramente são comparadas quatro versões do PH, que designam-se por PH1, PH2, PH3 e PH4. A partir das comparações feitas, a melhor versão do PH é selecionada. Em seguida, a versão de PH selecionada é comparada ao *solver* CPLEX. Todas as versões do PH são implementadas na linguagem de programação C++, utilizando as bibliotecas do CPLEX. Os subproblemas de cada versão são resolvidos utilizando-se o *solver* CPLEX.

A seguir, são detalhadas as quatro versões do PH.

5.2.1 PH1

A versão PH1 corresponde ao Algoritmo 1. Nessa versão, não há nenhuma das modificações apresentadas no Capítulo 3. A única modificação em relação ao algoritmo original está no critério de parada. Dessa forma, o PH1 é uma versão na qual os subproblemas são resolvidos sequencialmente e o parâmetro ρ é mantido fixo. Cada subproblema resolvido pelo PH1 tem a formulação

$$\max \sum_{i=1}^{N} p_{Ts} P_{iTs} X_{iTs} - \sum_{t=0}^{T-1} \lambda_t^{s,k} (X_{its} - \hat{X}_{it}) + \frac{\rho^k}{2} ||X_{its} - \hat{X}_{it}||_2^2 \tag{5.2}$$

$$s.a: \quad Q = \sum_{i=1}^{N} P_{i0} X_{i0} \tag{5.3}$$

$$X_{its} = X_{i(t-1)s} + B_{its} - V_{its} \qquad i = 1, ..., N, t = 1, ..., T \tag{5.4}$$

$$\sum_{i=1}^{N} P_{its} V_{its} - \sum_{i=1}^{N} P_{its} B_{its} + f_t = l_t, \qquad t = 1, ..., T, i = 1, ..., N \tag{5.5}$$

$$X_{its} P_{its} \leq \pi_i \sum_{i=1}^{N} X_{its} P_{its}, \qquad t = 1, ..., T \tag{5.6}$$

$$K(L_t - F_t) - \sum_{i=1}^{N} P_{its} X_{its} \leq MC_{ts}, \qquad t = 1, ..., T \tag{5.7}$$

$$\sum_{j=0}^{2} C_{t+j} s \leq 2, \qquad t = 1, ..., T-2 \tag{5.8}$$

$$X_{its}, B_{its}, V_{its} \geq 0, \qquad i = 1, ..., N, t = 1, ..., T \tag{5.9}$$

$$C_{ts} \in \{0, 1\}, \qquad t = 1, ..., T \tag{5.10}$$

Nesta formulação, optou-se por aplicar as restrições de não antecipatividade somente às variáveis contínuas, assim como fez Almeida et al. [2013] ao escolher um subconjunto das variáveis de decisão para aplicar as restrições de não antecipatividade. Baseando-se em Gonçalves et al. [2011], Almeida et al. [2013] afirma que a aplicação da relaxação às restrições de não antecipatividade envolvendo somente um subconjunto de variáveis de decisão mantém o acoplamento entre cenários que compartilham decisões comuns. Na modelo estudado neste trabalho, o acoplamento entre as variáveis contínuas e binárias é dado pela restrição (4.16) do modelo original.

Na formulação apresentada, $\lambda_t^{s,k}$ é o multiplicador lagrangiano do cenário s e estágio t e ρ^k é o parâmetro de penalidade da k-ésima iteração. A variável $\hat{X}_{it}$ é equivalente à dada pela equação (2.17). Para cada $s \in S$ (o conjunto de índices dos cenários), tem-se um subproblema.

Nos primeiros experimentos computacionais, observou-se que, ao escolher um valor de ϵ para o critério de parada, dado pela equação (2.24), a qualidade das soluções obtidas para as instâncias com muitos cenários era inferior à qualidade das soluções obtidas para as instâncias com poucos cenários. Dessa forma, considerou-se o critério de parada usado por Rockafellar & Wets [1991] como inapropriado. Por esse motivo, desenvolveu-se a uma modificação da desigualdade (2.24), que serve como critério de parada dessa e das demais versões (com uma pequena modificação nas versões PH2, PH3 e PH4, decorrente do agrupamento de cenários) do PH implementadas neste trabalho. Além disso, o critério de parada utilizado neste trabalho também impõe ao

algoritmos implementados um limite no número de iterações. O critério de parada utilizado é apresentado a seguir.

Dado um número real ϵ arbitrário, o algoritmo PH1 pára quando

$$\frac{\sqrt{\sum_{\bar{\omega}\in\Omega} p_{\bar{\omega}} \left\| x^{\bar{\omega},k+1} - \hat{x}^{\bar{\omega},k} \right\|_2^2}}{\sqrt{\sum_{\bar{\omega}\in\Omega} p_{\bar{\omega}} \left\| x^{\bar{\omega},1} - \hat{x}^{\bar{\omega},0} \right\|_2^2}} \leq \epsilon, \tag{5.11}$$

ou quando $k > M$, sendo M um inteiro positivo. Observando-se a equação 5.16 e comparando-a à equação 2.24, percebe-se que aquela se difere desta somente pela adição do termo que aparece no denominador.

5.2.2 PH2

Essa versão do PH traz três modificações apresentadas no Capítulo 4: agrupamento de cenários, fixação de variáveis e paralelização. O parâmetro ρ é mantido fixo durante as iterações do algoritmo. O agrupamento de cenários é feito dividindo-se a árvore de cenários em blocos com a mesma quantidade de cenários. Cada bloco representa um conjunto de cenários que compartilham uma história de parâmetros em comum até um determinado estágio. As Figuras 5.1-5.2 ilustram o processo de agrupamento.

Cada grupo de cenários constitui um subproblema a ser resolvido pelo PH. Na formulação de cada subproblema, as restrições de não antecipatividade são introduzidas para ligar as variáveis que representam os nós comuns aos diferentes cenários que compõem o mesmo grupo. As restrições de não antecipatividade que são relaxadas e adicionadas à função objetivo de cada subproblema envolvem somente as variáveis que representam os nós comuns aos diferentes grupos. No caso dos grupos da Figura 5.2, por exemplo, os dois grupos de cenários só possuem o nó s_0 em comum. Portanto, nesse caso, as restrições de não antecipatividade relaxadas só envolvem as variáveis associadas a esse nó. O PH com agrupamento de cenários é apresentado no Algoritmo 2.

As variáveis fixadas no modelo deste trabalho são as variáveis binárias. A fixação dessas variáveis ocorre assim que elas satisfaçam as restrições de não antecipatividade por um determinado número de iterações do algoritmo. O algoritmo é paralelizado nas instruções correspondentes à linha 2 do Algoritmo 2. Com a paralelização, em cada iteração do algoritmo, são distribuídos para as *threads* N subproblemas, em que N é dado por

$$N = \frac{|\Omega|}{|S_i|},$$

Algoritmo 2: PH2

1: No início, faça:

(i) Defina inicialmente $k = 0$ e divida os cenários de Ω entre os grupos $S_1, ..., S_N$.

(ii) Para cada S_i, defina uma indexação dos cenários de forma que $S_i = \{\omega_1^i, ..., \omega_r^i\}$, em que r é o número de cenários de S_i, e a probabilidade p^{S^i} como a soma das probabilidades dos cenários $p^{\omega_1^i}, ..., p^{\omega_r^i}$.

(iii) Seja t' o último estágio no qual todos os grupos de cenários compartilham o mesmo nó da árvore de cenários. Para cada um dos grupos de cenários, compute o vetor $\hat{x}^{S^i,0} = (\hat{x}_0^{S^i,0}, ..., \hat{x}_{t'}^{S^i,0})$ pela equação (5.2.3).

(iv) Para cada grupo S^i, inicialize o vetor de multiplicadores lagrangianos $\lambda^{S^i,0} = (\lambda_0^{S^i,0}, ..., \lambda_{t'}^{S^i,0}) = (0_0, 0_1, ..., 0_{t'})$. Escolha o fator de penalidade $\rho^0 > 0$.

2: Para cada grupo S^i, resolva o subproblema

$$\min_{x^{S^i}} f(x^{S^i}, S^i) + \sum_{t=0}^{t'} \left(\lambda_t^{S^i,k}(x_t^{S^i} - \hat{x}_t^{S^i,k}) + \frac{\rho^k}{2} \left\| x_t^{S^i} - \hat{x}_t^{S^i,k} \right\|^2\right) \tag{5.12}$$

$$\text{sujeito a} \quad x^{S^i} \in X(S^i) \tag{5.13}$$

e defina $x^{S^i,k+1} = (x_0^{S^i,k+1}, ..., x_{t'}^{S^i,k+1}) = (x_0^{\omega_1^i,k+1}, ..., x_{t'}^{\omega_r^i,k+1})$ como a solução do subproblema.

3: Para cada grupo S^i, $t = 0, ..., t'$, compute

$$\hat{x}_t^{S^i,k+1} = \frac{\sum_i p_{S^i} x_t^{\omega_1^i,k+1}}{\sum_i p_{S^i}}. \tag{5.14}$$

4: Escolha ρ^{k+1} e compute para cada S^i, $t = 0, ..., t'$,

$$\lambda_t^{S^i,k+1} = \lambda_t^{S^i,k} + \rho^k\big(x_t^{S^i,k+1} - \hat{x}_t^{S^i,k+1}\big). \tag{5.15}$$

5: Pare o algoritmo se, para um dado ϵ,

$$\frac{\sqrt{\sum_{S^i} p_{S^i} \left\| x^{S^i,k+1} - \hat{x}^{S^i,k} \right\|_2^2}}{\sqrt{\sum_{S^i} p_{S^i} \left\| x^{S^i,1} - \hat{x}^{S^i,0} \right\|_2^2}} \leq \epsilon, \tag{5.16}$$

ou quando $k > M$, sendo M inteiro positivo definido *a priori*. Do contrário, incremente k e volte para as instruções da linha 2.

em que $|\Omega|$ é quantidade total de cenários e $|S_i|$ é a quantidade de cenários do agrupamento S_i - todos os agrupamentos têm a mesma quantidade de cenários. Para paralelizar o código, foi utilizado o *framework* OpenMP.

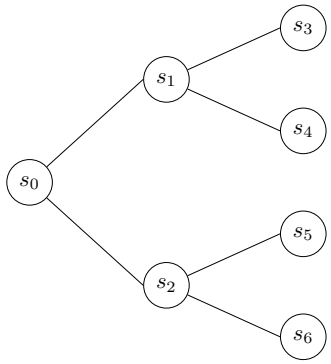

Figura 5.1: Árvore de cenários original.

A formulação do subproblema resolvido pelo PH2 no Passo 2 do algoritmo é

$$\max \sum_{s \in S^r} \sum_{i=1}^{N} p_{Ts} P_{iTs} X_{iTs} - \sum_{t=0}^{t'} (\lambda_t^{s,k} (X_{its} - \hat{X}_{it}) + \frac{\rho^k}{2} ||X_{its} - \hat{X}_{it}||_2^2) \tag{5.17}$$

$$s.a: \quad Q = \sum_{i=1}^{N} P_{i0} X_{i0} \tag{5.18}$$

$$X_{its} = X_{i(t-1)s} + B_{its} - V_{its} \qquad i = 1, ..., N, t = 1, ..., T, s \in S^r \tag{5.19}$$

$$\sum_{i=1}^{N} P_{its} V_{its} - \sum_{i=1}^{N} P_{its} B_{its} + f_t = l_t, \quad t = 1, ..., T, i = 1, ..., N \tag{5.20}$$

$$X_{its} P_{its} \leq \pi \sum_{i=1}^{N} X_{its} P_{its}, \qquad t = 1, ..., T, s \in S^r \tag{5.21}$$

$$K(L_t - F_t) - \sum_{i=1}^{N} P_{its} X_{its} \leq MC_{ts}, \quad t = 1, ..., T, s \in S^r \tag{5.22}$$

$$\sum_{j=0}^{2} C_{t+j} s \leq 2, \qquad t = 1, ..., T-2, s \in S^r \tag{5.23}$$

$$X_{its} = X_{its'}, \qquad s' \in \mathcal{H}^t(s), i = 1, ..., N, t = 0, ..., T-1, s \in S^r \tag{5.24}$$

$$X_{its}, B_{its}, V_{its} \geq 0, \qquad i = 1, ..., N, t = 1, ..., T, s \in S^r \tag{5.25}$$

$$C_{ts} \in \{0, 1\}, \qquad t = 1, ..., T, s \in S^r \tag{5.26}$$

Nessa formulação, S^r é um subconjunto de índices de S e tem-se uma nova definição de $\mathcal{H}^t(s)$, que é $\mathcal{H}^t(s) = \{s' \in S^r | \omega_r^s = \omega_r^{s'}, r = 1, ..., t\}$.

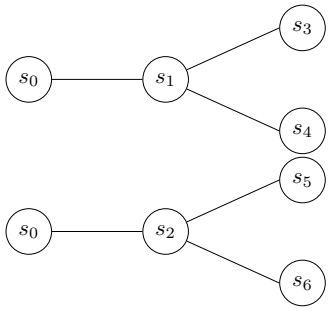

Figura 5.2: Agrupamentos de cenários a partir da árvore de cenários original. Os únicos parâmetros que os dois agrupamentos possuem em comum são os parâmetros do nó s_0.

5.2.3 PH3

A versão PH3 apresenta todas as modificações que aparecem no PH2. Além disso, há no PH3 um método de atualização do parâmetro ρ. O método de atualização de ρ implementado para versão PH3 é apresentado em seguida.

Seja ρ^k o parâmetro de penalidade da k-ésima iteração do PH3. A atualização do parâmetro de penalidade na $(k+1)$-ésima iteração é dada por

$$\rho^{k+1} = \begin{cases} \rho^k & \Delta_D^k - \Delta_D^{k-1} \leq 0 \\ \frac{1}{\delta}\rho^k & \Delta_D^k - \Delta_D^{k-1} > 0 \end{cases}$$

em que Δ_D^k é dado pela equação (3.4).

5.2.4 PH4

Semelhante à versão PH3, a versão PH4 apresenta todas as modificações que aparecem no PH2. Além disso, há no PH4 um método de atualização do parâmetro ρ, baseado no método proposto por Gul [2010] e que é apresentado na equação (3.2.1).

Capítulo 6

Resultados Experimentais

Neste capítulo, são apresentados e discutidos os resultados dos experimentos realizados durante o desenvolvimento desta dissertação. Na seção 6.1 são apresentadas a forma de divisões das instâncias, os propósitos envolvidos nessa divisão e os parâmetros utilizados. Na Seção 6.2 são abordados os métodos de pesquisa e definição dos parâmetros dos algoritmos implementados. Na Seção 6.3 são apresentados os recursos computacionais utilizados e as estruturas das tabelas com resultados dos experimentos. Na Seção 6.4 são comparados os resultados obtidos ao se aplicar as diferentes versões do PH às instâncias do grupo A. Em seguida, na Seção 6.5 são comparados os resultados obtidos ao se aplicar as versões PH2, PH3 e PH4 para resolver as instâncias do grupo B. Posteriormente, na Seção 6.6 são comparados os resultados obtidos ao se aplicar o CPLEX e a versão PH3 para resolver as instâncias do grupo C. Por fim, na Seção 6.7 é apresentada uma breve análise de sensibilidade do parâmetro ϵ, que faz parte do critério de parada dos algoritmos.

6.1 Instâncias

Neste trabalho, foram geradas instâncias para se estudar os algoritmos implementados. Tendo em vista o objetivo principal desse trabalho, foram geradas instâncias com variados números de cenários. As instâncias geradas são caracterizadas por dois parâmetros básicos: o número de nós por estágio, representado por N, e o número de estágios da árvore de cenários, representado por T. Para cada dupla de valores (N, T) escolhida para compor os grupos de instâncias, foram geradas 10 instâncias diferentes, exceto no caso das instâncias com $T = 3$ do grupo C, que serão abordadas mais à frente. A execução dos algoritmos sobre cada conjunto de 10 instâncias permitiu a obtenção de estatísticas sobre seus desempenhos. Além disso, optou-se por dividir as

instâncias geradas em três grupos, com o objetivo de apresentar de forma progressiva as etapas desenvolvidas nesse trabalho.

O primeiro grupo de instâncias, designado por A, foi utilizado para comparar todas as versões implementadas do PH . As instâncias do grupo A consistem em árvores de cenários binárias, com estágios que variam de $T = 8$ até $T = 13$.

O segundo grupo de instâncias, designado por B, é composto por instâncias com quantidades maiores de cenários e é usado para comparar as versões PH2, PH3 e PH4 entre si e ao CPLEX. O PH1 foi eliminado nessa etapa da análise porque seu tempo de execução tornou inviável sua aplicação nesse grupo de instâncias. Além disso, esse grupo de instâncias foi utilizado para se definir a melhor versão do PH.

As instâncias do grupo B são representadas por árvores com três ou quatro nós por estágio. O número de estágios das instâncias com três nós varia de $T = 8$ a $T = 10$. No caso das instâncias com quatro nós por estágio, essas foram geradas somente com $T = 8$ estágios.

O último conjunto de instâncias, grupo C, foi subdividido em dois grupos: um subgrupo de instâncias com $T = 5$ estágios e outro subgrupo de instâncias com $T = 3$ estágios. As instâncias com $T = 5$ estágios foram utilizadas para se fazer comparações entre o CPLEX e PH3, a versão que se mostrou a mais eficiente entre as quatro implementadas. As instâncias com $T = 3$ foram utilizadas para se analisar como a variabilidade do valor da função objetivo se comporta em função do esquema de ramificação das árvores de cenários. Para cada conjunto de instâncias definido por um valor de N e $T = 3$, foram geradas 1.000 instâncias.

Para todas as instâncias geradas, são listados a seguir os parâmetros fixos e seus valores:

i) Número de ativos: 2;

ii) Contribuições dos assegurados: 300;

iii) Valor do *funding ratio*: 1;

iv) Valor máximo de *underfunding*: 1.000;

v) Recursos financeiros do fundo no instante inicial: 10.000;

vi) Ponderação máxima permitida de cada tipo de ativo no portifólio: 0,7 para o ativo de renda fixa e 0,3 para o de renda variável.

As Tabelas 6.1, 6.2 e 6.3 apresentam mais informações sobre as instâncias. No restante deste capítulo, utiliza-se o nome genérico presente nessas tabelas para fazer

Tabela 6.1: Informações das instâncias do grupo A.

Nome genérico das instâncias	Grupo	N	T	Número de cenários
2-8	A	2	8	256
2-9	A	2	9	512
2-10	A	2	10	1.024
2-11	A	2	11	2.048
2-12	A	2	12	4.096
2-13	A	2	13	8.192

Tabela 6.2: Informações das instâncias do grupo B.

Nome genérico das instâncias	Grupo	N	T	Número de cenários
3-8	B	3	8	6.561
3-9	B	3	9	19.683
3-10	B	3	10	59.049
4-8	B	4	8	65.536

Tabela 6.3: Informações das instâncias do grupo C.

Nome genérico das instâncias	Grupo	N	T	Número de cenários
2-3	C	2	3	8
3-3	C	3	3	27
4-3	C	4	3	64
5-3	C	5	3	125
6-3	C	6	3	216
7-3	C	7	3	343
8-3	C	8	3	512
9-3	C	9	3	729
10-3	C	10	3	1.000
11-3	C	11	3	1.331
2-5	C	2	5	32
3-5	C	3	5	243
4-5	C	4	5	1.024
5-5	C	5	5	3.125
6-5	C	6	5	7.776
7-5	C	7	5	16.807
8-5	C	8	5	32.768
9-5	C	9	5	59.049
10-5	C	10	5	100.000
11-5	C	11	5	161.051

referência a um conjunto de 10 instâncias, todas definidas por uma dupla comum de valores N-T. Portanto, as instâncias 2-5, por exemplo, designam as 10 instâncias cujo número de ramificações por período, N, é 2 e o número de períodos, T, é 5.

6.2 Parâmetros dos algoritmos

Os algoritmos implementados neste trabalho são definidos por vários parâmetros. A seguir, esses parâmetros são listados, descritos e, posteriormente, discute-se a forma como cada um desses foi determinado para a realização dos experimentos computacionais. Os parâmetros são:

- ρ_0: é o fator de penalidade inicial do PH;
- μ: é o número de iterações consecutivas que uma variável binária deve satisfazer as restrições de não antecipatividade para que seu valor seja fixado;
- *Threads*: é o número de unidades de instruções sequenciais do código implementado que são executadas de forma independente e paralela;
- δ: é o fator de atualização do parâmetro de penalidade, utilizado na equação (3.2.1);
- G: números de cenários que compõem um agrupamento;
- M: número máximo de iterações permitido na execução dos algoritmos;
- ϵ: é o valor para a condição de parada do algoritmo, dada pela desigualdade (5.16).

Para determinar os valores apropriados dos parâmetros ρ, μ, G e M para realização dos experimentos computacionais, foram realizados testes com diferentes instâncias, escolhidas de forma arbitrária a partir de cada grupo de instâncias. A Tabela 6.4 mostra os valores definidos nos testes para os parâmetros ρ, μ e M. Além disso, essa tabela mostra também valores escolhidos para o parâmetro *Threads*.

Tabela 6.4: Parâmetros dos algoritmos.

Versão	ρ_0	μ	Threads	δ	M	ϵ
PH1	$\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$	X	X	X	50	0,05
PH2	$\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$	1	8	X	50	0,05
PH3	$\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$	1	8	2	50	{0,05, 0,1, 0,15, 0,2, 0,25, 0,3}

PH4	$\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$	1	8	2	50	0,05

X indica que o parâmetro não se aplica à versão.

Como discutido por Zehtabian & Bastin [2016], o parâmetro ρ é um importante fator na convergência do PH. Segundo Helgason & Wallace [1991], os valores de ρ devem ser pequenos, mas não devem ser escolhidos valores de ρ demasiadamente pequenos, pois isso supostamente tornaria a convergência do PH mais lenta. Tendo em conta essas observações, procurou-se testar um conjunto variado de valores de ρ_0 que permitissem observar a convergência dos algoritmos implementados.

Na fase preliminar de testes, definiu-se $\rho_0 = 10^p$ e testou-se os valores de ρ_0 para vários valores de $p \in \mathbb{Z}$. Constatou-se que, para $p > -3$, o valores de *gap*, definido pela equação

$$gap = \frac{SOL_H - SOL_O}{SOL_O} \tag{6.1}$$

em que SOL_H é a solução da heurística utilizada (ou sejas, as versões do PH) e SOL_O é a solução ótima obtida aplicando-se o *solver* do CPLEX, apresentam um caráter oscilatório ao longo das iterações dos algoritmos. Por outro lado, para $p < -6$, constatou-se que o os valores de *gap* diminuem nas primeiras iterações, mas mantêm-se estáveis em seguida. Por essa razão, os experimentos computacionais posteriores foram realizados somente para $-6 \leq p \leq -3$.

Os valores de μ testados foram $\{1, 3, 5\}$. A partir dos testes, observou-se que em todas as instâncias testadas, os valores de *gap* obtidos foram os mesmos para os três valores de μ. O tempo de execução, no entanto, foi menor para $\mu = 1$. Por essa razão, fixou-se $\mu = 1$ para os experimentos computacionais realizados posteriormente.

Com relação ao parâmetro G, como apontam Ryan et al. [2013], a determinação desse é basicamente feita de forma empírica, sendo necessário achar um valor nem tão grande, que torne o subproblema complexo demais, e nem tão pequeno, para que seja proveitosa a agregação de cenários. Dessa forma, o parâmetro G foi definido para cada grupo de instâncias de forma independente. Para as instâncias do grupo A, foi definido somente um valor de G para todas as instâncias. Já para as instâncias do grupo B foram definidos dois valores diferentes: um valor para as instâncias com $N = 3$ e outro para as instâncias com $N = 4$. Para o grupo C, foi escolhido um valor de G para cada conjunto de instâncias definido por valores comuns de N. A Tabela 6.5 mostra os valores de G determinados a partir da fase de testes.

Tratando-se do parâmetro δ, escolheu-se o valor $\delta = 2$, o mesmo utilizado por

Tabela 6.5: Valores do parâmetro G para cada conjunto de instâncias.

Grupo	Nome genérico das instâncias	G
A	2-8	128
A	2-9	128
A	2-10	128
A	2-11	128
A	2-12	128
A	2-13	128
B	3-8	2.187
B	3-9	2.187
B	3-10	2.187
B	4-8	1.024
C	2-5	8
C	3-5	27
C	4-5	64
C	5-5	125
C	6-5	216
C	7-5	343
C	8-5	512
C	9-5	729
C	10-5	1.000
C	11-5	1.331

Gul [2010], para avaliação dos algoritmos PH3 e PH4 sobre todas as instâncias.

Os parâmetros M e ϵ fazem parte da condição de parada de todos os algoritmos implementados. Na fase de testes, observou-se que para $\epsilon < 0,05$, os valores de *gap* obtidos de algumas instâncias diminuíam inicialmente, mas eventualmente, a partir de uma determinada iteração dos algoritmos, mantinham-se constantes. Por outro lado, para $\epsilon > 0,05$, observou-se que os valores de *gap* obtidos eram maiores do que os mesmos obtidos quando $\epsilon = 0,05$. Por essa razão, fixou-se $\epsilon = 0,05$ para a realização da maior parte dos experimentos deste trabalho. O parâmetro ϵ somente foi variado para se estudar o efeito desse parâmetro sobre o *gap* e o número de iterações da versão PH3 quando aplicada às instâncias 10-5. Em relação ao parâmetro M, fixado o valor de ϵ, verificou-se que, para a maioria das instâncias testadas, o algoritmos terminavam sua execução antes de se completarem 50 iterações. Nos casos em que os algoritmos não terminaram com menos de 50 iterações, observou-se, de maneira geral (e principalmente no caso da versão PH1), que o *gap* não diminuiu nas iterações acima desse limite.

O parâmetro *Threads* foi definido como o número de núcleos do computador no qual os experimentos deste trabalho foram realizados. Na próxima seção, são apresentados os recursos computacionais disponíveis para a realização deste trabalho.

6.3 Experimentos Computacionais

Após a fase de definição dos parâmetros dos algoritmos, partiu-se para a execução dos experimentos computacionais. A seguir, apresenta-se os recursos computacionais utilizados na execução desses experimentos e, em seguida, aborda-se a organização e estrutura das tabelas que contêm os resultados dos experimentos realizados.

6.3.1 Recursos computacionais

Os experimentos deste trabalho foram executados em computador com processador Intel Core i7-10510U com frequência 1.80GHz, 16 GB de memória RAM disponíveis e sistema operacional Ubuntu, versão 20.04, instalado. A versão utilizada do CPLEX foi a 12.9.0.

6.3.2 Organização das tabelas

As Tabelas 6.6, 6.7 6.8, 6.9, 6.10 e 6.11 mostram os resultados dos experimentos computacionais conduzidos neste trabalho.

Na Tabela 6.6, são mostrados os resultados da aplicação do CPLEX para cada conjunto de instâncias, definido pelo par de valores N-T. Nessa tabela, para cada dupla N-T, são mostrados os valores médios, obtidos dos resultados de 10 instâncias, do tempo de execução, em segundos, (indicado por "Tempo médio (s)") e valor da função objetivo (indicado por "FO média"). Além dessas informações, também é mostrado o valor do desvio médio absoluto (representado por *MAD*, de *mean absolute deviation*), calculado a partir do valor da função objetivo de cada instância e a média do valor da função objetivo obtida das 10 instâncias que possuem os mesmos valores de N e T. A Tabela 6.7 é semelhante à Tabela 6.6, porém os resultados se referem somente às instâncias com $T = 3$ estágios do grupo C. Além disso, nessa tabela, os valores apresentados foram calculados sobre 1.000 instâncias.

Na Tabela 6.8, são mostrados os resultados obtidos pela aplicação das versões PH1, PH2, PH3 e PH4 sobre as instâncias do grupo A. Nessa tabela, para cada conjunto de instâncias, definido por um par de valores N-T, são mostrados os valores médio, obtidos a partir dos resultados de 10 instâncias, da função objetivo ("FO média"), do número de iterações ("Iter. média"), do tempo de execução, em segundos, ("Tempo médio (s)") e do *gap*, em porcentagem ("GAP médio (%)"), além da versão do algoritmo ("Versão").

Nas Tabelas 6.9 e 6.10, as colunas representam as mesmas informações contidas nas colunas da Tabela 6.8. No entanto, a Tabela 6.9 mostra os resultados das aplicações

Tabela 6.6: Resultados da aplicação do CPLEX às instâncias.

N	T	Tempo médio (s)	FO média	MAD
2	5	0,0	19038,57	1,645,16
2	8	0,0	32748,71	4,056,49
2	9	0,0	33604,63	3,953,26
2	10	0,0	43068,82	7,023,02
2	11	2,0	51023,27	5,290,05
2	12	6,0	54340,42	5,228,14
2	13	19,0	60906,11	7,240,75
3	5	0,0	19405,51	1,125,85
3	8	5,1	29741,28	2,127,50
3	9	32,4	34678,01	3,302,05
3	10	185,6	39728,62	5,675,06
4	5	0,0	18497,82	1,353,00
4	8	132,9	26948,14	1,947,25
5	5	0,0	18989,21	1,442,53
6	5	2,0	18813,33	981,98
7	5	6,5	17978,05	866,49
8	5	17,2	18683,70	815,60
9	5	45,2	18237,83	676,13
10	5	100,7	18532,01	469,21

das versões PH2, PH3 e PH4 para resolver as instâncias do grupo B. Já a Tabela 6.10 mostra os resultados da versão PH3 quando aplicada às instâncias do grupo C.

Por fim, na Tabela 6.11, são mostrados os resultados da versão PH3 aplicada as instâncias 10-5, quando o valor de ϵ foi variado. As colunas "Iter. média"e "GAP médio (%)"referem-se, respectivamente, ao número médio de iterações e aos *gaps* médios obtidos.

Tabela 6.7: Resultados do CPLEX aplicado às instâncias com $T = 3$ estágios.

Ramificação	Períodos	MAD
2	3	1646,80
3	3	1159,28
4	3	949,68
5	3	819,78
6	3	742,59

Continua na próxima página

Ramificação	Períodos	MAD
7	3	665,62
8	3	612,28
9	3	567,52
10	3	538,51
11	3	492,90

Tabela 6.8: Resultados das versões PH1, PH2, PH3 e PH4 sobre as instâncias do grupo A.

				FO média	Iter. média	Tempo médio (s)	GAP médio (%)
N	T	Versão	ρ_0				
2	8	1	10^{-6}	36880,99	50,0	15,0	13,02
			10^{-5}	35541,27	50,0	15,1	8,76
			10^{-4}	32906,39	49,3	14,9	0,47
			10^{-3}	33550,61	50,0	15,1	2,62
2	8	2	10^{-6}	33398,22	26,0	0,5	2,20
			10^{-5}	33172,42	26,0	0,5	1,43
			10^{-4}	32753,09	15,3	0,2	0,02
			10^{-3}	32962,94	21,4	0,4	0,75
2	8	3	10^{-6}	33398,22	26,0	0,5	2,20
			10^{-5}	33172,42	26,0	0,5	1,43
			10^{-4}	32753,09	15,3	0,2	0,02
			10^{-3}	32962,94	21,4	0,4	0,75
2	8	4	10^{-6}	33374,14	26,0	0,5	2,12
			10^{-5}	32991,00	23,8	0,5	0,84
			10^{-4}	32770,26	17,5	0,3	0,08
			10^{-3}	33346,94	21,4	0,4	2,15
2	9	1	10^{-6}	38534,62	50,0	34,5	14,81
			10^{-5}	35876,38	50,0	34,6	6,74
			10^{-4}	33718,21	50,0	34,5	0,36
			10^{-3}	38286,44	50,0	34,4	14,62
2	9	2	10^{-6}	35366,07	35,6	1,4	5,29
			10^{-5}	34690,66	35,6	1,4	3,23

Continua na próxima página

N	T	Versão	ρ_0	FO média	Iter. média	Tempo médio (s)	GAP médio (%)
			10^{-4}	33608,11	18,6	0,5	0,01
			10^{-3}	33864,10	17,0	0,6	0,76
2	9	3	10^{-6}	35366,07	35,6	1,4	5,29
			10^{-5}	34717,62	35,6	1,4	3,30
			10^{-4}	33682,38	16,8	0,4	0,23
			10^{-3}	33614,30	12,9	0,4	0,03
2	9	4	10^{-6}	35283,40	35,6	1,4	5,05
			10^{-5}	34155,14	33,8	1,3	1,60
			10^{-4}	33616,13	16,1	0,4	0,04
			10^{-3}	34406,89	17,0	0,6	2,38
2	10	1	10^{-6}	47839,34	50,0	88,4	11,10
			10^{-5}	44273,49	50,0	87,8	2,73
			10^{-4}	43730,74	50,0	88,0	1,73
			10^{-3}	58563,35	50,0	88,2	39,46
2	10	2	10^{-6}	44664,66	50,0	4,5	3,78
			10^{-5}	43632,46	45,0	3,9	1,27
			10^{-4}	43103,91	27,6	2,3	0,10
			10^{-3}	44219,35	40,6	3,4	3,02
2	10	3	10^{-6}	44676,59	50,0	4,0	3,82
			10^{-5}	43906,37	47,0	3,9	1,95
			10^{-4}	43258,99	33,8	2,8	0,49
			10^{-3}	43130,74	23,5	1,7	0,14
2	10	4	10^{-6}	44466,53	50,0	4,5	3,29
			10^{-5}	43359,10	42,9	3,8	0,67
			10^{-4}	43083,26	19,0	1,4	0,03
			10^{-3}	43968,62	40,6	3,5	1,91
2	11	1	10^{-6}	56802,02	50,0	224,7	11,43
			10^{-5}	51615,91	50,0	225,3	1,16
			10^{-4}	52627,95	50,0	225,2	3,27
			10^{-3}	92366,20	50,0	225,3	83,35
2	11	2	10^{-6}	53638,29	50,0	11,2	5,22
			10^{-5}	51272,41	47,8	10,4	0,49

Continua na próxima página

N	T	Versão	ρ_0	FO média	Iter. média	Tempo médio (s)	GAP médio (%)
			10^{-4}	51071,00	41,5	9,3	0,10
			10^{-3}	55258,62	50,0	11,0	8,41
2	11	3	10^{-6}	53753,56	50,0	9,5	5,45
			10^{-5}	52940,71	50,0	9,4	3,85
			10^{-4}	51342,55	41,5	7,9	0,64
			10^{-3}	51024,54	19,2	3,4	0,00
2	11	4	10^{-6}	53337,84	50,0	11,0	4,64
			10^{-5}	51461,86	47,0	10,3	0,86
			10^{-4}	51335,77	28,1	6,0	0,75
			10^{-3}	55077,24	46,9	10,3	7,89
2	12	1	10^{-6}	58922,47	50,0	660,3	8,29
			10^{-5}	54692,69	50,0	660,8	0,63
			10^{-4}	62082,09	50,0	661,1	14,34
			10^{-3}	197411,60	50,0	657,9	266,73
2	12	2	10^{-6}	56585,91	50,0	24,0	4,06
			10^{-5}	54395,05	36,7	17,3	0,09
			10^{-4}	54697,32	46,0	22,1	0,66
			10^{-3}	73802,94	50,0	23,9	37,34
2	12	3	10^{-6}	57422,78	50,0	20,8	5,64
			10^{-5}	56551,73	50,0	21,5	4,03
			10^{-4}	54461,90	21,4	9,2	0,20
			10^{-3}	54351,40	24,0	10,2	0,02
2	12	4	10^{-6}	56697,85	50,0	24,0	4,31
			10^{-5}	54658,78	43,1	20,6	0,57
			10^{-4}	55326,85	36,3	17,3	1,77
			10^{-3}	55652,83	47,1	22,6	2,50
2	13	1	10^{-6}	64936,22	50,0	2058,0	6,57
			10^{-5}	61769,04	50,0	2059,0	1,50
			10^{-4}	83100,15	50,0	2059,0	38,54
			10^{-3}	408606,30	50,0	2057,3	595,37
2	13	2	10^{-6}	63537,57	50,0	51,7	4,34
			10^{-5}	60926,40	44,1	45,5	0,03

Continua na próxima página

N	T	Versão	ρ_0	FO média	Iter. média	Tempo médio (s)	GAP médio (%)
			10^{-4}	62426,59	50,0	51,7	2,70
			10^{-3}	118343,55	50,0	51,7	98,40
2	13	3	10^{-6}	66105,64	50,0	45,3	8,75
			10^{-5}	64424,77	50,0	47,0	5,86
			10^{-4}	60912,48	17,2	15,8	0,01
			10^{-3}	61027,19	22,6	19,9	0,24
2	13	4	10^{-6}	64591,92	50,0	51,8	6,21
			10^{-5}	62110,14	47,5	49,2	1,89
			10^{-4}	75405,02	45,7	47,2	27,31
			10^{-3}	108824,75	49,3	51,0	78,69

Tabela 6.9: Resultados das versões PH2, PH3 e PH4 sobre as instâncias do grupo B.

N	T	Versão	ρ_0	FO média	Iter. média	Tempo médio (s)	GAP médio (%)
3	8	2	10^{-6}	29741,28	20,8	22,1	0,00
			10^{-5}	29747,08	17,4	18,7	0,02
			10^{-4}	30172,23	26,2	27,5	1,54
			10^{-3}	42631,98	35,6	36,9	43,14
3	8	3	10^{-6}	29741,28	20,8	18,6	0,00
			10^{-5}	29741,28	13,2	11,8	0,00
			10^{-4}	29741,28	13,6	12,1	0,00
			10^{-3}	32147,50	24,4	21,7	7,34
3	8	4	10^{-6}	29741,28	16,9	16,4	0,00
			10^{-5}	29742,48	17,3	16,7	0,00
			10^{-4}	30107,66	22,0	21,4	1,33
			10^{-3}	40229,13	32,0	30,9	34,30
3	9	2	10^{-6}	34703,61	37,8	138,7	0,07
			10^{-5}	34720,22	23,8	89,1	0,12
			10^{-4}	38624,55	50,0	182,7	11,03
			10^{-3}	124349,20	50,0	181,9	254,45
3	9	3	10^{-6}	35125,43	45,3	136,3	1,27

Continua na próxima página

N	T	Versão	ρ_0	FO média	Iter. média	Tempo médio (s)	GAP médio (%)
			10^{-5}	34727,60	21,9	67,9	0,14
			10^{-4}	34940,57	18,6	58,6	0,76
			10^{-3}	36099,81	25,9	79,5	5,46
3	9	4	10^{-6}	34694,93	27,2	97,9	0,05
			10^{-5}	34731,98	19,9	72,8	0,16
			10^{-4}	35901,48	34,7	124,3	3,75
			10^{-3}	77452,40	48,0	170,7	120,82
3	10	2	10^{-6}	39738,33	31,3	348,5	0,02
			10^{-5}	40294,07	45,7	502,1	1,54
			10^{-4}	65286,75	50,0	551,7	68,16
			10^{-3}	403869,80	50,0	552,0	965,37
3	10	3	10^{-6}	41305,81	50,0	439,6	3,84
			10^{-5}	39739,22	13,3	124,8	0,02
			10^{-4}	39728,62	14,7	135,7	0,00
			10^{-3}	39728,76	25,9	232,7	0,00
3	10	4	10^{-6}	40328,07	43,3	474,5	1,33
			10^{-5}	40224,52	26,4	295,1	1,04
			10^{-4}	53643,21	45,3	497,2	32,03
			10^{-3}	66905,31	50,0	545,5	66,69
4	8	2	10^{-6}	26971,04	35,4	249,1	0,10
			10^{-5}	28349,85	50,0	349,0	5,51
			10^{-4}	61879,39	50,0	351,2	131,70
			10^{-3}	463610,70	50,0	349,3	1641,79
4	8	3	10^{-6}	27313,50	50,0	295,5	1,33
			10^{-5}	26948,14	8,9	55,5	0,00
			10^{-4}	26974,81	22,6	135,3	0,10
			10^{-3}	26948,62	25,2	150,4	0,00
4	8	4	10^{-6}	26972,86	37,2	268,9	0,09
			10^{-5}	34217,52	41,2	299,4	25,63
			10^{-4}	33244,38	50,0	361,6	23,89
			10^{-3}	227626,27	50,0	360,3	792,83

Tabela 6.10: Resultados da versão PH3 sobre as instâncias do $T = 5$ estágios do grupo C.

				FO média	Iter. média	Tempo médio (s)	GAP médio (%)
N	T	Versão	ρ_0				
2	5	3	10^{-6}	19791,47	45,2	0,0	4,11
			10^{-5}	19730,67	45,2	0,0	3,78
			10^{-4}	19402,56	36,8	0,0	1,97
			10^{-3}	19071,66	15,1	0,0	0,17
3	5	3	10^{-6}	20621,77	50,0	0,0	6,27
			10^{-5}	20372,92	50,0	0,0	4,99
			10^{-4}	19662,01	38,2	0,0	1,30
			10^{-3}	19405,51	9,4	0,0	0,00
4	5	3	10^{-6}	19490,16	50,0	2,0	5,27
			10^{-5}	19283,60	50,0	2,0	4,13
			10^{-4}	18542,32	26,1	0,8	0,23
			10^{-3}	18652,76	19,6	0,4	0,86
5	5	3	10^{-6}	20157,68	50,0	7,4	6,10
			10^{-5}	19692,54	50,0	7,5	3,64
			10^{-4}	18989,36	13,4	1,6	0,00
			10^{-3}	18989,21	17,0	2,3	0,00
6	5	3	10^{-6}	19766,10	50,0	18,2	5,07
			10^{-5}	19026,14	50,0	17,6	1,14
			10^{-4}	18815,31	14,5	4,9	0,01
			10^{-3}	21085,19	20,0	6,9	12,39
7	5	3	10^{-6}	18669,10	50,0	39,7	3,77
			10^{-5}	17988,74	26,9	21,6	0,06
			10^{-4}	17978,05	13,8	10,9	0,00
			10^{-3}	17978,05	19,8	15,8	0,00
8	5	3	10^{-6}	19346,73	50,0	79,9	3,51
			10^{-5}	18683,70	8,8	13,7	0,00
			10^{-4}	18684,24	17,6	27,9	0,00
			10^{-3}	18683,70	21,3	33,4	0,00
9	5	3	10^{-6}	18634,33	50,0	156,4	2,14
			10^{-5}	18239,29	19,0	59,8	0,01
			10^{-4}	18237,83	17,3	54,8	0,00
			10^{-3}	18237,83	23,6	73,9	0,00

10	5	3	10^{-6}	18736,45	50,0	291,0	1,09
			10^{-5}	18532,01	14,4	84,1	0,00
			10^{-4}	18532,01	19,2	113,1	0,00
			10^{-3}	18532,01	25,0	145,5	0,00
11	5	3	10^{-6}	18446,43	43,5	440,8	X
			10^{-5}	18395,75	13,6	141,6	X
			10^{-4}	18395,75	20,1	204,2	X
			10^{-3}	18396,67	31,1	315,6	X

X indica que não foi possível aplicar o CPLEX.

Tabela 6.11: Resultados das versão PH3 aplicada às instâncias 10-5, quando variou-se o valor de ϵ.

ϵ	ρ_0	Iter. média	GAP médio (%)
0,05	10^{-6}	50,0	1,09
0,05	10^{-5}	14,4	0,00
0,05	10^{-4}	19,2	0,00
0,05	10^{-3}	25,0	0,00
0,10	10^{-6}	50,0	1,09
0,10	10^{-5}	10,7	0,00
0,10	10^{-4}	17,7	0,00
0,10	10^{-3}	24,1	0,00
0,15	10^{-6}	45,7	1,10
0,15	10^{-5}	10,2	0,00
0,15	10^{-4}	17,1	0,00
0,15	10^{-3}	23,5	0,00
0,20	10^{-6}	32,4	1,13
0,20	10^{-5}	9,6	0,01
0,20	10^{-4}	16,7	0,02
0,20	10^{-3}	23,0	0,01
0,25	10^{-6}	27,9	1,16
0,25	10^{-5}	9,2	0,03
0,25	10^{-4}	16,3	0,03
0,25	10^{-3}	22,6	0,02

Continua na próxima página

ϵ	ρ_0	Iter. média	GAP médio (%)
0,30	10^{-6}	19,2	1,29
0,30	10^{-5}	9,0	0,05
0,30	10^{-4}	15,6	0,08
0,30	10^{-3}	22,1	0,08

6.4 Comparação das versões do PH1, PH2, PH3 e PH4 aplicadas às instâncias do grupo A

Nesta seção, são comparados os resultados obtidos da aplicação dos algoritmos PH1, PH2, PH3 e PH4 para resolver as instâncias do grupo A. Para se fazer essas comparações, analisa-se as Figuras 6.1–6.6, que foram criadas a partir dos dados da Tabela 6.8. A Figura 6.1 mostra as distribuições do valores médios de *gap* das quatro versões do PH. A partir da Figura 6.1, é possível observar que há maior variabilidade nos valores de *gap* obtidos a partir da aplicação da versão PH1 quando comparamos às distribuições dos *gaps* obtidas pelas versões PH2, PH3, e PH4. Além disso, é possível notar que a mediana dos *gaps* referentes à versão PH1 é consideravelmente maior que as medianas para as demais versões do PH.

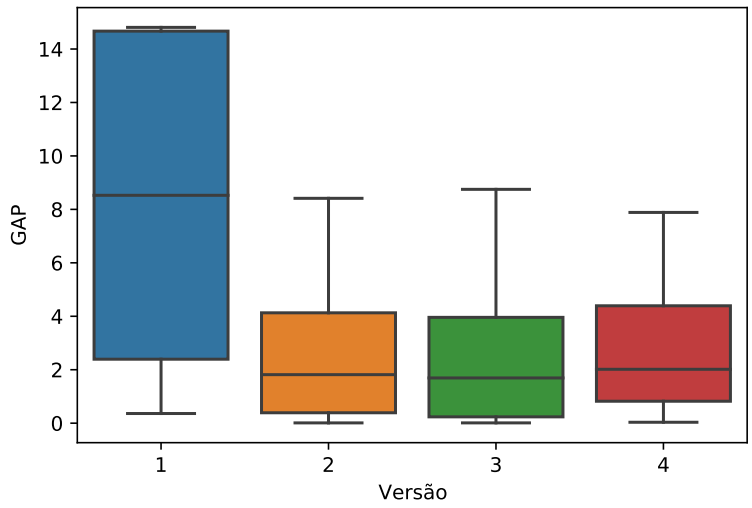

Figura 6.1: Distribuição dos *gaps* médios, extraídos da Tabela 6.8, para cada versão de PH.

Quanto ao número médio de iterações, a Figura 6.2 apresenta as distribuições dos

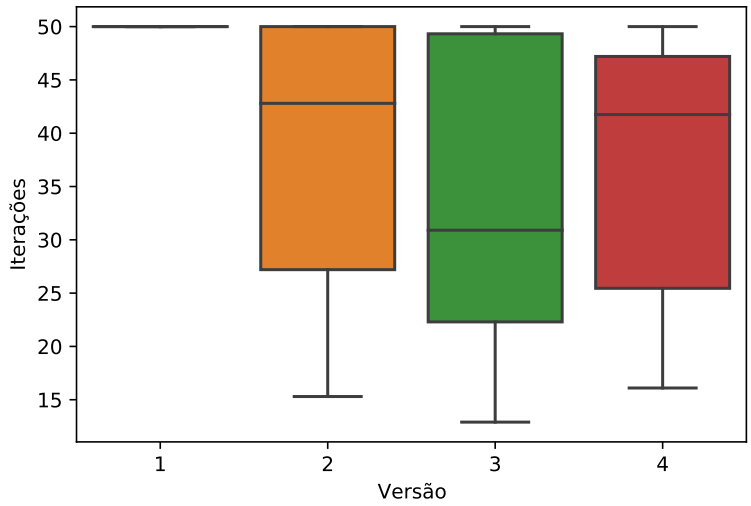

Figura 6.2: Distribuição dos números médios de iterações, extraídos da Tabela 6.8, para cada versão de PH.

números médios de iterações das quatro versões do PH. Pode-se observar, a partir da Figura 6.2, que a distribuição do número médio de iterações relativa à versão PH1 é a que apresenta menor variabilidade em relação a esse parâmetro. No entanto, observa-se também, a partir da Tabela 6.8, que, para a versão PH1, o número médio de iterações para a maioria dos conjuntos de instâncias atinge o limite de 50 iterações. Observa-se também na Figura 6.2 que a mediana relativa à versão PH3 é a menor, quando comparadas as medianas das quatro versões.

Para comparar os tempos médios de execução, as Figuras 6.3–6.6 mostram os tempos médios de execução das quatro versões do PH quando os valores de ρ_0 são, respectivamente, 10^{-3}, 10^{-4}, 10^{-5} e 10^{-6}. A partir dessas figuras, pode-se notar que PH1 se destaca das outras versões em termos de tempo de execução. Apesar dos tempos médios de execução das versões PH2, PH3 e PH4 crescerem à medida que são resolvidas instâncias com mais períodos, como se observa pela Tabela 6.8, esses crescimentos são imensamente menores quando comparados ao crescimento do tempo médio de execução da versão PH1. Para exemplificar essa diferença de comportamento dos algoritmos, compara-se o tempo médio de execução para resolver as instâncias com 12 períodos com o tempo médio pare resolver as instâncias de 11 períodos, quando escolhe-se $\rho_0 = 10^{-4}$. Enquanto há um aumento aproximado de 1,71 vezes no tempo médio de execução ao resolver as instâncias de 12 períodos com a versão PH3, o aumento no tempo médio de execução é de, aproximadamente, 3,11 vezes quando a versão PH1 é utilizada para resolver essas instâncias.

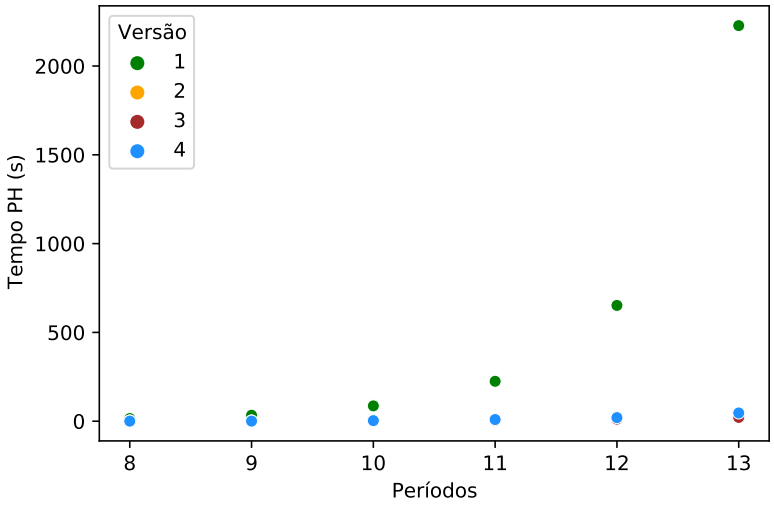

Figura 6.3: Tempo médio de execução em função do número de períodos da instância, para $\rho_0 = 10^{-3}$. Os dados foram extraídos da Tabela 6.8.

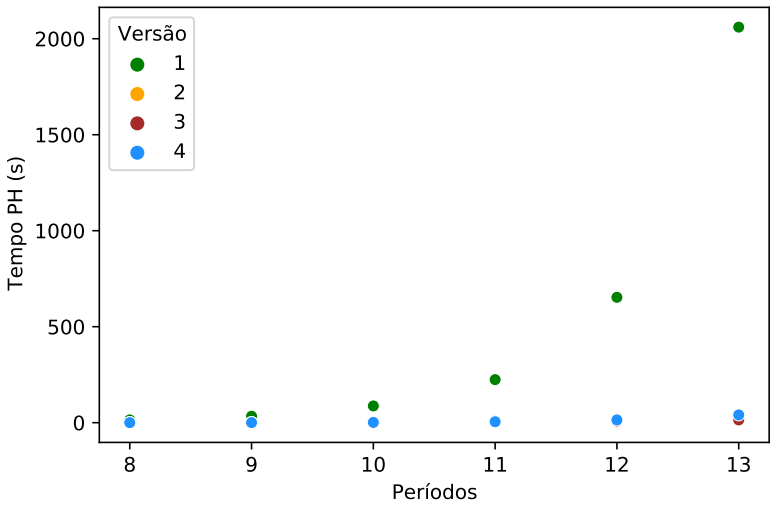

Figura 6.4: Tempo médio de execução em função do número de períodos da instância, para $\rho_0 = 10^{-4}$. Os dados foram extraídos da Tabela 6.8.

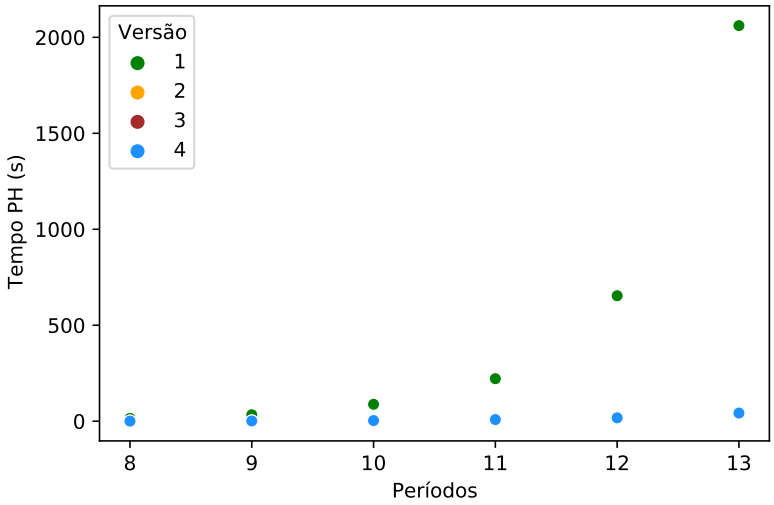

Figura 6.5: Tempo médio de execução em função do número de períodos da instância, para $\rho_0 = 10^{-5}$. Os dados foram extraídos da Tabela 6.8.

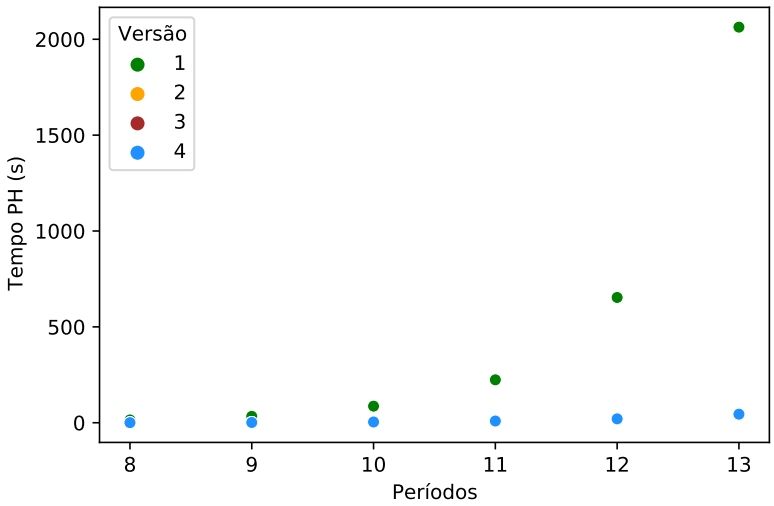

Figura 6.6: Tempo médio de execução em função do número de períodos da instância, para $\rho_0 = 10^{-6}$. Os dados foram extraídos da Tabela 6.8.

Para finalizar essa seção, pontua-se que os resultados apresentados constituem evidências de que as melhorias propostas para o PH original, brevemente apresentadas no Capítulo 3, podem afetar significativamente o desempenho desse algoritmo, tanto em termos de qualidade das soluções encontradas quanto do tempo de execução. Essas evidências somam-se a resultados já encontrados na literatura [Garcia-Gonzalo et al., 2020], [Ryan et al., 2013], [Zehtabian & Bastin, 2016]. Porém, como mostram Zehtabian & Bastin [2016], através de uma revisão da literatura, a eficácia da atualização do parâmetro de penalidade, como proposta de melhoria para o PH, depende do problema em estudo. Na próxima seção, investiga-se os efeitos dessa e de outras melhorias por meio da comparação entre as versões PH2, PH3 e PH4 quando aplicadas às instâncias do grupo B.

6.5 Comparação das versões PH2, PH3 e PH4, aplicadas às instâncias do grupo B

Nesta seção, comparam-se os resultados obtidos para as versões PH2, PH3 e PH4 quando aplicadas para resolver as instâncias do grupo B. As comparações são orientadas pela análise das Figuras 6.7-6.16. Essas figuras foram construídas a partir dos dados da Tabela 6.9.

A Figura 6.7 mostra as distribuições do valores médios de *gap* das três versões do PH. A partir dessa figura, é possível notar que a distribuição dos valores médios de *gap* relativa à versão PH3 é a que apresenta menor variabilidade dentre as três versões. Por meio de uma inspeção da Tabela 6.9, pode-se verificar que o valor máximo do *gap* médio para a versão PH3 é igual a 7,34%, enquanto que os valores máximos para as versões PH2 e PH4 são, respectivamente, iguais a 1641,79% e 792,83%.

A Figura 6.8 apresenta as distribuições dos números médios de iterações das três versões do PH. Pela análise dessa figura, é possível constatar que a distribuição do número médio de iterações da versão PH3 é a que apresenta menor variabilidade, além de apresentar a menor mediana, das três versões.

As Figuras 6.9–6.12 mostram os tempos médios de execução para instâncias 3-8, 3-9 e 3-10, quando os valores de ρ_0 são, respectivamente, 10^{-3}, 10^{-4}, 10^{-5} e 10^{-6}. Observando-se a Figura 6.9, constata-se que os tempos médios de execução da versão PH3 são consideravelmente menores que o das outras versões para as instâncias 3-9 e 3-10. Pode-se verificar resultados semelhantes na Figura 6.10, onde notam-se diferenças ainda mais expressivas entres os tempos médios da versão PH3 e das outras versões. Na Figura 6.11, essa diferença notável no tempo médio de execução entre a versão PH3

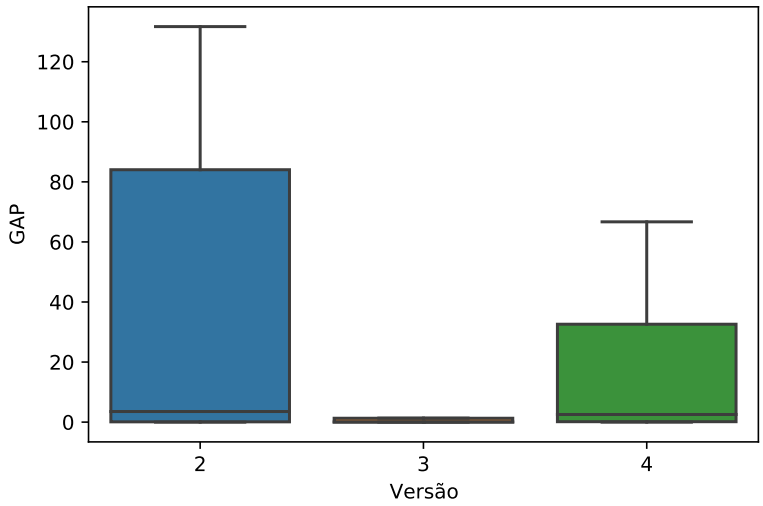

Figura 6.7: Distribuição dos *gaps* médios. Os dados foram extraídos da Tabela 6.9.

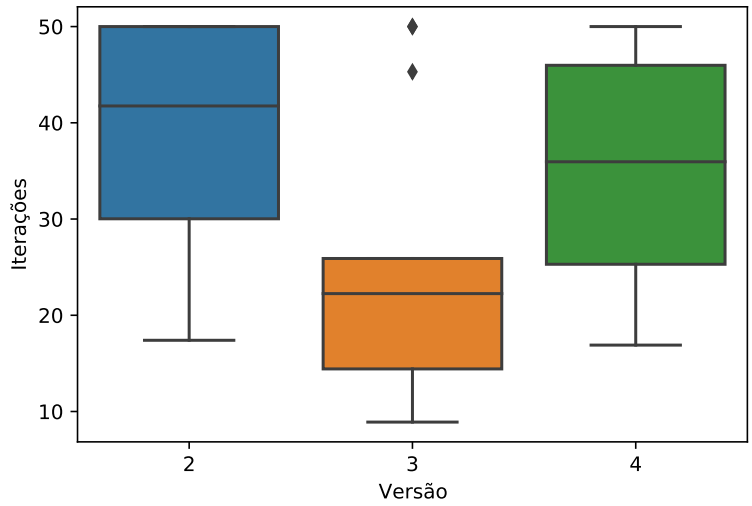

Figura 6.8: Distribuição dos números médios de iterações. Os dados foram extraídos da Tabela 6.9.

e as demais somente se verifica para as instâncias 3-10. Na Figura 6.12, verifica-se que o tempo médio de execução da versão PH3 já não é o menor para as instâncias em questão.

De forma semelhante às figuras apresentadas anteriormente, as Figuras 6.13–6.16 mostram os tempos médios de execução das versões PH2, PH3 e PH4. Esses tempos médios, no entanto, referem-se a aplicação dessas versões sobre as instâncias 4-8. Cada uma dessas figuras mostra resultados relativos a um valor específico de ρ_0.

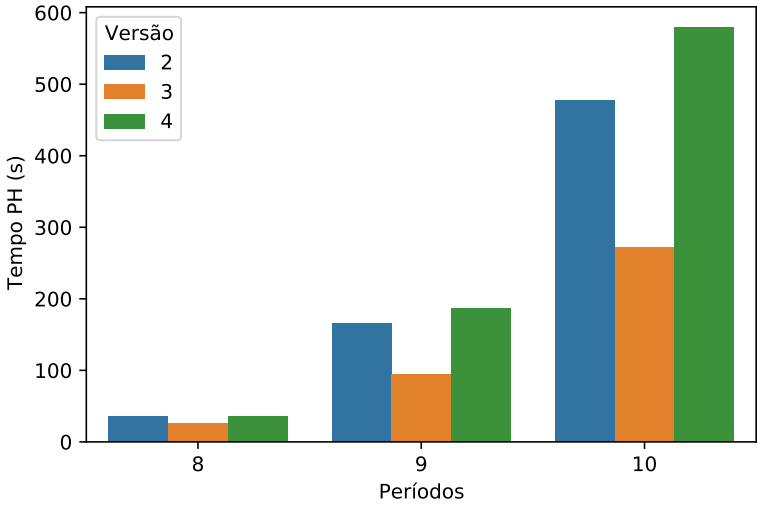

Figura 6.9: Tempo médio de execução em função do número de períodos da instância, com $N = 3$ e $\rho_0 = 10^{-3}$. Os dados foram extraídos da Tabela 6.9.

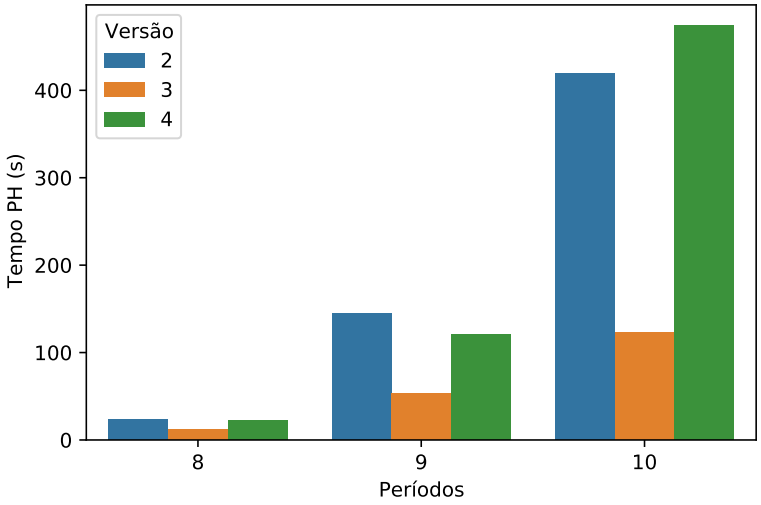

Figura 6.10: Tempo médio de execução em função do número de períodos da instância, com $N = 3$ e $\rho_0 = 10^{-4}$. Os dados foram extraídos da Tabela 6.9.

Ao analisar-se as Figuras 6.13–6.15, nota-se que o tempo médio de execução da versão PH3 é significativamente menor que o das outras duas versões. Com o auxílio da Tabela 6.9, verifica-se que o tempo médio consumido pela versão PH3 representa aproximadamente 43% e 41%, respectivamente, dos tempos médios consumidos pelas versões PH2 e PH4 no caso em que $\rho_0 = 10^{-3}$. A diferença entre os tempos médios de execução da versão PH3 em relação às demais é ainda mais notável para o caso em

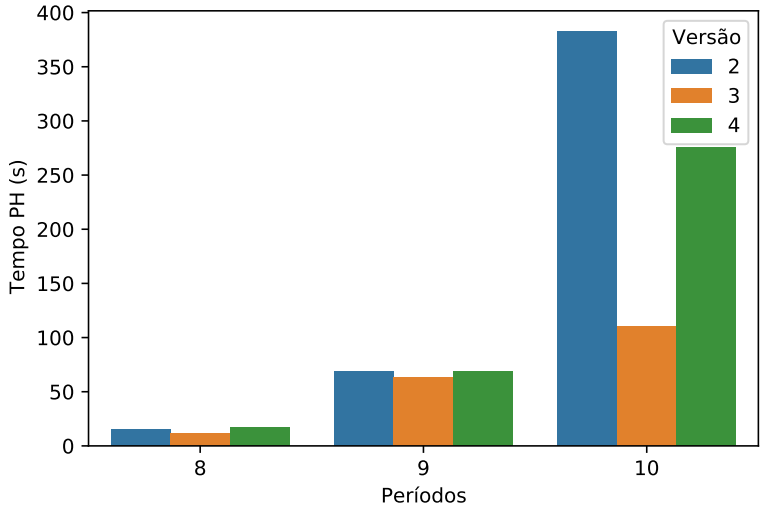

Figura 6.11: Tempo médio de execução em função do número de períodos da instância, com $N = 3$ e $\rho_0 = 10^{-5}$. Os dados foram extraídos da Tabela 6.9.

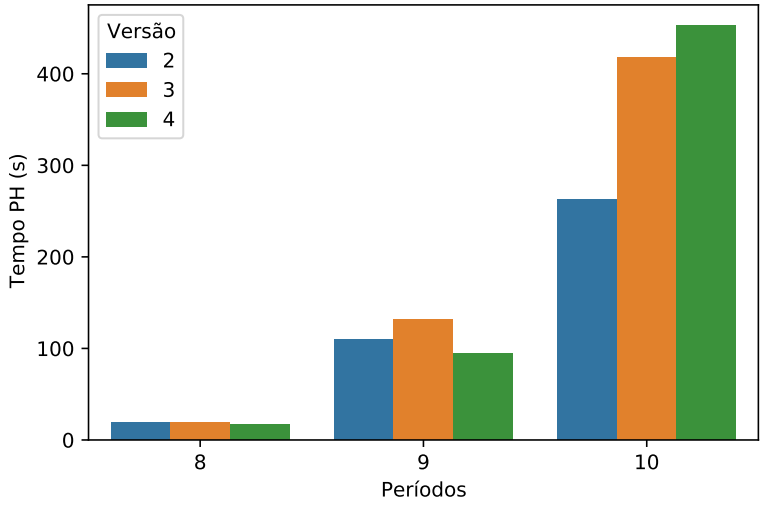

Figura 6.12: Tempo médio de execução em função do número de períodos da instância, com $N = 3$ e $\rho_0 = 10^{-6}$. Os dados foram extraídos da Tabela 6.9.

que $\rho_0 = 10^{-5}$. Nesse caso, é possível verificar, a partir da Tabela 6.9, que o tempo médio consumido pela versão PH3 representa aproximadamente 16% e 18% do tempo médio consumido pelas versões PH2 e PH4. Torna-se importante salientar que, tanto para as instâncias 3-8 quanto para as instâncias 4-8, os tempos médios de execução da versão PH3 são consideravelmente menores que os tempos das outras versões para a maior parte dos valores de ρ_0 utilizados.

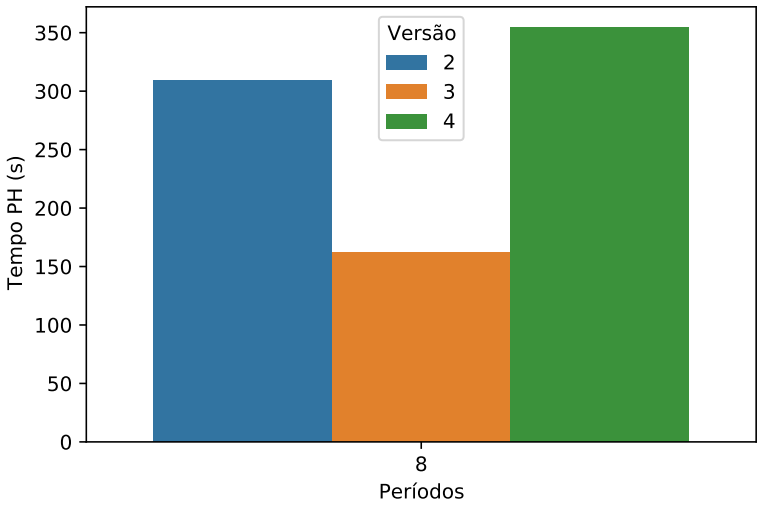

Figura 6.13: Tempo médio de execução na resolução das instâncias 4-8, quando $\rho_0 = 10^{-3}$. Os dados foram extraídos da Tabela 6.9.

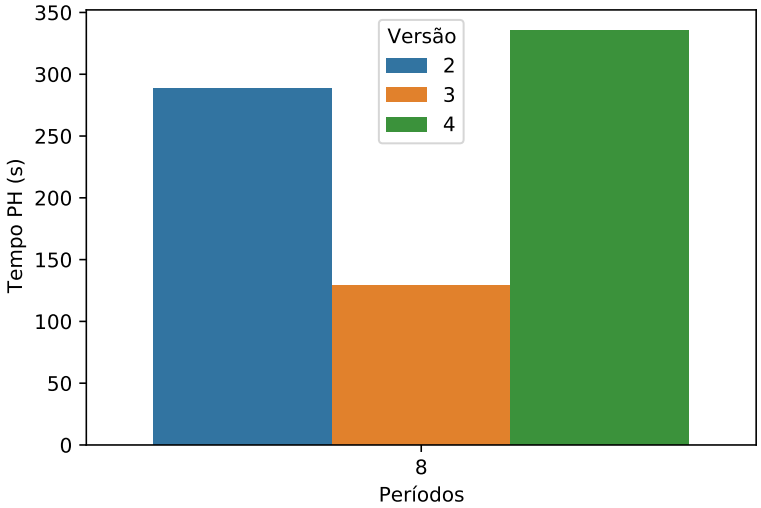

Figura 6.14: Tempo médio de execução na resolução das instâncias 4-8, quando $\rho_0 = 10^{-4}$. Os dados foram extraídos da Tabela 6.9.

Diante dos resultados apresentados nessa seção, que consideram tanto a qualidade das soluções encontradas como a eficiência dos algoritmos em termos de iterações e tempo de execução, elegeu-se a versão PH3 como a melhor versão do PH implementada neste trabalho. Além disso, esses resultados indicam que, para o problema estudado neste trabalho, a atualização do parâmetro de penalidade dado pela equação (5.2.3), de forma geral, é melhor que a forma de atualização proposta por Gul [2010], dada

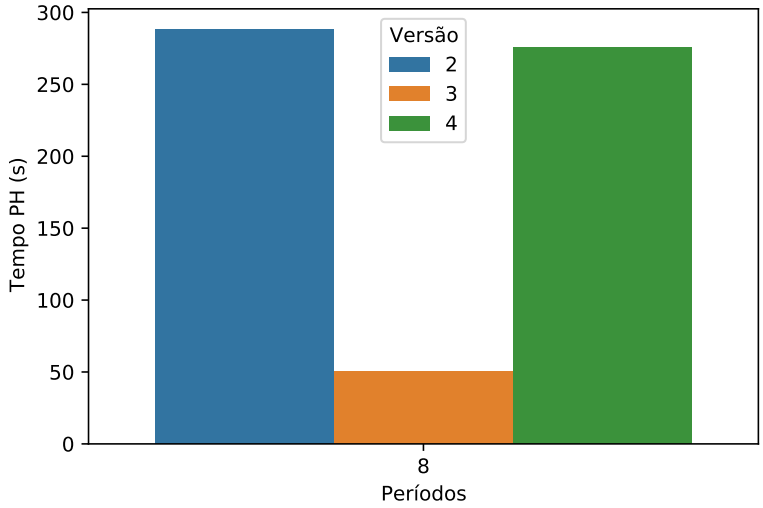

Figura 6.15: Tempo médio de execução na resolução das instâncias 4-8, quando $\rho_0 = 10^{-5}$. Os dados foram extraídos da Tabela 6.9.

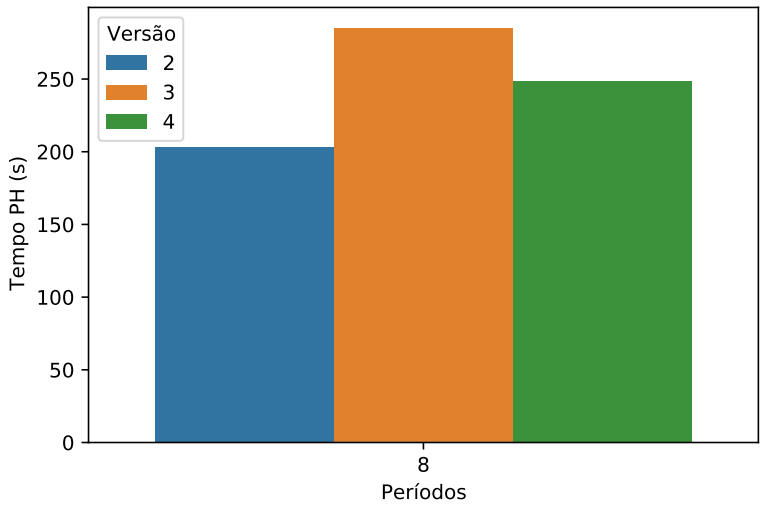

Figura 6.16: Tempo médio de execução na resolução das instâncias 4-8, quando $\rho_0 = 10^{-6}$. Os dados foram extraídos da Tabela 6.9.

pela equação (5.15).

Na próxima seção, compara-se os resultados versão PH3 com os do CPLEX, quando esses algoritmos foram utilizados para resolver as instâncias do grupo C.

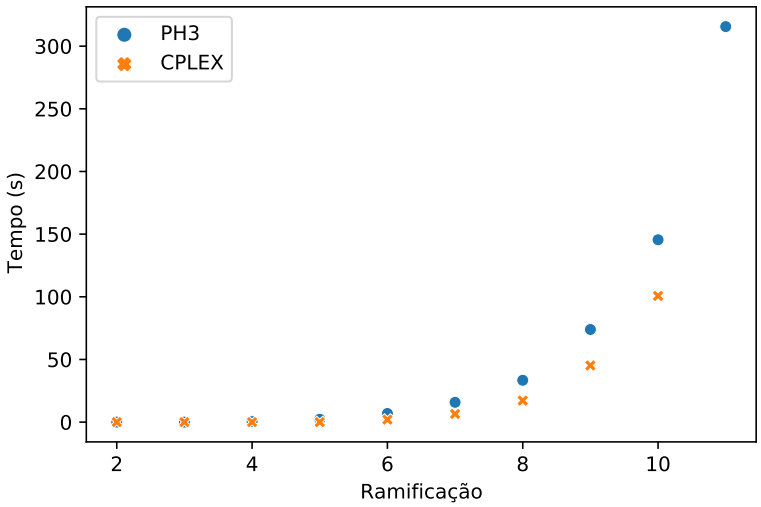

Figura 6.17: Tempos médios de execução da versão PH3 e do CPLEX para instâncias do grupo C. Esses resultados se referem aos experimentos realizados com $\rho_0 = 10^{-3}$. Esse gráfico foi construído com dados da Tabela 6.10.

6.6 Comparação da versão PH3 com CPLEX, aplicados às instâncias do grupo C

Nesta seção, compara-se os resultados obtidos através da versão PH3 com o resultados do CPLEX, quando esses algoritmos foram utilizados para resolver instâncias do grupo C. As comparações são apresentadas a partir da análise das Figuras 6.17–6.20, que mostram os tempos médios de execução obtidos da aplicação da versão PH3 e do CPLEX sobre todas as instâncias do grupo C, e da Tabela 6.10. Essas figuras foram produzidas a partir dos dados da Tabela 6.10.

Por meio de uma análise das Figuras 6.17– 6.20, observa-se que os tempos médios de execução do CPLEX são inferiores, de forma geral, aos da versão PH3. Observa-se também que os tempos desses algoritmos aproximam-se quando $\rho_0 = 10^{-4}$. Por sua vez, nota-se que as diferenças entre esses tempos são maiores quando $\rho_0 = 10^{-6}$. Verifica-se também que, para as instâncias 8-5 e 10-5 e escolhendo-se $\rho_0 = 10^{-5}$, o tempo médio de execução do PH3 foi inferior ao tempo médio do CPLEX. Além disso, pode-se verificar, a partir da Tabela 6.10, que os *gaps* médios parecem aumentar a medida que diminui-se os valores ρ_0.

É importante destacar que apesar do CPLEX gastar menos tempo computacional que o PH3 para resolver a maioria das instâncias, as instâncias 11-5, que são as maiores instâncias do conjunto, não puderam ser resolvidas pelo CPLEX por falta de memória.

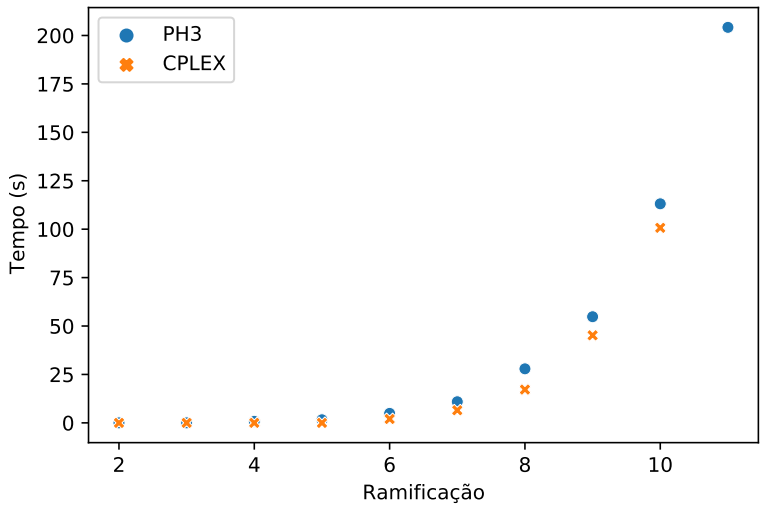

Figura 6.18: Tempos médios de execução da versão PH3 e do CPLEX para instâncias do grupo C. Esses resultados se referem aos experimentos realizados com $\rho_0 = 10^{-4}$. Esse gráfico foi construído com dados da Tabela 6.10.

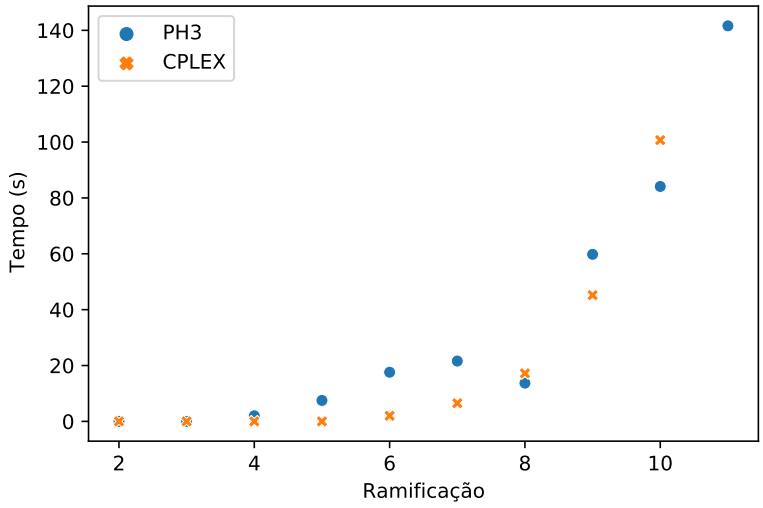

Figura 6.19: Tempos médios de execução da versão PH3 e do CPLEX para instâncias do grupo C. Esses resultados se referem aos experimentos realizados com $\rho_0 = 10^{-5}$. Esse gráfico foi construído com dados da Tabela 6.10.

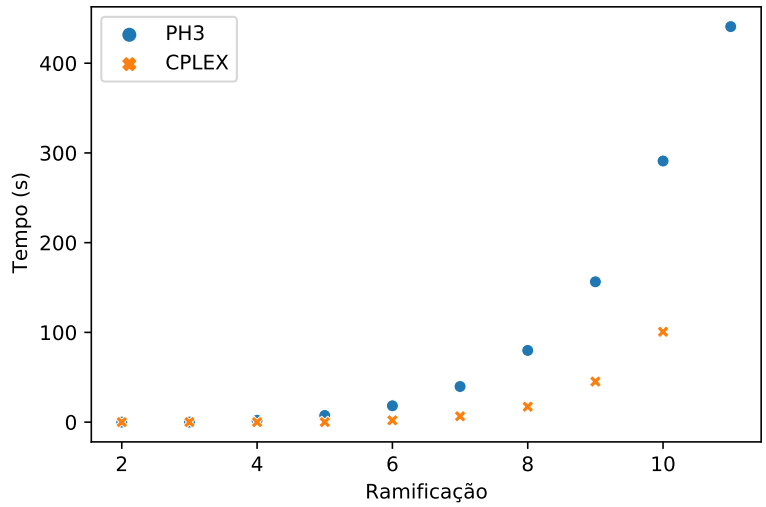

Figura 6.20: Tempos médios de execução da versão PH3 e do CPLEX para instâncias do grupo C. Esses resultados se referem aos experimentos realizados com $\rho_0 = 10^{-6}$. Esse gráfico foi construído com dados da Tabela 6.10.

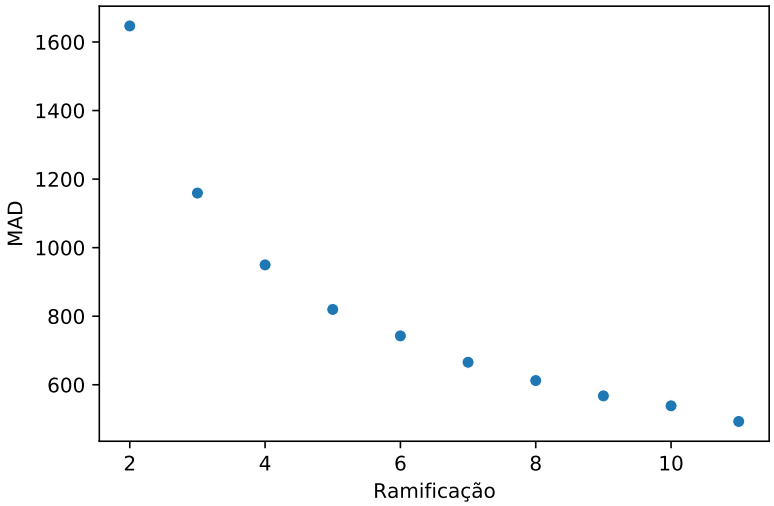

Figura 6.21: Valores de *MAD* obtidos da aplicação do CPLEX às instâncias com $T = 3$ estágios do grupo C.6.7.

No entanto, a versão PH3 foi capaz de resolver todas as instâncias que não puderam ser resolvidas pelo CPLEX. Esse resultado é importante pois, a partir da análise da Figura 6.21, que apresenta os valores de MAD para instâncias com $T = 3$ estágios, observa-se que a variabilidade do valor da função objetivo diminui a medida que aumenta-se o número de ramificações das árvores de cenários. Decorre então a importância de métodos de resolução que consigam resolver árvores com muitos cenários. Portanto, os

resultados apresentados, somados a outros [Perboli et al., 2017], reforçam a importância de algoritmos de decomposição como alternativas aos resolvedores de propósito geral. Pela capacidade de resolver instâncias grandes do problema, a utilização dessa versão do PH também pode ser uma alternativa à estratégia apresentada por de Oliveira et al. [2017] para evitar a utilização de árvores com muitos cenários.

Para finalizar essa seção, pode-se destacar, a partir dos resultados apresentados, a importância do valor de ρ_0 para o tempo de execução e para a qualidade das soluções obtidas pela versão PH3.

Na próxima seção, apresenta-se uma análise de sensibilidade do parâmetro ϵ na condição de parada da versão PH3.

6.7 Análise de sensibilidade do parâmetro ϵ para a versão PH3

Nesta seção, analisa-se como a variação do parâmetro ϵ, que faz parte do critério de parada dos algoritmos implementados neste trabalho, afeta o desempenho da versão PH3. Para fazer essa análise, primeiramente escolheu-se os valores de ϵ que estão apresentados na Tabela 6.11 para a versão PH3. Posteriormente, resolveram-se as instâncias 10-5 com a versão PH3 para cada um dos valores de ϵ escolhidos. Os resultados desses experimentos são mostrados na Tabela 6.11 e nas Figuras 6.22 e 6.23.

Analisando-se a Figura 6.22, observa-se que o *gap* médio cresceu de forma mais expressiva quando escolheu-se $\rho_0 = 10^{-6}$. Também pode-se observar que é aproximadamente na mudança de $\epsilon = 0,25$ para $\epsilon = 0,3$ que ocorre o maior aumento no *gap* médio. Outro ponto a se destacar é que, embora se tenha aumentado ϵ até 0,3, os valores de *gaps* médios permaneceram inferiores a 0,2% para $\rho_0 \in \{10^{-3}, 10^{-4}, 10^{-5}\}$.

Ao se analisar a Figura 6.23 e a Tabela 6.11, verifica-se que o número médio de iterações reduziu, para todos os valores de ρ_0, a medida que se aumentou ϵ. Além disso, nota-se que a redução mais significativa ocorreu para $\rho_0 = 10^{-6}$. Nesse caso, o número médio de iterações reduziu-se para menos da metade quando avalia-se a mudança de $\epsilon = 0,05$ para $\epsilon = 0,3$.

Por fim, destaca-se a partir dos resultados apresentados que há um *trade-off*, como haveria de se esperar, entre aumentar valor da tolerância do critério de parada, ϵ, e a redução no número de iterações do algoritmo PH3. Portanto, aumentando-se o valor de ϵ pode-se obter a resolução de instâncias do problema de forma mais rápida. Isso significa que se, no contexto do problema do fundo de pensão, o gestor do fundo aceitar soluções cujo *gap* seja maior que 0%, usar a versão PH3, fazendo-se ajustes

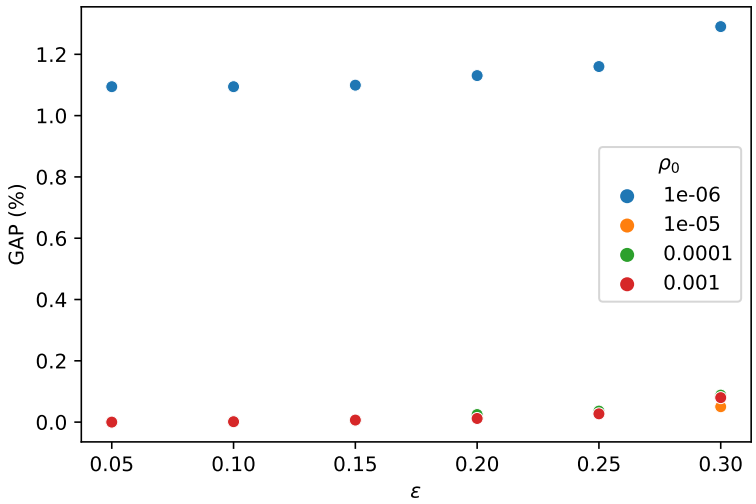

Figura 6.22: *Gaps* médios em função do parâmetro ϵ.

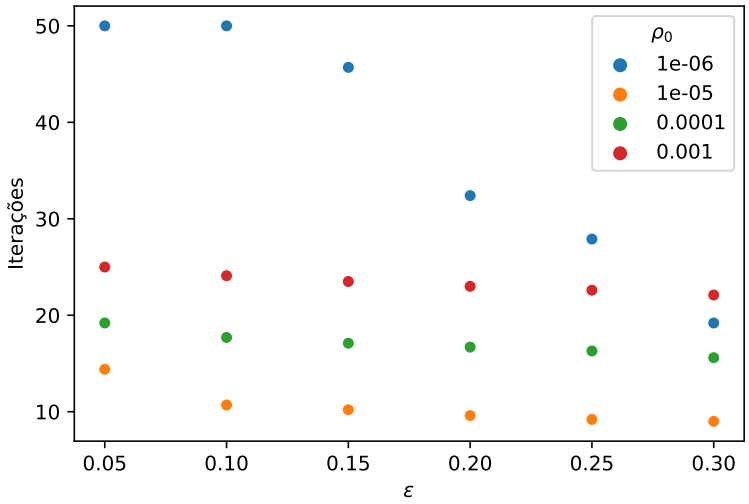

Figura 6.23: Números médios de iterações em função do parâmetro ϵ.

no valor de ϵ, pode ser uma alternativa mais eficiente que a utilização de *solvers* de propósito geral.

Capítulo 7

Conclusões e Sugestão de Trabalhos Futuros

7.1 Conclusões

Neste trabalho, o objetivo principal foi desenvolver métodos de resolução que resolvessem instâncias grandes de um problema de ALM de fundos de pensão, cujo o modelo foi proposto por de Oliveira et al. [2017]. No desenvolvimento deste trabalho, o modelo de otimização do problema foi implementado e os cenários das instâncias utilizadas foram criadas a partir de séries temporais históricas.

Para que o objetivo deste trabalho fosse alcançado, foram implementadas 4 diferentes versões do algoritmo PH, que foram denominadas PH1, PH2, PH3 e PH4, para que se escolhesse a que melhor cumprisse o objetivo proposto. A versão PH1 corresponde ao algoritmo original, proposto por Rockafellar & Wets [1991], sem as modificações apresentadas no capítulo 3. A versão PH2 consiste no algoritmo original, acrescido de três modificações propostas na literatura e já apresentadas no capítulo 3, que são: agrupamento de cenários, fixação de variáveis e paralelização. As versões PH3 e PH4 consistem, cada uma delas, em modificações específicas da versão PH2, que incluem formas de atualização do parâmetro de penalização.

Para eleger a melhor versão implementada do PH, primeiro comparou-se as diferentes versões desse algoritmo a partir das estatísticas dos valores médios dos *gaps* relativos às soluções encontradas, dos números de iterações até o término dos algoritmos e dos tempos de execução consumidos. O processo de comparação entre as diferentes versões do PH foi dividido em duas etapas sucessivas. Primeiro foram comparadas as versões PH1, PH2, PH3 e PH4, utilizando-se o grupo de instâncias A. Depois, foram

comparadas somente as versões PH2, PH3 e PH4, utilizando-se o grupo de instâncias B, composto por instâncias cujas árvores são maiores que as do grupo A.

Na fase em que todas as versões foram comparadas, observou-se que a versão PH1 apresenta tempos de execução consideravelmente maiores que os tempos das outras versões. Além disso, notou-se também que os *gaps* médios relativos às soluções obtidas pela utilização da versão PH1 são, de forma geral, maiores que os *gaps* médios obtidos utilizando-se as outras versões. Também observou-se que a versão PH1 atinge, para quase todas as instâncias e valores de ρ_0, o limite de 50 iterações, enquanto que o mesmo não ocorre para as outras versões do PH. Por esses motivos, a versão PH1 foi considerada ineficiente e descartada nas etapas de análise posteriores.

Na fase em que somente as versões PH2, PH3 e PH4 foram comparadas, foi observado que a distribuição dos *gaps* médios referentes às soluções encontradas pela versão PH3 foi a que apresentou menor variabilidade dentre as três versões. Além disso, o valor máximo do *gap* médio referente à versão PH3 é consideravelmente menor que os valores máximos dos *gaps* médios referentes às versões PH2 e PH4.

Ainda nessa última fase de comparação, constatou-se que a distribuição dos números médios de iterações da versão PH3 foi a que apresentou menor variabilidade, além de ter apresentado a menor mediana, das três versões. Constatou-se também que, de forma geral, a versão PH3 apresentou tempos médios de execução consideravelmente menores que as outras versões. Considerando todas observações, elegeu-se a versão PH3 como a melhor.

Em seguida, comparou-se a versão PH3 com o CPLEX na resolução das instâncias do grupo C. O propósito nessa etapa foi avaliar o desempenho desses algoritmos para resolver instâncias à medida que aumenta-se o número de cenários das árvores. Para tanto, foram criadas instâncias cujas árvores possuem um número fixo de períodos, mas com número variado de ramificações por período. Nessa fase de comparações, observou-se que os tempos médios de execução do CPLEX são inferiores, de forma geral, aos da versão PH3. No entanto, constatou-se que, para as instâncias 8-5 e 10-5, quando foi escolhido $\rho_0 = 10^{-5}$, os tempos médios de execução da versão PH3 foram inferiores aos do CPLEX. A partir dos resultados obtidos, foi também possível observar a importância do valor de ρ_0 para o tempo de execução e para a qualidade das soluções obtidas pela versão PH3. Cabe também destacar que a versão PH3 foi capaz de resolver instâncias que não puderam ser resolvidas pelo CPLEX, por falta de memória.

Após a fase de comparação entre a versão PH3 e o CPLEX, realizou-se uma breve análise de sensibilidade do parâmetro ϵ na versão PH3. A partir dos resultados obtidos, observou-se que, aumentando-se o valor de $\epsilon = 0,05$ para $\epsilon = 0,3$, houve uma diminuição considerável no número médio de iterações para que o algoritmo terminasse,

principalmente no caso em que $\rho_0 = 10^{-6}$. No entanto, o aumento em ϵ resultou também em um aumento nos *gaps* médios referentes às soluções obtidas.

Para finalizar, este trabalho apresenta um algoritmo alternativo para que gestores de fundos de pensão possam resolver grandes instâncias do problema de ALM.

7.2 Sugestão de Trabalhos Futuros

Tendo em vista os recursos computacionais disponíveis para a realização deste trabalho, como sugestão para trabalhos futuros, recomenda-se um estudo mais minucioso da estratégia de paralelização do PH3 em ambientes de computação de alto desempenho. Recomenda-se também a investigação de outras estratégias de atualização do parâmetro ρ_0, dado que na literatura da área existem diversas estratégias descritas e testadas. Outra possibilidade de trabalho futuro consiste no estudo comparativo entre o PH e o algoritmo *Nested-Benders*. Por fim, uma outra alternativa de trabalho futuro consistiria na aplicação dos algoritmos implementados a instâncias reais.

Referências Bibliográficas

Almeida, J. S. d. et al. (2013). Análise de desempenho de estratégias no algoritmo de progressive hedging quando aplicado na solução do problema de planejamento da operação energética.

Beraldi, P. & Bruni, M. E. (2014). A clustering approach for scenario tree reduction: an application to a stochastic programming portfolio optimization problem. *Top*, 22(3):934--949.

Birge, J. R. & Louveaux, F. (2011). *Introduction to stochastic programming*. Springer Science & Business Media.

Boender, G. C. (1997). A hybrid simulation/optimisation scenario model for asset/liability management. *European Journal of Operational Research*, 99(1):126--135.

Bradley, S. P. & Crane, D. B. (1972). A dynamic model for bond portfolio management. *Management science*, 19(2):139--151.

Carino, D. R.; Kent, T.; Myers, D. H.; Stacy, C.; Sylvanus, M.; Turner, A. L.; Watanabe, K. & Ziemba, W. T. (1994). The russell-yasuda kasai model: An asset/liability model for a japanese insurance company using multistage stochastic programming. *Interfaces*, 24(1):29--49.

Consigli, G. & Dempster, M. A. (1998). Dynamic stochastic programmingfor asset-liability management. *Annals of Operations Research*, 81:131--162.

Cornuejols, G. & Tütüncü, R. (2006). *Optimization methods in finance*, volume 5. Cambridge University Press.

de Oliveira, A. D. & Filomena, T. P. (2020). Compacting multistage stochastic programming models through a new implicit extensive framework. *Journal of Computational Science*, p. 101128.

de Oliveira, A. D.; Filomena, T. P.; Perlin, M. S.; Lejeune, M. & de Macedo, G. R. (2017). A multistage stochastic programming asset-liability management model: an application to the brazilian pension fund industry. *Optimization and Engineering*, 18(2):349--368.

de Oliveira, A. D.; Filomena, T. P. & Righi, M. B. (2018). Performance comparison of scenario-generation methods applied to a stochastic optimization asset-liability management model. *Pesquisa Operacional*, 38(1):53--72.

Dong, C.; Huang, G.; Tan, Q. & Cai, Y. (2014). Coupled planning of water resources and agricultural landuse based on an inexact-stochastic programming model. *Frontiers of Earth Science*, 8(1):70--80.

Garcia-Gonzalo, J.; Pais, C.; Bachmatiuk, J.; Barreiro, S. & Weintraub, A. (2020). A progressive hedging approach to solve harvest scheduling problem under climate change. *Forests*, 11(2):224.

Gonçalves, R. E. C. et al. (2011). Desenvolvimento de modelos de programação estocástica aplicados à programação mensal da operação energética: uma análise comparativa de desempenho computacional.

Gondzio, J. & Kouwenberg, R. (2001). High-performance computing for asset-liability management. *Operations Research*, 49(6):879--891.

Gul, S. (2010). *Optimization of surgery delivery systems*. Tese de doutorado, Arizona State University.

Helgason, T. & Wallace, S. W. (1991). Approximate scenario solutions in the progressive hedging algorithm. *annals of Operations Research*, 31(1):425--444.

Høyland, K. & Wallace, S. W. (2001). Generating scenario trees for multistage decision problems. *Management science*, 47(2):295--307.

King, A. J. & Wallace, S. W. (2012). *Modeling with stochastic programming*. Springer Science & Business Media.

Kouwenberg, R. & Zenios, S. A. (2008). Chaper 6 - stochastic programming models for asset liability management. Em Zenios, S. & Ziemba, W., editores, *Handbook of Asset and Liability Management*, pp. 253–303. North-Holland, San Diego.

Lauria, D. & Consigli, G. (2017). A defined benefit pension fund alm model through multistage stochastic programming. *International Journal of Finance & Managerial Accounting*, 2(7):1--10.

Majidi-Qadikolai, M. & Baldick, R. (2017). A generalized decomposition framework for large-scale transmission expansion planning. *IEEE Transactions on Power Systems*, 33(2):1635--1649.

Peng, Z.; Zhang, Y.; Feng, Y.; Rong, G. & Su, H. (2019). A progressive hedging-based solution approach for integrated planning and scheduling problems under demand uncertainty. *Industrial & Engineering Chemistry Research*, 58(32):14880--14896.

Perboli, G.; Gobbato, L. & Maggioni, F. (2017). A progressive hedging method for the multi-path travelling salesman problem with stochastic travel times. *IMA Journal of Management Mathematics*, 28(1):65--86.

Pichler, A. & Tomasgard, A. (2016). Nonlinear stochastic programming–with a case study in continuous switching. *European Journal of Operational Research*, 252(2):487--501.

Rockafellar, R. T. & Wets, R. J.-B. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119--147.

Ryan, S. M.; Wets, R. J.-B.; Woodruff, D. L.; Silva-Monroy, C. & Watson, J.-P. (2013). Toward scalable, parallel progressive hedging for stochastic unit commitment. Em *2013 IEEE Power Energy Society General Meeting*, pp. 1–5.

Shapiro, A.; Dentcheva, D. & Ruszczyński, A. (2014). *Lectures on stochastic programming: modeling and theory*. SIAM.

Valladão, D. & Veiga, Á. (2008). Optimum allocation and risk measure in an asset liability management model for a pension fund via multistage stochastic programming and bootstrap. International Conference on Engineering Optimization.

Wallace, S. W. & Fleten, S.-E. (2003). Stochastic programming models in energy. *Handbooks in operations research and management science*, 10:637--677.

Watson, J.-P. & Woodruff, D. L. (2011). Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, 8(4):355--370.

Watson, J.-P.; Woodruff, D. L. & Strip, D. R. (2007). Progressive hedging innovations for a stochastic spare parts support enterprise problem. Relatório técnico SAND-2007-3722J, Sandia National Laboratories, Albuquerque, NM, USA.

Xu, D.; Chen, Z. & Yang, L. (2012). Scenario tree generation approaches using k-means and lp moment matching methods. *Journal of Computational and Applied Mathematics*, 236(17):4561--4579.

Zehtabian, S. & Bastin, F. (2016). *Penalty parameter update strategies in progressive hedging algorithm.* CIRRELT.

Zenios, S. A. & Ziemba, W. T. (2007). *Handbook of Asset and Liability Management: Applications and case studies.* Elsevier.