Pontifícia Universidade Católica
do Rio de Janeiro

Cristiane Salgado Pereira

Métodos de otimização multiobjetivo para programação de petróleo em refinaria utilizando programação genética

Tese de Doutorado

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós–graduação em Engenharia Elétrica da PUC-Rio.

Orientador : Prof.ª Marley Maria Bernardes Rebuzzi Vellasco
Co-orientador: Prof. Douglas Mota Dias

Rio de Janeiro
Dezembro de 2018

Pontifícia Universidade Católica
do Rio de Janeiro

Cristiane Salgado Pereira

Métodos de otimização multiobjetivo para programação de petróleo em refinaria utilizando programação genética

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós–graduação em Engenharia Elétrica da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof.ª Marley Maria Bernardes Rebuzzi Vellasco
Orientador
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Douglas Mota Dias
Co-orientador
Universidade do Estado do Rio de Janeiro – UERJ

Prof.ª Karla Tereza Figueiredo Leite
Universidade do Estado do Rio de Janeiro – UERJ

Prof. Luis Martí Orosa
Universidade Federal Fluminense – UFF

Dr. Marcus Vinicius de Oliveira Magalhães
Industrial – Petrobras

Dr. Paulo Cesar Ribas
CENPES – Petrobras

Prof. Márcio da Silveira Carvalho
Coordenador Setorial do Centro Técnico Científico – PUC-Rio

Rio de Janeiro, 20 de Dezembro de 2018

Cristiane Salgado Pereira

Graduou-se em Engenharia Química pela Universidade Federal do Rio de Janeiro (UFRJ) em abril de 2002. Tornou-se Mestre na área de Métodos de Apoio à Decisão pelo Programa de Pós-Graduação em Engenharia Elétrica, da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) em novembro de 2012.

Ficha Catalográfica

Salgado Pereira, Cristiane

Métodos de otimização multiobjetivo para programação de petróleo em refinaria utilizando programação genética / Cristiane Salgado Pereira; orientador: Marley Maria Bernardes Rebuzzi Vellasco; co-orientador: Douglas Mota Dias. – 2018.

160 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, 2018.

Inclui bibliografia

1. Engenharia Elétrica - Teses;. 2. Otimização multiobjetivo;. 3. Algoritmos evolucionários multiobjetivo;. 4. Programação genética linear com inspiração quântica;. 5. Programação genética orientada à gramática;. 6. Programação de petróleo em refinaria.. I. Vellasco, Marley Maria Bernardes Rebuzzi. II. Dias, Douglas Mota. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. IV. Título.

CDD: 621.3

Agradecimentos

Agradeço a Deus por todas as suas formas de manifestação e por me conceder o privilégio de ter uma vida feliz até aqui.

Aos meus orientadores Marley Vellasco e Douglas Dias, por me guiarem e estarem sempre presentes durante todo o desenvolver deste trabalho.

À Petrobras por ter como um dos seus valores o investimento na capacitação de seus funcionários. Ao Antonio Vicente, Leonardo e Luiz Claudio por estimularem a minha dedicação para que esta tese pudesse ser concluída. Aos companheiros do trabalho que são exemplos de competência e conhecimento.

Aos profissionais da área de programação da produção da Petrobras por sua dedicação e comprometimento neste demandante trabalho e por compartilharem seu conhecimento.

Aos meus pais, Ilda e Valdemir, meus amores. Seu exemplo moldou minha vida e a maneira como me relaciono com o mundo. Seus valores de educação e respeito estão plantados e espero que sempre colham frutos. Incluo aqui os meus amores de quatro patas, estejam por aqui ou não, porque são todos meus grandes companheiros de vida. São família.

Ao Wilson, meu grande companheiro. Obrigada por cada momento que eu precisei buscar força para seguir em frente. Obrigada por pesquisar comandos no Latex, mesmo preferindo o Word (o que eu jamais entenderei).

À família que tantas vezes ouviu "não vai dar, to doutorando". Agora deve vai ficar mais fácil!

Aos amigos... Muitas fases, encontros e desencontros, semelhanças e diferenças. E a certeza de que tê-los em minha vida me faz feliz. A amizade forma um ciclo que nos torna a todos mais completos.

Resumo

Salgado Pereira, Cristiane; Vellasco, Marley Maria Bernardes Rebuzzi; Dias, Douglas Mota. **Métodos de otimização multiobjetivo para programação de petróleo em refinaria utilizando programação genética**. Rio de Janeiro, 2018. 160p. Tese de Doutorado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

A programação de produção em refinaria pode ser compreendida como decisões que buscam otimizar alocação de recursos, o sequenciamento de atividades e a sua realização temporal, respeitando restrições e visando ao atendimento de múltiplos objetivos. Apesar da complexidade e natureza combinatória, a atividade carece de sistemas sofisticados que auxiliem o processo decisório, especialmente baseadas em otimização, pois as ferramentas utilizadas são planilhas ou softwares de simulação. A diversidade de objetivos do problema não implica em equivalência de importância. Pode-se considerar que existem grupos, onde os que afetam diretamente a capacidade produtiva da refinaria se sobrepõem aos associados à maior continuidade operacional. Esta tese propõe o desenvolvimento de algoritmos multiobjetivos para programação de petróleo em refinaria. As propostas se baseiam em conceituadas técnicas da literatura multiobjetivo, como dominância de Pareto e decomposição do problema, integradas à programação genética com inspiração quântica. São estudados modelos em um ou dois níveis de decisão. A diferenciação dos grupos de objetivos é avaliada com base em critérios estabelecidos para considerar uma solução proposta como aceitável e também é avaliada a influência de uma população externa no processo evolutivo. Os modelos são testados em cenários de uma refinaria real e os resultados são comparados com um modelo que trata os objetivos de forma hierarquizada. As abordagens baseadas em dominância e em decomposição apresentam vantagem sobre o algoritmo hierarquizado, e a decomposição é superior. Numa comparação com o modelo em dois níveis de decisão, apenas o que utiliza estratégia de decomposição em cada nível apresenta bons resultados. Ao final deste trabalho é obtido mais de um modelo multiobjetivo capaz de oferecer um conjunto de soluções que atendam aos objetivos críticos e deem flexibilidade de análise *a posteriori* para o programador de produção, o que, por exemplo, permite que ele pondere questões não mapeadas no modelo.

Palavras-chave

Engenharia Elétrica - Teses; Otimização multiobjetivo; Algoritmos evolucionários multiobjetivo; Programação genética linear com inspiração quântica; Programação genética orientada à gramática; Programação de petróleo em refinaria.

Abstract

Salgado Pereira, Cristiane; Vellasco, Marley Maria Bernardes Rebuzzi (Advisor); Dias, Douglas Mota (Co-Advisor). **Multiobjective optimization methods for refinery crude scheduling applying genetic programming**. Rio de Janeiro, 2018. 160p. Tese de doutorado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Refinery scheduling can be understood as a set of decisions which aims to optimize resource allocation, task sequencing, and their time-related execution, respecting constraints and targeting multiple objectives. Despite its complexity and combinatorial nature, the refinery scheduling lacks more sophisticated support decision tools. The main systems in the area are worksheets and, sometimes, simulation software. The multiple objectives do not mean they have the same importance. Actually, they can be grouped whereas the objectives related to the refinery production capacity are more important than the ones related to a smooth operation. This thesis proposes the development of multiobjective algorithms applied to crude oil refinery scheduling. The proposals are based on the major technics of multiobjective literature, like Pareto dominance and problem decomposition, integrated with a quantum-inspired genetic programming approach. One and two decision level models are studied. The difference between groups is handled with conditions that define what can be considered a good solution. The effect of using an archive population in the evolutionary process is also evaluated. The results of the proposed models are compared with another model that handles the objectives in a hierarchical logical. Both decomposition and dominance approaches have better results than the hierarchical model. The decomposition model is even better. The bilevel decomposition method is the only one, among two decision levels models, which have shown good performance. In the end, this work achieves more than one multiobjective model able to offer a set of solutions which comprises the critical objectives and can give flexibility to the production scheduler does his analysis. Therefore, he can consider aspects not included in the model, like the forecast of crude oil batches not scheduled yet.

Keywords

Multiobjective optimization; Evolultionary multiobjective algorithms; Quantum-inspired linear genetic programming; Grammar-based genetic programming; Refinery crude oil scheduling.

Sumário

Lista de figuras

Lista de tabelas

Lista de Abreviaturas

AC-DQIGLGP *Archive constrained decomposition-based quantum inspired grammar-based linear genetic programming*

AC-NSQIGLGP *Archive constrained non-dominated sort quantum inspired grammar-based linear genetic programming*

BiC-CNSQIGLGP *Bilevel constraint constrained non-dominated sort quantum inspired grammar-based linear genetic programmin*

BiC-QIGLGP *Bilevel constraint quantum inspired grammar-based linear genetic programmin*

BiGA *Bilevel genetic algorithm*

BiO-CDQIGLGP *Bilevel objective constrained decomposition-based quantum inspired grammar-based linear genetic programmin*

BLEAQ *Bilevel evolutionary algorithm based on quadratic approximation*

BOP *Bilevel optimization problem*

CD *Crowding distance do NSGA-II*

C-DQIGLGP *Constrained decomposition-based quantum inspired grammar-based linear genetic programming*

C-NSQIGLGP *Constrained non-dominated sort quantum inspired grammar-based linear genetic programming*

CSA Corridas com soluções aceitas

CoBRA *Coevolutionary Bilevel method using repeated algorithm*

DE *Differential evolution*

DL diesel leve

DP diesel pesado

DSL *Domain specific language*

DM *Decision Maker*

DQIGLGP *Decomposition-based quantum inspired grammar-based linear genetic programming*

EMOA *Evolutionary multiobjective algorithm*

FP Frente de Pareto

GA *Genetic Algorithm*

GLP gás liquefeito de petróleo

GP *Genetic Programming*

GP-HH *Genetic Programming based Hyper-Heuristic*

HBLEMO *Hybrid bilevel evolutionary multiobjective algorithm*

HV Hipervolume

IAT Índice de acidez total

IBEA *Indicator based evolutionary algorithm*

ICA *Imperial Colony Algorithm*

IGD *Inverted Generational Distance*

KKT Karush-Kuhn-Tucker

LGP *Linear Genetic Programming*

MaOP *Manyobjective optimization problem*

MDVRP *Multidepot Vehicle Routing Problem*

MOBOP *Multiobjective bilevel optimization problem*

MOEA/D *Multiobjective evolutionary algorithm based on decomposition*

MOEA/D-DE *Multiobjective evolutionary algorithm based on decomposition and differential evolution*

MOGA *Multiobjective genetic algorithm*

MOP *Multiple objective problem*

MOPSO *Multiobjective particle swarm optimization*

NL nafta leve

NP nafta pesada

NP-Hard *Non-deterministic polynomial-time hardness*

NSGA *Non-dominated sorting genetic algorithm*

NSQIGLGP *Non-dominated sort quantum inspired grammar-based linear genetic programming*

PBI *Penalty-based boundary intersection approach*

Q querosene

QIGLGP *Quantum-inspired grammar-based linear genetic programming*

RAT resíduo atmosférico

SMS-EMOA *S metric selection - EMOA*

SOP *Single objective problem*

SPEA *Strength Pareto evolutionary algorithm*

UDA Unidade de destilação atmosféria

VEGA *Vector evolutionary genetic algorithm*

1
Introdução

A atividade de programação de produção existe em diversos segmentos da indústria, onde é responsável por definir como os produtos são produzidos através da alocação de materiais, máquinas e operações numa sequência e momentos definidos. É, portanto, a programação de recursos e processos necessários para uma empresa oferecer serviços ou produtos. Por exemplo, na indústria têxtil, as decisões estão relacionadas ao tamanho, à sequência, ao tempo e à alocação de cada lote de produção e quais fardos de algodão devem ser utilizados na produção [1]. Já na indústria farmacêutica, as decisões são responsáveis por sequenciar os compostos químicos em máquinas de mistura, embaladeiras, análises do controle de qualidade, além da gestão de estoques das matérias-primas e produtos [2]. Em linhas gerais, programar a produção comumente se resume às seguintes decisões: "que tarefas executar?", "onde realizar as tarefas de produção?"(alocação de tarefas aos recursos), "em que sequência produzir?" e "quando realizar as tarefas de produção?" [3].

Decisões na cadeia de suprimento são feitas em três principais níveis, chamados Estratégico, Tático e Operacional [4]. O Estratégico está relacionado aos objetivos de longo prazo, concentrado no caminho que deve ser percorrido e nos recursos necessários para seu alcance. O Tático está relacionado à utilização dos recursos existentes de forma a maximizar a rentabilidade da empresa no curto/médio prazo. Neste nível de decisão, as variáveis tem maior abrangência geográfica, contemplando diferentes unidades produtivas e os principais arcos de movimentação entre elas. O horizonte de visualização e decisão é de meses ou até anos. Por último, o Operacional está focado em realizar as orientações definidas no nível Tático, maximizando a unidade produtiva [1], no caso, a refinaria. A programação de produção encontra-se neste nível.

A Figura 1.1 representa um comparativo entre o planejamento e a programação em uma refinaria, ambos no nível operacional de decisão. O lado esquerdo da figura mostra a base de uma refinaria com a informação necessária para seu planejamento. Esta atividade tem por objetivo a maximização da margem de lucro da unidade produtiva, respeitando as condições e diretrizes definidas no nível tático de decisão, através do planejamento e programação corporativos. Comparativamente ao nível tático, as informações manipuladas

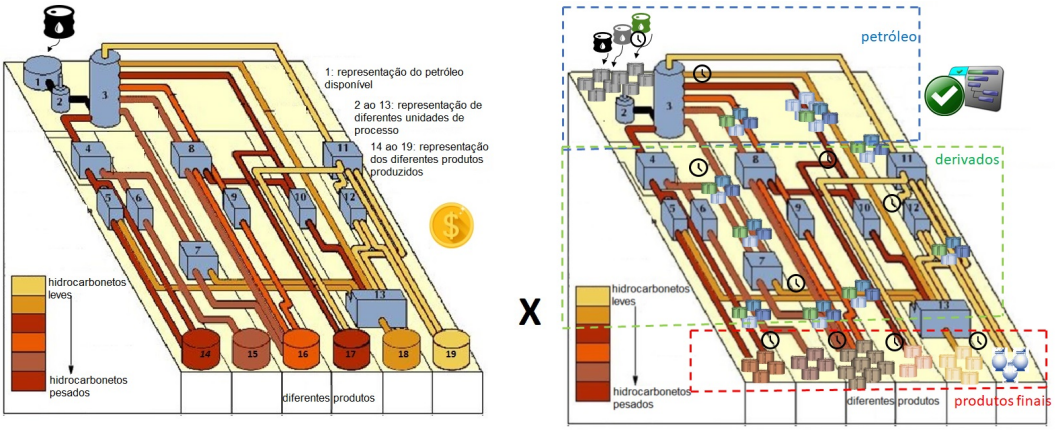

Figura 1.1: Planejamento e Programação na Refinaria

neste nível têm maior detalhamento e somente o que está diretamente relacionado à refinaria é modelado. Graças a uma representação mais aderente dos perfis de rendimento e qualidade das correntes de unidades de processo, das restrições e das flexibilidades de mistura, obtém-se previsões mais acuradas sobre a produção da refinaria para um dado elenco de petróleo. Neste nível são atualizados, principalmente, os volumes totais e produtos intermediários para composição de produto final, além do nível de utilização e modo de operação das unidades de processo. Em geral, este planejamento é refeito com periodicidade mínima mensal.

No lado direito da Figura 1.1, a base do planejamento ganha mais informação e o foco passa a ser a capacidade de transformar as diretrizes do planejamento em um cronograma de atividades viável no dia a dia. A programação na refinaria é responsável por fazer a ligação entre o planejamento e as operações da planta de processo através da distribuição temporal das atividades que foram previstas em etapas anteriores [5]. Além do trabalho para sua elaboração inicial, a programação de produção envolve revisões e atualizações frequentes para responder às variabilidades do ambiente produtivo e mantê-la aderente aos objetivos estabelecidos no planejamento. É comum que a programação da refinaria seja dividida em áreas, cada qual sob a responsabilidade de diferentes programadores, que devem trabalhar integradamente. As linhas pontilhadas desta mesma figura representam o escopo de uma divisão típica: programação de petróleo (azul), programação de derivados (verde) e programação de misturas e produtos finais (vermelho). O horizonte de programação e, portanto, de visualização, geralmente está entre 5 e 10 dias. Este nível hierárquico de decisão é o tema central desta tese e será detalhado ao longo do trabalho, a começar pela seção 1.1.

As refinarias de petróleo constituem o mais importante exemplo de sis-

tema de processamento contínuo, gerador de múltiplos produtos, em função da quantidade e do valor dos produtos gerados [6]. No Brasil, em 2017 foram processados mais de 1,74 milhões de barris de petróleo por dia [7]. Essa informação, combinada com a quantidade de diferentes produtos, preços, rotas de processamento e movimentação, faz com que as atividades de planejamento e programação de produção em refinarias envolvam decisões relacionadas a bilhões de dólares anuais. Sob esta perspectiva, é compreensível que haja interesse no desenvolvimento de métodos que suportem as decisões de programação para maior captura deste potencial.

1.1 Programação de produção em refinaria de petróleo

Para desdobrar os objetivos do plano de produção em atividades de dia a dia, o programador de produção precisa obter, tratar e manipular uma grande quantidade de informações de diferentes naturezas. Por exemplo, é necessário conhecer, minimamente:

- composição, qualidade e níveis de estoque em tanque dos diferentes produtos;
- previsão de recebimento de matéria-prima: data de chegada, composição, qualidade e volume;
- entrega de produtos: datas de entrega e volume;
- equipamentos em manutenção, entrando ou retornando dela;
- variações no desempenho dos equipamentos da planta;
- desvios operacionais relacionados ao não cumprimento da última programação de produção vigente.

Além desses itens de variabilidade diária, o programador de produção deve: conhecer como as diferentes unidades de processo e demais equipamentos interagem; e estar familiarizado com limitações de vazão de carga, rendimentos e quaisquer outros parâmetros que representem eficiência, economia ou restrições físicas do processo produtivo com o qual trabalha [8]. O programador precisa reagir tanto às variabilidades do processo como às do negócio e deve ser capaz de prever as consequências de desvios em relação às atividades programadas. Um dos papéis da programação é conseguir identificar possíveis situações de crise (por exemplo, baixo estoque) antes que elas aconteçam e tentar evitá-las [5].

Ainda que a programação de toda a refinaria seja uma tarefa compartilhada por mais de um programador, o esforço de manipular e conciliar todas essas informações demanda muito tempo, especialmente considerando que

muitas vezes é necessário refazê-la mais de uma vez ao dia. Comercialmente é possível encontrar sistemas que auxiliam no processo de elaboração da programação. No entanto, são soluções baseadas em simulação por eventos com limitado escopo de aplicação de otimização [9]. Associando as características do problema à carência de um ferramental mais complexo para suporte às decisões, o trabalho do programador de produção resulta, geralmente, em um conjunto de atividades operacionalmente viáveis, sem maiores recursos para avaliações de cenário ou otimizações.

Como apresentado na Figura 1.1, em uma refinaria típica brasileira, a programação é dividida nas áreas de: programação de petróleo, programação de derivados e programação de misturas para produtos finais.

O programador de misturas é responsável por definir a participação das correntes de componentes intermediários nas misturas que formam os produtos finais. Para tal, deve considerar a disponibilidade desses intermediários (vazão de produção e estoque) e suas respectivas qualidades. Precisa considerar os potenciais destinos pois, algumas vezes, o mesmo intermediário pode ser usado para dois produtos finais diferentes, como por exemplo, nafta pesada, que pode compôr a mistura de gasolina ou de diesel, dependendo de suas propriedades e/ou da necessidade de produção. O principal objetivo do programador de misturas é garantir que os produtos estejam prontos e certificados (após análises de propriedades que garantem que o produto respeita todas as especificações da Agência Nacional de Petróleo que o qualificam para comercialização) na data em que devem ser entregues. Ainda que não seja possível considerar que existe uma solução consolidada de apoio a esta atividade, este segmento da programação de produção concentra alguns estudos de soluções de otimização, tanto comerciais (por exemplo: MBO da empresa Aspentech [10], Optimix da empresa Princeps [11]) quanto acadêmicas [12–15].

O programador de derivados é responsável por programar as unidades de conversão e tratamento da refinaria. Para tal, deve definir o nível de vazão de carga que as unidades operam, que correntes intermediárias compõem a carga e com que participação. Também deve definir qual o destino das correntes produzidas em todas as unidades (podem ir para tanque, serem processadas em outras unidades imediatamente, compor mistura de produto final ou ter mais de um desses destinos simultaneamente). Considerando que uma refinaria tem dezenas de unidades de processo além da destilação atmosférica e que o programador de derivados deve conhecer as flexibilidades, limitações de operação e previsão de desempenho de todas elas, é possível imaginar a complexidade de preparar esta programação. Entretanto, esta área é o meio da refinaria e sua programação é muito dependente da programação de petróleo e

derivados, o que significa que esforços para uma programação otimizada desta área faz mais sentido se houver uma programação otimizada de petróleo e/ou derivados. Alguns trabalhos propõem a integração planejamento-programação de forma que a representação da programação de petróleo e/ou de derivados é detalhada enquanto a área de intermediários têm modelagem agregada, numa abordagem semelhante ao que é feito nos modelos de planejamento [16–18].

O programador de petróleos é responsável por todas as atividades necessárias ao recebimento de petróleo e seu processamento nas unidades de destilação. Dependendo da autonomia de decisão da refinaria, o escopo de programação é diferente. Por exemplo, uma refinaria que compartilha terminal ou malha de duto com outras refinarias não tem autonomia para programar movimentações antes de suas fronteiras, pois as decisões devem ser tomadas por uma área da empresa que tenha uma visão integrada dos aspectos que envolvem todas as refinarias que compartilham o recurso. Neste cenário, a refinaria terá suas decisões começando em "como receber a matéria-prima na tancagem da refinaria". Já uma refinaria com terminal dedicado deve decidir sobre o recebimento do petróleo no terminal, sua movimentação até a refinaria e seu processamento nas unidades de destilação. A inclusão de mais uma área no escopo de decisão (terminal) aumenta significativamente a complexidade da programação. Contribuições já foram propostas através de desenvolvimentos tanto utilizando otimização matemática quanto algoritmos evolucionários [5, 8, 19–22]. No entanto, apesar da contribuição relevante, a otimização da programação de petróleo ainda não é um problema resolvido. Algumas limitações associadas aos modelos encontrados na literatura são: detalhamento da representação da refinaria (por ex: tanques de armazenamento que não contemplam misturas, não considerar lastro), limitações na representação de composição e propriedades (por ex: não garante que a composição da movimentação é a composição da mistura em tanque), não considerar aspectos de continuidade na programação (movimentações de pouco volume, resultando em muitas trocas de equipamentos onde não se observa benefício).

Independente do método proposto, os trabalhos que representam problemas de programação de produção, em geral, utilizam funções objetivo que representam benefício econômico e/ou com apenas um objetivo. Entretanto, no refino brasileiro, o elenco a ser processado é uma definição da companhia e, portanto, já foi decidido o que a refinaria vai receber. Dessa forma, orientar a programação com base no valor econômico dos petróleos provavelmente fará com que cenários futuros tenham pior valor agregado ou se tornem mais complexos. Aliado a isto, ainda não é comum haver informação suficiente para balizar uma avaliação econômica no tempo de tomada de decisão do nível da

programação.

Em seu dia a dia, o programador trabalha com múltiplos objetivos que representam componentes da missão de manter a refinaria operando continuamente e da forma mais estável possível, ou seja, minimizar variações da carga programada nas unidades de processo, receber os itens de petróleo conforme previsto, minimizar o número de operações de movimentação entre tanques, minimizar a sobrespecificação de produtos (*giveaway*) - que acontece quando produtos são entregues com qualidade muito melhor do que o definido pela regulamentação e não há nenhuma compensação financeira - entre outros.

Dessa forma, são apresentadas as seguintes observações:

- os objetivos operacionais não têm todos a mesma importância e poderiam ser agrupados seguindo este critério. Por exemplo, manter as unidades de processo operando continuamente e minimizar movimentações de troca de tanque são objetivos que estão em níveis de importância diferentes, pois o primeiro é quem garante a produção de derivados, enquanto o segundo é uma característica desejável para evitar muitas operações e regimes transientes na planta. Dessa forma, manter a unidade operando faz parte de um grupo de objetivos mais importante e troca de tanque faz parte de outro grupo de objetivos, com menor importância.
- entre os objetivos que fazem parte de um mesmo grupo de importância, não há uma diferenciação clara sobre o benefício de melhorar um objetivo em detrimento de outro. Por exemplo, há vantagem clara em minimizar troca de tanque incondicionalmente em detrimento de uma operação ininterrupta do duto? Este tipo de dilema indica que é provável a existência de um conjunto de soluções otimizadas que representam uma boa programação (também conhecida pelo termo *schedule*) para a refinaria.

Considerando a natureza de múltiplos objetivos do problema, a carência de tempo e de ferramental de apoio, pode-se afirmar que a solução operacionalmente viável resultante do trabalho quase manual do programador de produção não é a única resposta. Ainda que seja uma solução boa numa relação de compromisso entre objetivos, há outras que podem ser obtidas e apresentadas ao programador para que ele avalie as opções e selecione a que lhe parecer mais adequada sob critérios que talvez não estejam contemplados no modelo.

Problemas de programação de produção pertencem à classe de problemas *NP Hard* [23]. São bastante desafiadores, pois combinam representações discreta e contínua do tempo com centenas de outras variáveis de decisão também discretas (alocação) e contínuas (volumes), além de possuir natureza

combinatória em cada decisão e multiplicidade de objetivos [22]. A computação evolucionária é amplamente utilizada em problemas deste grau de complexidade graças à sua capacidade de gerar soluções otimizadas sem a necessidade de uma representação matemática complexa [24]. A estrutura populacional da computação evolucionária representa outra vantagem destes algoritmos para aplicação em problemas com múltiplos objetivos, pois diversos indivíduos evoluem simultaneamente em busca do ótimo, o que permite que, ao final, seja obtida mais de uma solução otimizada [25].

Dentre as técnicas de computação evolucionária, em [5] é proposto um modelo baseado em programação genética e com inspiração quântica para resolver um problema de programação de produção. Os resultados obtidos por este modelo (nominado *Quantum-Inspired Grammar-based Linear Genetic Programming* - QIGLGP) foram superiores aos obtidos por um modelo baseado em algoritmos genéticos utilizando a abordagem clássica de operadores de cruzamento e mutação. Segundo [26], a inspiração quântica é uma metaheurística promissora para lidar com problemas de programação multiobjetivo, pois apresenta uma efetiva exploração do espaço de busca.

A complexidade do problema, seus múltiplos objetivos, a categorização existente entre eles e a carência de ferramentas que ofereçam diferentes soluções de qualidade ao programador de produção motivaram o desenvolvimento desta tese que propõe uma nova abordagem para programação de petróleo em refinaria, desenvolvida sob os preceitos da programação genética com inspiração quântica e dos algoritmos evolucionários multiobjetivos.

1.2 Objetivos da Pesquisa

Com base nas considerações apresentadas anteriormente, esta tese tem como objetivo principal propor e desenvolver modelos evolucionários com inspiração quântica para otimização de problemas multiobjetivos e que possam ser aplicados a problemas de programação de petróleo em refinaria. As principais características desejadas são:

- Otimização dos diferentes objetivos que são independentes entre si.
- Evolução considerando a diferença de importância entre dois grupos de objetivos.
- Capacidade de apresentar o conjunto das melhores soluções de compromisso entre os objetivos. Estas soluções devem representar uma programação de petróleo viável durante todo o horizonte do cenário e, portanto, também respeitar as restrições do problema que estiverem estabelecidas.

– Capacidade de oferecer soluções de qualidade mesmo em cenários de média ou alta complexidade.
– Gerar todas as soluções em tempo equivalente ao que o programador de produção leva para gerar uma solução utilizando as ferramentas que ele tem disponíveis.
– Flexibilidade para representar diferentes configurações de topologia da refinaria. Por exemplo, se novos tanques estão disponíveis e/ou outros são retirados de operação, o modelo não precisa ser refeito. Se capacidades mínimas e/ou máximas de equipamentos são alteradas, também não afeta a execução do modelo, apenas sua parametrização. Estas informações não têm uma dinâmica de alteração frequente, mas são parte da realidade da refinaria.
– Flexibilidade para considerar diferentes restrições do problema de programação de petróleo em refinaria, como por exemplo, limites de propriedades da mistura de petróleo ou de produtos intermediários derivados da unidade de destilação.
– Aplicável a quaisquer configurações de cenário de programação, como por exemplo, dias de horizonte, número de petróleos manipulados, vazões de equipamentos ou vezes em que diferentes petróleos são recebidos na refinaria.

Com estas características atingidas, o programador de produção que atualmente tem no Excel sua principal ferramenta de apoio às decisões, poderá avaliar um conjunto de diferentes soluções, todas de qualidade, aumentando seu potencial de análise e, dessa forma, oferecendo mais robustez ao processo decisório *a posteriori*.

1.3 Contribuições

A proposta deste trabalho foi desenvolver novos modelos evolucionários com inspiração quântica para resolver problemas de programação de produção multiobjetivo os quais, além dos objetivos serem independentes entre si, também podem ser identificados em dois grupos distintos de importância. Dessa forma, as principais contribuições deste trabalho são:

– Desenvolvimento de modelo evolucionário em programação genética com inspiração quântica integrado aos fundamentos da otimização multiobjetivo baseada em dominância da população. Nesta abordagem a aptidão de cada indivíduo é medida por sua não dominância em relação aos demais indivíduos da população.

– Desenvolvimento de modelo evolucionário em programação genética com inspiração quântica integrado aos fundamentos da otimização multiobjetivo baseada em decomposição. Nesta abordagem o problema é dividido em subproblemas e a aptidão dos indivíduos é medida através de uma função de decomposição dos objetivos originais do problema.

– Desenvolvimento de modelo evolucionário *bilevel* em programação genética com inspiração quântica. Nesta abordagem cada nível resolve, sequencialmente, um problema com algumas características diferentes e há troca eventual de informação entre os níveis.

– Análise comparativa entre as duas abordagens baseadas em otimização multiobjetivo e a otimização *bilevel*. Os modelos foram avaliados considerando diferentes cenários reais de programação.

Esta tese também contribui com:

– Discussão sobre os objetivos da programação de petróleo em refinaria. Avaliação do uso de objetivos operacionais ao invés de econômicos, além da consideração de dois grupos de importância diferentes entre os objetivos.

– Discussão sobre as características que impactam o grau de complexidade de cenários da programação. Avaliação e apresentação de cinco cenários que representam diferentes desafios. Também é apresentado o impacto que as restrições modeladas para o problema provocam nos cenários de programação.

– Análise comparativa dos melhores resultados nesta tese com a programação real proposta pelo programador de produção da refinaria utilizada como caso de estudo.

1.4 Organização do Texto

Os capítulos desta tese estão estruturados da seguinte forma:

– Capítulo 2: é apresentada uma revisão bibliográfica sobre os métodos de solução encontrados para problemas multiobjetivo, principalmente os desenvolvidos com algoritmos evolucionários e baseados em dominância de soluções ou decomposição do problema. Também é apresentada uma revisão sobre métodos de solução em dois níveis hierárquicos de decisão, chamados *bilevel*.

- Capítulo 3: são apresentados os objetivos da programação de produção, as consequências de seu não-atendimento e os principais tipos de movimentação que representam a programação de petróleo.
- Capítulo 4: é apresentada uma revisão sobre o modelo de *Quantum-Inspired Grammar-based Linear Genetic Programming* (QIGLGP) desenvolvido em [5] para tratar o problema de programação de petróleo em refinaria e utilizado como algoritmo evolucionário base para a proposta multiobjetivo apresentada.
- Capítulo 5: são apresentados os diferentes modelos multiobjetivo propostos, integrando programação genética com inspiração quântica e as principais estratégias de otimização multiobjetivo. Também são apresentados os modelos de otimização *bilevel* tanto baseado em priorização dos objetivos (como na QIGLGP original) quanto utilizando princípios de otimização multiobjetivo.
- Capítulo 6: são apresentados os resultados para todos os modelos propostos no Capítulo 5, bem como uma análise comparativa entre eles.
- Capítulo 7: são apresentadas as conclusões e observações obtidas durante o desenvolvimento desta tese e sugestões de trabalhos futuros.

2
Abordagem Evolutiva na Otimização Multiobjetivo

Neste capítulo primeiramente são apresentados conceitos de problemas multiobjetivo e as principais abordagens para tratar este tipo de problema: dominância, indicadores e decomposição. Em seguida, é apresentada a evolução dos algoritmos multiobjetivos para tratar problemas com mais de três objetivos, que representa a classe de problemas *Manyobjective*. Por último, são apresentados conceitos de otimização em dois níveis hierárquicos (chamados de *bilevel* em todo este trabalho). É dado maior enfoque às técnicas de solução utilizando metaheurísticas, principalmente baseadas em computação evolucionária.

2.1
Otimização Multiobjetivo

Resolver um problema de otimização implica em determinar soluções que minimizem (ou maximizem) os objetivos em questão, respeitando as restrições que definem o espaço de busca desta solução. Quando estes objetivos podem ser representados por apenas um elemento, por exemplo minimizar custo de produção, configura-se uma classe de problemas chamada *single-objective problem* (SOP), onde é natural determinar se uma solução é melhor do que outra. Nos SOPs, é desejado encontrar apenas uma solução para o problema. Muitas vezes não é possível ou é muito custoso encontrar a melhor solução, mas, pela natureza da abordagem do problema, o que se deseja é obter apenas uma boa solução. SOPs trabalham apenas com o espaço de busca, pois o espaço de solução é representado por uma dimensão.

No entanto, há diversos exemplos de problemas caracterizados por buscar soluções que atendam a objetivos tipicamente conflitantes. Por exemplo, em investimentos financeiros se deseja maximizar o retorno e minimizar o risco. São dois objetivos diferentes e entre eles se observa conflito, pois os chamados investimento de alto risco (como mercado de ações), nos quais se pode perder tudo, também são os que podem oferecer alta recompensa. Do outro lado, os investimentos conservadores irão oferecer recompensa mais modesta, mas o risco de perder o dinheiro investido é mínimo.

Há duas formas de lidar com o dilema dos problemas de natureza multiobjetiva [27]:

– estabelecer uma preferência entre os objetivos, seja através da definição de pesos multiplicadores para cada termo, ou através de uma priorização hierárquica explícita, de forma que o otimizador sempre consiga estabelecer que uma solução é melhor que outra. Nesta abordagem cabe ao decisor definir, *a priori*, o nível de importância dos objetivos. A função objetivo, ainda que naturalmente multiobjetivo, é tratada como um escalar;

– utilizar algoritmos que permitam que diversas soluções igualmente boas sejam apresentadas, dado que o algoritmo não é capaz de avaliar, neste grupo, que haja uma solução melhor do que a outra. Neste caso, todas as soluções otimizadas são apresentadas ao decisor, que as analisa e seleciona, *a posteriori*, qual adotar. A função objetivo é tratada como um vetor de objetivos com valores independentes.

Problemas multiobjetivo (*MultiObjective Problem* - MOP) tratados por estratégias *a posteriori* terão que lidar com soluções incomparáveis. Por exemplo, no problema de investimento, uma solução que apresenta alto risco e alta recompensa não é comparável com uma solução que apresenta baixo risco e baixa recompensa. Outra característica das estratégias *a posteriori* é a utilização do espaço de objetivos (que possui tantas dimensões quanto o número de objetivos) para conduzir às soluções.

A formulação de problemas multiobjetivos é dada por [27]:

$$\min_{(x)} \quad F(x) = (F_1(x), \cdots, F_M(x)) \tag{2-1}$$

sujeito a:

$$G(x) \geq 0$$
$$H(x) = 0,$$

onde $F(x)$ é a função multiobjetivo composta pelos M diferentes objetivos do problema, representados, cada um, através de sua $F_i(x)$; $G(x)$ representa o conjunto de restrições de desigualdade; e $H(x)$ representa o conjunto de restrições de igualdade. As variáveis do problema são representadas por x.

As técnicas tradicionais para a resolução de MOPs baseiam-se na sua conversão em problemas SOP. Dentre as técnicas desenvolvidas, encontram-se [27]:

– **somatório de objetivos ponderado por pesos**: vetor de pesos escalares é multiplicado ao vetor de objetivos, resultando num valor

também escalar que representa o objetivo único. Os pesos representam uma forma de atribuir importância aos objetivos;

- **método de restrições**: apenas um objetivo é mantido e os demais são transformados em restrições de desigualdade, para as quais são estimados valores. Cada combinação é resolvida como um problema de otimização simples. O processo é repetido até que um critério de parada seja atingido;
- **minimização por metas**: os objetivos são substituídos pelo somatório dos desvios entre o valor calculado do objetivo e o valor esperado.

Ao longo dos anos, os algoritmos evolucionários receberam maior atenção em problemas multiobjetivos, devido à sua característica de manipular uma população de indivíduos independentes que evoluem a cada geração, a qual é aderente à ideia de decisão *a posteriori*. Cada indivíduo representa uma solução para o problema, o que permite que se resolva problemas *single objective* e *multiobjective*, pois no primeiro caso a evolução ocorrerá de forma que todos os indivíduos convirjam para uma solução ótima e, no segundo caso, a população evoluirá com flexibilidade para melhor atender aos diferentes objetivos e, ao final da evolução, possuirá indivíduos que representam diferentes soluções otimizadas para a avaliação do decisor [27].

Um histórico da evolução dos algoritmos evolucionários multiobjetivo (*Evolutionary Multiobjective Algorithms* - EMOA) é apresentado em [28], o qual indica como primeiro algoritmo evolucionário para tratar MOPs o *Vector Evolutionary Genetic Algorithm* - VEGA [29], proposto em 1985. Entretanto, somente em 1993 surgem os primeiros algoritmos preocupados em garantir diversidade da população e que utilizam conceitos de dominância: *Nondominated Sorting Genetic Algorithm* (NSGA) [30] e *Multiobjective Genetic Algorithm* (MOGA) [31]. A partir de 1995 o elitismo é trazido aos algoritmos, através do desenvolvimento do NSGA-II [32], que representa, puro ou com variações, o algoritmo baseado em dominância mais encontrado na literatura multiobjetivo. Entre os modelos baseados em metaheurísticas, além do conceito de dominância das soluções, encontram-se estudos de algoritmos baseados em indicadores da população e algoritmos baseados na decomposição do problema em subproblemas. Os conceitos básicos dessas três linhas são descritos nas seções a seguir.

2.1.1 Técnicas baseadas em dominância

O primeiro grupo de algoritmos utiliza o conceito de dominância de Pareto, que é definido em [27]:

Definição 1 *É dito que a solução x_1 domina x_2 ($x_1 \preceq x_2$), se duas condições são verdadeiras:*

- *A solução x_1 não é pior do que a solução x_2 em nenhum dos M objetivos*
- *A solução x_1 é melhor do que a solução x_2 em pelo menos um dos M objetivos*

Se uma das condições é violada, x_2 é não-dominada por x_1.

Dado um conjunto de soluções **P**, o conjunto de soluções não-dominadas **P'** é dado por aquelas soluções que não são dominadas por nenhuma outra solução de **P**. Quando **P** representa todo o espaço de busca, **P'** é definido como o *conjunto Pareto Ótimo*. A representação destas soluções no espaço de objetivos é denominada *Frente* (ou *Fronteira*) *de Pareto*.

A Figura 2.1 [27] exemplifica a comparação por dominância entre soluções. Dado que é um problema de minimização e tomando como exemplo a solução C, sabe-se que todas as soluções que tiverem f_1 e f_2 maiores do que $f1$ e f_2 de C estarão no quadrante superior direito de C (com uma coloração mais escura no lado esquerdo da figura), onde se encontram as soluções E e F. Neste caso, pode-se afirmar que E e F são piores do que C pois os valores de nenhum de seus objetivos é melhor do que os valores dos correspondentes objetivos de C. Neste caso, E e F são soluções dominadas por C. Pelo mesmo raciocínio, pode-se afirmar que C é dominado pelas soluções A e B. A frente de Pareto ótima do problema é dada pelo conjunto de soluções que são não dominadas por qualquer outra solução do espaço de busca. Esta frente é apresentada na linha pontilhada do lado direito da figura.

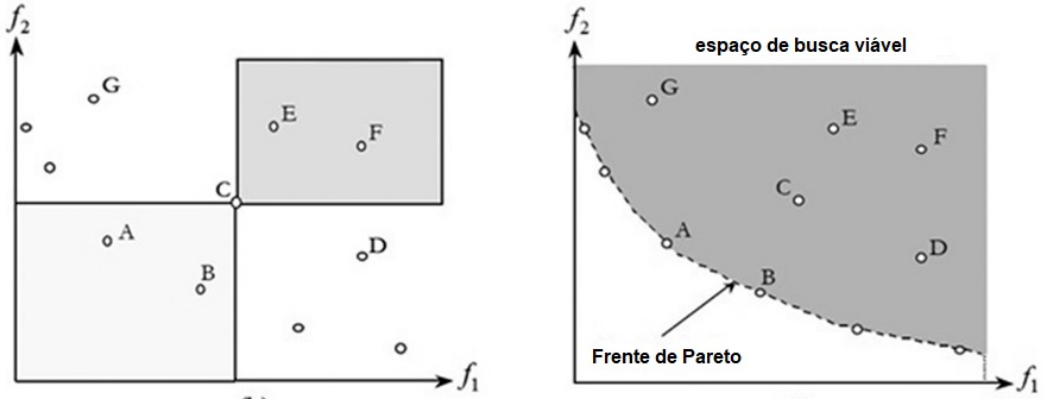

Figura 2.1: Soluções dominadas, não-dominadas e frente de Pareto.

Os maiores desafios da otimização multiobjetivo estão em encontrar o conjunto de soluções que esteja mais próximo da frente de Pareto ótima (convergência), na distribuição das soluções encontradas ao longo da frente

(diversidade) e em alcançar estes objetivos em menor tempo computacional [27].

Um esquema representativo de algoritmos genéticos clássicos começa pela inicialização da população para, em seguida, apresentar um *loop* entre as etapas de avaliação dos indivíduos, substituição da população, seleção de indivíduos e aplicação de operadores, que persistem em execução iterativa até que um critério de parada seja atingido.

Um algoritmo genético clássico para otimização multiobjetivo segue este mesmo processo. No entanto, as etapas de avaliação de indivíduos, substituição da população e seleção dos indivíduos são realizadas de forma essencialmente diferentes.

Na avaliação dos indivíduos, o conceito de dominância de uma solução é identificado pelo *rank* da frente de Pareto à qual ela pertence ou pela quantidade de outros membros da população que aquela solução domina e por quantos ela é dominada.

A cada geração do algoritmo NSGA-II [32], por exemplo, a população (composta por pais e filhos) é avaliada e ordenada de acordo com os valores de seus objetivos pelo critério de não-dominância entre seus membros, formando frentes de Pareto com um valor de *rank*. Por exemplo, a frente de Pareto *rank* 0 é composta por todos os indivíduos que são não-dominados por qualquer outro indivíduo da população. Isto significa que, entre aqueles indivíduos, não há nenhum que não atenda aos critérios estabelecidos na Definição 1. Na sequência, a frente de Pareto *rank* 1 é composta por todos os indivíduos que são não-dominados por qualquer outro indivíduo da população, exceto pelos que pertencem à frente de Pareto *rank* 0. Este processo se repete até que todos os indivíduos da população estejam classificados em alguma frente de Pareto. Uma segunda métrica, *Crowding Distance*, foi desenvolvida para atuar sobre a diversidade da população, promovendo um desempate entre indivíduos que pertencem à mesma frente de Pareto. A *Crowding Distance* $CD_{i,j}$ é dada por:

$$CD_{i,j} = \sum_{k=1}^{M} \frac{f_{(j,k,i+1)} - f_{(j,k,i-1)}}{\left| f_{(j,k)max} - f_{(j,k)min} \right|}, \tag{2-2}$$

onde M é o número de objetivos, i é o indivíduo avaliado, j é o *rank* da frente de Pareto a qual ele pertence, $f_{(j,k,i+1)}$ é o valor do objetivo k no indivíduo superior mais próximo de i neste objetivo e pertencente ao mesmo *rank*, e $f_{(j,k,i-1)}$ é o valor do objetivo k no indivíduo inferior mais próximo de i neste objetivo e pertencente ao mesmo *rank*, $f_{(j,k)max}$ e $f_{(j,k)min}$ são, respectivamente, o maior e o menor valor do objetivo k no *rank* j.

Com essas duas métricas determinadas, a substituição da população ocorre de acordo com uma ordenação crescente dos *ranks* das frentes de Pareto. Caso a população tenha tamanho fixo e, ao atingir esse limite, não acomode todos os membros de um dado *rank*, são incluídos na nova população os indivíduos com maior *Crowding Distance*, seguindo uma ordenação decrescente.

A etapa de seleção dos indivíduos sobre a qual os operadores serão aplicados também é realizada de acordo com as métricas estabelecidas. No torneio, vence o indivíduo com menor *rank* e, em caso de empate, de maior *Crowding Distance*.

Strength Pareto Evolutionary Algorithm (SPEA2) [33] também é um algoritmo por dominância, mas ao invés de organizar os indivíduos em *ranks* de frentes de Pareto, a avaliação da solução é feita com base no total de indivíduos que são dominados pelos indivíduos que dominam a solução em questão. A preocupação com a diversidade se reflete no estabelecimento de uma medida de densidade, que guarda relação inversa com a distância euclidiana de cada objetivo do indivíduo com o objetivo do indivíduo mais próximo. De forma equivalente ao NSGA-II, a substituição da população e a seleção dos indivíduos seguem, sequencialmente, essas duas métricas.

Quando os problemas tratados possuem restrições, uma abordagem usada é a aplicação de regras de viabilidade como a proposta em [32], que estabelece que ao comparar duas soluções:

- se uma for viável e a outra não, a viável sempre será melhor do que a inviável;
- se as duas forem viáveis, são aplicadas as regras de dominância de Pareto já apresentadas;
- se as duas forem inviáveis, a melhor será a que tiver menor violação das restrições.

Durante anos o mecanismo de seleção dos EMOAs esteve, majoritariamente, baseado na dominância de Pareto, que permite diferenciar bem soluções até 3 objetivos. No entanto, estudos comprovaram que, ao aumentar o número de funções objetivo, a dominância de Pareto tem dificuldade para alcançar convergência e diversidade, uma vez que, à medida que o número de objetivos aumenta, a proporção de indivíduos não-dominados também aumenta, o que deteriora a capacidade da dominância de Pareto em diferenciar soluções [34].

2.1.2 Técnicas baseadas em indicadores

O uso de indicadores em otimização multiobjetivo começou pelo questionamento de como comparar os EMOAs para avaliar se um algoritmo tem desempenho melhor que outro. Em diversas áreas, indicadores são utilizados com a proposta de transformar um conjunto de dados em uma informação única, escalar, de fácil comparação. Um desdobramento, que se tornou o segundo grupo de metodologias de EMOAs, foram as propostas de aplicar indicadores à população durante o processo evolutivo para, com isso, identificar os indivíduos que melhoram a qualidade da população e assim persistem para a próxima geração.

O indicador de hipervolume (I_H) é o mais popular utilizado nos EMOAs deste grupo de abordagem devido à sua compatibilidade com a dominância de Pareto e à comprovação de que seu melhoramento na população equivale a aproximar-se das soluções Pareto-ótimas [35]. O hipervolume (I_H) é definido como o volume da figura formada, no espaço de objetivos, entre um ponto de referência e o posicionamento de uma solução ou de todas as não-dominadas da população [34]. Dependendo do posicionamento do ponto de referência (que pode ser arbitrado) e se é um problema de maximização ou minimização dos objetivos, o que se deseja pode ser aumentar ou diminuir o hipervolume.

Outros dois indicadores utilizados são ϵ-*additive* e *Inverted Generational Distance* (IGD). O primeiro representa a distância mínima que uma solução ou um conjunto de soluções precisa percorrer para dominar (ou ser dominado por) outro [36], enquanto o segundo representa uma medida de distância entre as soluções da curva de Pareto aproximada e a verdadeira frente de Pareto do problema. No entanto, em diversos problemas não é fácil, ou mesmo possível, determinar a frente de Pareto verdadeira e, por isso, se utiliza uma frente de Pareto representativa à verdadeira e que deve atender aos critérios de diversidade e boa distribuição ao longo das soluções.

Dois algoritmos populares com indicadores são o *Indicator Based Evolutionary Algorithm* (IBEA) [36] e o *S metric selection evolutionary multiobjective optimization algorithm* (SMS-EMOA) [37].

No IBEA, a ideia principal é definir a otimização em termos de um indicador de desempenho (os mais comumente usados são hipervolume ou ϵ-*additive*) e depois utilizar esta medida no processo de seleção dos indivíduos. A etapa de avaliação dos indivíduos é feita através da seguinte função objetivo, que mede a perda de qualidade da população se o indivíduo x_i for retirado

dela:

$$F'(x_i) = \sum_{x_j \in \quad P \setminus x_i}^{j=nIndiv} -e^{\frac{-I(\{x_j\},\{x_i\})}{k}}, \tag{2-3}$$

onde i é o indivíduo para o qual a função está sendo calculada, j são todos os demais indivíduos da população, P é a população, I é o indicador adotado e k é o fator de escala da função de aptidão. A etapa de substituição da população se utiliza da propriedade *dominance preserve* (que os indicadores devem respeitar), que garante que se $x_1 \preceq x_2$, então $F'(x_1) > F'(x_2)$. Ou seja, a retirada de x_2 causa menos perda de qualidade na população. Por exemplo, após a aplicação dos operadores genéticos, ocorre o processo de determinação de $F'(x)$ para todos os indivíduos e aquele que apresentar menor $F'(x)$ é retirado. Este processo é repetido até que seja atingido o tamanho definido da população para iniciar a próxima geração. IBEA não utiliza nenhuma técnica de preservação da diversidade na população e é mais rápido do que técnicas que comparam toda a população, dado que IBEA o faz apenas entre pares de indivíduos [36].

No SMS-EMOA a função de avaliação é tratada como um vetor de objetivos e os indivíduos da população são ordenados por não-dominância, como utilizado no NSGA-II. A cada iteração, apenas um novo membro da população é criado a partir da aplicação de operadores. Para manter o tamanho fixo da população é necessário definir se o novo indivíduo substituirá algum existente. Para tal, em ordem crescente de *rank*, é verificado se o novo indivíduo é não-dominado pelos que pertencem à frente de Pareto em questão. Quando esta frente é localizada, a equação 2-4 é utilizada para calcular a contribuição de cada indivíduo para o hipervolume da frente para, então, remover da população o que tiver menor contribuição.

$$\Delta_H(x_i, FP_r) = \quad H(FP_r) - H(FP_r \setminus \{x_i\}), \tag{2-4}$$

onde r representa o *rank* de organização das frentes de Pareto, FP_r representa a frente de Pareto *rank* r, H representa o hipervolume formado pelos indivíduos em questão e Δ_H representa a diferença entre o hipervolume da frente de Pareto *rank* r, com e sem o indivíduo i em questão.

2.1.3 Técnicas baseadas em decomposição

A abordagem mais recente utilizada pelos EMOAs trata da decomposição do problema multiobjetivo em um conjunto de subproblemas de otimização com apenas um objetivo. Ainda que métodos com este princípio já tenham recebido atenção principalmente utilizando técnicas matemáticas, somente após a publicação do algoritmo *Multiobjective Evolutionary Algorithm based on Decomposition* (MOEA/D) [38], começaram a ser mais explorados na computação evolucionária [34].

Em MOEA/D os múltiplos objetivos são decompostos em subproblemas *single objective* e a população é dividida em subpopulações, onde cada indivíduo é associado a um dos subproblemas de otimização. Cada subproblema é vinculado a um número de subproblemas vizinhos, que contribuem com seu processo de otimização. A função objetivo de cada subproblema é a função de decomposição, modelada de forma que incorpore todos os objetivos do problema original. A função de decomposição de Tchebycheff, apresentada na Equação 2-5, tem sido comumente utilizada devido à sua menor parametrização e porque já foi comprovado que, nesta decomposição, para toda solução $x_{FP}*$ pertencente à frente de Pareto, existe um vetor de pesos λ para o qual $x_{FP}*$ também é o ótimo da equação 2-5 [38][34]:

$$\min \quad g^{Te}(x|\lambda, z^*) = \max_{1\leq i\leq m} \{\lambda_i \, |f_i(x) - z^*{}_i|\}, \tag{2-5}$$

onde m é o número de objetivos, λ é o vetor de pesos dos objetivos do subproblema em questão, $f_i(x)$ é o valor da função de avaliação do objetivo i e $z^* = (z_1^*,...,z^*{}_m)$ é o vetor de pontos de referência válido para todos os subproblemas. $z^*{}_i$ é dado pelo mínimo $f_i(x)$ obtido até então.

O algoritmo possui uma população externa elitista que preserva os melhores indivíduos encontrados até então. A execução do MOEA/D é feita em duas etapas: na primeira, de inicialização, é necessário definir o número de vizinhos (parâmetro) de cada subproblema, estabelecer os vetores peso de cada subproblema (há técnicas e metodologias propostas para tal), calcular a distância euclidiana, em pares de subproblema, de todos os vetores-peso e, com essa informação, identificar os vizinhos mais próximos de cada-subproblema (menor distância). Na segunda etapa ocorre a evolução onde, para cada geração e em cada subproblema, acontece a reprodução (utilizando-se dois indivíduos do vetor de vizinhos), cálculo do novo x_i e sua atualização no vetor de referência (se for o caso), cálculo da função de decomposição e atualização das soluções

de vizinhança e, por último, a retirada de indivíduos dominados da população elitista.

Uma característica positiva do MOEA/D é que como cada subproblema é resolvido com interferência de um número reduzido de vizinhos, este algoritmo tem menor complexidade computacional quando comparado com o NSGA-II, por exemplo [38]. Uma preocupação neste algoritmo é evitar que uma solução substitua indivíduos em todos os seus vizinhos, o que levaria à perda de diversidade e pode tornar difícil aos operadores de evolução gerar soluções melhores [34].

Desde a proposição do MOEA/D, diversos trabalhos já foram conduzidos na literatura com o objetivo de superar as limitações de parametrização do algoritmo, melhorar seu desempenho, adaptá-lo a diferentes tipos de problema e propor novos algoritmos baseados em decomposição. Foram estudadas propostas para a geração dos vetores peso, outras funções de decomposição, modificação dos operadores de reprodução, entre outros [39–41].

Uma versão do MOEA/D baseada em evolução diferencial foi proposta em [42] e chamada MOEA/D-DE. Esse algoritmo é baseado em operadores de evolução diferencial e técnicas para evitar a chance de perda de diversidade, como por exemplo, a utilização de um parâmetro que controla o número máximo de vizinhos que são substituídos e também a permissão, com baixa probabilidade, da migração de indivíduos para subproblemas que não são vizinhos a ele.

O impacto que a definição dos pontos de referência na função de decomposição de Tchebycheff pode causar no desempenho do algoritmo é estudado em [43]. São consideradas três abordagens: pessimista (pequeno escalar diminuído do vetor de mínimo dos objetivos), otimista (escalar diminuído é um pouco maior) e dinâmica (caminha da otimista para a pessimista). Resultados mostraram que a abordagem pessimista explora melhor a região do espaço de busca já mapeada, a otimista explora melhor novas regiões e a dinâmica encontra um bom equilíbrio.

2.1.4 Problemas "Manyobjective"

Diversos problemas são caracterizados por mais de três objetivos e esta classe de problemas, que vem sendo tratada pela alcunha de *Many objective problem* (MaOP), apresenta três grandes desafios, mesmo com a evolução dos algoritmos evolutivos. O primeiro e, talvez, mais importante, é que quanto mais objetivos, maior proporção da população será não dominada, e assim, qualquer EMOA elitista terá dificuldade em acomodar novos membros na população.

Em segundo, a implementação de um operador de diversidade (como *crowding distance* ou *cluster*) será computacionalmente custoso. Por último, a visualização de frentes de Pareto com alta dimensão é muito complicada, dificultando a análise *a posteriori* das soluções [44]. Em [45], os autores apresentaram que a preservação da diversidade e a busca pela convergência são dois objetivos conflitantes e que operadores genéticos tradicionais não são adequados para atingir os dois simultaneamente, principalmente em MaOPs. Mudanças sobre os EMOAs tradicionais podem torná-los mais eficientes para tratar problemas *many objectives*, principalmente no que tange aos dois primeiros grandes desafios mencionados.

O primeiro algoritmo a caminhar no sentido de escapar da baixa pressão seletiva decorrente das diversas soluções não dominadas foi o MOEA/D, através da divisão do problema em subproblemas. Alguns algoritmos, como em [46], foram propostos na linha de utilizar ou propôr novos princípios de dominância. Outra linha foi desenvolvida a partir do uso de pontos de referência para orientar a busca. Nesta linha foi desenvolvido o reconhecido algoritmo NSGA-III [44] e sua evolução para tratar problemas com restrições [47].

A estrutura básica do NSGA-III segue a do NSGA-II, estando as principais diferenças associadas ao operador de seleção. A manutenção da diversidade entre os membros da população é ajudada pelo fornecimento de um certo número de pontos de referência bem distribuídos no espaço de objetivos.

Os pontos de referência usados pelo modelo podem ser fornecidos pelo usuário ou determinados a partir de um procedimento estruturado. Em [44] é adotado um posicionamento sistemático dos pontos em um hiperplano normalizado [48], no qual o usuário precisa informar em quantos pontos os hiperplanos devem ser cortados (maior ou igual ao número de objetivos), o que é usado para determinar o número de pontos de referência. Quando o problema possui mais de oito objetivos, a definição do número de pontos de referência é calculada em dois estágios para diminuir a quantidade de pontos e, consequentemente, a complexidade do modelo. Para atuar na diversidade dos membros da população e ter uma boa distribuição na frente de Pareto, o algoritmo inclui a normalização dos objetivos e a associação de cada indivíduo da população a um ponto de referência. O modelo é avaliado em problemas com três a quinze objetivos e comparado com o algoritmo MOEA/D, apresentando resultado superior em diversos casos, principalmente quando a frente de Pareto ótima não é uniformemente distribuída, como nas classes de problemas DTLZ3 (possui frentes de Pareto locais) e DTLZ4 (possui uma densidade parcial de pontos em uma região). As etapas de normalização e construção

de hiperplanos do NSGA-III se mostraram eficientes dado que o algoritmo não perdeu desempenho em problemas em que os valores dos objetivos eram de ordens de grandeza diferentes.

Para lidar com restrições, o NSGA-III foi modificado através da alteração de operadores, gerando o algoritmo *Constrained NSGA-III* (C-NSGA III) [47]. Uma das mudanças é uma variação no conceito de dominância equivalente ao que foi usado no NSGA-II também para lidar com restrições. Os indivíduos da população que não violam restrições são organizados em frentes de Pareto seguindo a mesma metodologia do NSGA-III sem restrições. Já os indivíduos que violam as restrições são colocados em novas frentes de Pareto (a partir da subsequente à última das soluções viáveis), ordenados em ordem crescente no valor da violação (calculada a partir da normalização das restrições). Nos demais casos, segue o princípio da dominância modificada, ou seja, o indivíduo viável tem maior aptidão que o inviável e, caso os dois sejam inviáveis, terá maior aptidão o que violar menos as restrições. Outra mudança é o retorno do torneio para seleção de indivíduos. A seleção entre dois indivíduos viáveis é aleatória, assim como no NSGA-III. Este mesmo trabalho propõe uma modificação nos algoritmos MOEA/D e MOEA/D-DE de acordo com os mesmo princípios usados no NSGA-III. Uma comparação dos modelos mostrou que os três algoritmos são eficientes para lidar com restrições, havendo vantagem do NSGA-III pois atingiu a frente de Pareto ótima bem distribuída em mais casos e teve melhor desempenho nos casos com maiores números de objetivos.

Para diminuir o esforço computacional e tratar casos em que o algoritmo não resultou em uma frente de Pareto ótima uniformemente distribuída, foi proposto um NSGA-III adaptativo onde os pontos de referência são reavaliados a cada geração. A expectativa é que haja um indivíduo associado a um ponto de referência e, se há pontos sem indivíduos, é possível que sejam pontos inúteis, devendo ser removidos para que outros pontos sejam criados. Os novos são criados próximos aos pontos que concentrarem mais de um indivíduo associado [47].

A associação entre um ponto de referência e um indivíduo da população se mostrou o indicativo mais forte da obtenção de uma frente de Pareto ótima bem distribuída. Nesse sentido, foi desenvolvido o algoritmo *EliteNSGA-III* [49] com o intuito de, através da preservação dos melhores indivíduos de cada ponto de referência, melhorar a distribuição da frente de Pareto ótima. Este algoritmo traz duas modificações em relação ao NSGA-III original: um mecanismo para preservação dos melhores indivíduos de cada ponto de referência e mudança na forma de seleção dos pais. A preservação dos indivíduos é feita através de uma

população *archive* que armazena, para cada ponto de referência, o indivíduo com menor distância dele. A cada geração, os indivíduos da nova população são comparados, por ponto de referência, aos presentes no *archive*. Se, para um dado ponto de referência, um novo indivíduo se aproxima mais, ele substituirá o equivalente que existia no *archive*. Se o do *archive* continuar mais próximo, ele será mantido e, se na nova população não houver indivíduos para um ponto de referência qualquer, será mantido o que já existia no *archive* associado a ele. Em relação ao mecanismo de seleção dos pais, é proposto que estes sejam selecionados, com igual probabilidade entre indivíduos da população atual ou da população *archive*. Dessa forma, é menor a probabilidade de ter muitos filhos que, ao usar os mesmos pais, acabem associados aos mesmos pontos de referência (o fato do *archive* guardar apenas um membro em cada ponto de referência, contribui) e assim diminuir a diversidade. Uma comparação com o NSGA-III mostrou que este algoritmo teve melhor desempenho em casos de três a quinze objetivos, produzindo frentes de Pareto ótimas melhor distribuídas [49]. Um estudo realizado sobre o impacto dos métodos de seleção de indivíduos na diversidade da população de problemas multiobjetivo mostrou que, para os casos de algoritmos baseados em não-dominância de Pareto ou em indicadores da população, o algoritmo NSGA-III e uma versão modificada do NSGA-II (na medida de *crowding distance*) apresentaram melhores resultados em problemas *benchmark* com diferentes números de objetivos [50].

Uma abordagem híbrida que combina o processo de evolução da *Genetic Programming based Hyper-Heuristic* (GP-HH), em árvore, com o mecanismo de seleção do NSGA-III é proposto em [23] para criar regras de decisão para um problema *job shop schedule*, utilizando diversas instâncias. Em seguida, para cada instância, soluções são geradas a partir de cada regra de decisão e a aptidão deste indivíduo é medida através de uma função de avaliação *manyobjective* (testes com quatro e com cinco objetivos). A seleção por torneio é a adotada e são mantidos os operadores de cruzamento e mutação comuns à programação genética em árvore. Durante a evolução, o mecanismo de atualização da população é feito seguindo os conceitos de ordenação por dominância, associação a pontos de referência e preocupação com diversidade utilizados no NSGA-III. O paralelo feito é que se um indivíduo gerado a partir de uma regra domina outro indivíduo gerado a partir de uma regra diferente, então, a primeira regra domina a segunda regra. Resultados foram comparados com algoritmos híbridos de NSGA-II e SPEA2 também adaptados para programação genética e o desempenho obtido com o NSGA-III mostrou-se muito superior.

Algoritmos baseados nos princípios do NSGA-III ou o uso direto do

método são encontrados em diversos casos de aplicação em diferentes áreas. Um problema de *schedule* de fabricação de peças com três objetivos: tempo total de execução das tarefas - *makespan*, antecipação em relação à data limite - *tardiness* e tempo total de produção e liberação das peças; é tratado com um algoritmo proposto baseado numa adaptação do NSGA-III, que mostra melhores resultados do que os obtidos pela versão multiobjetivo do algoritmo de enxame de partículas (*Multiobjective Particle Swarm Optimization* - MOPSO) [51]. O NSGA-III também é utilizado como referência na proposição de um algoritmo multiobjetivo para tratar um problema de cadeia de suprimentos entre empresas interconectadas e competidoras entre si [52]. MOEA/D e NSGA-III apresentaram desempenho semelhante na definição da espessura e posição do furo de um absorvedor de impacto (para dissipar energia cinética, com diversas aplicações) feito de aço cônico com tampa perfurada. Nesta aplicação, os objetivos são otimizados para atingir a máxima absorção de energia e mínima força de pico [53]. Em [54] o NSGA-III é aplicado em um problema de abastecimento de energia considerando aspectos econômicos, balanço energético e confiabilidade do abastecimento para o usuário final.

A combinação das técnicas de não-dominância de Pareto (NSGA-III) e decomposição (MOEA/D) é proposta em [55] para problemas com mais de três objetivos e sem restrições. O objetivo é utilizar os méritos das duas abordagens para balancear convergência e diversidade durante o processo evolucionário. A integração ocorre na etapa de atualização da população do MOEA/D, que é avaliada considerando todos os seus indivíduos, com os princípios de não-dominância de Pareto. Em [56] MOEA/D com função de decomposição *Penalty-based boundary intersection approach* (PBI) invertida obteve melhores resultados do que com a PBI convencional, Tchebycheff ou o NSGA-III em problemas *manyobjective* quando aplicada em problemas *benchmark*. Em [57] é proposta a integração de algoritmos baseados na preferência do usuário e em decomposição para resolver problemas *benchmark* com mais de três objetivos. A decomposição agrega a vantagem de ser menos suscetível a problemas de perda de pressão seletiva quando comparado com a não-dominância e a inclusão de técnicas para definição de preferência contribui com a distribuição do vetor de pesos, favorecendo a diversidade da população.

2.2 Otimização em 2 níveis de decisão (Bilevel)

O estudo de técnicas e métodos de otimização em mais de um nível hierárquico é motivado pela observação de que diversos problemas reais, em diferentes áreas, seguem uma estrutura hierárquica de decisão com diferentes

atores. As decisões são tomadas sem cooperação entre os agentes, que controlam subconjuntos das variáveis, e cada decisão tomada em um nível impacta diretamente os espaços de busca e de solução dos demais [58]. Um exemplo ilustrativo de otimização hierárquica é o problema do "fabricante X varejista", onde ambos os atores desejam maximizar seus lucros. O fabricante pode ser representado como um líder, pois sua decisão sobre o preço do produto é a primeira a ser feita. O varejista pode ser encarado como um seguidor, pois irá reagir ao preço estabelecido pelo líder através da compra do produto, em maior ou menor quantidade, e do preço que atribuirá para revenda. Para maximizar seu lucro, o fabricante deseja cobrar o preço mais alto possível desde que não implique em diminuição da quantidade de produto comprada pelo varejista. Dessa forma, o fabricante tentará antecipar o comportamento do varejista para determinar o valor de venda de seu produto [59].

A obtenção de soluções ótimas em problemas multi-nível é computacionalmente custosa, tornando-os praticamente intratáveis [58][60]. A otimização em dois níveis é a classe deste grupo de problemas onde se concentram as pesquisas devido à sua menor complexidade e boa representatividade nos desafios do mundo real. O termo *bilevel programming* foi usado pela primeira vez em [61] e é adotado até hoje para nominar os problemas de otimização em 2 níveis hierárquicos [62]. Neste exemplo do varejista x fabricante, ambos os atores e, consequentemente, ambos os níveis de decisão têm na função objetivo apenas um termo, que é a maximização de seu lucro. No entanto, há diversos problemas com hierarquia de decisão multi-nível que podem ser representados de forma que nos dois níveis haja funções multiobjetivo.

Um *Bilevel Optimization Problem* (BOP) é encarado como um jogo líder-seguidor onde a cooperação não acontece. Cada jogador otimiza seus próprios objetivos, mas suas decisões afetam o espaço de decisão do outro jogador.

Matematicamente, um BOP pode ser formulado como um problema de otimização (seguidor) que faz parte do conjunto de restrições de outro problema de otimização (líder), como [63]:

$$\min_{(x_u,\ x_l)} \quad F(x) = (F_1(x), \cdots, F_M(x)) \tag{2-6}$$

sujeito a:

$$\begin{aligned} x_l \quad \in \quad & \underset{(x_l)}{argmin} \quad f(x) = (f_1(x), \cdots, f_N(x)) \\ & sujeito \quad a: \\ & \qquad g(x) \geq 0 \\ & \qquad h(x) = 0 \end{aligned}$$

$$G(x) \geq 0$$
$$H(x) = 0,$$

onde $F(x)$ é a função multiobjetivo composta pelos M diferentes objetivos do problema líder, representados, cada um, através de sua $F_i(x)$; e $G(x)$ e $H(x)$ representam, respectivamente, as restrições de igualdade e desigualdade também do líder. De forma equivalente, para o problema seguidor, $f(x)$ é a função multiobjetivo composta pelos N diferentes objetivos, representados, cada um, através de sua $f_i(x)$, sendo $g(x)$ e $h(x)$, respectivamente, as restrições de igualdade e desigualdade. Já x_u e x_l representam as variáveis do líder e do seguidor, respectivamente.

Encontrar a solução de um BOP é encontrar o conjunto x, pertencente ao conjunto de soluções viáveis, que otimize $f(x_l)$ para um x_u fixo e, respeitando esta e as demais restrições do problema líder, otimize $F(x)$. Para cada valor do vetor x_u do líder, as restrições do seguidor definem o espaço viável de busca do seguidor [58] [64].

Ainda que todos os objetivos e restrições do líder e do seguidor sejam lineares, o BOP não será nem contínuo em qualquer ponto e nem convexo para a função objetivo (FO) do problema líder. Mesmo o mais simples dos BOPs é não convexo e *NP-Hard* [58][64][62][65]. A não convexidade do problema sugere a possibilidade de diversos ótimos locais. Estas características tornam muito difícil encontrar uma solução ótima global [66][67][62].

Quando um ou ambos os níveis do BOP são multiobjetivo, não há uma resposta que represente a solução para o problema. Conjuntos de soluções (x_u,x_l) devem ser tratados e, neste caso, a complexidade aumenta significativamente para a solução de *Multi-Objective Bilevel Otimization Problems* (MOBOP).

Diversas categorias de aplicações podem ser modeladas como problemas BOP, por exemplo:

- Transporte: *upper-level* decide sobre controle, gerenciamento, investimentos na malha de rodovias, cobrança de pedágios; enquanto *follower* (usuários da rede) decide sobre a rota que irá realizar em função das decisões tomadas pelo líder.
- Gerenciamento de rentabilidade: é uma das aplicações mais populares [58] e consiste em identificar a percepção de valor que consumidores têm sobre um produto. Através da previsão que se faz sobre a resposta do consumidor, o produto é oferecido no momento certo pelo maior preço possível que não desestimule a quantidade de venda.
- Projetos de engenharia: diversas aplicações de engenharia química e mecânica podem ser tratadas como problemas de otimização líder onde

as soluções viáveis devem respeitar condições físicas, como estabilidade ou equilíbrio, representadas por um problema de otimização seguidor.

– Cadeia de suprimento: a modelagem pode passar pela resposta à localização de centros de distribuição em função dos clientes que poderão ser atendidos por quais centros e as rotas de cada um destes centros aos postos de atendimentos [68].

Como qualquer problema de otimização, BOPs podem ser divididos nas categorias: contínuo (apenas variáveis contínuas), combinatórios (com variáveis discretas) ou mistos (com variáveis contínuas e discretas). Nos últimos 30 anos, a maioria dos estudos e desenvolvimentos foi feito sobre BOPs lineares contínuos (tratando-os com programação matemática linear), especialmente no problema seguidor. Se o seguidor e/ou líder de um BOP tem características combinatórias, o problema se torna consideravelmente mais difícil e as referências literárias se tornam escassas.

Uma comparação entre BOPs e problemas de teoria dos jogos pode levar à interpretação do BOP como uma versão estática de dois jogadores para o não-cooperativo jogo de Stackelberg, muito estudado especialmente no contexto econômico [58]. Realmente há similaridade nos princípios das duas áreas de conhecimento. No entanto, o jogo de Stackelberg observa o problema buscando um equilíbrio, enquanto BOPs tratam de um problema de otimização como restrição de outro problema de otimização. O jogo de Stackelberg gera apenas um resultado em cada movimento, enquanto BOPs podem ter diversas soluções no seguidor. Nessas condições, uma observação estática do jogo de Stackelberg pode não representar uma solução BOP [60].

Também não produzirão os mesmos resultados um problema com dois objetivos e a conversão deste problema num BOP, representando-o como dois níveis com um objetivo. A solução de um problema BOP não é necessariamente parte do conjunto de Pareto de um problema de otimização multiobjetivo em apenas um nível (MOP) e vice-versa [64]. A otimização *bilevel* busca identificar soluções (x_u,x_l) que resultem em bons valores de função objetivo para o líder, enquanto também estão próximas do ótimo no seguidor com as variáveis fixadas. Esta característica leva ao fato de que existem boas soluções que não estão na fronteira de Pareto e soluções que estão na fronteira de Pareto, mas que não são boas soluções para o problema completo [64]. Estudos já foram realizados na tentativa de encontrar uma conexão entre problemas MOP e BOP, que não foi estabelecida até hoje. Em contra-partida, alguns autores já demonstraram que otimalidade em MOP e em BOP são ideias diferentes [58].

BOPs nos quais os dois níveis têm apenas um objetivo e são lineares foram alvo de atenção de pesquisas considerando uma abordagem de solução

matemática, onde podem ser encontrados quatro grupos de métodos de solução [66]:

- Reformulação para problema de apenas um nível usando condições KKT (*Karush-Kuhn-Tucker*) de otimalidade do seguidor.
- Abordagens *vertex enumerations*, que desenvolve uma versão modificada do método simplex, explorando o fato de que a solução ótima global estará em um ponto extremo do espaço de soluções viáveis.
- Algoritmos que extraem informação de gradiente do problema seguidor e usam isso como uma diretiva computacional para a FO do problema líder
- Algoritmos de penalização, que introduzem um termo de penalidade associado à violação de certas condições de otimalidade, dessa forma, levando a busca em direção à solução ótima.

Um MOBOP é convertido em um MOP não suave (*nonsmooth*) a partir das condições KKT de otimalidade do seguidor em [69]. Então, uma sequência de MOPs suaves (convexos, contínuos e diferenciáveis) que aproxima progressivamente o MOP não suave é introduzido. É mostrado que as soluções ótimas de Pareto do problema aproximado convergem para as soluções ótimas de Pareto do MOBOP original. Os mesmos autores, em [70], propõem a introdução de uma nova função de penalização para resolver um problema MOBOP. O método se baseia na inclusão do *duality gap* do seguidor entre os objetivos do líder, com uma penalização. Os resultados mostraram que essa abordagem pode ser interessante. Uma revisão das abordagens que foram desenvolvidas utilizando representações matemáticas determinísticas para resolver BOPs lineares ou misto-inteiros é apresentada em [66].

Se, por um lado, encontrar soluções ótimas de um BOP é um verdadeiro desafio computacional, por outro, na maior parte das aplicações reais é suficiente encontrar uma boa solução. Dessa forma, abordagens que geram boas soluções mas não garantem o ótimo são interessantes focos de pesquisa [58]. Técnicas metaheurísticas são candidatas naturais deste desenvolvimento pois têm como uma de suas principais características a capacidade de lidar com problemas de grande dimensão, gerando soluções em um tempo razoável, mas onde não há garantia de ser encontrada a solução ótima [62][65].

As diversas técnicas metaheurísticas possuem subclassificações, por exemplo: podem ser baseadas em apenas um indivíduo que evolui ao longo das iterações (*local search*, *simulated annealing*, *tabu search* etc.) ou em populações (*diferencial evolution*, *ant colony*, *particle swarm*, *genetic algorithms*,

genetic programming etc.); a evolução pode acontecer "com" ou "sem" memória do caminho percorrido até aquela geração e também pode acontecer através de uma distribuição de probabilidade explícita (*estimation of distribution algorithms*) ou por distribuição de probabilidade implícita, que acontece através da aplicação de operadores (principal abordagem de *genetic algorithms*).

Algoritmos baseados em metaheurísticas têm sido pesquisados para resolver BOPs, podendo ser divididos em quatro abordagens:

- Transformação do problema BOP em apenas um nível: através de metodologias exatas ou de aproximação que substituam o problema seguidor (métodos baseados em penalizações, enumerações, regiões de confiança). Quando o problema seguidor tem restrições e objetivo diferenciáveis, este pode ser substituído por suas condições KKT como restrições para o líder e ter suas variáveis adicionadas ao líder, representando-as por multiplicadores de Lagrange. Esta transformação só pode ser aplicada em problemas convexos e diferenciáveis, mas a partir daí, podem ser usados quaisquer métodos metaheurísticos clássicos para a solução [58]. No entanto, apesar da representação matemática possível, a presença de muitos multiplicadores de Lagrange e um termo abstrato envolvendo coderivadas para representar o seguidor fazem com que esta abordagem seja difícil de ser aplicada na prática [60].
- Transformação do problema BOP em MOP: dado que BOP são, necessariamente, problemas com pelo menos dois objetivos, uma abordagem natural poderia ser transformá-lo em um problema multiobjetivo de apenas um nível (MOP). No entanto, as condições de otimalidade de problemas MOP e BOP são diferentes e, portanto, uma solução na frente de Pareto pode não ser uma solução do problema com estrutura BOP. Não é surpresa que uma solução do BOP possa ser dominada em termos das funções líder e seguidor. No entanto, em [71] é proposta uma forma de fazer a transformação BOP/MOP usando o conceito de *cone dominance*, para condições muito específicas de aplicação. Neste trabalho um BOP com apenas um objetivo em cada nível é transformado em um MOP de quatro objetivos. Derivadas do objetivo original são envolvidas na formulação do problema, o que faz com que a abordagem seja limitada a problemas diferenciáveis. Teoricamente, a ideia poderia ser estendida a MOBOPs, mas nenhuma sugestão com formulação matemática foi feita até então [58].
- Resolução sequencial: se propõe a resolver os dois níveis (líder e seguidor) de forma sequencial. O problema líder gera uma solução ou população

de soluções (x_u,x_l), orientado por uma função objetivo $F(x_u,x_l)$. Estas soluções são passadas ao problema seguidor, que atua apenas sobre as variáveis de decisão x_l (as variáveis x_u, definidas para o problema líder, são tratadas como constantes) para, através da atuação de uma metaheurística, promover melhorias na população. Após a conclusão da otimização do seguidor, a nova população (x_u,x_l^*) retorna indivíduos ao problema líder, representando a população inicial de um novo ciclo iterativo. Este processo se repete até que um critério de parada para todo o problema seja obtido [58].

– Co-evolução: não exige condições muito restritivas para aplicação. Duas populações diferentes evoluem em paralelo, cada uma com parte das variáveis de decisão. A solução completa é obtida a partir de uma troca cooperativa entre as populações, ou seja, os dois níveis trocam informação para manter uma visão completa sobre o BOP. As principais preocupações nesta abordagem estão em definir qual informação será trocada, como esta troca acontecerá e qual o critério para definir quando as trocas ocorrerão [64].

2.2.1 Metaheurísticas em BOPs com um objetivo em cada nível

Dentro do campo de pesquisas com metaheurísticas, há trabalhos que apresentaram bons desempenhos usando diferentes técnicas, muitas vezes com abordagens híbridas, para resolver problemas de otimização bilevel em que há apenas um objetivo em cada nível. Para BOPs complexos, as técnicas clássicas são de difícil aplicação devido ao fato de não serem convexos, contínuos ou diferenciáveis em todo o espaço de solução. Geralmente, metaheurísticas são técnicas computacionalmente custosas. Dessa forma, alguns autores desenvolvem modelos híbridos entre as duas classes de técnicas com o objetivo de ter melhor aproveitamento de ambas [72].

O problema do fabricante x varejista é estudado em [59] através de um modelo híbrido, desenvolvendo-o em duas formas de representação: primeiro o líder sendo o fabricante e depois é o varejista que assume este papel. Ambos os atores buscam encontrar o melhor preço de seu produto como objetivo. Nos dois casos, utiliza metaheurística baseada em *Imperial Colony Algorithm* (ICA) para o problema líder e resolve o problema seguidor com um método matemático. Compara o resultado com uma estratégia evolucionária no líder e mostra que ICA apresentou menor esforço computacional, mas ainda é muito custoso. Os resultados também mostraram que o ator que atuava como líder alcançava maior benefício.

Um algoritmo de otimização bilevel baseado em *Particle Swarm Optimization* (PSO) é proposto por [67] para resolver dois clássicos problemas *NP-Hard* de cadeia de suprimento: roteamento de veículos e localização de centros. No problema de roteamento de veículos, o líder define os clientes que serão atendidos por cada frota, garantindo a viabilidade desta alocação, com restrições de capacidade, de viagem máxima para cada caminhão, mas sem considerar a sequência em que cada veículo irá visitar os clientes. Já o seguidor é responsável por otimizar as rotas de atendimento e alocar os veículos. Dado que o custo de cada rota só pode ser definido pelo seguidor, o líder apenas faz uma estimativa deste valor para considerar entre seus objetivos.

No problema de Localização de Centros de Distribuição, o líder define onde serão localizados os centros de distribuição de mercadorias para os clientes e o seguidor define todo o roteamento de veículos para este atendimento. O modelo proposto é um híbrido que envolve PSO no líder e para o problema seguidor é usado o algoritmo *nearest neighborhood* combinado com *Expanding Neighborhood Search Method* para melhorar as soluções de cada partícula [67].

Para os dois problemas, os resultados são comparados com o modelo [73], também *bilevel* e que utiliza GA. A abordagem com PSO apresenta resultados melhores em quase todas as instâncias avaliadas. A métrica de comparação é baseada na distância à melhor solução já obtida em cada instância [67]. No entanto, para tratar os dois problemas de cadeia de suprimentos, este modelo mostrou desempenho médio inferior a trabalho anterior do mesmo autor [74], um PSO baseado em topologia de expansão combinatória local e global, que não envolvia abordagem *bilevel*.

O primeiro algoritmo proposto, encontrado por esta revisão, para tratar diferentes classes de BOP em um único *framework* utilizando *genetic algorithm* (GA) foi denominado *Bilevel Genetic Algorithm* (BiGA) [75]. São manipuladas duas populações, uma para cada nível hierárquico, que evoluem paralelamente num processo típico do GA clássico. A cada dado número de gerações, acontece a coevolução com a troca de indivíduos que estão relacionados nas duas populações. Uma população elitista é mantida externamente para garantir a preservação dos melhores membros de cada população.

Em [65] é proposto um algoritmo hierárquico baseado em estratégia evolucionária, cuja solução vai sendo adaptada ao longo dos níveis. Cada nível tem apenas um objetivo e o algoritmo começa resolvendo o problema líder, levando em consideração todas as restrições de todas as camadas hierárquicas e manipulando todas as variáveis, como se todas fossem do nível de decisão do líder. Segundo os autores, esta solução será pior ou igual à obtida caso o segundo nível fosse considerado, por exemplo. O próximo passo é fixar as

variáveis do primeiro nível e resolver o segundo nível de forma equivalente ao que foi usado no líder, ou seja, usar a função objetivo do segundo nível e todas as restrições e variáveis do problema a partir deste nível (terceiro, quarto, até onde houver). Este procedimento se repete até que o último nível hierárquico do problema seja resolvido.

O algoritmo *Coevolutionary Bilevel Method using Repeated Algortithm* (CoBRA) [68] foi desenvolvido a partir dos conceitos apresentados pelo BiGA [75]. É baseado em metaheurística nos dois níveis de decisão como uma alternativa para resolver problemas de grande dimensão com um ou mais objetivos em cada nível. Para testar o método, utiliza um problema de cadeia de suprimento e propõe uma nova métrica, chamada *Rationality*, para avaliá-lo comparativamente a abordagens clássicas. *Rationality* mede a distância ao ótimo das variáveis do problema seguidor (dado que as variáveis do líder estão fixas) pela determinação de quantas vezes o algoritmo foi capaz de melhorar a solução em um período definido (*direct rationality*) e também o quanto foi melhorada (*weighted rationality*).

Os componentes do método CoBRA são: um algoritmo de otimização para cada nível (se o subproblema tem apenas um objetivo, é usado um algoritmo evolucionário clássico e, se o subproblema é multiobjetivo, é usado NSGA-II), uma estratégia coevolucionária, uma estratégia de *archiving* (para persistência das melhores soluções de cada nível na sua respectiva população), um critério de parada e operadores evolucionários adequados para cada tipo de problema.

Em uma estrutura inversa à apresentada em [67], nesta abordagem, os problemas de localização de centros de distribuição e de roteamento de veículos (MDVRP - *MultiDepot Vehicle Routing Problem*) são tratados de forma que o problema líder é transportar as mercadorias dos depósitos para os varejistas, respondendo à demanda do varejista, e o problema seguidor é produzir as mercadorias nas fábricas e transportá-las aos depósitos [68].

Na versão multiobjetivo *bilevel* modelada, o líder se mantém com apenas um objetivo, mas o seguidor deve encontrar soluções que sejam Pareto para seus dois objetivos que retratam dois tipos diferentes de custos operacionais [68]. Foram feitas análises comparativas para o problema MDVRP com abordagem *bilevel* de apenas um objetivo (BiMDVRP) com os modelos CoBRA e abordagem reparativa (não há *archiving* e operador de coevolução), assim como para o problema *bilevel* multiobjetivo (M-BiMDVRP) com os mesmos dois tipos de modelagem. Na abordagem BiMDVRP a modelagem reparadora apresentou a melhor média de função de avaliação, bem como o melhor caso, na maioria das instâncias. No entanto, a métrica de racionalidade foi melhor

para o CoBRA em todas as instâncias. Na abordagem M-BiMDVRP, o modelo CoBRA teve melhor desempenho em todos os critérios e para todas as instâncias. O tempo computacional para obtenção das soluções usando método CoBRA foi inferior a um terço daquele da abordagem reparadora em qualquer instância [68].

Em [62] é proposto um algoritmo baseado em evolução diferencial (*Differential Evolution* - DE) para tratar problemas *bilevel* e este é aplicado a um conjunto de problemas de diferentes áreas para avaliar seu desempenho, demonstrando ser uma boa heurística de busca. O modelo aposta na sinergia da boa capacidade de busca da DE para otimizar o problema líder combinada com ferramentas determinísticas NLP (*Non Linear Programming*) focadas em resolver o problema seguidor. Todos os problemas possuem funções com apenas um objetivo em cada nível.

Em [72] é apresentado um algorítimo híbrido, *Bilevel Evolutionary Algorithm based on Quadratic Approximations* (BLEAQ), que utiliza aproximação quadrática no seguidor e um algoritmo evolucionário no líder. O algoritmo BLEAQ foi testado em dois conjuntos de testes, mostrando melhora no número de funções de avaliação nos dois níveis quando comparado ao algoritmo em [76], que propôs uma transformação do problema *bilevel* em problema multiobjetivo, usados como referência do artigo.

2.2.2 Metaheurísticas em BOPs multiobjetivos

BOPs com apenas um objetivo receberam mais atenção da comunidade científica, pois é a área onde se encontram mais trabalhos tanto nas linhas de modelagem matemática como metaheurística (incluindo algoritmos evolucionários) [58]. Dessa forma, pode-se considerar que estudos sobre *Multiobjective Bilevel Optimization Problems* (MOBOPs) ainda são escassos. Esta observação não reflete falta de interesse ou de casos práticos de aplicação, mas sim é atribuída à grande complexidade adicionada pelos múltiplos objetivos combinada com a complexa interação entre os dois níveis [60].

Duas metodologias são propostas para resolver MOBOPs em [60]: uma é um algoritmo híbrido e auto-adaptativo que usa algoritmo evolucionário e busca local para gerar a frente de Pareto do MO e a outra metodologia se baseia na primeira e incorpora preferências do decisor do problema líder em passos intermediários do problema seguidor de forma a diminuir o custo computacional.

Dado que o problema de otimização seguidor pertence ao grupo de restrições do líder e, dessa forma, restringe seu espaço de solução, pode-se

afirmar que uma solução só pode ser viável no líder se é Pareto-ótima no seguidor [60].

Tomando como exemplo hipotético um MOBOP com dois objetivos em cada nível, a Figura 2.2 apresenta, nos quatro cantos, a região viável do seguidor para diferentes valores das variáveis x_u (recebida do Líder e fixada no Servidor) e, no centro da figura, é apresentada uma possível relação entre membros de cada região viável do Seguidor e seu posicionamento na região do Líder [60].

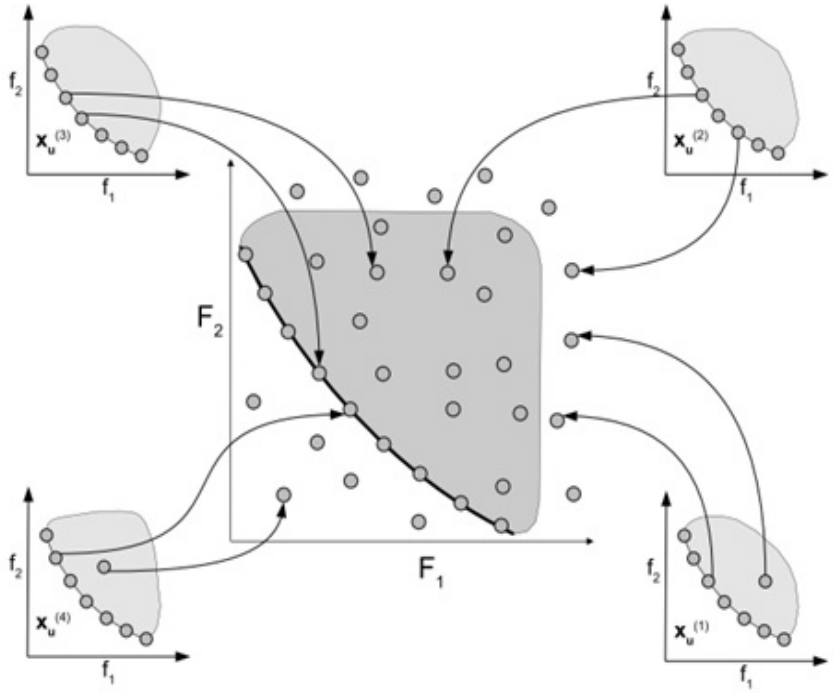

Figura 2.2: Representação gráfica dos espaços de solução de um MOBOP. O gráfico no centro da imagem representa a região viável do problema líder, com indivíduos dentro e fora dela. Os gráficos laterais representam quatro regiões viáveis do problema seguidor, com indivíduos dentro e fora dela. As regiões do seguidor são diferentes porque foram criadas a partir de quatro valores diferentes de x_u.

A observação dessa figura permite a discussão das seguintes situações [60]:

- a região viável do Seguidor muda dependendo dos valores de x_u recebido do Líder e, portanto, pode resultar num conjunto de soluções diferentes. É o que se observa para os quatro diferentes valores de x_u ($x_u^{(1)}$, $x_u^{(2)}$, $x_u^{(3)}$ e $x_u^{(4)}$) na figura;
- para as soluções membro do Seguidor que não são ótimas, a correlação com a região viável do Líder sempre é a de exclusão. Ou seja, os indivíduos que não pertencerem à frente de Pareto ótima do Seguidor

não irão pertencer à região viável do Líder e, por isso, não devem ser migradas pois irão apenas dificultar a solução do MOBOP. Este exemplo é apontado no gráfico da variável $x_u^{(4)}$ que apesar de dominar as soluções do Líder, não é viável;

- nos conjuntos $x_u^{(1)}$, $x_u^{(2)}$ e $x_u^{(4)}$ observa-se que ser Pareto-ótimo do líder é condição necessária, mas não suficiente para pertencer à região viável do líder, pois os 3 casos apresentam membros que atendem à primeira condição, mas não são viáveis. Isto pode acontecer devido a restrições estabelecidas no líder;
- nos conjuntos $x_u^{(2)}$ e $x_u^{(3)}$ observa-se soluções de Pareto do seguidor que pertencem à região viável do líder, mas não são Pareto deste problema. Ou seja, ser Pareto do seguidor e viável no líder não implica em ser Pareto no líder;
- nos conjuntos $x_u^{(3)}$ e $x_u^{(4)}$ observa-se soluções que são Pareto no seguidor e também no líder.

Em [77] é apresentado o problema de definição de pedágios de uma malha rodoviária sob a perspectiva de MOBOPs, onde os objetivos do líder são minimizar o tempo de viagem no sistema e também as emissões veiculares, enquanto o seguidor é um problema de equilíbrio de fluxo de tráfego que resulta da política de pedágios definida pelo líder. Em essência, o seguidor são os usuários finais, que possuem como objetivo minimizar seu tempo total de viagem e o custo do pedágio. Nessa abordagem MO, um conjunto de valores de pedágio e emissões para a malha rodoviária é obtido.

Em [63] é proposto um algoritmo baseado em NSGA-II para resolver os dois níveis de decisão de forma síncrona. O algoritmo é genérico a ponto de poder ser usado para problemas com um ou mais objetivos em cada nível, convexos/não-convexos, lineares/não-lineares, diferenciáveis/não-diferenciáveis. A abordagem não é sequencial, mas sim, emprega uma estrutura de evolução entrelaçada das populações líder e seguidor. O algoritmo é computacionalmente muito custoso, mas permitiu entender as complicadas influências que um problema tem sobre o outro. Por isso, em trabalhos seguintes, os autores investiram em um algoritmo menos estruturado, auto-adaptativo e computacionalmente mais rápido, que é formado por um híbrido de algoritmo evolucionário (NSGA-II) e busca local para tratar os MOBOP, chamado (*Hybrid Bilevel Evolutionary Multi-Objective algorithm* (HBLEMO).

O HBLEMO foi testado com um conjunto de problemas, chamado *DS test suite* com diferentes graus de dificuldade e escalonáveis (avaliou-se com 10, 20, 30 e 40 variáveis). Em todos os casos, HBLEMO se mostrou capaz de encontrar

as frentes de Pareto dos problemas. Também foi realizada uma comparação do HBLEMO com um algoritmo aninhado que usa NSGA-II combinado com busca local. Este algoritmo trabalha com uma população fixa e, para cada x_u, o seguidor é terminado pela execução de uma busca local em todas as soluções não-dominadas identificadas pela última iteração do NSGA-II. Para comparação do HBLEMO com este algoritmo foram usados dois problemas do *DS test suite* com os mesmos quatro tamanhos de variáveis e os resultados indicaram que o HBLEMO necessita de, no mínimo, dez vezes menos avaliações para encontrar um conjunto de soluções com a mesma medida de hipervolume.

Em trabalho posterior [78], com o objetivo de diminuir a quantidade de avaliações necessárias, foi proposto alterar a abordagem *a posteriori* do HBLEMO para uma abordagem "progressivamente interativa"onde o decisor do problema líder interage com o HBLEMO para orientar a busca para uma "solução preferida" e assim, um único ponto da frente de Pareto é buscado. Esta abordagem é indicada pelos autores em problemas *manyobjective*, quando há dificuldade de convergência e de manutenção da diversidade. O algoritmo desenvolvido *Progressively interactive-HBLEMO* (PI-HBLEMO) [78] tem como principais mudanças o critério de dominância e critério de parada em relação ao HBLEMO.

A mudança no critério de parada se deve à diferença essencial entre os algoritmos. O do HBLEMO se baseia no hipervolume, obtido a partir das diversas soluções da frente de Pareto. Já no PI-HBLEMO, a busca é por uma única solução de preferência e, por isso, o cálculo com base no hipervolume não pode ser feito. No PI-HBLEMO, o critério passa a ser calculado com base na distância entre uma solução melhorada a partir das melhores soluções de gerações anteriores. Para avaliar este modelo, foram usados os cinco casos do *DS test suite*, convertendo-os em problemas de maximização para que o PI-HBLEMO pudesse ser utilizado. Os resultados mostraram uma redução efetiva no número de avaliações. Nos diferentes casos, o decisor do problema líder foi chamado entre 7 e 29 vezes para avaliar as soluções durante o processo evolutivo.

O próximo capítulo apresenta os múltiplos objetivos da programação de petróleo, qual a sua importância e as consequências quando eles não são adequadamente atendidos. Também apresenta as principais decisões que o programador deve tomar para atingir esses objetivos e o benefício em ter mais de uma solução de qualidade para avaliação *a posteriori*.

3
Programação de Petróleo em Refinaria

Este capítulo apresenta as principais atividades da programação de petróleo em uma refinaria, bem como uma discussão sobre os objetivos desta área de atuação.

Uma refinaria é constituída por diversas unidades de processo, onde cada uma desempenha um papel de separação, conversão ou tratamento de fluxos materiais (correntes de carga) de forma a gerar outros fluxos materiais (correntes de produtos intermediários ou produtos finais) com propriedades diferentes. Ao final de todas etapas de processamento, o petróleo adquirido pela refinaria é convertido numa série de derivados com maior valor agregado, que são vendidos.

As correntes que saem de uma unidade de processo são destinadas a outra(s) e esse percurso pode ser realizado diretamente, ou com residência temporária em tanques nos quais ocorre mistura entre os produtos que já estão no tanque e os que chegam. Essa mistura altera as propriedades dos produtos em tanque que serão, em algum momento, destinados a unidades de processo.

A programação de produção trata de resolver o complexo problema que se forma para movimentar diversas correntes simultaneamente, conectando equipamentos de origem e de destino, respeitando as restrições de qualidade e operacionais associadas a cada movimentação e observando os diversos objetivos operacionais que estão relacionados à manutenção da carga das unidades de processo e à operação mais estável possível da refinaria como um todo. O resultado da atividade de programar a produção é um cronograma de atividades com duração e recursos alocados, chamado de programação de produção ou *schedule* por sua tradução em inglês.

3.1
Atividades da programação de petróleo

A programação de petróleo engloba as atividades desde o recebimento das misturas de petróleo até a produção das correntes de produtos intermediários na saída da(s) unidade(s) de destilação. Em termos de unidades de processo, apenas as primeiras da refinaria estão neste escopo. São a(s) unidade(s) de destilação atmosférica e a(s) unidade(s) de destilação a vácuo. A unidade de

destilação atmosférica (UDA) promove a separação física dos hidrocarbonetos do petróleo em diferentes correntes, baseando-se nos diferentes pontos de ebulição na condição de pressão atmosférica dentro da unidade. Em geral, as correntes de gás liquefeito de petróleo (GLP) - a mais leve -, nafta leve (NL), nafta pesada (NP), querosene (Q), diesel leve (DL), diesel pesado (DP) e resíduo atmosférico (RAT) - a mais pesada -, são produzidas nesta unidade. A corrente de RAT é encaminhada diretamente para a unidade de destilação a vácuo (UDV) que possui o mesmo princípio de funcionamento da UDA, mas opera em condição de vácuo e, por isso, é possível fracionar o RAT em outras correntes como gasóleo leve de vácuo (GOL), gasóleo pesado de vácuo (GOP) e resíduo de vácuo (RV), reduzindo a carga térmica necessária para essa separação quando comparada à condição atmosférica. Em geral, UDA e UDV operam como um par e, frequentemente, são modeladas como uma única unidade. Dependendo da refinaria específica, pode haver alguma variação na quantidade de correntes produzidas, mas as apresentadas neste parágrafo são uma representação genérica bem frequente.

O final da envoltória da programação de petróleo é, muito frequentemente, a produção das unidades UDA + UDV. No entanto, o início não é bem definido, pois depende de onde começa a autonomia de decisão da refinaria. O principal fator de definição é se há compartilhamento de recursos (equipamentos) para o recebimento de petróleo. Este recebimento acontece em tanques geralmente fora da área geográfica da refinaria, em um terminal, que pode estar a dezenas ou até centenas de quilômetros de distância. Dessa forma, se duas ou mais refinarias compartilham o mesmo terminal, é provável que exista uma área da empresa responsável por programar as movimentações do terminal de forma a atender a todas as refinarias, garantindo que todas recebam seus petróleos adequadamente, ou seja, em quantidade, qualidade e disponibilidade necessárias para manter as unidades de processo operando. Neste caso, o escopo de decisão da refinaria e, portanto, a programação de petróleo, começa no recebimento das bateladas de mistura (definidas pela empresa) nos tanques da refinaria. Por outro lado, se uma refinaria tem um terminal dedicado, ela pode programar desde o recebimento do petróleo nos tanques do terminal e movimentá-lo para a refinaria através de um oleoduto. Neste caso, seu escopo de decisão é mais amplo, o que pode conferir mais flexibilidade pois possui mais recursos, e certamente mais complexo pois aumenta a natureza combinatória e o horizonte do problema de programação. A Figura 3.1 [5] apresenta uma representação esquemática do escopo de decisão da programação de petróleo de uma refinaria com terminal dedicado. Nela pode ser observado que o programador decidirá sobre o descarregamento dos navios petroleiros nos

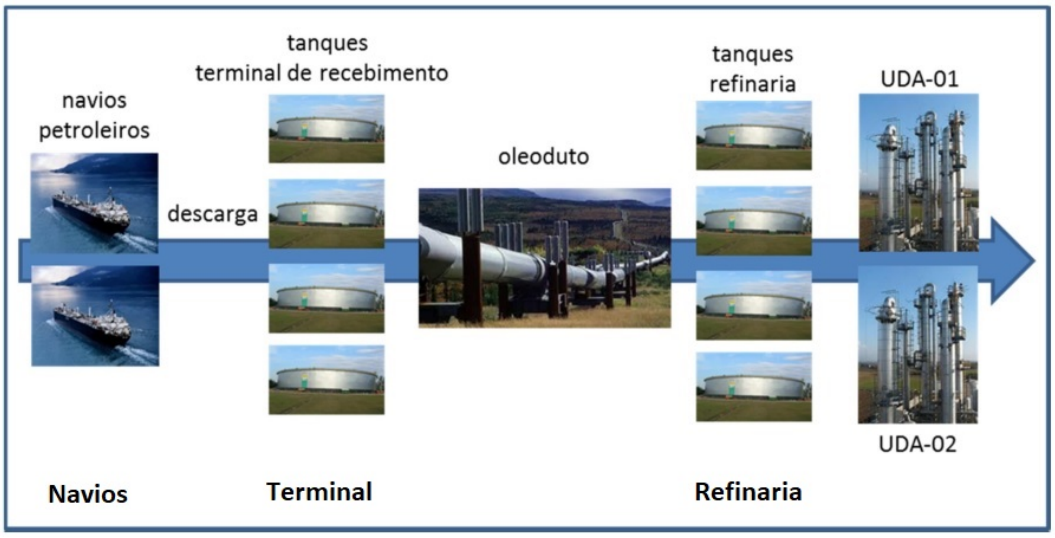

Figura 3.1: Representação esquemática do escopo de decisão da programação de petróleo de uma refinaria com terminal dedicado.

.

tanques de recebimento do terminal, a movimentação destes tanques para os tanques da refinaria através de oleoduto e a movimentação dos tanques da refinaria para, enfim, seu processamento nas unidades de destilação. Refinarias com este escopo de decisão são investigadas nesta tese e, por isso, seus tipos de movimentação são apresentados nas próximas seções.

3.1.1 Descarregamento dos navios petroleiros

Esta atividade trata do recebimento de petróleo e conecta duas áreas lógicas da programação: "Navios" e "Terminal", através de um duto para descarregamento.

O programador da refinaria, diariamente, é atualizado da previsão de navios que chegarão ao terminal nos próximos dias, conhecendo data/hora de chegada, composição e volume de cada mistura de petróleo disponível no navio. O descarregamento deve respeitar a ordem de chegada dos navios ao terminal e deve acontecer, preferencialmente, dentro de uma janela de operação conhecida, que tem início no horário de atracação do navio.

Os navios possuem compartimentos de grande capacidade que permitem trazer mais de uma mistura de petróleo separadamente. Como simplificação, estes compartimentos podem ser entendidos como tanques cuja capacidade é o volume da mistura que transportam. Nesta tese é usado o termo *tanqueCN* para diferenciar os compartimentos de um mesmo navio.

É importante definir o termo "mistura de petróleo" utilizado. "Petróleo" é

um termo genérico que representa a *commodity*, mas sempre que uma produção de petróleo é estabelecida, o hidrocarboneto ali retirado tem um conjunto de características e propriedades e a ele é dado um nome de batismo, por exemplo "Marlim P-33" ou "Sapinhoá" em casos dos petróleos brasileiros. Cada petróleo recebe um nome de batismo diferente porque cada um tem composição e, consequentemente, características e propriedades diferentes. Para generalizar o termo, uma mistura de petróleo representa qualquer razão de mistura entre petróleos ou mesmo um petróleo puro. Por exemplo, um compartimento com 40% de Sapinhoá e 60% de Marlim P-33 contém uma mistura de petróleo, assim como um compartimento com 100% de Marlim P-33 também contém uma mistura de petróleo.

Dessa forma, se um navio contiver apenas um *tanqueCN*, ele transportará apenas uma mistura de petróleo e se tiver mais *tanqueCNs* transportará tantas diferentes misturas de petróleo quanto seu número de *tanqueCNs*. Alguns exemplos de configurações possíveis em um navio qualquer são:

- apenas um *tanqueCN* transportando 100% de volume de um único petróleo
- apenas um *tanqueCN* transportando mistura homogênea de $x\%$ de um petróleo com $y\%$ de outro petróleo e $z\%$ de um terceiro petróleo.
- um *tanqueCN* transportando 100% de um petróleo e outro *tanqueCN* com mistura homogênea de $y\%$ de um petróleo e $z\%$ de outro petróleo.
- um *tanqueCN* contendo mistura homogênea com $y\%$ de um petróleo e $x\%$ de um petróleo e um outro *tanqueCN* com mistura homogênea de $w\%$ de um petróleo e $z\%$ de outro petróleo.

O descarregamento é feito através de um duto de pequena capacidade, que representa uma linha submarina entre o navio e o terminal. Uma linha é tratada como um duto homogêneo, ou seja, em todo instante de decisão em que se observe seu inventário, ela estará uniformemente preenchida com uma única mistura de petróleo. Independentemente de sua capacidade, os dutos nunca estão vazios. Dessa forma, no instante inicial de descargamento de cada *tanqueCN*, a linha submarina está cheia da mistura que foi retirada do *tanqueCN* anterior e este é o primeiro volume a ser deslocado. Por exemplo, se o último recebimento aconteceu no dia anterior a partir de um navio com petróleo $P03$ e hoje já está atracado para descarregamento um navio com apenas um *tanqueCN* com 80.000 m^3 de uma mistura de 35% de $P06$ e 65% de $P08$, o programador de produção deve primeiro decidir o destino do volume que está na linha submarina, pois ele será deslocado pelo descarregamento. A linha passa a ser preenchida pela mistura de $P06$ e $P08$ e assim permanece

até que um outro navio qualquer comece a descarregar. Supondo que a linha submarina tenha capacidade de 10.000 m^3, ao final do descarregamento, o navio estará completamente vazio, a linha preenchida com a mistura homogênea de 35% de $P06$ e 65% de $P08$, e o terminal terá recebido 10.000 m^3 de $P03$ e 70.000 m^3 dessa mesma mistura de $P06$ e $P08$.

Em cada navio o programador deve decidir qual dos *tanqueCNs* será descarregado primeiro e esta decisão se mantém até que o compartimento esteja vazio. Somente então será iniciado o descarregamento do próximo *tanqueCN* escolhido pelo programador de petróleo. O descarregamento completo de cada *tanqueCN* pode acontecer em mais de uma movimentação, destinado a mais de um tanque do terminal. Cada movimentação precisa ter definido o seu volume e o tanque do terminal que será seu destino. O volume total das movimentações que tem o mesmo *tanqueCN* como origem deve corresponder ao volume do *tanqueCN*, assim como o volume total de todas as movimentações de todos os *tanqueCNs* de um mesmo navio deve corresponder ao volume do navio.

Estas decisões afetam diretamente a disponibilidade dos equipamentos para outras atividades, pois os tanques do terminal executam apenas uma tarefa em cada instante de tempo, ou seja, eles não podem receber e enviar misturas de petróleo simultaneamente (essa habilidade, chamada operação pulmão, é evitada em tanques de petróleo). Este conjunto de decisões não tem impacto apenas volumétrico, pois as propriedades da mistura formada nos tanques do terminal pode, por exemplo, pelo alto teor de acidez, dificultar a programação das unidades de destilação da refinaria.

Resumidamente, é um tipo de atividade que se repete em diversas movimentações até que cada navio esteja completamente descarregado e, para cada uma delas, o programador deve definir: *tanqueCN* de origem, tanque de destino no terminal, vazão e volume da movimentação.

3.1.2 Movimentação de terminal para refinaria

Este tipo de atividade trata da movimentação entre os tanques do terminal e os da refinaria. Portanto, conecta as áreas "Terminal" (onde se encontram os tanques do terminal) e "Refinaria" (onde se encontram os tanques da refinaria) através de um oleoduto. Cada movimentação do terminal para a refinaria envolve três equipamentos: tanque do terminal, oleoduto de transferência e tanque da refinaria.

Essencialmente, qualquer tanque do terminal pode bombear para o oleoduto, assim como qualquer tanque da refinaria pode receber mistura de petróleo do oleoduto. Variações sobre essa premissa dependerão de especificidades

da refinaria. No entanto, embora seja fisicamente possível, por razões de segurança não se admite o envio simultâneo de mais de um tanque do terminal para o oleoduto e nem o recebimento simultâneo em mais de um tanque da refinaria.

Diferentemente de *tanqueCN*, usado apenas para diferenciar o armazenamento de misturas de petróleo em navio, os tanques do terminal e da refinaria são equipamentos sujeitos a restrições de operação e sobre os quais são consideradas regras de mistura. Estes tanques possuem uma capacidade de armazenamento máxima que deve ser respeitada quando forem destino de uma movimentação. Estes tanques também possuem uma restrição volumétrica mínima (além do zero de armazenamento), denominada lastro. O lastro define o volume mínimo de armazenamento que um tanque pode atingir. A diferença entre o volume total do tanque e o lastro é chamado volume operacional, ou seja, o volume máximo que efetivamente o tanque pode enviar em uma movimentação. O lastro existe para garantir que a sucção da bomba que retira produto do tanque opere sempre imersa no produto. Do ponto de vista de disponibilidade de material, é um volume a ser ignorado, devendo-se estar sempre atento ao volume operacional. No entanto, o volume do lastro impacta a composição final do tanque após um recebimento, pois o produto ali contido se mistura ao volume recebido e, por isso, deve ser considerado nos cálculos de composição e propriedades da nova mistura em tanque.

O oleoduto que conecta terminal e refinaria é de grande capacidade, percorrendo uma grande distância e, por isso, não pode ser considerado que esteja sempre cheio da mesma mistura. Assumir homogeneidade neste duto seria uma restrição grande e que não representaria a realidade, pois é frequente que esteja preenchido por misturas de petróleo diferentes. A cada mistura de petróleo bombeada para o duto é dado o nome de "item de bombeio" (IB). Por estar preenchido por itens de bombeio que representam misturas de petróleo diferentes, este duto é considerado heterogêneo. Uma premissa adotada no âmbito deste trabalho é que as interfaces entre os itens que preenchem o duto não se misturam, possuindo composições e volumes diferentes. No entanto, a soma dos volumes de todos os itens de bombeio totalizam o volume do duto em qualquer instante de tempo.

O duto opera sempre do terminal para a refinaria, seguindo o padrão FIFO (*First In First Out*) nos itens de bombeio. A pressão pela entrada de um item na extremidade do terminal movimenta todos os que estão à frente, fazendo com que o item na extremidade da refinaria comece a ser recebido nos tanques. Por exemplo, a Figura 3.2 apresenta o perfil do interior de um duto de 20.000 m^3 num dado instante t_1 qualquer. Ele está preenchido por dois itens de

bombeio IB01 e IB02, em que cada um possui 10.000 m^3. No instante seguinte, é iniciado o bombeio de 6.000 m^3 (volume da atividade), representando o item IB03, de um tanque do terminal para o oleoduto. Ao final desta movimentação, um retrato do interior deste duto indicaria 4.000 m^3 do IB01 (mais próximo à refinaria e que não foi completamente descarregado dado que o volume de IB03 é menor que IB01), 10.000 m^3 de IB02 e, por último, 6.000 m^3 de IB03 com composição igual à do tanque do terminal de onde ele foi bombeado. Ainda na Figura 3.2 pode ser observado que o volume útil do tanque do terminal (a esquerda do duto) diminui o volume correspondente ao da atividade (6.000 m^3), assim como também diminui o espaço disponível (S_{disp}) no tanque da refinaria (na extremidade direita). Enquanto o duto está em movimento, os itens não se misturam devido à pressão exercida sobre eles pelo bombeio. No entanto, se houver parada do equipamento, por fenômenos associados à mecânica dos fluidos, começará a ocorrer mistura entre os itens de bombeio. Este desdobramento não é desejado pois as misturas resultantes não serão homogêneas e, portanto, suas composições e propriedades serão desconhecidas e variáveis. A vazão de deslocamento dentro do duto é dada pela vazão do item que está sendo bombeado para ele e esta vazão é influenciada por propriedades da mistura, como por exemplo, viscosidade.

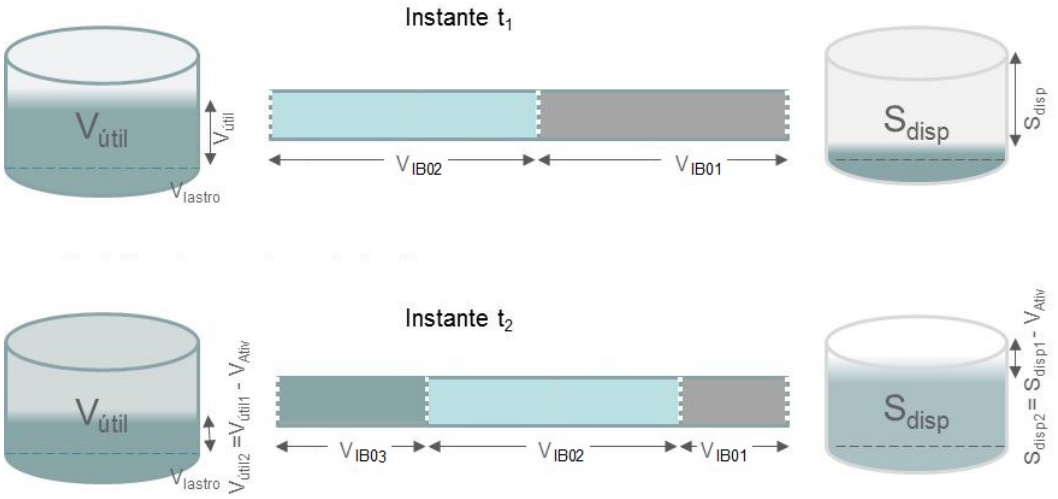

Figura 3.2: Retrato do interior do duto antes e depois de movimentação

Resumidamente, é um tipo de atividade que se repete em diversas movimentações de transferência entre terminal e refinaria e, para cada uma, quatro definições do programador: tanque de origem no terminal, tanque de destino na refinaria, vazão e volume da movimentação.

3.1.3 Carga das Unidades de Destilação

O programador de produção conhece a vazão média que as UDAs devem processar ao longo do mês. Essa é uma das informações resultantes do processo de planejamento da produção que a empresa realiza mensalmente de forma a estabelecer as principais diretrizes de produção que maximizem sua rentabilidade. São exemplos dessas diretrizes: vazão média mensal das unidades de processo, campanha de algumas unidades de processo e volume total de produção de alguns produtos.

Cada petróleo tem uma composição diferente de hidrocarbonetos e, para cada conjunto de condições operacionais (por exemplo, temperatura e pressão dos estágios de separação) a que for submetido, terá um perfil de rendimentos e propriedades dos produtos diferente. A esse conjunto de condições operacionais que caracterizam um modo de operar a unidade, denomina-se campanha. Em geral, as refinarias possuem poucas campanhas para suas unidades de destilação e pode haver mudança entre campanhas dentro do mesmo mês. Diferentes petróleos em uma mesma campanha resultarão em perfis de rendimentos e propriedades diferentes nos produtos. Da mesma forma, o mesmo petróleo em campanhas diferentes irá resultar também num perfil de rendimentos e propriedades diferentes de seus produtos. Essa diferença de perfil entre os produtos dos petróleos também confere flexibilidade e complexidade ao processo de programação, pois a composição de misturas é um recurso utilizado para o atendimento de restrições ou metas de produção.

Os tanques da refinaria são usados para dar carga nas UDAs e, em geral, no máximo dois estão simultaneamente na carga da mesma unidade em um dado instante de tempo. Dependendo da flexibilidade de alinhamento da refinaria em questão, os mesmos tanques podem alimentar mais de uma unidade de processo simultaneamente. Quando dois tanques estão na carga da mesma unidade de destilação, isso representa uma configuração chamada "injeção", onde um tanque participa com menos de 50% da carga (tanque de injeção) e o outro a complementa (tanque base). Este arranjo fornece maior flexibilidade para montar a composição da carga, mas dificulta a programação dos recursos, dado que durante a operação de injeção dois tanques estão alocados para a atividade. Cabe ao programador decidir sobre a relação custo $\times$ benefício deste dilema.

Os tanques da refinaria operam com restrições de capacidade e lastro, como já mencionado, e também não podem executar mais de uma tarefa no mesmo instante. No entanto, as atividades de carga de unidade trazem uma complexidade extra, chamada "Preparação". As UDAs possuem limitação no

teor de água e salmoura que pode entrar como contaminante da corrente de petróleo, pois em caso contrário, introduz problemas operacionais como, por exemplo, desarme da dessalgadora, desbandejamento da torre, além de, no médio prazo, aumentar a corrosão na unidade e, consequentemente, reduzir sua vida útil entre manutenções. Para garantir que o petróleo descarregado do navio e movimentado do terminal fique abaixo desse limite inferior, sempre que um tanque da refinaria recebe itens de bombeio do terminal, deve haver um intervalo entre o final dos recebimentos naquele tanque e a disponibilidade da mistura de petróleo para ser carga das UDAs. Neste intervalo o tanque não pode realizar nenhuma operação e ele deve ser suficiente para decantação e drenagem da salmoura do petróleo.

Além da restrição em relação à "Preparação", as UDAs geralmente também têm restrições de qualidade da mistura de carga a ser processada e/ou sobre os produtos produzidos. Estas restrições não se aplicam aos tanques, mas sim à mistura que entra na unidade, ou seja, é equivalente ao tanque se a movimentação usar somente um ou então se aplica à mistura resultante da participação de dois ou mais tanques que estejam na carga da unidade no mesmo instante.

Essencialmente há dois tipos de atividade que se repetem nas diversas movimentações que alimentam as UDAs. São elas: a carga será feita com apenas um tanque de origem ou a carga será feita com dois tanques de origem. Para cada movimentação do primeiro caso, o programador deve definir o tanque de origem na refinaria, UDA de destino, vazão e volume da movimentação. Para o segundo caso, o programador deve decidir os dois tanques de origem na refinaria, o percentual de participação de cada um deles, UDA de destino, vazão e volume da movimentação.

3.2 Objetivos

O objetivo de toda refinaria de petróleo é gerar lucro para a empresa através da transformação de petróleo em diferentes produtos finais que são comercializados. Esta referência econômica é utilizada para orientar as decisões do planejamento da produção, como, por exemplo, o elenco que será processado nas refinarias ao longo do mês e a vazão de carga das unidades (ambos afetados pela previsão de demanda de produtos finais). No entanto, quanto mais próximo do dia-a-dia operacional, mais difícil é monetizar as decisões no ambiente produtivo. Alguns aspectos podem contribuir com esse maior distanciamento:

– a definição do elenco de petróleo adquirido e a oferta e movimentação dos

derivados produzidos respondem pela maior parcela do cálculo financeiro da refinaria. No entanto, no ambiente de decisão da programação estas decisões já foram tomadas. O elenco a ser processado ao longo do mês está previsto, assim como os compromissos de produção dos produtos;

– não há uma metodologia consolidada para valoração de correntes intermediárias e a programação de produção é uma atividade dividida em áreas, onde o início e/ou final do escopo de atuação do programador é limitado por correntes intermediárias;

– não há uma metodologia estabelecida para monetizar as operações em si, como por exemplo, trocar o tanque alinhado na movimentação.

No nível de decisão da programação de produção, os objetivos da atividade estão mais próximos de fatores operacionais, como: a manutenção da vazão e campanha das unidades de processo próximas às definições do planejamento da produção; o atendimento aos compromissos (data e volume) de recebimento de matéria-prima e entrega de produtos; além de uma preocupação com uma operação mais suave (com menos transições) dos equipamentos. Na programação de petróleo estes objetivos estão associados à operação das unidades de destilação, descarregamento de navios petroleiros, operação do oleoduto e número de movimentações. Cada um destes itens é apresentado nas próximas seções.

3.2.1 Operação das unidades de destilação atmosférica

Dois aspectos são considerados para a operação das UDAs: a campanha e a vazão destas unidades. Ambas são informações resultantes do planejamento. No entanto, a vazão tem uma variabilidade muito maior do que a campanha, ou seja, enquanto é provável que a unidade opere durante muitos dias com a mesma campanha, a vazão da unidade tem oscilação diária. Esta variação pode estar associada às perturbações do ambiente produtivo, como por exemplo, a curva das bombas que succionam a mistura de petróleo do(s) tanque(s) para a unidade, ou pode ser uma decisão do programador de petróleo, que tem liberdade para tal, de forma a tornar o *schedule* viável. Por exemplo, se um navio petroleiro tiver previsão de atraso ou com volume menor do que o originalmente previsto, o programador pode definir uma vazão de carga mais baixa para compensar a mudança do recebimento, evitando a parada da unidade. De forma equivalente, se, por qualquer razão, houver estoque de petróleo na refinaria além do previsto e houver margem para armazenamento ou maior venda de produtos acabados, o programador pode programar uma

vazão aumentada por alguns dias. As variações na vazão diária não devem ser altas pois, em caso contrário, será difícil não se distanciar da meta estabelecida no planejamento.

É importante esclarecer que variações na vazão das unidades são permitidas, mas as UDAs devem operar continuamente. A parada total de unidades é um evento programado com anos de antecedência na refinaria, obedece um calendário de manutenção e demanda diversas ações para que a refinaria se prepare para os dias de indisponibilidade de produção da unidade em questão. Em empresas integradas, a programação de parada de uma unidade pode impactar até outras refinarias. Infelizmente, podem haver eventos que obriguem a parada das unidades fora das janelas programadas. No entanto, a parada de uma unidade nunca é um recurso que o programador de produção tem à sua disposição para decidir incluir no horizonte de poucos dias de um *schedule*.

Manter as UDAs operando continuamente e próxima à meta de vazão está na categoria dos objetivos mais importantes da programação pois desvios na sua meta afetam diretamente a produção da refinaria. A lógica é simples: se há menor processamento de matéria-prima, haverá menor produção de produtos e, portanto, menor venda e menor receita para a empresa. A meta de vazão das unidades e a previsão de volume produzido dos produtos finais são informações definidas no planejamento tático e ajustadas no planejamento operacional, o que significa dizer que são resultantes do que representa o maior benefício econômico para a empresa e, portanto, desvios dos seus valores, em princípio, representam desvios da maior lucratividade. Outro aspecto que pode vir de uma operação insuficiente das UDAs é o destino no petróleo não processado. Se ele já estava na refinaria, implica no aumento de estoque, o que pode aumentar a complexidade da programação dos cenários futuros dependendo do nível atingido. Se ele não pode ser recebido, é preciso avaliar as implicações que este não recebimento pode ter para a empresa, como por exemplo, a movimentação do petróleo para outro destino.

3.2.2 Descarregamento de navios petroleiros

A programação de chegada ao terminal de navios petroleiros, em geral, não é uma decisão da refinaria, mas sim de outros setores da empresa que a controla. Isso acontece porque essa definição é um outro problema de programação completo e complexo que considera características da frota de navios disponíveis, propriedade ou afretamento de navios, disponibilidade do petróleo pelo fornecedor, distância da unidade produtiva ao destino, entre outros.

A refinaria recebe a informação sobre a previsão de chegada de navios petroleiros para os próximos dias e estão aí incluídas:

- data-hora de chegada de cada navio;
- a janela de operação do navio (máximo de horas em que o navio deve ser liberado para deixar o porto após sua atracação).
- volume e composição de cada compartimento ($tanqueCN$) do navio.

O programador de petróleo deve se preocupar em concluir todo o descarregamento do navio dentro da respectiva janela de operação. Caso contrário, se houver atraso, a indisponibilidade do recurso na data prevista afetará diretamente o problema de programação de navios, pois é provável que já houvesse outra atividade alocada para este navio que nada tem a ver com aquela refinaria, como por exemplo, o atendimento a outras refinarias, e que poderá sofrer atraso. Dependendo do tipo de afretamento do navio e do tamanho do atraso, ainda pode existir uma penalização financeira. Ao não cumprimento da janela de operação do navio e consequente penalização associada, utiliza-se o termo "pagamento de sobrestadia".

Dentro da refinaria, o atraso no descarregamento do navio pode comprometer a disponibilidade do petróleo para o processamento nas UDAs. Esse comprometimento pode ser diretamente volumétrico ou indiretamente por propriedades, pois o petróleo atrasado pode ser necessário para misturar com outros já em terra e enquadrar limites de propriedades antes de poder ser processado nas UDAs. Devido às consequências complicadoras do atraso, o descarregamento completo do navio petroleiro dentro de sua janela de operação compõe, juntamente com a operação contínua das UDAs, os dois objetivos mais importantes da programação de petróleo pois podem impactar diretamente a produção da refinaria e até mesmo outras áreas da companhia.

3.2.3 Operação do oleoduto

A operação do oleoduto garante a movimentação do petróleo do terminal para a refinaria. A programação de suas atividades em geral vai além do horizonte de descarregamento de todos os navios e aquém da programação das UDAs (que se estende por todo o cenário). O modelo de decisão do programador de produção é de privilegiar a movimentação do óleo para a refinaria pois, dada a natureza variável do ambiente produtivo, em caso de qualquer problema no duto ou terminal, com o petróleo na refinaria, diminui a probabilidade ou severidade do impacto da operação das unidades enquanto a operação normal não é restabelecida.

Para este equipamento também é desejada uma operação contínua pois existem dois fenômenos indesejados associados à sua parada:

- se os itens de petróleo dentro do duto não estiverem em movimento, a pressão sobre eles é diminuída e assim facilita a mistura entre itens. Esta mistura não será homogênea e o perfil de composição do petróleo dentro do duto fica desconhecido, dificultando qualquer previsão de composição e qualidade dos tanques da refinaria para compôr a carga das UDAs;
- em pequenas paradas do duto o gasto energético para retomar sua operação é maior do que o de mantê-lo operando.

Este objetivo tem um aspecto auxiliar ao de operação contínua das UDAs pois, do ponto de vista de garantir petróleo na refinaria, a própria demanda das unidades tem atuação forte. Um papel desempenhado apenas por ele é de atuar numa programação mais contínua das bateladas do duto de forma a mantê-lo sempre pressurizado e evitar as interfaces de composição desconhecida.

Este objetivo não é considerado do mesmo nível de criticidade dos dois já apresentados pois o objetivo das unidades cumpre este papel de forma mais ativa. Movimentar uma hora a mais ou a menos de petróleo no duto, na maior parte dos casos, não terá grandes consequências. Dessa forma, esse objetivo tem dois efeitos combinados:a movimentação do petróleo e uma operação mais estável do oleoduto, o que certamente é uma característica desejada e buscada pelo programador de petróleo.

3.2.4 Trocas de Tanque

Cada operação altera a refinaria. Pode ser através da geração de um regime transiente na unidade, pois é uma nova composição que está entrando, ou por envolver uma manobra de equipamentos (válvulas, linhas) em painel ou em campo, o que envolve risco.

O programador de petróleo tende a ser conservador no número de movimentações, ou seja, se não há um motivo para mudar os recursos envolvidos (por exemplo, por capacidade máxima ou mínima atingida dos equipamentos, misturar propriedades de petróleos em tanque), ele provavelmente irá manter os mesmos recursos. Ou seja, quando um tanque é alinhado na carga de uma unidade, em princípio, ele continuará nesta atividade até que seu lastro seja atingido, quando, então, poderá ser utilizado em nova(s) movimentação(ões) do oleoduto. Nas atividades de carga de unidade é fácil compreender esta orientação, pois se um tanque for substituído da carga de uma unidade com, por exemplo, 50% de seu volume ainda disponível e ele for usado num recebimento

de oleoduto, ao final da nova atividade, o tanque terá que passar novamente pelo tempo de preparação descrito na seção 3.1.3 antes de estar disponível para uma nova atividade de carga. Dessa forma, supondo um mesmo volume final fornecido, o tempo indisponível do equipamento foi maior pois dois tempos de preparação aconteceram. Este comportamento pode tornar as decisões de alocação dos equipamentos mais complexas.

Assim como o objetivo de operação do oleoduto, este também busca um *schedule* mais contínuo como um todo. Não está diretamente associado a um impacto no processamento e, por isso, não tem o mesmo nível de criticidade da operação contínua das UDAs e descarregamento do navio.

4
Modelo Evolutivo base

Este capítulo descreve o modelo *Quantum-inspired Grammar-based Linear Genetic Programming* (QIGLGP), desenvolvido para propor soluções otimizadas para a programação de petróleo dentro do escopo de decisão de uma refinaria. Ainda que esteja publicado em [5] é importante seu detalhamento aqui para maior compreensão deste trabalho, pois esta tese propõe uma abordagem multiobjetivo para o modelo em questão.

O QIGLGP é um modelo evolutivo que possui quatro entidades principais: uma gramática para representar as atividades da programação em si; indivíduos quânticos que representam todas as possíveis soluções de programação; indivíduos clássicos que representam soluções produzidas; e um operador quântico que promove a evolução ao longo das gerações. Cada uma dessas entidades será detalhada nas próximas seções deste capítulo.

4.1
Gramática para a Refinaria

Como discutido no Capítulo 3, a programação de produção trata de realizar um conjunto de atividades ao longo de um horizonte de tempo. Essencialmente, as atividades representadas neste trabalho são movimentações para as quais é necessário definir também os recursos envolvidos, a duração e o volume. Sob esta perspectiva é possível desenhar uma Linguagem Específica de Domínio (*Domain Specific Language - DSL*) que represente as principais atividades de programação da refinaria que se estiver modelando. A DSL é uma linguagem de programação ou linguagem de especificação que oferece, através de notações adequadas e abstrações, poder expressivo voltado e, geralmente, restrito a um determinado domínio de problema, sendo, para tal, baseada nos conceitos relevantes e características desse domínio [79]. Portanto, para desenvolver uma DSL adequadamente, o passo mais importante é mapear as principais atividades da programação. A notação escolhida para definição desta gramática é baseada na BNF (*Backus-Naur Form*), que descreve estruturas gramaticais admissíveis na construção de uma linguagem através da quádrupla {S, N, T, P}, onde: S denota o símbolo inicial, N denota o conjunto de símbolos não terminais, T denota o conjunto de símbolos terminais e P são as produções

[80]. A Figura 4.1 apresenta a estrutura BNF de uma gramática apenas de exemplo. O símbolo inicial $\langle func \rangle$ define a lista de funções que estão disponíveis na gramática. O símbolo | separa alternativas existentes para seleção da função. Cada uma destas funções pode ter elementos não terminais, representados pelos argumentos $\langle arg0X \rangle$ que as acompanham. O símbolo $\langle\ \rangle$ significa que a gramática tem mapeada uma lista de alternativas para este argumento. O símbolo " " identifica os elementos terminais que podem ser utilizados em cada argumento. Uma produção que pode resultar da estrutura da Figura 4.1 é $Func03(texto04, valor11 opcao02, texto11)$. Seguindo esta estrutura para o desenvolvimento da linguagem é possível garantir que apenas instruções topologicamente válidas sejam geradas, ou seja, no exemplo em questão, não é possível produzir uma função do tipo $Func02(valor03, texto12)$ porque $texto12$ não é um elemento terminal para $\langle arg02 \rangle$ que é o argumento de $Func02$. Aplicando esta lógica na programação de petróleo, não há risco de uma atividade que representa carga da UDA ser criada utilizando outros tanques que não os que armazenem petróleo para alimentar esta unidade.


```
<func>::=   "Func01"    |
            "Func02" <arg01> <arg02>  |
            "Func03" <arg02> <arg03> <arg04> |
            •
            •
            •
            "FuncN"  <arg01> <arg04> <arg07>


<arg01>::=      "valor01" | "valor02" | "valor03"
<arg02>::=      "texto01" | "texto02" | "texto03" | "texto04"
<arg03>::=      "valor11" <subarg01> | "valor12" <subarg02>
<subarg01>::=   "opcao01" | "opcao02"
<subarg02>::=   "opcao11" | "opcao12" | "opcao13"
<arg04>::=      "texto11" | "texto12"
<arg07>::=      "int01"   | "int02" | "int03" | "int04"
```

Figura 4.1: Estrutura para criação da gramática.

4.2 Gene quântico e indivíduo quântico

Um gene quântico guarda em si todas as produções que podem ser criadas com a gramática. A unidade básica de informação com a qual o gene quântico trabalha é o *qudit*, que pode ser descrito por um vetor de estado em um sistema mecânico quântico de d níveis, onde d representa o número de estados nos quais

o *qudit* pode ser medido. Isto é, representa a cardinalidade do *token* que terá seu valor determinado pela observação do seu respectivo *qudit*. O estado de um *qudit* (Ψ) é uma superposição linear dos d estados e pode ser representado pelo somatório das probabilidades de que o *qudit* seja encontrado em cada estado quando observado [81]. Um *qudit* representa todas as probabilidades até o momento em que ele é observado. A Figura 4.2 representa a forma como o estado do *qudit* é implementado: uma estrutura de dados como uma roleta, cujo valor de saída indica o *token* observado. *Token* é a nomenclatura adotada no modelo Programação Genética Linear com Inspiração Quântica (PGLIQ) [81] para definir o identificador único que representa internamente as funções e os terminais, respectivamente.

Utilizando a roleta da Figura 4.2 para representar as funções da gramática da Figura 4.1, tem-se que a fatia azul mapeia a região do *token* $k = 0$, vinculado à primeira função da gramática, $Func01$. A fatia laranja mapeia o *token* $k = 1$, vinculado à segunda função, $Func02$, e assim sucessivamente até que a fatia amarela mapeia a região do último *token* $k = d - 1$, vinculado à última função $FuncN$. Neste exemplo de roleta $Func01$ tem 12% de probabilidade de ser observada (intervalo [0 , 0,12]), $Func02$ tem 37% de probabilidade (intervalo (0,12 , 0,49]), e assim sucessivamente até que a última função tem 18% de probabilidade de ser observada (intervalo (0,82 - 1,0]). O somatório existente na figura representa todas as demais funções (da terceira até a penúltima) que não foram explicitadas no exemplo. Dessa forma, se o valor sorteado for 0,24 a roleta indica o *token* de função 1, que corresponde a $Func02$, pois a implementação é base 0.

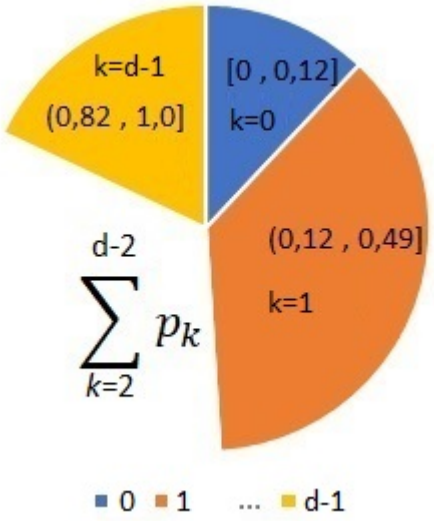

Figura 4.2: Exemplo de probabilidade de estados, token associado e instrução equivalente para observação de um *qudit* de função)

O gene quântico implementa uma estrutura de roleta em cada definição demandada pela gramática, com a representação de todos os elementos terminais disponíveis. A primeira roleta é associada à definição da função. Depois, para cada função, há outras roletas que mapeiam os elementos terminais com *tokens* de terminal. Seguindo o exemplo, $Func02$ tem outras duas roletas associadas a ela. A primeira representa a distribuição de probabilidades dos elementos terminais de $arg01$ e, de forma análoga, a segunda representa a distribuição de probabilidades dos elementos terminais de $arg02$.

O gene quântico pode ser visualizado como uma árvore de possíveis estados, cada um com sua probabilidades e mapeamentos dos *tokens* com as funções e terminais. A Figura 4.3 apresenta a estrutura de um gene quântico. É utilizado o mesmo exemplo de gramática da Figura 4.1, podendo-se observar que todas as possíveis produções desta gramática estão representadas. As linhas contínuas conectam cada função com seus argumentos. Cada seleção (função ou argumento) tem um *token* associado para o número de elementos que ela pode representar. *prob* representa a probabilidade de que aquele elemento seja observado e ela depende do gene, da função e do argumento. Por exemplo, o argumento $arg2$ é necessário nas funções $Func02$ e $Func03$. Em ambos os casos, $arg2$ disponibiliza os mesmos Y elementos terminais como alternativa. No entanto, a probabilidade de que o elemento 1 seja observado como $arg2$ da função $Func02$ é diferente da probabilidade de que este mesmo elemento 1 seja observado como $arg2$ de $Func03$ durante o processo de observação do gene quântico em questão. Na figura, a letra p indica que se trata de uma probabilidade, $a\langle num\rangle$ representa o argumento ao qual a probabilidade se refere, g representa o índice do gene quântico em questão, o numeral na sequência se refere ao *token* de função para o qual a probabilidade é válida e o último numeral se refere ao *token* do elemento terminal para o qual a probabilidade é válida. $pa2_{g31}$ significa a probabilidade do elemento terminal 1 do argumento 2 na função 3 ser observado no gene g. Havendo mudança de gene, função, ou elemento terminal do argumento, a probabilidade deste outro conjunto provavelmente terá um valor diferente.

O indivíduo quântico nada mais é do que uma lista de genes quânticos com tamanho fixo. Como o gene quântico representa a superposição de todas as produções que podem ser geradas pela gramática, o indivíduo quântico acaba por representar todas as soluções possíveis de serem geradas para o problema em questão. No caso da programação de petróleo, o gene quântico guarda a distribuição de probabilidades para que qualquer atividade possa ser gerada. O *schedule* é exatamente um sequenciamento e repetição de atividades de programação, portanto, o indivíduo quântico representa todos os *schedules*

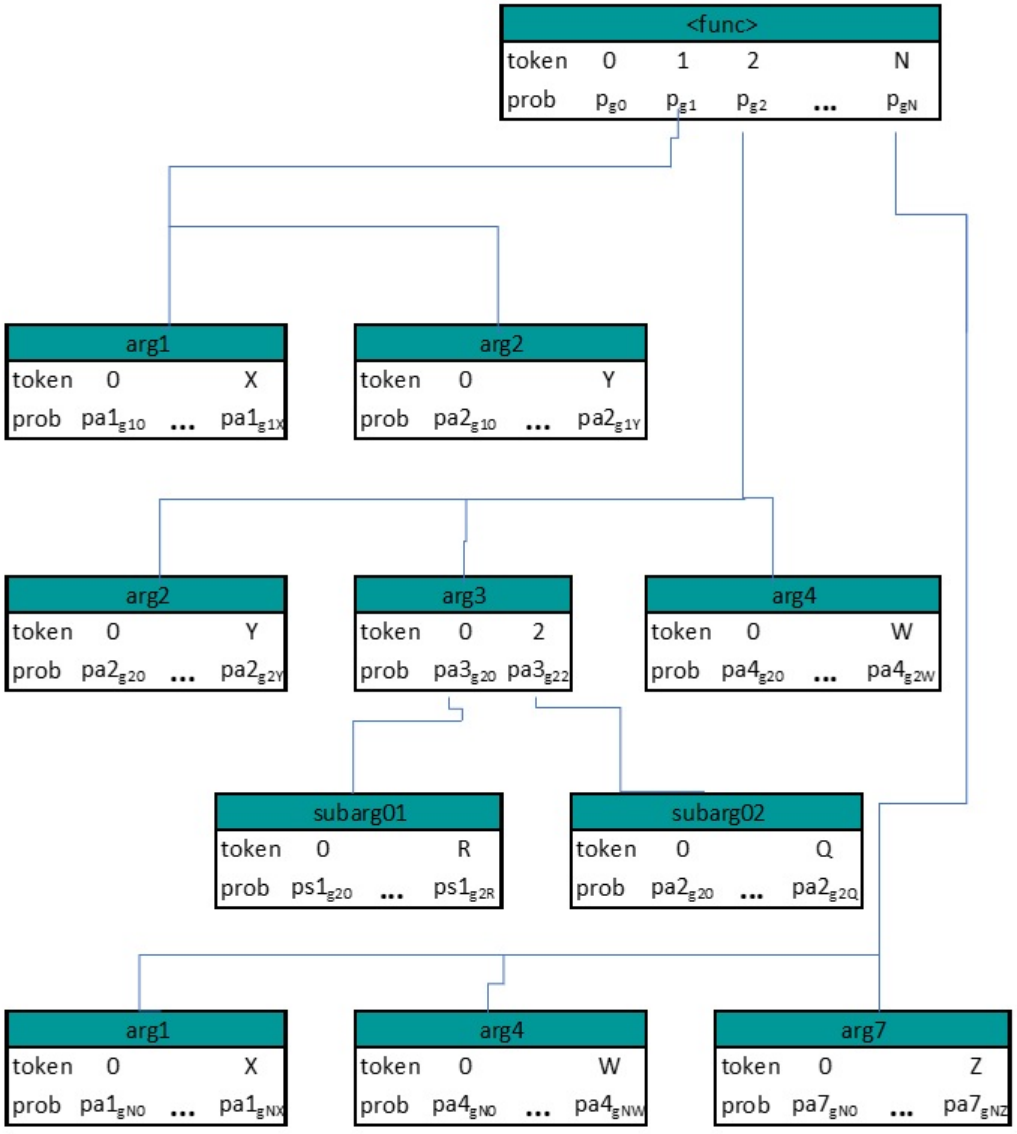

Figura 4.3: Exemplo da estrutura de um gene quântico

que podem ser gerados.

4.3 Gene clássico e indivíduo clássico

O gene clássico é o resultado de todas as observações feitas no gene quântico necessárias para gerar uma produção. Mantendo o exemplo utilizado na seção 4.2, se, para o gene quântico q a observação da roleta de funções indicar o valor 0,25, então $Func02$ foi selecionada. Esta função tem dois argumentos, portanto, duas novas observações nas suas respectivas roletas precisarão ser feitas até que se defina a estrutura completa da função. Esta forma completa, por exemplo, $Func02(valor02,texto04)$ é o gene clássico. A observação completa de um gene quântico resulta em apenas um gene clássico e há um mapeamento direto entre eles, ou seja, o enésimo gene quântico gera o enésimo gene clássico.

O indivíduo clássico é a disposição linear de todos os genes clássicos. Indivíduo clássico e indivíduo quântico têm o mesmo número de genes. A execução do programa composto pelo indivíduo clássico é feita da primeira para a última linha, como é característico da programação genética linear (*Linear Genetic Programming - LGP)*. Cada gene clássico é submetido às restrições que lhe competem de acordo com o tipo de função que representa no problema.

Se um gene quântico representa todas as possíveis produções, o gene clássico representa a função observada. Se o indivíduo quântico representa todas as possíveis soluções, o indivíduo clássico representa a solução resultante de todas as observações de todos os genes quânticos. Na programação de petróleo, cada gene clássico é uma movimentação e cada indivíduo clássico representa uma solução completa de *schedule*, que pode ser boa ou não.

Para dar flexibilidade ao indivíduo clássico e permitir que seu tamanho varie (em relação ao número de instruções), na gramática é disponbilizada a função *NOp* como um *intron* do modelo. Os *introns* são elementos que não representam nada no domínio do problema e que, no caso da LGP clássica, surgem espontaneamente na criação da população inicial e que emergem ao longo da evolução, como resultado de operações de cruzamento e mutação [82]. Na QIGLGP, o *intron* explícito *NOp* foi incorporado à gramática na forma de uma função que não representa nenhuma movimentação na refinaria, uma *NoOperation - NOp*. Dessa forma, em qualquer momento da evolução a observação do gene quântico pode levar à substituição de *NoOperation* por uma movimentação no gene clássico, incluindo, por exemplo, uma movimentação entre terminal e refinaria entre dois genes que representavam carga de unidade. É importante salientar que um gene clássico (instrução) que não resulta em

uma movimentação também representa um *intron*. Porém, nesse caso, trata-se de um tipo de *intron* que surge ou emerge espontaneamente, a exemplo do que ocorre na LGP clássica.

4.4 Operador quântico

A evolução dos indivíduos acontece pela atuação do operador quântico (OpQ) sobre as probabilidades de observação do *qudit* nos indivíduos quânticos. Todos os indivíduos clássicos têm suas aptidões medidas de acordo com os valores dos objetivos na função de avaliação e, de acordo com este resultado, o OpQ incrementa a probabilidade, no indivíduo quântico correspondente, de observar todos os elementos que geraram cada instrução desses indivíduos clássicos. A atuação do OpQ é dada por

$$p_{(i,g)} = p_{(i,g-1)} + sOpQ \times p_{(i,g-1)}, \tag{4-1}$$

onde $p_{(i,g)}$ representa a probabilidade de observação da função (ou terminal) i na geração g que represente a função (ou terminal) observado no melhor indivíduo na geração $g-1$ e $sOpQ$ representa um passo, ou seja, o incremento na probabilidade de observação desta função ou terminal. Os valores das probabilidades são padronizados de modo a garantir soma 1 para cada conjunto de funções e terminais.

4.5 Entidades específicas para a programação de produção

A QIGLGP tem ainda outras entidades específicas para o domínio do problema de programação de petróleo:

- Planta: representa o estado da planta de processo (equipamentos em geral) num determinado instante. Sempre que uma movimentação for feita, é utilizada e atualizada a informação do inventário (volume, composição e propriedades) dos equipamentos envolvidos, que está armazenada na estrutura de dados da planta de processo. Cada equipamento da planta também conhece a informação de quando está disponível para uma nova movimentação baseado na atualização de quando foi concluída sua última movimentação.
- Restrições: lista de restrições do domínio do problema que devem ser respeitadas antes que uma movimentação possa ser feita. Por exemplo, tempo de preparação que deve ser aplicado antes da carga das UDAs, limites de propriedades em tanque etc.

– Simulador: executa os cálculos que permitem avaliar se uma dada movimentação não viola as restrições do problema e também os que atualizam a planta de processo caso a movimentação possa ser realizada, ou seja, volume, composição e propriedades após a movimentação, além da nova data-hora de disponibilidade dos equipamentos envolvidos.

4.6 Funcionamento do Modelo

O modelo inicia com a criação dos indivíduos da população quântica. A distribuição inicial das probabilidades das funções é dada através de um parâmetro do modelo que define a probabilidade da instrução *NOp* ser observada. O complemento é uniformemente distribuído entre as demais funções da gramática, ou seja, se *NOp* for definida com 0,60 e houver outras 8 funções na gramática, *Func01*, ... , *Func08* terão, cada uma, 0,05 de probabilidade de serem observadas. Na roleta de probabilidades, *NOp* ocupa o intervalo [0,0 - 0,60], *Func01* ocupa (0,60 - 0,65], e assim sucessivamente, até que *Func08* completa o intervalo de sorteio com (0,95 - 1,0].

Para os argumentos das funções, os elementos terminais, não há nenhum parâmetro. Dentro de cada grupamento, as probabilidades são uniformemente distribuídas entre os elementos, ou seja, se o argumento $arg02$ de uma função tem quatro possíveis valores (quatro elementos terminais), cada um terá uma probabilidade inicial de 0,25 de ser observado.

A partir da observação de cada indivíduo quântico, o indivíduo clássico correspondente é criado. Para criar o *schedule* que este indivíduo representa, a cada gene clássico são executadas duas etapas: a instrução que o gene clássico representa é submetida às restrições do problema e, se ela puder ser executada, esta instrução representa uma atividade do *schedule* e a planta de processo é atualizada.

O modelo QIGLGP também aceita a inclusão de atividades pré-definidas no início do *schedule*. Estas atividades representam movimentações que já estão acontecendo no início do horizonte do cenário (por exemplo, um tanque que já está na carga de uma UDA) e que não serão trocadas; por isso, o otimizador não pode alterá-las. Elas devem ser representadas seguindo os padrões da gramática e formam parte da solução. Todos os cenários de programação possuem atividades pré-definidas e elas não ocupam mais do que algumas horas.

O conjunto de atividades pré-definidas juntamente com a lista de instruções do indivíduo clássico que puderam ser executadas compõem o *schedule*

obtido pela interpretação do indivíduo. A qualidade desta solução é avaliada de acordo com a função de aptidão do problema.

No modelo QIGLGP, os múltiplos objetivos do problema são tratados de forma independente e hierarquizada, ou seja, *a priori* é definido qual objetivo é mais importante, o segundo mais importante, terceiro e assim sucessivamente. Os valores de cada objetivo são calculados independentemente e a ordenação da população se dá de acordo com o desempenho do indivíduo, seguindo a priorização definida dos objetivos. O indivíduo melhor avaliado será aquele cujo *schedule* resultar no menor valor do objetivo 1, não importando o valor dos demais objetivos. O segundo melhor indivíduo será o de segundo menor valor do objetivo 1 e assim sucessivamente. Quando houver empate no valor do objetivo 1, o desempate é o valor do objetivo 2, sendo definido como melhor o indivíduo com menor valor para o objetivo 2.

A QIGLGP também é um algoritmo elitista que trabalha com uma população auxiliar. A cada geração, essa população auxiliar é gerada a partir da observação dos indivíduos quânticos e essa população é ordenada juntamente com a população da geração anterior de forma que os melhores indivíduos desse conjunto completo seja definido como a população da geração atual.

Esta ordenação orientará a aplicação do operador quântico nos indivíduos quânticos. O indivíduo quântico "índice 1" terá sua distribuição de probabilidades alterada de forma que o próximo indivíduo clássico resultante de sua observação tenha maior probabilidade de refletir as escolhas que resultaram no melhor indivíduo clássico da geração anterior. O indivíduo quântico "índice 2" será atualizado de acordo com o segundo melhor indivíduo clássico e assim sucessivamente.

Este processo de observação dos indivíduos quânticos para geração dos indivíduos clássicos correspondentes, avaliação destes indivíduos, ordenação da população e atualização da distribuição de probabilidades dos indivíduos quânticos se repete até que um critério de parada é atingido. Em todos os modelos desta tese, este critério é um número fixo de gerações.

Como forma de evitar convergência prematura do modelo e aumentar sua capacidade de percorrer o espaço de soluções, foi estabelecido um critério para reinicialização da população: sempre que o valor de cada objetivo de todos os indivíduos das populações clássica e auxiliar for o mesmo, as populações quântica e clássica são reiniciadas, preservando somente o melhor indivíduo encontrado até o momento.

O funcionamento do modelo QIGLGP para tratar o problema de programação de petróleo é apresentado pelo Algoritmo 1, onde NI representa o número de indivíduos das populações clássica (PC) e quântica (PQ), g repre-

senta a geração em questão, nGP o número total de gerações, IC_{Mg} é o melhor indivíduo da geração e IC_M é o melhor indivíduo gerado até então.

1: geração g=0
2: Criar os NI indivíduos da população quântica inicial($\mathrm{PQ_g}$)
3: Criar os NI indivíduos da população clássica inicial($\mathrm{PC_g}$) a partir da observação de $\mathrm{PQ_g}$
4: Construir *schedule* de cada indivíduo da ($\mathrm{PC_g}$)
5: Medir aptidão do *schedule* de cada indivíduo da ($\mathrm{PC_g}$) de acordo com a avaliação da função multi-objetivo
6: $g \leftarrow g + 1$
7: **while** $g <= nGP$ **do**
8: Criar população clássica auxiliar ($\mathrm{PC_{aux}}$) a partir da observação de $\mathrm{PQ_{g\text{-}1}}$
9: Construir *schedule* de cada indivíduo da ($\mathrm{PC_{aux}}$)
10: Calcular a aptidão do *schedule* de cada indivíduo da ($\mathrm{PC_{aux}}$) de acordo com a avaliação de cada termo da função multiobjetivo
11: Ordenar os indivíduos da $\mathrm{PC_{g\text{-}1}}$ e da PCaux de acordo com suas aptidões e a priorização definida para os objetivos.
12: $PC_{\mathrm{g}} \leftarrow NI$ melhores indivíduos de $\mathrm{PC_{aux}}$ + $\mathrm{PC_{g\text{-}1}}$
13: Aplicar o operador quântico (OpQ) a cada um dos NI indivíduos quânticos de acordo com o indivíduo clássico correspondente
14: Comparar o melhor indivíduo clássico obtido da geração g ($\mathrm{IC_{Mg}}$) com o melhor indivíduo clássico obtido pela evolução até o momento ($\mathrm{IC_M}$)
15: **if** IC_{Mg} is better than IC_{M} **then**
16: $IC_M \leftarrow IC_{\mathrm{Mg}}$
17: **end if**
18: **if** Avaliação de todos os indivíduos de $\mathrm{PC_g}$ são iguais **then**
19: Voltar à etapa 2
20: **end if**
21: $g \leftarrow g + 1$
22: **end while**

Algoritmo 1: Funcionamento do QIGLGP

5
Modelos multiobjetivo para programação de petróleo

Esta tese propõe uma abordagem multiobjetivo baseada em programação genética para o problema de programação de petróleo em refinaria capaz de oferecer um conjunto de soluções de qualidade para o programador de produção, geradas em tempo computacional equivalente ao que hoje ele leva para produzir uma solução com base na falta de ferramentas de apoio.

Ao oferecer mais de uma alternativa, em que todas representam a melhor relação de compromisso entre os objetivos, é dada ao programador a flexibilidade de avaliar estes *schedules* e decidir *a posteriori* qual será adotado na refinaria considerando aspectos que podem não estar representados, pois todo modelo é uma simplificação da realidade.

Este capítulo apresenta os quatro objetivos operacionais que guiam a programação de petróleo e as estratégias de solução multiobjetivo propostas, desenvolvidas utilizando os princípios de ordenação não-dominada (*non-dominated sort*) dos indivíduos e de decomposição do problema em subproblemas, que são a base de algoritmos evolutivos multiobjetivo importantes na literatura atual.

5.1
Objetivos do problema

Quanto mais próximo o processo decisório estiver do dia a dia operacional, mais ele é orientado para manter a operação da planta o mais estável possível e não comprometer a produção. Como discutido na seção 3.2, os objetivos operacionais não têm todos o mesmo nível de criticidade, pois os que estão diretamente relacionados com a manutenção da produção são considerados acima dos que estão mais associados à um perfil de movimentações mais suave.

Neste trabalho, o processo evolucionário dos algoritmos propostos é orientado por quatro objetivos, todos de minimização: desvio da carga programada das UDAs ("objetivo de UDA"), descarregamento dos navios petroleiros fora da janela de operação ("objetivo de navio"), tempo e número de paradas do oleoduto terminal-refinaria ("objetivo de duto"); e, por último, número total

de trocas de tanques ("trocas de tanque"). A metodologia para o cálculo de cada objetivo é detalhada nas próximas seções.

5.1.1 Desvio da carga programada das UDAs

O programador decide a vazão com que cada UDA vai operar de forma a se manter aderente às diretrizes do planejamento da produção. Esse é um dado de entrada do modelo. A vazão planejada integrada no tempo do horizonte da programação fornece o valor de carga pretendido em todo o período.

Este objetivo visa, portanto, a minimizar o desvio entre a carga programada e a das movimentações resultantes das decisões do algoritmo evolutivo. Para determinar este desvio, é definido que toda movimentação só pode ser criada com o valor de vazão programado para a UDA em questão. Quando, por qualquer razão, não é possível operar a UDA na vazão definida, a movimentação não é alocada neste instante de tempo, passando a fazê-lo somente quando for possível. Por exemplo, se um tanque x está na carga de uma UDA até o instante h e, ao seu final, não há outro tanque disponível para carga na unidade (somente z horas depois), o modelo não contempla baixar a vazão de carga da UDA para que o tanque x possa ser mantido por mais z horas. O que acontecerá é que a unidade ficará parada durante essas z horas. Na refinaria esta parada não aconteceria, pois a decisão do programador seria diminuir a vazão de carga da unidade. Nesta proposta não foi adotada esta abordagem, pois seria mais uma variável manipulada, aumentando a complexidade do modelo. É assumida hipótese simplificadora de que pequenas paradas de UDA sugeridas pela otimização são absorvidas pelas variações do ambiente operacional e pode ser desconsideradas. Matematicamente, este objetivo mede o número de horas totais paradas das UDAs. Na prática, ele representa a diferença entre o volume de processamento pretendido e o que foi conseguido pelo modelo, ou seja, representa o desvio em relação à carga programada. A formulação deste objetivo é dada por:

$$UDAs = (Horiz \times nUDA) - \sum_{UDA}^{UDA=nUDA} \sum_{m_{UDA}}^{m=tm_{UDA}} (HF_{m,UDA} - HI_{m,UDA}), \quad (5\text{-}1)$$

onde *UDAs* é o número total de horas que todas as UDAs estiveram paradas ao longo do cenário; *Horiz* é o número de horas do horizonte do cenário; *nUDA* é a quantidade de UDAs da refinaria; *UDA* é o índice da UDA em questão; m é o índice da movimentação (atividade) que está sendo avaliada; tm_{UDA} é o número total de movimentações da UDA em questão; *HF* é a hora final da movimentação m da UDA em questão; e *HI* é a hora inicial da movimentação m da UDA em questão.

5.1.2 Atraso no descarregamento de navios petroleiros

A programação de chegada ao terminal de navios petroleiros não faz parte do escopo de decisões deste tipo de refinaria, ou seja, ela recebe a informação da previsão sobre a janela de operação destes navios, bem como o volume e composição de cada mistura de petróleo que está sendo transportada, conforme descrito na seção 3.2.2.

A vazão de descarregamento do navio também é um parâmetro do modelo e pode assumir valores diferentes para cada navio descarregado. Na prática, sabe-se que essa vazão é afetada por características da mistura sendo bombeada, como densidade e viscosidade. No entanto, não há uma formulação para esta correlação. Com base na experiência, as refinarias utilizam dois ou três valores, dependendo da mistura de petróleo que estão recebendo. Pela falta de uma formulação ou heurística mais precisa, nem o QIGLGP original e nem suas variações multiobjetivo propostas nesta tese modelam correlações de como as propriedades da mistura impactam a vazão máxima de bombeio. Esta informação é um dado de entrada para os modelos.

Durante a janela de operação, o descarregamento do navio deve ser completo, mas a operação não precisa ser ininterrupta, ou seja, o navio pode ser parcialmente descarregado, interromper a atividade e retomá-la algumas horas depois, completando-a. Se todo o processo ocorrer dentro da janela de operação, não há penalização.

Este objetivo é definido como uma soma entre qualquer volume não descarregado dos navios e o número de horas após o final da janela de operação que foi utilizado para o completo descarregamento do petróleo. Matematicamente é definido da seguinte forma:

$$Navio = \sum_{n}^{n=nNavio} VolBordo_n + \sum_{n}^{n=nNavio} (HF_n - FJO_n), \qquad (5\text{-}2)$$

onde $Navio$ é o valor do objetivo relacionado ao descarregamento de navios; n é o navio em questão; $nNavio$ é o número total de navios no cenário; $VolBordo$ é o volume de mistura de petróleo não descarregado do navio até o final do horizonte de programação; HF é a data-hora final de descarregamento total do navio; e FJO é a data-hora final da janela de operação programada para o navio.

O descarregamento segue a fila de navios estabelecida como dado de entrada, ou seja, se o cenário possui dois navios, o descarregamento do segundo só pode começar após a conclusão do descarregamento (com atraso ou não) do primeiro navio. E, se um dado navio tiver mais de um *tanqueCN*, cabe

ao otimizador decidir qual descarregar primeiro e este será mantido até sua conclusão para depois seguir a descarga do próximo *tanqueCN*. O modelo permite e penaliza a descarga atrasada do navio, mas não é permitida a antecipação, ou seja, nenhum navio começa a operar antes da hora de início de sua janela de operação.

Este objetivo avalia dois aspectos: a primeira parcela da equação informa se há algum volume não descarregado no navio (infração mais grave), enquanto a segunda parcela informa sobre o número de horas em atraso quando o descarregamento é completado. Embora sejam grandezas diferentes (volume e horas), o objetivo cumpre seu papel no sentido de que qualquer indivíduo que não descarregue completamente o navio será pior do que qualquer indivíduo que atrase a descarga completa, porque o volume não entregue, quando acontecer, será da ordem de grandeza de milhares de m^3 e o número de horas em atraso será da ordem de grandeza máxima de centenas de horas. Assim, a evolução será orientada a garantir a entrega do volume total e depois o cumprimento da janela de operação.

5.1.3 Parada na movimentação do oleoduto

Como discutido na seção 3.2.3, quanto menos paradas houver na programação do oleoduto, de melhor qualidade ela será. Quanto mais tempo o duto operar, mais ele estará contribuindo com a manutenção da carga das UDAs. Dessa forma, este objetivo foi modelado para buscar a máxima operação do duto e o comportamento contínuo desejado.

São avaliados dois aspectos:

- o número de paradas no duto: uma parada no duto é identificada pelo fato de que, durante um intervalo de tempo qualquer, não houve movimentação no duto;
- o número de horas que o duto ficou parado ao longo do horizonte, enquanto há programação para ele.

Assim como aconteceu para o descarregamento de navio, estes itens possuem ordens de grandeza diferentes (repetições inteiras e horas). O número de paradas é da ordem de unidades, enquanto o número de horas paradas varia de unidades a dezenas, com maior probabilidade para o segundo grupo. Como é desejado priorizar a minimização do tempo total de paradas e depois o número de paradas, adota-se um recurso matemático de utilizar o valor inteiro das horas paradas e de dividir o número de paradas por 1.000 para garantir que esta informação fique como decimal do valor do objetivo enquanto

o tempo de parada representa o inteiro. Dessa forma, a evolução será orientada primeiro para diminuir o tempo total de paradas e depois o número de paradas. Matematicamente este objetivo é apresentado assim:

$$Duto = \text{truncate}\left(\sum_{m}^{m=mDuto-1}(HI_{m+1} - HF_m)\right) + \frac{CP}{1000}, \qquad (5\text{-}3)$$

onde *Duto* é o valor do objetivo; *CP* representa o número de paradas do duto; m representa uma atividade do duto; *mDuto* representa o número total de atividades do duto; *HF* e *HI* representam a hora de fim e hora de início de uma atividade m do duto. A escolha do valor 1.000 como parâmetro divisor do número de paradas do duto deve-se a uma das restrições do modelo (detalhadas no Capítulo 6), a qual define que cada movimentação do oleoduto tem duração mínimo de três horas. No limite, se houvesse 1.000 movimentações mínimas intercaladas por paradas (de qualquer duração), mais de 3.000 horas de horizonte poderiam ser programadas antes que este termo interferisse no valor das unidades do objetivo. Este horizonte representa mais de três meses de programação, o que é muito superior ao praticado.

Assim como a vazão de descarregamento do terminal, a vazão de bombeio do oleoduto também pode sofrer alguma variação em função de propriedades da mistura. Pelas mesmas razões ali expostas, esta vazão ainda é um parâmetro do modelo.

5.1.4 Trocas de Tanque

Este objetivo busca um *schedule* mais contínuo como um todo, onde, se não houver benefício em trocar tanques (seja por indisponibilidade de seguir no uso dos recursos originais ou para se adequar aos impactos provocados por mudanças em outras atividades), que a troca não aconteça. Ele é definido pelo somatório das vezes em que o equipamento de origem e/ou destino de uma atividade foi alterado. Matematicamente, este objetivo nada mais é do que um contador incrementado sempre que:

- um tanque estava na carga de uma UDA e na movimentação seguinte é utilizado outro tanque na mesma unidade;
- um tanque estava na carga de uma UDA e na movimentação seguinte ele está na carga de outra UDA;
- um tanque estava recebendo item de petróleo de um descarregamento de navio e na movimentação seguinte é utilizado outro tanque;
- um tanque estava bombeando para o oleoduto e na movimentação seguinte é utilizado outro tanque;

– um tanque estava recebendo do oleoduto e na movimentação seguinte é utilizado outro tanque.

Certamente algumas trocas de tanque são necessárias. Por exemplo, a carga da unidade deve ser substituída de um tanque para outro se o primeiro já não possuir volume para seguir com a movimentação. A minimização deste objetivo não tem a proposta de eliminá-las, mas sim de evitar situações como um tanque estar na carga de uma unidade, passar para outro tanque e voltar ao primeiro tanque sem que nenhuma outra atividade que modificasse esses recursos tenha acontecido.

5.1.5 Métricas de avaliação dos algoritmos propostos

Para avaliar o desempenho dos algoritmos propostos, este trabalho utiliza duas métricas: hipervolume da população, que mede a convergência do algoritmo e a diversidade dos indivíduos gerados; e percentual de corridas que produziram soluções de *schedule* aceitas (%CSA), que mede a eficácia do algoritmo em encontrar soluções válidas.

O hipervolume (HV) é uma métrica frequentemente utilizada para medir e comparar a qualidade de soluções geradas por algoritmos multiobjetivos. É definido como o hipervolume da figura formada pelas soluções não-dominadas da população e, por isso, representa o tamanho do espaço dominado [83]. Muito de seu mérito advém de ser o único indicador de desempenho unário que é compatível com a dominância de Pareto e sobre o qual já foi possível demonstrar que sua maximização implica na convergência do modelo [84].

O percentual de corridas que geraram soluções aceitas (%CSA) foi proposto em [5] e mede a relação entre a quantidade de corridas cujo melhor indivíduo é uma solução de *schedule* viável e o número total de corridas que foram executados para construir a estatística de desempenho. Cada corrida é uma réplica do mesmo modelo com a mesma parametrização, rodado para o mesmo cenário, onde a única diferença está na semente de inicialização dos números aleatórios. Por exemplo, se um cenário de programação foi executado 50 vezes, com diferentes inicializações, e em 30 corridas o melhor indivíduo é uma solução de *schedule* viável, o %CSA do modelo neste cenário é 60%. Nesta tese, que aborda algoritmos multiobjetivos, a métrica original é adaptada de forma a considerar corridas aceitas aquelas nas quais os indivíduos da frente de Pareto de menor *rank* representam *schedules* viáveis.

A definição de viabilidade de uma solução de programação está associada ao adequado atendimento dos objetivos críticos discutidos nas seções 3.2.1 e 3.2.2, cujas formulações são apresentadas nas seções 5.1.1 e 5.1.2. Idealmente,

as soluções propostas pelos algoritmos não devem apresentar valor diferente de zero para os objetivos de "Desvio da carga programada para as UDA" e "Atraso no descarregamento de navios" pois, no primeiro caso, significa que não houve nenhuma perda de processamento de petróleo e, no segundo caso, que não houve penalidade por sobrestadia dos navios e nem comprometimento da alocação do recurso para atender outras demandas. No entanto, dada a variabilidade do ambiente operacional (como, por exemplo, imprecisão de leitura de medidores de vazão ou nível de tanque, problemas de operação dentro ou fora da refinaria, mudanças na programação de entregas, etc.), soluções que tenham valores um pouco superiores a zero não devem ser desconsideradas pois é provável que representem programações viáveis no desdobrar diário das movimentações. Dessa forma, é adotada uma tolerância sobre o valor destes objetivos para que uma solução seja considerada viável (aceita). São aceitas programações com atraso total de descarregamento de navio inferior a 1 hora e até 2% de desvio em relação à carga programada das UDAs.

A tolerância de uma hora no atraso de descarregamento do navio objetiva não comprometer soluções apenas em função de precisão matemática, pois valores inferiores não significam um atraso real sobre o qual vá haver penalização. A tolerância de 2% sobre a carga programada das UDAs vem de um levantamento de dados histórico que compara o processamento programado no *schedule* e o processamento efetivamente realizado durante o horizonte da programação.

A Figura 5.1 apresenta, para sete diferentes cenários de programação (totalizando 80 dias) de uma refinaria real, o desvio percentual entre a vazão efetivamente processada (realizada) e vazão total programada para as UDAs no dia. Para os dados utilizados, em quase 70% dos casos o desvio percentual foi negativo, ou seja, o processamento real foi menor do que o programado. Este resultado indica que a premissa de não descartar soluções cujo valor do objetivo "Desvio de carga programada das UDAs" não é 0,0 é uma estratégia válida. Considerando todos estes pontos, o desvio médio entre o volume efetivamente processado e o programado foi -3,4%.

Ainda na Figura 5.1 é possível observar que o conjunto de valores negativos tem uma amplitude de variação maior que o desvio positivo. A Figura 5.2 apresenta a frequência do desvio diário do percentual de carga, onde se observa que podem ser encontrados desvios positivos ou negativos até a faixa de 6 a 8%, mas há 13 pontos apenas negativos, abaixo de -8%. Após a remoção destes pontos, realizada com o intuito de não absorver grandes variações na tolerância, uma nova média indicou um desvio de -1,9% entre o processamento realizado e o programado. Ainda na Figura 5.2 é possível observar que o

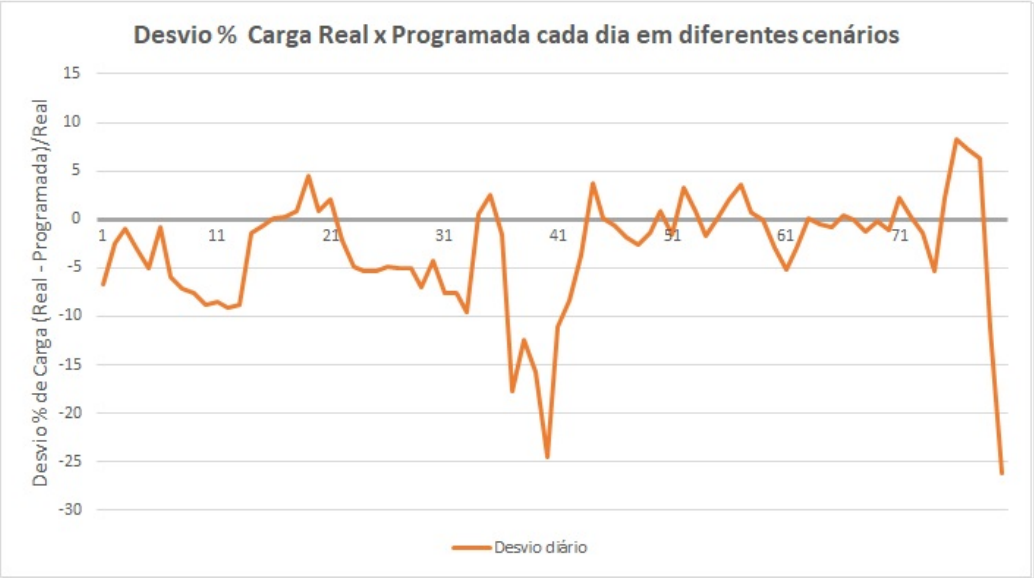

Figura 5.1: Desvio percentual diário entre a carga processada e a carga programada

intervalo [-2,0) é o mais frequente. Dessa forma, foi estabelecida a tolerância de -2% para o objetivo "Desvio da carga programada das UDAs". Como este objetivo mede o número total de horas parado das UDAs, um processamento até 2% inferior é representado por um tempo total de parada das UDAs de até 2% do horizonte de programação. Por exemplo, num cenário de 240 horas de duração, soluções que tenham tempo total de parada das UDAs por até 4,8 h são consideradas válidas (aceitas).

Como já discutido, os objetivos de "Parada na movimentação do oleoduto" e "Trocas de tanque" têm como função fazer com que a solução proposta seja uma programação mais suave operacionalmente. Não são críticos e não há um valor que defina a aceitação de uma solução em função deles.

Estabelecidos os objetivos do problema, as métricas de avaliação dos algoritmos e as condições que definem que uma solução pode ser aceita, as estratégias de solução para resolver o problema de programação de petróleo em refinaria são apresentadas nas próximas seções.

5.2 Propostas de solução

Em diversos cenários é provável que haja mais de uma solução de programação que atenda aos objetivos críticos com diferentes valores e também tenham diferentes valores de número de trocas de tanque e horas de movimentação do duto. Entre algumas dessas soluções é subjetivo definir se uma é

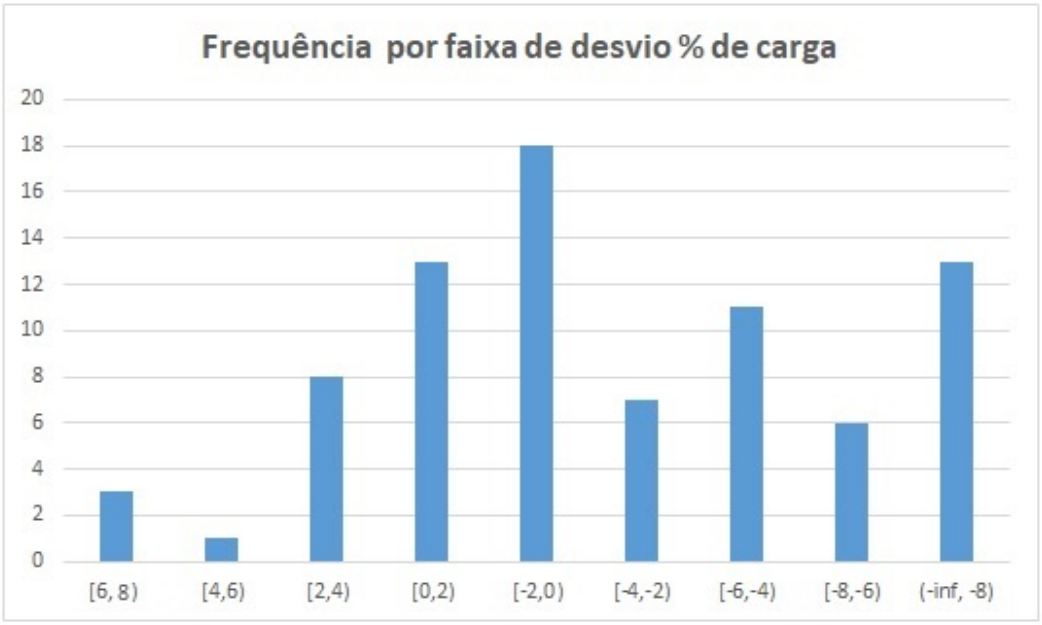

Figura 5.2: Frequência do desvio percentual de carga das UDAs

melhor do que a outra. Por exemplo, se uma solução alcança 0,0 h de parada de UDAs, 0,9 h de atraso de navio, 47 h de parada de oleoduto e 22 trocas de tanque, e outra solução alcança 2,0 h de parada de UDA, 0,0 de atraso de navio, 35 h de parada de oleoduto e 27 trocas de tanque, ambas as soluções representam *schedules* viáveis e determinar uma hierarquia clara entre elas não é um processo determinístico. Essas soluções devem ser oferecidas ao programador de produção para que ele as avalie e decida qual adotar.

Para lidar com os múltiplos objetivos do problema e a característica de diferentes níveis de importância entre eles, esta tese tem a proposta de desenvolver um modelo de otimização que atue respeitando a diferenciação e seja capaz de oferecer as melhores relações de compromisso entre os objetivos. Para tal, são estudadas abordagens para lidar com os quatro objetivos em um ou dois níveis hierárquicos de decisão. Foram consideradas três estratégias:

- E-Prz: Priorização de objetivos. Os objetivos são ordenados linearmente de forma explícita, seguindo uma definição feita *a priori*, e todos são tratados no mesmo nível de decisão.
- E-MO: *Manyobjective*. Essencialmente os objetivos possuem o mesmo nível de importância. Não há uma diferenciação explícita de criticidade e todos são tratados no mesmo nível de decisão. Os modelos desenvolvidos se baseiam em dois conceituados algoritmos multiobjetivos: MOEA/D e NSGA-III.

– E-Bi: *Bilevel*. O problema é tratado em dois níveis de decisão. São testadas duas abordagens: *bilevel* das restrições, onde o problema seguidor evolui a busca de soluções de um problema sem restrições para, em seguida, fornecer indivíduos para o problema líder que evolui considerando as restrições do problema de programação de produção; e *bilevel* dos objetivos onde o problema seguidor evolui apenas com base nos objetivos críticos e o problema líder considera todos.

5.2.1
Estratégia E-Prz: Priorização de Objetivos

Esta estratégia é o modelo QIGLGP apresentado no Capítulo 4, caso base deste trabalho e proposto em [5]. É usado para efeito de comparação com as estratégias propostas nesta tese. No QIGLGP o usuário define explicitamente a prioridade de cada objetivo antes do início da otimização e esta definição é utilizada para ordenar os indivíduos e orientar a evolução. Neste método é definido que o objetivo de parada de UDA é o mais importante, seguido do objetivo de descarregamento de navio, na sequência por movimentação do oleoduto e, por último, o número de trocas de tanque.

A Figura 5.3 apresenta o processo de comparação entre dois indivíduos I_i e I_j de acordo com o modelo QIGLGP. UDA_{Ii} e UDA_{Ij} representam, respectivamente, os valores do objetivo de desvio de carga das UDAs para os indivíduos I_i e I_j. NV_{Ii} e NV_{Ij} são os valores do objetivo de desvio no descarregamento dos navios destes mesmos indivíduos. OD_{Ii} e OD_{Ij} se referem ao objetivo de movimentação de oleoduto e, por último, TT_{Ii} e TT_{Ij} são o número de troca de tanques de cada indivíduo. A comparação define que se um dos indivíduos tem o objetivo de desvio de UDA menor do que o outro, este indivíduo é melhor, não importando o valor de nenhum dos demais objetivos. Somente em caso de empate é avaliado o objetivo seguinte e, nesse caso, será melhor o indivíduo que tiver menor valor de descarga de navio, não importando os demais. Se um novo empate ocorrer, o objetivo de movimentação de oleoduto é usado como critério de desempate e, somente se ainda houver empate, será avaliado o número de trocas de tanque e, assim, será melhor o indivíduo que fizer menos trocas. Se todos os objetivos forem iguais, não há diferenciação entre posicionar antes I_i ou I_j na ordenação pois eles são equivalentes.

5.2.2
Estratégia E-MO: Manyobjective

A essência do uso de estratégias multiobjetivo é a busca de um conjunto de soluções que representam as melhores relações de compromisso entre os

$$
\begin{cases}
UDA_{Ii} < UDAIj \rightarrow I_i \\
UDA_{Ii} > UDAIj \rightarrow I_j \\
UDA_{Ii} = UDAIj \rightarrow \begin{cases}
NV_{Ii} < NVIj \rightarrow I_i \\
NV_{Ii} > NVIj \rightarrow I_j \\
NV_{Ii} = NVIj \rightarrow \begin{cases}
OD_{Ii} < OD_{Ij} \rightarrow I_i \\
OD_{Ii} > ODIj \rightarrow I_j \\
OD_{Ii} = ODIj \rightarrow \begin{cases}
TT_{Ii} < TT_{Ij} \rightarrow I_i \\
TT_{Ii} > TT_{Ij} \rightarrow I_j \\
TT_{Ii} = TTIj \rightarrow I_i \; ou \; I_j
\end{cases}
\end{cases}
\end{cases}
\end{cases}
$$

Figura 5.3: Comparação dos indivíduos de acordo com o método QIGLGP

objetivos. Todos são considerados igualmente importantes. Esta ideia é oposta à apresentada na estratégia E-Prz, que se concentra em um objetivo de cada vez, numa hierarquia clara.

A representação com quatro objetivos torna a programação de petróleo parte de uma classe de problemas tratada na literatura como *manyobjective*, para a qual são reconhecidos os desafios de se manter a pressão seletiva, decorrente da falta de diferenciação dos indivíduos especialmente em algoritmos baseados em não-dominância. Quanto mais objetivos são considerados, maior a probabilidade de que indivíduos não sejam dominados e se agrupem em poucas frentes de Pareto.

Dentro das abordagens baseadas na não-dominância dos indivíduos para lidar com problemas *Manyobjectives* foi desenvolvido o algoritmo NSGA-III [44] que propôs o uso de pontos de referência espalhados ao longo do espaço dos objetivos para promover diversidade nas soluções da frente de Pareto. O NSGA-III é um algoritmo genético que utiliza operadores de cruzamento e mutação. Os estudos da estratégia E-MO são voltados para a proposição de um modelo de programação genética com inspiração quântica que utilize os princípios de não-dominância de soluções e pontos de referência apresentados pelo NSGA-III. Esta proposta é descrita na seção 5.2.2.1.

O algoritmo multiobjetivo MOEA/D foi a primeira proposta para evitar a perda de pressão seletiva em problemas com muitos objetivos através da decomposição do problema em subproblemas [38]. Também se trata de um algoritmo genético que utiliza operadores de cruzamento e mutação durante sua evolução. Este algoritmo também inspirou a proposição de um novo modelo de programação genética com inspiração quântica baseado em decomposição para tratar o problema de programação de petróleo em refinaria. Esta proposta é descrita na seção 5.2.2.2.

5.2.2.1 *Non-dominated sort* QIGLGP

O Capítulo 4 descreve os principais elementos do modelo QIGLGP, o detalhamento do algoritmo (Algoritmo 1) e seu processo evolutivo hierarquizado. Para resolver problemas multiobjetivo, a etapa 11 deste algoritmo é substituída pelo processo de ordenação por não-dominância e novas etapas são incluídas. O modelo proposto, denominado *Non-dominated Sort Quantum Inspired Grammar-based Linear Genetic Programming* (NSQIGLGP), é apresentado no Algoritmo 2.

A inicialização do NSQIGLGP inclui a obtenção dos pontos de referência que promovem a diversidade da população. De acordo com [44] estes pontos podem ser fornecidos, caso o usuário tenha conhecimento sobre a região do espaço de objetivos onde estão as soluções de interesse, ou calculados seguindo uma metodologia estruturada. Neste trabalho é adotado o método apresentado em [48], também utilizado na proposição do algoritmo NSGA-III. De acordo com este método, os pontos de referência são posicionados em um hiperplano normalizado igualmente inclinado para todos os eixos dos objetivos e há interseção entre todos os eixos. O número de pontos de referência é resultante da expressão de combinação que relaciona o número de objetivos e o número de divisões do hiperplano de acordo com:

$$H = \binom{M + p - 1}{p}, \tag{5-4}$$

onde p representa o número de divisões do hiperplano, M o número de objetivos e H o número de pontos de referência calculado. Quanto maior o número de divisões do hiperplano, maior será o número de pontos de referência e, consequentemente, também maior a complexidade computacional para execução do algoritmo.

Enquanto o critério de parada do NSQIGLGP (dado por um número fixo de gerações) não é atingido, em cada geração, primeiramente, os quatro objetivos são calculados para todos os indivíduos da população clássica auxiliar de acordo com as seções 5.1.1, 5.1.2, 5.1.3 e 5.1.4. Na sequência, as etapas 12 a 25 do Algoritmo 2 são executadas para ordenar e selecionar os indivíduos englobando a população clássica auxiliar (PC_{aux}) e a população clássica da geração anterior ($PC_{g\text{-}1}$).

Todos os indivíduos de PC_{aux} e $PC_{g\text{-}1}$ são comparados entre si de acordo com a Definição 1 apresentada no capítulo 2. Ou seja, a cada par de indivíduos, se nenhum dos quatro objetivos do indivíduo I_i possui valor maior do que o correspondente objetivo do indivíduo I_j e em pelo menos um objetivo o valor de I_i é menor do que I_j, então I_i domina I_j. Caso contrário, I_j é uma solução

não-dominada por I_i. O resultado da comparação entre todos os indivíduos permite a identificação dos que são não-dominados por qualquer membro da população. Estes indivíduos compõem a frente de Pareto *rank 0* (FP_0). Ainda na mesma etapa 12 do Algoritmo 2, uma nova rodada de ordenação é feita desconsiderando os indivíduos da FP_0. Os indivíduos agora não-dominados compõem a próxima frente de Pareto, FP_1. O processo de desconsiderar indivíduos já classificados e identificar os não-dominados do novo conjunto persiste até que todos os indivíduos estejam agrupados em frentes de Pareto. Como o total de indivíduos é o dobro do tamanho definido para a população, é necessário preservar apenas os melhores indivíduos, ou seja, os que estão nas frentes de Pareto de menor *rank*. Enquanto a soma de indivíduos das frentes de Pareto (em ordem crescente de *rank*) for menor do que o tamanho da população, todos os seus indivíduos persistirão (etapas 13 a 18 do Algoritmo 2). Na frente de Pareto cuja inclusão de todos os elementos representaria um excedente na população, se abrem as etapas 21 a 24 que atuam na preservação da diversidade através da seleção de indivíduos desta frente com base na densidade de soluções em torno dos pontos de referência.

Primeiramente é identificada a quantidade de indivíduos necessários para completar o tamanho de PC_g (variável K no Algoritmo 2) e que devem ser selecionados da frente de Pareto *rank l* (índice que representa a que excederia). Na sequência, cada um dos indivíduos já alocados em PC_g é associado ao ponto de referência mais próximo. Para que a medida da distância não seja influenciada pela ordem de grandeza dos objetivos, é necessário que os valores estejam normalizados. Após esta etapa, é possível identificar e ordenar os pontos de referência pela menor quantidade de indivíduos de PC_g associados. Por exemplo, pode ser identificado que o ponto de referência pr_k não possui nenhum indivíduo de PC_g associado a ele, e os pontos pr_j e pr_m possuem um indivíduo associado a cada.

Seguindo a ordem dada pelos pontos de referência com o menor número de indivíduos de PC_g associados, são buscados indivíduos em FP_l para que possam ser integrados a PC_g. Dessa forma, cada indivíduo de FP_l também é associado ao ponto de referência mais próximo. Mantendo o exemplo do parágrafo anterior, primeiramente é identificado se há indivíduos de FP_l que foram associados a pr_k. Se houver, o indivíduo de FP_l mais próximo a ele é incluído em PC_g e o número de indivíduos associados a pr_k é atualizado para um. Na nova condição, há três pontos de referência (pr_k, pr_j e pr_m) com um indivíduo de PC_g associado. Se ainda faltarem três ou mais indivíduos para completar o tamanho de PC_g, os indivíduos de FP_l associados a estes pontos de referência são candidatos à migração para PC_g. No entanto, se a necessidade for

menor, são sorteados quais dos pontos de referência cujos indivíduos associados de FP_l serão os candidatos. Nestes casos em que os pontos de referência já possuem indivíduos de PC_g associados, não é utilizado o critério de migrar o indivíduo de FP_l mais próximo do ponto de referência, mas sim um sorteio entre os indivíduos de FP_l associados a este ponto de referência.

No Algoritmo 2, f é o *rank* (índice) de uma frente de Pareto, FP_{f} se refere aos indivíduos da população pertencentes à frente de Pareto f, FP_{l} se refere aos indivíduos da população pertencentes à frente de Pareto l cuja inclusão total em PC_g excederia o tamanho da população e NI é o tamanho da população.

Diferenciando os grupos de objetivos no modelo NSQIGLGP. Em sua estrutura básica, apresentada no Algoritmo 2, o modelo NSQIGLGP não permite nenhum tratamento diferenciado entre os dois níveis de importância dos quatro objetivos porque essa ideia diverge do conceito original dos algoritmos multiobjetivos. Entretanto, este trabalho propõe que essa diferenciação exista no processo evolutivo através do critério de aceitação de soluções apresentado na seção 5.1.5.

Dessa forma, os limites mínimos para que a solução seja considerada um *schedule* viável são expressos como uma violação de restrição. O nível de violação impacta a comparação por dominância de Pareto entre os indivíduos e, consequentemente, a sua distribuição em frentes de Pareto.

A proposta original de mudança da definição de dominância para problemas com restrição foi realizada em [32] e seus preceitos foram mantidos nas versões *constrained* dos principais algoritmos multiobjetivos, como o *Constrained-NSGAIII* e o *Constrained-MOEA/D*. Em problemas com restrições, a comparação entre dois indivíduos I_i e I_j se dá da seguinte forma:

- se I_i não viola as restrições do problema e I_j viola, então I_i domina I_j independentemente do valor dos objetivos;
- se I_i e I_j violam restrições do problema, então será dominado o indivíduo que tiver maior violação, independentemente do valor dos objetivos;
- se I_i e I_j não violam as restrições do problema, então a comparação segue a Definição 1 original, que dita as regras de dominância de Pareto.

A medida da violação se dá através do estabelecimento de uma equação ou regra que permita o cálculo de um valor único, facilmente comparável entre os indivíduos. Nesta tese a métrica estabelecida é dada por uma relação entre os valores dos objetivos de UDA e Navio do indivíduo e os valores máximos possíveis que estes objetivos podem ter, ou seja:

1: geração g=0
2: Criar os *NI* indivíduos da população quântica inicial ($\mathrm{PQ_g}$)
3: Criar os *NI* indivíduos da população clássica inicial ($\mathrm{PC_g}$) a partir da observação de $\mathrm{PQ_g}$
4: Construir *schedule* de cada indivíduo da ($\mathrm{PC_g}$)
5: Medir aptidão do *schedule* de cada indivíduo da ($\mathrm{PC_g}$) de acordo com a avaliação da função multi-objetivo
6: $g \leftarrow g + 1$
7: Obter pontos de referência (fornecidos pelo usuário ou calculados)
8: **while** $g \leq nGP$ **do**
9: Criar população clássica auxiliar ($\mathrm{PC_{aux}}$) a partir da observação de $\mathrm{PQ_{g\text{-}1}}$
10: Construir *schedule* de cada indivíduo da ($\mathrm{PC_{aux}}$)
11: Calcular a aptidão do *schedule* de cada indivíduo da ($\mathrm{PC_{aux}}$) de acordo com a avaliação de cada termo da função multiobjetivo
12: Ordenação por não-dominância dos indivíduos de ($\mathrm{PC_{g\text{-}1}} \cup \mathrm{PC_{aux}}$).
13: $f \leftarrow 0$
14: $|PC_{\mathrm{g}}| \leftarrow 0$
15: **while** $|PC_{\mathrm{g}}| + |PC_{\mathrm{f}}| \leq NI$ **do**
16: $PC_{\mathrm{g}} \leftarrow (PC_{\mathrm{g}})U(FP_{\mathrm{f}})$
17: $f \leftarrow f + 1$
18: **end while**
19: $l \leftarrow f$ (índice da FP que já não é toda comportada em PC_g)
20: **if** $|PC_{\mathrm{g}}| < NI$ **then**
21: Quantidade de indivíduos para completar $\mathrm{PC_g}$, selecionando de $\mathrm{FP_l}$: $K = (NI - |PC_{\mathrm{g}}|)$
22: Normalizar objetivos.
23: Associar cada indivíduo de PC_{g} a um ponto de referência, pelo critério de menor distância entre ambos.
24: Selecionar K indivíduos de FP_{l} através de um mapeamento dos que estão mais próximos dos pontos de referência mais "desocupados" (possui menos indivíduos de $\mathrm{PC_g}$ associados a eles). Incorporar estes K indivíduos à $\mathrm{PC_g}$.
25: **end if**
26: Aplicar o operador quântico (OpQ) a cada um dos *NI* indivíduos quânticos de acordo com o indivíduo clássico correspondente
27: **if** Avaliações de todos os indivíduos de $\mathrm{PC_g}$ são iguais **then**
28: Voltar à etapa 2
29: **end if**
30: $g \leftarrow g + 1$
31: **end while**

Algoritmo 2: Algoritmo NSQIGLGP.

$$Violacao = \frac{objUDA}{nUDAs \times Horizonte} + \frac{objNavio}{\sum_{n=1}^{nNavio} VolNavio_n}, \tag{5-5}$$

onde: $Violacao$ é o tamanho da violação que o indivíduo faz aos objetivos críticos; $objUDA$ é o valor do objetivo de desvio de carga da UDA; $objNavio$ é o valor do objetivo de desvio no descarregamento de Navios; $nNavio$ é a quantidade de navios planejados para o horizonte de programação; $VolNavio_n$ é o volume total do Navio n; $nUDA$ é o número de UDAs da planta; e $Horizonte$ é o número de horas do horizonte de programação.

Nos casos em que a violação for a mesma, a metodologia original leva a uma classificação de não-dominância entre os indivíduos. Este trabalho avalia uma modificação nesta regra de forma que, quando a violação for a mesma, seja realizada a comparação entre os indivíduos aplicando a definição de dominância aos objetivos não-críticos, ou seja, "Parada de oleoduto" e "Trocas de tanque". A etapa 12 do Algoritmo 2 é executada de acordo com a seguinte lógica:

- se I_i não viola as restrições do problema e I_j viola, então I_i domina I_j independentemente do valor dos objetivos;
- se I_i e I_j violam restrições do problema, então será dominado o indivíduo que tiver maior violação, independentemente do valor dos objetivos;
- se I_i e I_j possuem o mesmo valor de violação das restrições do problema: se (OD_{Ii} e TT_{Ii}) não são piores do que (OD_{Ij} e TT_{Ij}) e, ainda, pelo menos um destes objetivos de I_i é melhor do que o correspondente de I_j, então I_i domina I_j.
- se I_i e I_j não violam as restrições do problema, então a comparação segue a definição 1 original, que dita as regras de dominância de Pareto.

O capítulo 6 apresenta os resultados das duas metodologias propostas para lidar com a violação dos objetivos críticos: a que não utiliza os objetivos não-críticos em caso de empate na violação e a que os utiliza. A proposta é, de acordo com esses resultados, definir qual dos dois métodos deve representar o modelo *Constrained Non-dominated Sort Quantum Inspired Grammar-based Linear Genetic Programming* (C-NSQIGLGP).

Participação de população externa na evolução do C-NSQIGLGP. NSQIGLGP e C-NSQIGLGP são algoritmos elitistas e, portanto, preservam seus melhores indivíduos clássicos gerados ao longo da evolução. No entanto, inspirada no algoritmo Elite-NSGAIII [49], é proposta a utilização de uma população *archive* externa que armazena um indivíduo associado a cada ponto de referência. Ao longo da evolução, se um novo indivíduo gerado e associado a um dado ponto de referência domina o indivíduo da população *archive*

associado ao mesmo ponto de referência, o novo indivíduo é copiado para a população *archive*, substituindo o anterior. Para garantir que a população não seja influenciada por soluções inviáveis, é definido que apenas soluções viáveis podem ser incorporadas à população *archive*.

Os indivíduos da população *archive* (PC_A) são candidatos a participar da evolução com o objetivo de aumentar a diversidade das soluções, pois podem fornecer indivíduos associados a pontos de referência que estão em menor densidade na população clássica da geração. Este trabalho avalia três métodos para a seleção de indivíduos clássicos preservados na população *archive* para sua utilização na evolução da C-NSQIGLGP. Em todos os casos, a influência se dá na etapa 26 do Algoritmo 2, onde ocorre a atualização dos indivíduos quânticos. Quando é definido que um indivíduo de PC_A deve ser utilizado, o operador quântico atualiza as probabilidades do indivíduo quântico em questão de acordo com as instruções do indivíduo clássico da PC_A e não do indivíduo clássico da população da geração.

As três metodologias que avaliam as condições para que um indivíduo de PC_A seja utilizado são:

- independente dos indivíduos de PC_g. A probabilidade de buscar um indivíduo em PC_A é proporcional ao tamanho desta população e obedece à equação 5-6;
- dependente de que em PC_g haja, para uma mesma frente de Pareto, mais de um indivíduo associado ao mesmo ponto de referência. Quando esta condição acontece, existe 50% de probabilidade de atualizar o indivíduo quântico a partir do indivíduo clássico da geração e 50% de probabilidade de que seja a partir de um indivíduo clássico de PC_A aleatoriamente selecionado. O valor de 50% como taxa de probabilidade de que o indivíduo quântico seja atualizado pelo indivíduo da PC_A é herdado do algoritmo NSGA-III;
- dependente de que em PC_g haja mais de um indivíduo associado ao mesmo ponto de referência na PC_g. Quando esta condição acontece, existe 50% de probabilidade de atualizar o indivíduo quântico a partir do indivíduo clássico da geração e 50% de probabilidade de que seja a partir de um indivíduo clássico de PC_A aleatoriamente selecionado.

A probabilidade pPC_A de que a atuação do operador quântico aconteça de acordo com um indivíduo da população externa PC_A, é dada por:

$$pPC_A = \quad 0{,}05 \quad + \quad \frac{(nPC_A - 1)}{nPC}, \tag{5-6}$$

onde nPC é o tamanho da população clássica da geração e nPC_A é o tamanho da população *archive* externa.

No Capítulo 6 são apresentados os resultados da avaliação do impacto dessas metodologias no desempenho do algoritmo C-NSQIGLGP e da sua contribuição para o aumento da diversidade de soluções. Ao modelo resultante, que utiliza uma população *archive* externa para armazenar indivíduos de diferentes pontos de referência que podem atuar no núcleo evolutivo, é dado o nome de *Archive Constrained Non-dominated Sort Quantum Inspired Grammar-based Linear Genetic Programming* (AC-NSQIGLGP).

A Figura 5.4 resume as propostas realizadas para o desenvolvimento de um modelo multiobjetivo de programação genética com inspiração quântica baseado em dominância de Pareto e capaz de representar diferentes grupos de objetivos.

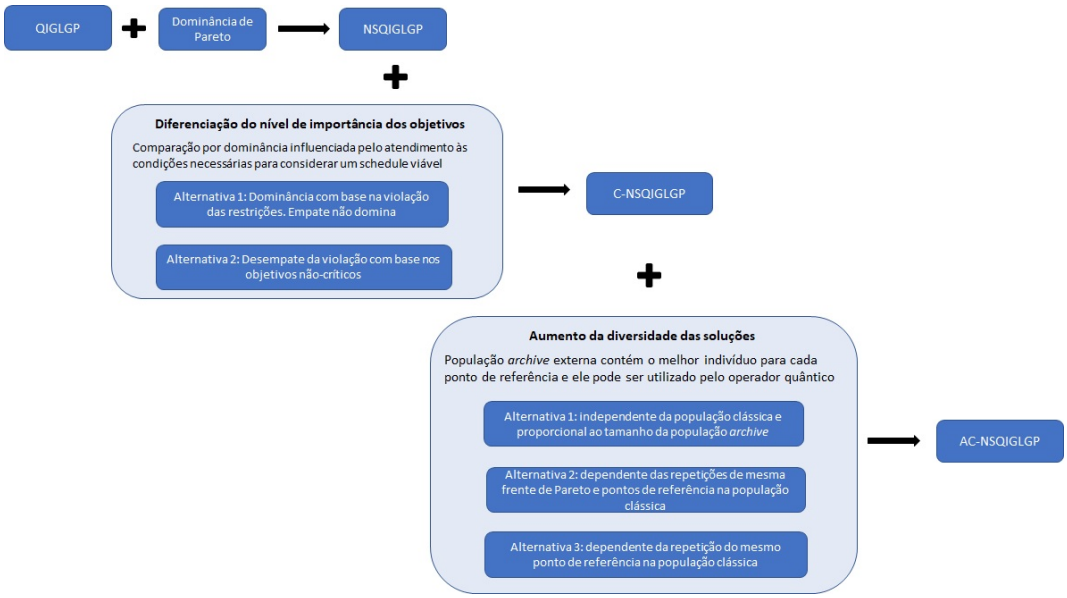

Figura 5.4: Resumo das proposições para um algoritmo multiobjetivo de programação genética com inspiração quântica baseado em dominância.

5.2.2.2 *Decomposition-based* QIGLGP

O princípio dos algoritmos baseados em decomposição é dividir o problema de otimização multiobjetivo em diversos subproblemas de otimização com apenas um objetivo, onde a função de aptidão de cada um destes subproblemas é a própria função de decomposição.

Dentre as funções de decomposição mais utilizadas estão as abordagens por Tchebycheff e *penalty-based boundary intersection* (PBI) [55]. Apesar do bom desempenho da decomposição por PBI em diversos problemas [47],

este trabalho propõe um algoritmo com decomposição por Tchebycheff que apresenta duas vantagens sobre a abordagem PBI: menor necessidade de parametrização (visto que PBI tem dois parâmetros a mais) e porque [38] sugere que PBI tem melhores resultados em problemas multiobjetivo contínuos, o que não é o caso da programação de petróleo, pois há variáveis de decisão discretas no modelo, além do objetivo de trocas de tanque, que é um valor inteiro positivo. A decomposição de Tchebycheff foi apresentada na equação 2-5.

O modelo proposto nesta seção integra a programação genética com inspiração quântica do QIGLGP aos princípios do algoritmo de decomposição mais utilizado, MOEA/D [38], como a própria função de decomposição em si, a criação e utilização de um vetor de pesos na função objetivo de cada subproblema, a definição da quantidade e quais são os vizinhos mais próximos de cada subproblema, além da existência de uma população *archive* externa elitista que armazena os indivíduos não dominados ao longo da evolução.

Este desenvolvimento modifica etapas fundamentais da QIGLGP como a ordenação dos indivíduos clássicos, que deixa de existir. Já a aplicação do operador quântico, que era realizada de acordo com a ordenação dos indivíduos clássicos, passa a ser realizada de acordo com o indivíduo clássico do subproblema. A preservação dos melhores indivíduos é feita pela população *archive* externa. Este modelo, denominado *Decomposition-based Quantum Inpired Grammar-based LinearGenetic Programming* (DQIGLGP) é apresentado no Algoritmo 3.

A inicialização do modelo DQIGLGP contempla etapas para a criação dos vetores peso (λ), definição do parâmetro T que representa o número de vizinhos de cada subproblema e o cálculo da distância Euclidiana entre os vetores peso de todos os subproblemas, de forma a identificar quem são os T vizinhos mais próximos. Os T vizinhos mais próximos do subproblema i compõem o vetor de vizinhos (B_i) desse subproblema. Para a determinação do vetor de pesos foi adotada a metodologia [48] utilizada na proposição do MOEA/D e também nos modelos com abordagem NSQIGLGP para a geração dos pontos de referência.

Os modelos propostos baseados na decomposição da QIGLGP não precisam de uma população clássica auxiliar. As populações clássica e quântica de cada subproblema são compostas de um único indivíduo. Em cada geração, a aptidão do indivíduo clássico do subproblema i (I_i) é avaliada comparativamente com a aptidão de indivíduos clássicos vizinhos a i para resolver o subproblema i. A medida da aptidão é dada pela Equação 2-5 que considera os valores dos objetivos do indivíduo e também seu vetor de pesos. Entre I_i e

seus vizinhos, o que tiver maior aptidão para resolver o subproblema i será a nova população deste subproblema. Quando este ciclo se completa para todos os subproblemas, muitos indivíduos clássicos migraram dos subproblemas onde estavam originalmente vinculados para novos.

O reposicionamento dos indivíduos clássicos orienta a evolução. O indivíduo quântico do subproblema i não migrou, continua vinculado a este subproblema. No final de cada geração, a distribuição de probabilidades deste indivíduo quântico é atualizada de acordo com as instruções do indivíduo clássico que representa a melhor solução para aquele subproblema.

Em relação ao parâmetro T, uma avaliação realizada em [38] concluiu que o MOEA/D não é muito sensível ao tamanho do conjunto de vizinhos, especialmente em configurações em que a população não é grande. A única preocupação descrita é de que T não seja muito grande de forma a promover uma convergência prematura da evolução, pois o mesmo indivíduo pode migrar para muitos subproblemas vizinhos. Uma das vantagens da programação genética com inspiração quântica é seu bom desempenho em populações pequenas, o que faz com que todos os modelos propostos nesta tese não tenham populações grandes. Dessa forma, é arbitrado um tamanho de T da ordem de 10% da população de forma a evitar que um mesmo indivíduo se replique em muitos subproblemas.

À exceção do objetivo de desvio no descarregamento de navios, que pode ter valor de centenas de milhares, principalmente no início da evolução, os demais objetivos do problema variam da ordem de grandeza de unidades até centenas. Existe portanto uma diferença de ordem de grandeza, mesmo que não seja das mais expressivas. Dessa forma, é adotada a normalização dos objetivos para os modelos de decomposição.

A etapa 26 do Algoritmo 3 trata da preservação dos melhores indivíduos através de uma população *archive* externa PC_A, que não interfere no processo evolutivo. Ao final de cada geração, os indivíduos clássicos de todos os subproblemas são ordenados em frentes de Pareto. Esta etapa não altera o subproblema do qual são solução, apenas permite identificar quem são os indivíduos não-dominados considerando os objetivos do problema. Os indivíduos não-dominados da PC_g são submetidos a uma comparação por dominância de Pareto com os indivíduos da PC_A. Deste conjunto, os indivíduos não-dominados de PC_g que também forem não-dominados pelos indivíduos de PC_A são incluídos nesta população e os indivíduos de PC_A que forem dominados por indivíduos de PC_g são removidos de PC_A. Quando o critério de parada do algoritmo é atingido, os indivíduos que compõem PC_A representam a solução do problema.

Assim como nas proposições feitas com base no modelo NSQIGLGP, para o DQIGLGP também foi analisado o impacto de diferenciar grupos de objetivos e a proposta de que PC_A influencie no processo evolutivo.

Diferenciando os grupos de objetivos no modelo DQIGLGP: No DQIGLGP há duas etapas de comparação entre indivíduos que podem ser impactadas pela proposta feita na seção 5.2.2.1 de diferenciar grupos de importância dos objetivos com base nos critérios de considerar um *schedule* viável.

A primeira está no próprio núcleo evolutivo, representada pela etapa 20 do algoritmo 3, quando a função de decomposição de Tchebycheff do indivíduo clássico do subproblema i é comparada aos valores obtidos por seus vizinhos. Em [47] é proposto o algoritmo C-MOEA/D que usa a mesma abordagem dada ao C-NSGAIII, ou seja, uma equação de violação é definida e terá maior aptidão o indivíduo com menor violação (se houver) e, se nenhum dos indivíduos violar restrições, então a aptidão fica somente a cargo da função de decomposição. O indivíduo com menor valor da equação 2-5 será mais apto.

A segunda (etapa 26 do algoritmo 3) acontece na atualização da PC_A, quando indivíduos não-dominados da PC_g são comparados com a PC_A para que persista em PC_A somente os não-dominados de todo o conjunto.

De forma análoga ao estudo realizado para o C-NSQIGLGP, este trabalho propõe a modificação no modelo DQIGLGP de forma a considerar o atendimento aos objetivos críticos como uma restrição. A equação de violação 5-5 é a mesma utilizada para o modelo por dominância. É avaliado se no modelo baseado em decomposição também há benefício no desempate quando a violação tiver o mesmo valor.

Forma parte deste estudo considerar que a influência da violação aconteça apenas na etapa de atualização da PC_A ou também durante o processo evolutivo. De acordo com os resultados obtidos, e apresentados no Capítulo 6, é definida a metodologia mantida neste modelo para diferenciar os objetivos críticos e não-críticos do problema. O modelo resultante é denominado *Constranined Decomposition-based Quantum Inspired Grammar-based Linear Genetic Programmin* (C-DQIGLGP).

Participação de população externa na evolução do C-DQIGLGP. Nos modelos DQIGLGP e C-DQIGLGP a população externa PC_A não influencia na evolução, tendo o papel apenas de preservar os indivíduos não-dominados até então. Este trabalho avalia o impacto de utilizar indivíduos de PC_A na etapa 27 do Algoritmo 3. Duas metodologias são propostas para definir a probabilidade de que um indivíduo de PC_A seja utilizado para atualizar a

1: geração g=0
2: Definir o número de subproblemas, o que também representa definir o tamanho da população
3: Calcular o vetor de pesos dos subproblemas λ
4: Definir o número de vizinhos de cada subproblema (T), calcular quem são os vizinhos mais próximos de cada subproblema (B_i)
5: Criar vetor de referência dos objetivos z que armazena o menor valor encontrado em cada objetivo
6: Criar os NI indivíduos da população quântica inicial($\mathrm{PQ_g}$)
7: Criar população *archive* externa PC_A vazia.
8: $g \leftarrow g + 1$
9: **while** $g <= nGP$ **do**
10: Criar população clássica ($\mathrm{PC_g}$) a partir da observação de $\mathrm{PQ_{g\text{-}1}}$
11: Construir *schedule* de cada indivíduo da ($\mathrm{PC_g}$)
12: Calcular os valores dos objetivos de cada indivíduo da ($\mathrm{PC_g}$)
13: $i \leftarrow 1$
14: **while** $i <= NI$ **do**
15: Criar indivíduo clássico I_i a partir da observação do indivíduo quântico IQ_i
16: Construir *schedule* e calcular os valores dos objetivos de I_i.
17: Atualizar vetor de referência dos objetivos z
18: Calcular aptidão f do indivíduo I_i e de seus vizinhos (indicados no vetor B_i) utilizando a decomposição de Tchebycheff e os vetores pesos dos objetivos de I_i e de seus vizinhos (λ_i e λ_{Bi}).
19: **for** Cada subproblema em B_i **do**
20: **if** $f_i < f_{\mathrm{Bi}}$ **then**
21: $I_{\mathrm{Bi}} \leftarrow I_i$
22: **end if**
23: **end for**
24: $i \leftarrow i + 1$
25: **end while**
26: Comparar por dominância todos os indivíduos de PC_g e PC_A e preservar em PC_A os indivíduos não dominados até então.
27: Aplicar o operador quântico (OpQ) a cada um dos NI indivíduos quânticos de acordo com o indivíduo clássico correspondente de PC_g
28: **if** Avaliação de todos os indivíduos de $\mathrm{PC_g}$ são iguais **then**
29: Voltar à etapa 2
30: **end if**
31: $g \leftarrow g + 1$
32: **end while**

Algoritmo 3: Algoritmo DQIGLGP

distribuição de probabilidades do indivíduo quântico de um subproblema no lugar do indivíduo clássico deste subproblema. Estas metodologias são:

– probabilidade fixa. Quando o operador quântico for atuar sobre o indivíduo quântico, é sorteado se as instruções que nortearão a atualização serão lidas do indivíduo clássico do subproblema ou de um indivíduo aleatório de PC_A. A probabilidade de que seja de PC_A possui um valor fixo, que é um parâmetro do modelo;

– dependente do número de indivíduos em PC_A. A probabilidade de que o indivíduo quântico do subproblema i seja atualizado de acordo com as instruções de um indivíduo aleatório de PC_A aumenta proporcionalmente conforme aumenta o número de indivíduos em PC_A, de acordo com a equação 5-6.

De acordo com os resultados obtidos para este estudo e apresentados no Capítulo 6, é avaliado o impacto das metodologias no desempenho do algoritmo C-DSQIGLGP. Ao modelo que utiliza a população *archive* externa para para influenciar na evolução é dado o nome de *Archive Constrained Decomposition Quantum Inspired Grammar-based Linear Genetic Programming* (AC-DQIGLGP).

A Figura 5.5 resume as proposições realizadas para o desenvolvimento de um modelo multiobjetivo de programação genética com inspiração quântica baseado em decomposição e capaz de representar diferentes grupos de objetivos.

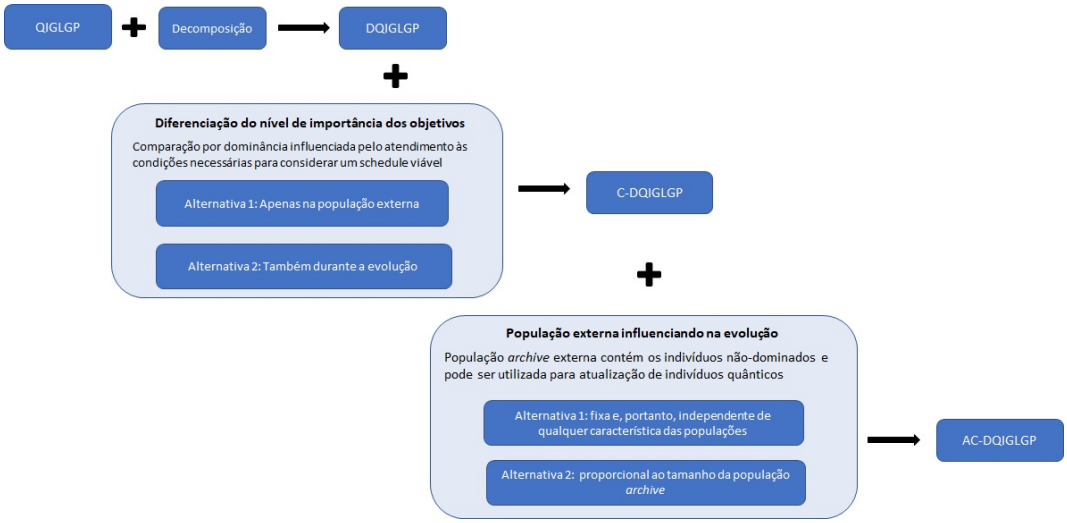

Figura 5.5: Resumo das proposições para um algoritmo multiobjetivo de programação genética com inspiração quântica baseado em decomposição

5.2.3
Estratégia E-Bi: Bilevel

Sob esta estratégia são feitos os primeiros ensaios de gerar soluções em dois níveis de otimização. Preliminarmente, foi avaliada uma divisão dos níveis baseada em se considerar ou não as restrições do problema. Foram gerados dois modelos: *Bilevel* em restrições de QIGLGP (BiC-QIGLGP) que utiliza a ordenação por priorização dos indivíduos, como no modelo original QIGLGP, e *Bilevel* em restrições de NSQIGLGP (BiC-CNSQIGLGP) que ordena os indivíduos em frentes de Pareto de acordo com os princípios de não-dominância, como no modelo C-NSQIGLGP. Estes modelos são apresentados na Seção 5.2.3.1.

Também é proposto um modelo em que os níveis não tratam exatamente os mesmos objetivos. O Seguidor visa a gerar soluções que atendam aos objetivos críticos e deve passá-las ao Líder, o qual visa a atender aos objetivos de homogeneidade mantendo o progresso alcançado pelo Seguidor. Este modelo (BiO-CDQIGLGP), apresentado na seção 5.2.3.2, utiliza o conceito de decomposição para lidar com os diferentes objetivos em cada nível.

5.2.3.1
Bilevel das restrições: Modelos BiC-QIGLGP e BiC-CNSQIGLGP

Nestes modelos, a divisão ainda não ocorre por diferenciação dos objetivos, mas sim por considerar ou não as restrições características e modeladas para o problema, como duração mínima de movimentações ou qualidade de correntes. Restrições como lastro e capacidade de tanque sempre são respeitadas, pois, caso contrário, não seria caracterizado como um problema de *schedule*. É importante ressaltar que a tolerância sobre os objetivos de UDA e descarregamento de navios representam métricas de aceitação das soluções usadas para avaliação dos algoritmos (Seção 5.1.5) e não restrições do problema. Por isso, não há tratamento diferenciado entre eles nos níveis Líder e Seguidor.

O Algoritmo 4 representa os modelos BiC-QIGLGP e BiC-CNSQIGLGP propostos. Nele, o nível em que as restrições estão ativas é tratado como Líder e o nível em que as restrições não estão ativas é tratado como Seguidor. Duas metodologias são avaliadas. Na primeira, a base do núcleo evolutivo, tanto do Líder como do Seguidor, é dada pelo princípio de priorização dos objetivos da QIGLGP original. Já a segunda metodologia utiliza a abordagem por não-dominância proposta pelo C-NSQIGLGP atuando nos dois níveis.

Líder e Seguidor têm suas populações clássica e quântica independentes (PCL, PCS, PQL e PQS) e todas são criadas na inicialização do algoritmo.

Entre as etapas 8 e 16 do Algoritmo 4, um ciclo *otimização Seguidor* → *otimização Líder* completo se repete por um número fixo de repetições, *nLoop*.

A primeira etapa do ciclo é a execução do núcleo evolutivo (QIGLGP ou C-NSQIGLGP) do nível Seguidor por gS gerações. Ao final, é avaliado o perfil das soluções de PCS. Se menos de 50% dos indivíduos de PCS representam soluções de *schedule* viáveis, então os indivíduos quânticos correspondentes desses indivíduos clássicos são migrados para a população do Líder. Por outro lado, se mais de 50% representarem soluções viáveis, são migrados para a população do Líder a quantidade de indivíduos que corresponda aos 50%, definidos aleatoriamente. Com a PQL semeada por indivíduos da PQS, o nível Líder evolui (QIGLGP ou C-NSQIGLGP), considerando as restrições, por gL gerações. Ao final, parte da PQL migra para PQS seguindo os mesmos critérios de quando a migração teve sentido contrário, ou seja, até 50% de PQL migra para PQS dependendo do percentual de indivíduos que representam *schedules* viáveis. O estabelecimento do valor de 50% para o parâmetro que define o máximo de indivíduos migrados se inspira no valor utilizado em outros métodos, como o MOEA/D onde, na criação de um novo indivíduo (filho) há 50% de probabilidade de que os pais venham da população da geração e 50% de que venha da população externa elitista. A metodologia para substituição também define que a população destino da migração tem os seus piores indivíduos substituídos pelos que estão sendo recebidos da população de origem. Este processo completa um ciclo e quando *nLoop* é atingido, as soluções finais são representadas pelos indivíduos da população clássica do nível Líder.

5.2.3.2 Bilevel dos objetivos: Modelo BiO-CDQIGLGP

Neste estudo Líder e Seguidor são tratados pelo núcleo evolutivo do modelo C-DQIGLGP. O Seguidor evolui considerando apenas os dois objetivos críticos do problema e o Líder evolui com os quatro objetivos originais.

O algoritmo começa com a resolução do nível Seguidor. O vetor de pesos dos subproblemas resultantes da decomposição do Seguidor e, consequentemente, o tamanho da população são definidos pela Equação 5-4. Como apenas dois objetivos são tratados, o tamanho deste vetor e da população do Seguidor resulta cerca de quatro vezes menor do que do nível Líder. Estas populações clássica e quântica (PCS e PQS) são iniciadas pelo mesmo método adotado no C-DQIGLGP.

O tamanho do vetor de vizinhos de cada subproblema havia sido definido como 10% da população na família de modelos DQIGLGP. No entanto, como a

1: iteracao do *loop* LL=0
2: Inicializacao das populações:
3: Criar os NI indivíduos da população quântica inicial do Líder(PQL_0) e os indivíduos da população quântica inicial do Seguidor(PQS_0)
4: Criar os NI indivíduos da população clássica inicial do Líder (PCL_0) a partir da observação de PQL_0 e os indivíduos da população clássica inicial do Seguidor (PCS_0) a partir da observação de PQS_0
5: Construir *schedule* de cada indivíduo da PCL_0 e de cada indivíduo da PCS_0
6: Medir aptidão do *schedule* de cada indivíduo da PCL_0 e da PCS_0 de acordo com a avaliação da função multiobjetivo
7: $LL \leftarrow LL + 1$
8: **while** $LL <= nLoop$ **do**
9: Evoluir (QIGLGP ou C-NSQIGLGP) nível Seguidor (sem restrições) por gS gerações.
10: Substituir membros da população quântica do Líder por membros da população quântica do Seguidor: se o número de indivíduos clássicos aceitos no Seguidor é inferior a 50% da população, copia todos os quânticos correspondentes para o Líder. Se for superior, mantém 50% da população quântica do Líder e copia os quânticos dos 50% melhores indivíduos clássicos do Seguidor para o Líder.
11: Evoluir (QIGLGP ou C-NSQIGLGP) nível Líder (com restrições) por gL gerações
12: **if** $LL < nLoop$ **then**
13: Substituir membros da população quântica do Seguidor por membros da população quântica do Líder: se o número de indivíduos clássicos aceitos no Líder é inferior a 50% da população, copia todos os quânticos correspondentes para o Seguidor. Se for superior, mantém 50% da população quântica do Seguidor e copia os quânticos dos 50% melhores indivíduos clássicos do Líder para o Seguidor.
14: **end if**
15: $LL \leftarrow LL + 1$
16: **end while**
17: Apresentar o(s) melhor(es) indivíduo(s) da população clássica Líder

Algoritmo 4: Abordagem Líder-Seguidor para ativação das restrições com ordenação baseada na QIGLGP ou C-NSQIGLGP

a população do Seguidor é muito pequena, resultaria em um vetor de vizinhos muito pequeno, diminuindo a probabilidade de migração. Dessa forma, para o Seguidor é arbitrado que o vetor de vizinhos mede metade do tamanho do vetor de vizinhos do Líder.

O número de gerações definido como critério de parada do algoritmo completo é dividido entre os níveis Líder e Seguidor proporcionalmente ao tamanho de suas populações. Por exemplo, se o número total de gerações for 10.000, o tamanho da população do Líder for 60 indivíduos e do Seguidor 15 indivíduos, então o Líder irá evoluir por 7.500 gerações $(10.000 \times (1 - (15/60)))$.

No entanto, o Seguidor não irá evoluir por apenas 2.500 gerações, pois, de forma a manter o número total de avaliações igual aos modelos das demais estratégias, a menor população do Seguidor é compensada por um aumento proporcional no número de gerações deste nível. Portanto, mantendo o exemplo anterior, sem a compensação o Seguidor deveria evoluir por 2.500 gerações, mas como sua população é quatro vezes menor, ele evoluirá por 10.000 gerações.

Ao final da evolução do Seguidor, todos os indivíduos de sua população *archive* externa são migrados para a população clássica do Líder. A atribuição de que indivíduo clássico será associado a que subproblema do nível Líder é aleatória.

O Líder evolui por gL gerações segundo o modelo C-DQIGLGP. Ao final da evolução, a população *archive* externa deste nível contém os indivíduos não-dominados neste nível e que representam a solução final do modelo.

O Algoritmo 5 representa o modelo proposto *Bilevel Objective Constrained Decomposition-based Quantum Inspired Grammar-based Linear Genetic Programmin* (BiO-CDQIGLGP).

1: Definir o número de subproblemas resultantes da decomposição nos níveis Seguidor e Líder, o que também representa definir o tamanho das populações.
2: Definir o número de vizinhos dos subproblemas da decomposição nos níveis Seguidor e Líder (T_S e T_L)
3: Calcular os vetores de pesos dos subproblemas λ_S e λ_L
4: Calcular o número de gerações que os níveis Líder e Seguidor evoluirão (gL e gS, respectivamente)
5: Executar a evolução do algoritmo C-DQIGLGP para o Seguidor por gS gerações
6: Migrar aleatoriamente os indivíduos da população *archive* do Seguidor para a população clássica do Líder
7: Executar a evolução do algoritmo C-DQIGLGP para o Líder por gL gerações
8: Apresenta(r) o(s) melhor(es) indivíduo(s) da população *archive* Líder

Algoritmo 5: Algoritmo BiO-CDQIGLGP: abordagem Líder-Seguidor da C-DQIGLGP considerando 2 e 4 objetivos

As proposições de modelo derivadas das estratégias E-MO e E-Bi foram avaliadas em cinco cenários de programação de petróleo. A diversidade das soluções entre as propostas foi comparada utilizando a métrica de hipervolume. A capacidade do modelo em gerar soluções válidas foi avaliada com base no percentual de corridas que geraram *schedules* viáveis. Esta última métrica também foi utilizada para comparar estas propostas com a estratégia E-Prz, que é o caso base representado pelo modelo base QIGLGP. Os resultado são apresentados no Capítulo 6.

6
Resultados

Neste capítulo são descritos a área de programação de petróleo de uma refinaria, os cenários de programação utilizados como estudo de caso, a DSL construída para o problema e os resultados obtidos nos diferentes modelos propostos pelas estratégias consideradas. Cada avaliação foi feita com base na execução de 50 corridas (réplicas) de cada cenário para cada proposta. Por não ser um método determinístico, as réplicas são necessárias para garantir significância estatística das observações que sejam feitas.

6.1
Estudo de caso

6.1.1
Informações sobre a refinaria

O escopo de decisão da programação de petróleo estudada nesta tese engloba: as decisões sobre o descarregamento de navios através de um duto de pequena capacidade (homogêneo) em tanques de um terminal; as movimentações deste petróleo para os tanques da refinaria através de oleoduto; e as movimentações que fazem com que as duas UDAs da refinaria sejam alimentadas pelos tanques da refinaria. As decisões de movimentação devem respeitar as restrições operacionais e de qualidade das correntes apresentadas nesta seção.

Os cenários usados são da mesma refinaria, cuja representação esquemática foi apresentada na Figura 3.1 e por isso possuem essencialmente os mesmos equipamentos para utilizar nas decisões de movimentação. A exceção decorre do fato de que esses cenários representam datas diferentes e, entre alguns, há anos de diferença. Isso fez com que, apesar da refinaria ter a mesma topologia, dois cenários (chamados 8119 e 8205) tivessem um tanque a mais no terminal e dois tanques a mais na refinaria disponíveis para utilização. A Tabela 6.1 apresenta lastro e capacidade dos tanques e dutos que compõem a topologia da planta e, no caso dos tanques, também informa em quais cenários eles estavam disponíveis.

A base de dados de petróleos utilizada e suas propriedades é exibida na Tabela 6.2. Ela representa o conjunto de todos os petróleos processados por

Tabela 6.1: Configuração dos equipamentos: lastro e capacidade.

Área	Equipamento	Lastro (m^3)	Capacidade (m^3)	Cenários disponíveis
Linha submarina	LS1	0	11.666	
Terminal	TqT-01	6.093	52.527	todos
Terminal	TqT-02	6.938	61.338	todos
Terminal	TqT-03	7.393	62.763	todos
Terminal	TqT-04	7.820	63.228	todos
Terminal	TqT-05	4.115	37.052	8119, 8205
Oleoduto	DUTO	0	19.000	
Refinaria	TqR-01	8.339	62.822	todos
Refinaria	TqR-02	5.193	37.268	todos
Refinaria	TqR-03	8.369	62.796	todos
Refinaria	TqR-04	8.827	63.255	todos
Refinaria	TqR-05	8.625	53.980	8119, 8205
Refinaria	TqR-06	8.666	63.093	8119, 8205

todos os cenários utilizados como estudo de caso, mas é um subconjunto do elenco de petróleos que podem ser processados na refinaria. Todos os modelos propostos nesta tese são flexíveis quanto à inclusão de novos petróleos. Dentre os explicitados na Tabela 6.2, estão petróleos com valores de propriedades que podem tornar a resolução do cenário de programação mais complexo para fazer as misturas que garantam o enquadramento das restrições de propriedade. A Tabela 6.3 exemplifica a estrutura de dados e valores de rendimentos e propriedades dos cortes de alguns petróleos em uma campanha de uma unidade de destilação atmosférica.

Além das informações de topologia da planta e bases de dados de produtos, a refinaria também opera sujeita a um conjunto de restrições operacionais e de propriedades que devem ser consideradas em todos os cenários de programação. As condições do cenário (composição dos tanques, volume total, programação de navios, etc) podem tornar mais difícil ou não o atendimento às restrições, mas elas sempre devem ser observadas. Para esta refinaria devem ser consideradas:

- em qualquer instante, o Índice de Acidez Total (IAT) da mistura processada nas unidades UDA-01 e UDA-02 deve ser inferior a 1,3 mgKOH/g óleo;
- em qualquer instante, o IAT das correntes de DL e DP produzidas na UDA-01 deve ser inferior a 1,9 mgKOH/g de óleo;
- em qualquer instante, o IAT da corrente de RAT produzida na UDA-01 deve ser inferior a 1,7 mgKOH/g de óleo;

Tabela 6.2: Base de petróleos utilizada.

Óleo	API	Densidade	IAT (mgKOH/g óleo)	Teor de Enxofre (%m/m)
PT01	47,6	0,7901	0,05	0,05
PT02	45,2	0,8008	0,05	0,06
PT03	26,7	0,8944	0,19	0,50
PT04	19,9	0,9346	1,19	0,74
PT05	22,8	0,9170	0,74	0,72
PT06	23,1	0,9153	0,59	0,67
PT07	20,6	0,9303	1,24	0,64
PT08	18,6	0,9428	1,35	0,76
PT09	27,0	0,8927	0,41	0,59
PT10	28,3	0,8855	0,09	0,58
PT11	28,0	0,8871	0,14	0,54
PT12	22,1	0,9212	1,82	0,42
PT13	20,0	0,9340	1,92	0,59
PT18	30,4	0,8740	0,08	0,31
PT21	32,0	0,8626	1,71	0,98
PT22	16,8	0,9541	3,35	0,56
PT23	13,4	0,9770	1,13	0,89
PT24	18,0	0,9465	2,69	0,69
PT29	22,2	0,9177	1,25	0,69

Tabela 6.3: Exemplos de rendimentos e propriedades de petróleos dos cenários.

Rendimento (%)				
Corte	PT01	PT02	...	PT12
GLP	3,40	2,43	...	0,68
NL	22,82	26,46	...	3,63
NP	23,23	23,17	...	8,42
Q	4,89	4,96	...	2,53
DL	27,96	24,36	...	19,14
DP	1,62	1,38	...	1,83
RAT	16,08	17,24	...	63,77

Densidade						
...	PT04	...	PT10	...		
...	0,4756	...	0,5451	...		
...	0,7402	...	0,7139	...		
...	0,8137	...	0,7929	...		
...	0,8519	...	0,8355	...		
...	0,8919	...	0,8689	...		
...	0,9333	...	0,9120	...		
...	0,9951	...	0,9745	...		

IAT				
Corte	...	PT08	PT09	...
NL	...	0	0	...
NP	...	0	0	...
Q	...	0	0	...
DL	...	1,32	0,56	...
DP	...	1,47	0,63	...
RAT	...	1,01	0,34	...

Enxofre				
...	PT05	...	PT11	...
...	0,0029	...	0,0012	...
...	0,1072	...	0,0715	...
...	0,2021	...	0,168	...
...	0,4952	...	0,4189	...
...	0,7373	...	0,593	...
...	0,9011	...	0,7472	...

- em qualquer instante, o IAT das correntes de DL e DP produzidas na UDA-02 deve ser inferior a 1,4 mgKOH/g óleo;
- em qualquer instante, o IAT da corrente de RAT produzida na UDA-02 deve ser inferior a 1,5 mgKOH/g óleo;
- após recebimento de item do oleoduto, qualquer tanque da refinaria precisa esperar, no mínimo, 24 horas antes de estar disponível para processamento nas unidades de destilação. Este é o "tempo de preparação", utilizado para que haja decantação e drenagem de água contaminada (salmoura) que vem misturada ao petróleo. A remoção desta salmoura se faz necessária para que não comprometa os equipamentos da refinaria, principalmente a dessalgadora e UDA;
- nenhum tanque do terminal ou da refinaria pode receber volume que o leve para acima de sua capacidade máxima de estocagem;
- nenhum tanque do terminal ou da refinaria pode transferir volume que o leve para baixo de sua capacidade mínima de estocagem (lastro);
- nenhuma movimentação entre terminal e refinaria, realizada via oleoduto, pode ser inferior a três horas. Esta heurística é adotada na refinaria para evitar que os itens de bombeio dentro do duto sejam muito pequenos, resultando em muitas interfaces.

Considerando a base de dados utilizada, pode-se comentar que, em termos de características de qualidade, há um conjunto representativo de petróleos que demandam maior atenção, pois seus valores de IAT estão acima dos limites especificados para a carga das UDAs. É o caso dos petróleos PT08, PT12, PT13, PT21, PT22 e PT24. Destes, somente PT21 e PT24 não violam também o limite para os cortes de DL, DP e RAT oriundos deles. Os petróleos PT04, PT07 e PT29 têm valores de IAT próximos ao permitido na carga, mas geram um corte DP que está acima do limite permitido na UDA-02. Os demais petróleos da base de dados possuem propriedades abaixo do limite máximo e, portanto, ajudam na especificação da mistura pois diluem os que estão acima. Conseguir o atendimento às restrições dependerá das movimentações realizadas que promoverão a mistura entre os petróleos. Estas informações têm o objetivo de reforçar a importância da composição de misturas para processamento nas UDAs pois, na grande maioria dos casos, petróleos sozinhos não atendem aos limites de restrições de processamento ou características desejadas nos cortes produzidos.

Características de topologia da refinaria como alinhamentos ou limitações de tancagem também influenciam na complexidade de construção do *schedule*,

mas os cenários de uma mesma refinaria estarão sujeitos a condições semelhantes. No entanto, há outros componentes que fazem com que a construção de uma solução de *schedule* seja um desafio com diferentes graus de complexidade, dependendo das condições de contorno do problema, como por exemplo: o horizonte de programação desejado; os níveis de estoque; composição e propriedade da tancagem no início do cenário; além da previsão de recebimento de matéria-prima sob os aspectos de tempo estimado de chegada, volume, composição e propriedades dos navios. Dessa forma, foram selecionados diferentes cenários com o intuito de representar diferentes graus de complexidade e assim permitir uma avaliação mais completa dos modelos quando submetidos a diferentes condições.

6.1.2 Cenários de programação de petróleo

Cinco cenários foram selecionados para avaliação: 2309, 2327, 2416, 8119 e 8205. Eles possuem características diferentes como, por exemplo, o horizonte de programação, perfil de elenco de petróleo, níveis de inventário em tanques do terminal e da refinaria e cronograma de chegada de navios, o que testa a aptidão dos modelos em lidar com diferentes situações. Em todos os cenários, a vazão de movimentação entre Terminal e Refinaria foi definida como 1.250m^3/h. A Tabela 6.4 apresenta informações sobre o horizonte de cada cenário e a programação do navio, enquanto a Tabela 6.5 traz informações sobre volume e fração de óleo restritivo (ácido) total nas áreas. Ambas são utilizadas para embasar algumas considerações feitas a seguir.

Tabela 6.4: Horizonte de programação e informações de recebimento de petróleo de cada cenário.

Cenário	Horiz (h)	Navio	Janela (h)	Vazão (m^3/h)	tanqueCN	Volume (m^3)	Petróleo
2309	240	1	82-114	5.000	1	129.167	PT01
2327	192	1	60-80	3.750	1	66.667	PT09
2416	240	1	11-48	3.750	1	75.000	PT03
					2	58.333	PT04
		2	89-121	4.167	1	129.168	PT01
8119	432	1	13-43	5.000	1	92.000	PT30
		2	157-187	5.000	1	92.000	PT30
		3	204-234	4.170	1	83.000	PT08
		4	325-355	5.000	1	50.000	PT18
					2	92.000	PT30
8205	480	1	12-42	4.167	1	87.500	PT08
		2	280-310	5.000	1	95.833	PT18
		3	352-382	5.000	1	70.000	PT13

Tabela 6.5: Perfil de inventário e carga programada das UDAs.

Cenário	Navios		Terminal		Refinaria		UDA (m3/h)
	Vol (m3)	fração restrit	Vol útil (m3)	fração restrit	Vol útil (m3)	fração restrit	
2309	129.167	0	181.798	0,14	93.601	0,15	1006,9
2327	66.667	0	110.075	0,21	92.624	0,10	1024,2
2416	262.500	0	10.940	0,23	91.224	0,30	945,7
8119	409.000	0,20	15.727	0,63	189.301	0,40	729,2
8205	253.333	0,62	31.167	0,05	158.492	0,18	729,3

Supostamente, o cenário 2327 deve ser o de mais fácil resolução. Seu horizonte de programação é o menor, o que diminui a quantidade de movimentações que precisam ser programadas. O inventário na refinaria não chega a ser um problema, pois há volume disponível para mais de três dias de operação, calculado com base na vazão total das UDAs e o volume útil na Refinaria apresentados na Tabela 6.5. Os três dias são considerados uma referência de estoque mínimo pelas refinarias. A fração de óleo restritivo (ácido) neste elenco é baixa. O Terminal tem nível de estoque perto de 40% de sua capacidade (calculado com base no volume útil do Terminal e a capacidade dos tanques deste terminal apresentado na Tabela 6.1) e alguma restrição. No entanto, o único navio previsto não tem nenhuma restrição, o que facilita a composição com o elenco em solo.

Os cenários 2309 e 2416 possuem dois dias a mais de horizonte de programação. No que diz respeito aos seus estoques, são opostos e ambos representam grandes desafios para o *scheduling*. O cenário 2309 tem inventário alto no terminal e precisa abrir espaço nos tanques para receber os quase 130.000 m^3 que chegam de uma única vez. É esperado que haja considerável movimentação no oleoduto, especialmente nos primeiros dias de horizonte, de forma a transferir petróleo do terminal para a refinaria. Por outro lado, o ciclo completo de utilização de um tanque da refinaria é equivalente ao seu tempo de recebimento, tempo de preparação e tempo na carga da unidade. Como a vazão da unidade é menor do que a vazão de recebimento e ainda existe o tempo de preparação, a transferência terminal-refinaria pode precisar ser interrompida.

No cenário 2416, um complicador é o baixo estoque do terminal, que não tem nenhum volume significativo para transferir até descarregar parcialmente o navio. A ociosidade do terminal, o alto volume do primeiro navio e as propriedades favoráveis do elenco facilitam esse descarregamento. Na chegada do segundo navio, o volume equivalente ao primeiro já foi cerca de 70% consumido (considerando as 78 horas entre a chegada dos dois navios e

movimentação a 1.250 m_3/h, que é a vazão do oleoduto), o que pode criar condições favoráveis para o seu descarregamento. O desafio deste cenário está na disponibilização do óleo na refinaria em tempo de seu processamento, lembrando que a movimentação no oleoduto leva cerca de 15 horas e, após recebimento, o tanque da refinaria precisa ficar em preparação por, no mínimo, 24 horas.

O cenário 8119 apresenta um horizonte significativamente maior que os dos cenários apresentados até agora. No entanto, este não é seu único desafio. Assim como no 2416, o 8119 tem baixo estoque no terminal e um volume expressivo de petróleo a ser descarregado ao longo de quatro navios. O cronograma destes navios é mais desfavorável porque o primeiro navio é menor e há um intervalo maior até a chegada do segundo. A participação de petróleos com algum tipo de restrição é alta.

O cenário 8205 é o mais longo, com 20 dias de horizonte, o que certamente resulta numa quantidade significativa de decisões de movimentação. Cenários de horizonte longo são mais difíceis de serem resolvidos sem uma ferramenta de apoio porque acumulam muitos níveis de decisões impactadas por decisões tomadas anteriormente e, a mudança de uma delas pode demandar a revisão de dezenas de condições. Em termos de composição do elenco, dois navios trazem uma fração alta de óleo restritivo. Entretanto, um deles é bem próximo ao limite, o que não deve tornar muito mais complexas as movimentações necessárias para garantir os limites de propriedades até a carga das UDAs.

Com base nesta discussão, pode-se supor que o cenário 2327 seja o mais fácil, os cenários 2309, 2416 e 8205 sejam intermediários, mas apresentando diferentes desafios, e que o cenário 8119 seja o mais complexo por reunir diversos desafios.

Nas próximas seções são apresentados os resultados dos estudos feitos com esses cenários com o objetivo de resolver, com eficiência, estas programações e oferecer diferentes soluções ao programador.

6.2 Gramática e parâmetros do modelo

Uma das entidades básicas do modelo QIGLGP é a DSL que deve representar as instruções do problema que se pretende tratar. A Seção 3.1 apresentou as principais atividades da programação de petróleo. Esta seção apresenta a DSL resultante da representação destas atividades aplicada à refinaria do estudo de caso, e também um exemplo do gene quântico gerado pela interpretação dessa gramática.

As atividades que devem ser representadas tratam de: “descarregamento

de navio", "carga da UDA" e "transferência por oleoduto". Cada uma destas atividades pode ser executada com o volume máximo que a movimentação pode ter ou uma fração dele. A carga de UDA pode ser feita com um ou dois tanques e, no último caso, caracteriza uma tarefa de carga com tanque de injeção. Dessa forma, além da função $\langle NOp \rangle$ discutida na Seção 4.3, a lista de funções disponíveis na gramática fica definida como:

- $\langle RcvItemMax \rangle$ e $\langle RcvItemProp \rangle$: para o descarregamento de navios. Ambas têm como argumentos o identificador do navio, o *tanqueCN* deste navio e o tanque de destino no terminal;
- $\langle PipeTransferMax \rangle$ e $\langle PipeTransferProp \rangle$: para a transferência entre terminal e refinaria. Ambas têm como argumentos o tanque de origem no terminal e o tanque de destino na refinaria;
- $\langle FeedCDUMax \rangle$ e $\langle FeedCDUProp \rangle$: para a carga de uma UDA a partir de um tanque. Ambas têm como argumentos o tanque de origem na refinaria e a UDA de destino;
- $\langle FeedCDUMaxInj \rangle$ e $\langle FeedCDUPropInj \rangle$: para a carga de uma UDA a partir de dois tanques. Ambas têm como argumentos o tanque principal de origem na refinaria, o tanque de injeção de origem na refinaria, a UDA de destino e a fração de participação do tanque de injeção na carga total da UDA.

O sufixo Max presente em algumas funções indica que estas são as que movimentam o volume máximo possível da instrução no momento de sua execução. As instruções com sufixo $Prop$ indicam que o volume movimentado será uma fração do máximo e, por isso, esse conjunto de funções precisa de um argumento a mais que é o valor desta fração. Dependendo do tipo de função, os argumentos $Prop$ são diferentes e, portanto, tem elementos terminais diferentes. A Figura 6.1 apresenta os elementos terminais para as proporções de atividades onde: $\langle itemProp \rangle$ se refere à fração para o descarregamento do navio, $\langle pipeProp \rangle$ à fração para transferência de oleduto, $\langle feedProp \rangle$ à fração de carga de unidade e $\langle injProp \rangle$ à fração de participação do tanque de injeção. Os elementos terminais destes argumentos foram determinados em [5] e mantidos neste trabalho. Conhecendo estas funções e a topologia da planta, é possível construir a gramática do cenário.

O volume máximo de uma movimentação depende do tipo de atividade que ela representa e dos equipamentos envolvidos no momento de sua execução. Seu princípio de cálculo é o menor valor entre o volume disponível no equipamento de origem e a demanda (espaço em tanque ou horizonte de carga para

```
<func>::=   "NOp" |
            "RcvItemMax"  <navioeTnq> <tnqT> |
            "RcvItemProp" <navioeTnq> <tnqT> <itemProp> |
            "FeedCDUMax" <tnqR> <uda>
            "FeedCDUProp" <tnqR> <uda> <feedProp> |
            "FeedCDUMaxInj" <tnqR> <tnqInj> <injProp> <uda>|
            "FeedCDUPropInj" <tnqR><tnqInj> <injProp> <feedProp> <uda>|
            "PipeTransfMax" <tnqT> <tnqR> |
            "PipeTransfProp" <tnqT> <tnqR> <pipeProp>

<navioeTnq>::= "navio1" <navio1Tanq> | "navio2" <navio2Tanq>
<navio1Tanq> ::= "tanqCN1" | "tanqCN2"
<navio2Tanq> ::= "tanqCN1"
<tnqT>       ::= "TqT01" | "TqT02" |"TqT03" |"TqT04"
<tnqR>       ::= "TqR01" | "TqR02" |"TqR03" |"TqR04"
<tnqInj>     ::= "TqR01" | "TqR02" |"TqR03" |"TqR04"
<uda>        ::= "UDA-01" | "UDA-02"
<itemProp>   ::= "0.1"|"0.2"|"0.3"|"0.4"|"0.5"|"0.6"|"0.7"|"0.8"|"0.9"
<pipeProp>   ::= "0.2"|"0.4"|"0.6"|"0.8"
<feedProp>   ::= "0.2"|"0.4"|"0.5"|"0.6"|"0.8"
<injProp>    ::= "0.1"|"0.2"|"0.3"| "0.4"| "0.5"
```

Figura 6.1: Gramática para o cenário 2416

.

UDA, por exemplo) disponível no equipamento de destino. Esta informação deve ser obtida da planta no horário associado à execução da instrução.

O cálculo do volume máximo *VolumeMaxAtiv* para as movimentações de descarregamento de navio é dado por:

$$VolumeMaxAtiv_{Navio} = \text{Min}(tnqT_{esp}, navioeTnq_{vol}), \quad (6\text{-}1)$$

onde $tnqT_{esp}$ é o espaço disponível no tanque do terminal definido para recebimento e $navioeTnq_{vol}$ é o volume ainda a ser descarregado do *tanqueCN* escolhido para o navio da vez. Já para as transferências de oleoduto, tem-se que:

$$VolumeMaxAtiv_{Duto} = \text{Min}(tnqT_{vol}, tnqR_{esp}, itemBombeio_{vol}), \quad (6\text{-}2)$$

onde $tnqT_{vol}$ é o volume útil disponível no tanque selecionado no terminal, $tnqR_{esp}$ é o espaço disponível no tanque selecionado da refinaria e $itemBombeio_{vol}$ é o volume do item de bombeio que está na extremidade da refinaria, prestes a ser recebido. Por sua vez, para uma carga de UDA com um único tanque tem-se que:

$$VolumeMaxAtiv_{UDA} = \text{Min}(tnqR_{vol}, (uda_{vaz} \times (dtHr_{fC} - dtHr_{iA}))), \quad (6\text{-}3)$$

onde, $tnqR_{vol}$ é o volume útil disponível no tanque selecionado na refinaria, uda_{vaz} é a vazão da unidade de destilação que o tanque deve alimentar, $dtHr_{fC}$ é a data-hora do fim do cenário de programação e $dtHr_{iA}$ é a data-hora do início da atividade que está sendo programada. E, finalmente, para carga de UDA com tanque de injeção:

$$VolumeMaxAtiv_{UDAinj} = \text{Min}\left(\frac{tnqR_{vol}}{1-\langle injProp\rangle}, \frac{tnqInj_{vol}}{\langle injProp\rangle}, uda_{vaz} \times (dtHr_{fC} - dtHr_{iA})\right), \tag{6-4}$$

onde $tnqInj_{vol}$ é o volume útil disponível no tanque de injeção selecionado na refinaria e $\langle injProp\rangle$ é o valor da fração da carga selecionado pelo algoritmo evolutivo, que é de responsabilidade do tanque de injeção.

Algumas heurísticas garantem a consistência do modelo. Note que o volume da linha submarina não é considerado para determinação do volume máximo da atividade de descarregamento de navio. As heurísticas implementadas garantem que qualquer volume maior do que o da linha submarina pode ser movimentado, mesmo que não seja um múltiplo de seu inventário e também garantem que, após qualquer movimentação, nunca existe um volume restante em *tanqueCN* inferior ao volume da linha submarina para não violar sua característica de homogeneidade. Em último caso, se o espaço disponível no tanque selecionado for inferior ao volume da linha submarina, esta instrução não será executada e o algoritmo evolutivo será responsável por encontrar uma solução mais adequada que evite esta situação.

A configuração com tanque de injeção fornece maior flexibilidade para montar a composição da carga, mas dificulta a programação dos equipamentos, dado que durante a operação de injeção dois tanques estão alocados para a atividade. Cabe ao programador decidir sobre a relação custo × benefício deste dilema. Com sua representação como uma função da gramática, esta decisão cabe ao modelo evolutivo.

Para melhor visualização do gene quântico, a Tabela 6.6 apresenta a estrutura de um gene construído a partir da gramática do cenário 2416. Nela podem ser observadas todas as possibilidades de estado do *qudit*. As observações deste gene produzirão uma instrução de gene clássico. Uma lista de genes clássicos forma um indivíduo clássico. Este processo é exemplificado na Figura 6.2, gerando instruções para o cenário 2416.

Os parâmetros comuns herdados do modelo QIGLGP foram mantidos

Tabela 6.6: Todas as possibilidades de estado do *qudit* de um gene quântico.

<func>		
TF	Prob	Conteúdo
0	p0’	NOp
1	p1’	RcvItemMax
2	p2’	RcvItemProp
3	p3’	FeedCDUMax
4	p4’	FeedCDUProp
5	p5’	FeedCDUMaxInj
6	p6’	FeedCDUPropInj
7	p7’	PipeTransfMax
8	p8’	PipeTransfProp

TnqT		
TT	Prob	Conteúdo
0	p0’	TqT01
1	p1’	TqT02
2	p2’	TqT03
3	p3’	TqT04

shipAndTank					
navio1			navio2		
TT	Prob	Conteúdo	TT	Prob	Conteúdo
0	p0’	tanqCN1	0	p0’	tanqCN1
1	p1’	tanqCN2			

ItemProp		
TT	Prob	Conteúdo
0	p0’	0.1
1	p1’	0.2
2	p2’	0.3
3	p3’	0.4
4	p4’	0.5
5	p5’	0.6
6	p6’	0.7
7	p7’	0.8
8	p8’	0.9

TnqR		
TT	Prob	Conteúdo
0	p0’	TqR01
1	p1’	TqR02
2	p2’	TqR03
3	p3’	TqR04

uda		
TT	Prob	Conteúdo
0	p0’	UDA-01
1	p1’	UDA-02

feedProp		
TT	Prob	Conteúdo
0	p0’	0.2
1	p1’	0.4
2	p2’	0.5
3	p3’	0.6
4	p4’	0.8

TnqInj		
TT	Prob	Conteúdo
0	p0’	TqR01
1	p1’	TqR02
2	p2’	TqR03
3	p3’	TqR04

InjProp		
TT	Prob	Conteúdo
0	p0’	0.1
1	p1’	0.2
2	p2’	0.3
3	p3’	0.4
4	p4’	0.5

TT	Prob	Conteúdo
0	p0’	0.2
1	p1’	0.4
2	p2’	0.5
3	p3’	0.6
4	p4’	0.8

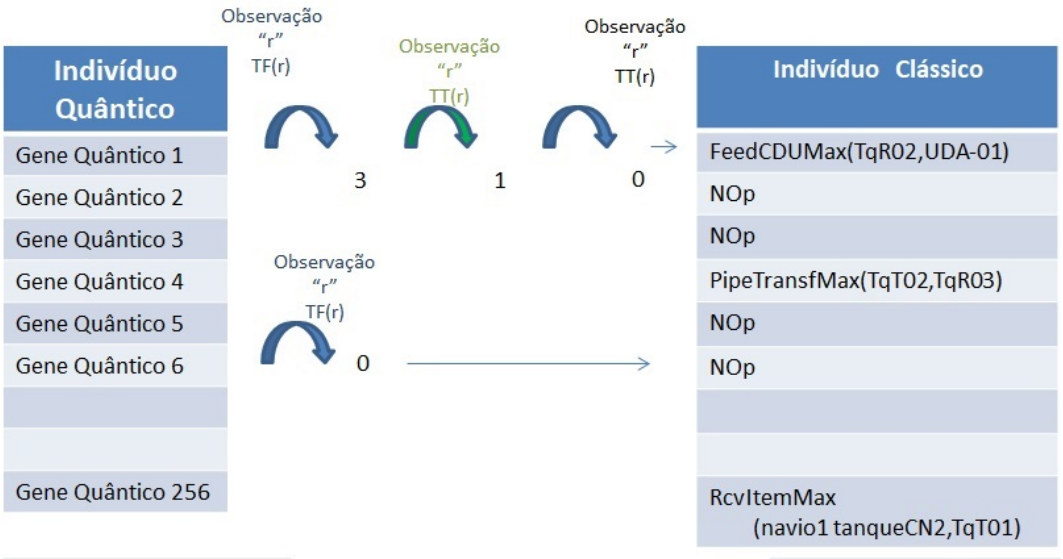

Figura 6.2: Representação da geração de um indivíduo clássico.

.

baseados no apresentado em [5], ou seja, tamanho do passo do operador quântico em 0,008, probabilidade inicial da instrução NOp em 0,8 e tamanho dos indivíduos clássico e quântico em 256 genes.

O tamanho da população foi determinado com base na Equação 5-4, que resulta em 56 indivíduos para os modelos que consideram quatro objetivos, ou seja, fica excluído apenas o problema Seguidor do modelo BiO-CDQIGLGP. Neste caso, o tamanho da população é de 13 indivíduos, determinado pela mesma equação. O número de gerações foi definido em 10.000 para todas as propostas, pois esta evolução leva tempo equivalente ao que o programador leva para gerar a sua solução.

A Seção 6.3 apresenta os resultados obtidos para os diferentes modelos considerando o estudo de caso e parâmetros discutidos nessa seção.

6.3 Resultados Experimentais

6.3.1 E-Prz: Priorização de Objetivos

Esta seção apresenta os resultados do modelo QIGLGP, que prioriza os objetivos e os compara hierarquicamente. A representação e evolução baseada em programação genética e inspiração quântica desenvolvidas neste modelo se mantiveram como base de todos os estudos feitos nesta tese. Estes resultados têm o propósito de estabelecer uma base de comparação para avaliação das abordagens multiobjetivo e *bilevel* nas próximas seções.

A Figura 6.3 permite visualizar a distribuição do melhor indivíduo de corridas que resultaram em *schedules* viáveis e a Tabela 6.7 apresenta algumas estatísticas destas corridas. Com base nelas, é possível observar que o cenário 2416 é o que apresenta maior dispersão para o objetivo de UDA, o que pode ser associado ao baixo nível do inventário deste cenário, como visto na Seção 6.1.2, que faz com que a convergência deste objetivo seja mais desafiadora. O cenário 2327 tem uma grande nuvem de pontos, sendo a maioria concentrada na origem. Este comportamento é coerente com a expectativa de que seja de mais fácil resolução. O cenário 8205 tem um patamar mais alto e disperso para o objetivo de duto, indicando menor compactação da programação deste equipamento. Finalmente, o 8119 é nitidamente um cenário desafiador pois se observam apenas dois pontos de solução para ele. O diâmetro das esferas representa o valor do objetivo de descarregamento de Navio e nenhuma variação sensível é observada, o que é confirmado nas estatísticas da Tabela 6.7.

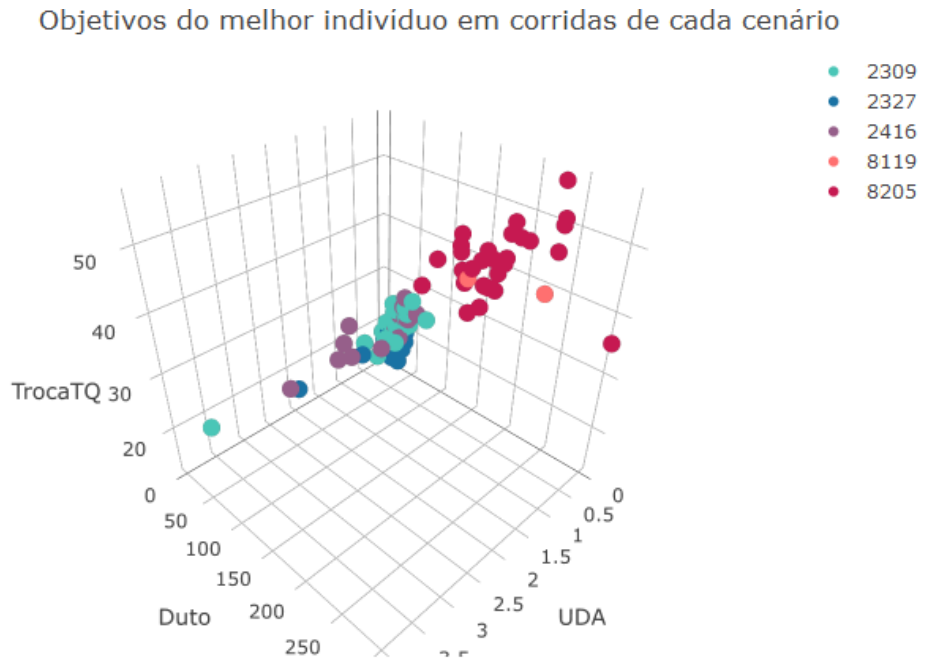

Figura 6.3: Melhor indivíduo das corridas com soluções aceitas.

A Figura 6.4 apresenta a evolução do melhor indivíduo nos quatro objetivos considerando a média das réplicas válidas. Os gráficos estão apresentados seguindo a prioridade dos objetivos. Para minimizar o efeito de escala, o eixo do valor dos objetivos é dividido pelo tamanho do horizonte em três objetivos. É possível observar uma degeneração da convergência dos objetivos à medida que sua classificação na escala de prioridades é menor. O objetivo de UDA tem um perfil de evolução característico de uma convergência. No objetivo

Tabela 6.7: Estatísticas das corridas aceitas da QIGLGP.

Métrica	Modelo	Cenário				
		2309	2327	2416	8119	8205
UDA	Média	0,17	0,06	0,53	0,00	0,00
	DesvPad	0,67	0,31	0,78	0,00	0,00
Navio	Média	0,00	0,00	0,00	0,00	0,00
	DesvPad	0,03	0,00	0,00	0,00	0,00
Duto	Média	25,86	16,12	41,11	178,0	163,13
	DesvPad	19,66	8,71	16,92	68,59	49,89
Troca Tanque	Média	20,5	16,1	23,6	37,5	40,3
	DesvPad	3,76	1,68	2,64	2,12	6,07

de Navio este perfil também é observado, mas ele é interrompido em alguns momentos nos quais o objetivo de UDA alcança valores menores. A evolução do duto é suave, mas ainda se observa um perfil de redução. Já no objetivo de troca de tanque ela praticamente não acontece. Estes resultados sinalizam que a abordagem por hierarquia está, essencialmente, tratando dos objetivos críticos, mas não alcança os não-críticos no número de gerações definido.

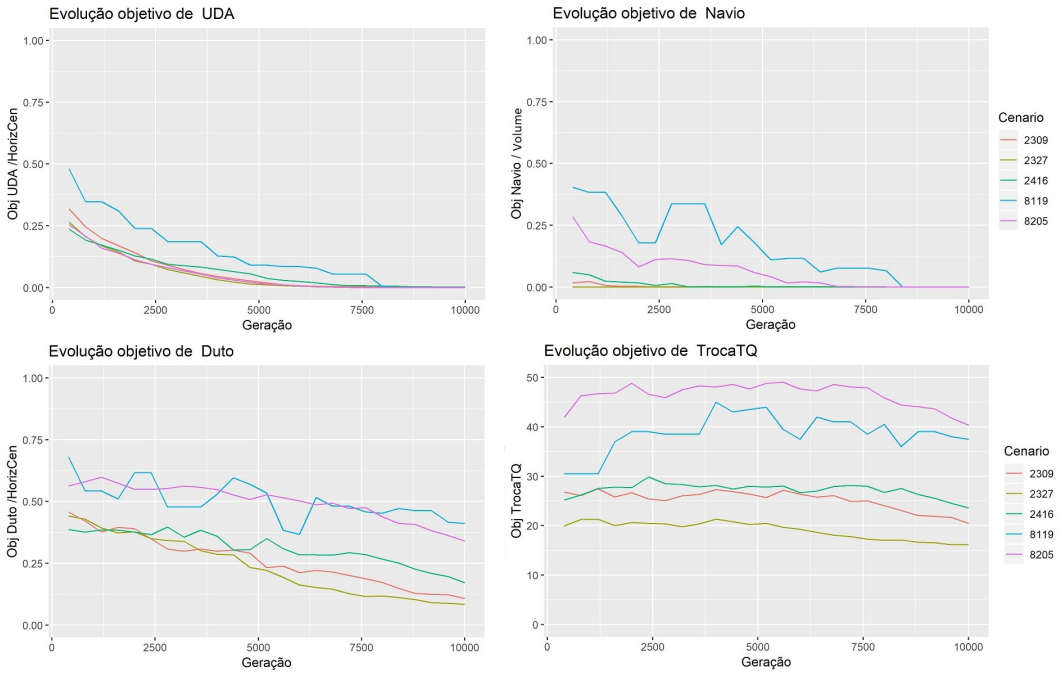

Figura 6.4: Média da evolução do melhor indivíduo das corridas com soluções aceitas no modelo QIGLGP.

O percentual de corridas que resultaram em soluções aceitas (%CSA) é apresentado na Figura 6.5, na parte azul das barras. A parte laranja se refere ao aumento do %CSA para estes mesmos cenários quando as restrições do problema são relaxadas. Ou seja, limites de propriedades ou tempo mínimo

de movimentação não são considerados. Apenas as restrições de lastro e capacidade dos tanques são mantidas, pois em caso contrário, o problema de *schedule* estaria descaracterizado. Assim, os valores dentro das barras azuis se referem ao %CSA do modelo QIGLGP e os valores dentro das barras laranjas se referem ao incremento do %CSA resultante da relaxação das restrições.

Também pela Figura 6.5 se observa o grande impacto que existe no cenário 8119, sobre o qual foi mapeado maior desafio em relação à composição de petróleo. Com a relaxação das restrições, a capacidade de resolução deste cenário é aumentada. O segundo cenário com maior impacto em relação à relaxação das restrições é o 2309, que apresentava níveis altos de estoque, e pode ter sido favorecido pela relaxação do tempo mínimo de movimentações, o qual permite transferências de pequenos volumes e pode aumentar o aproveitamento da capacidade dos tanques.

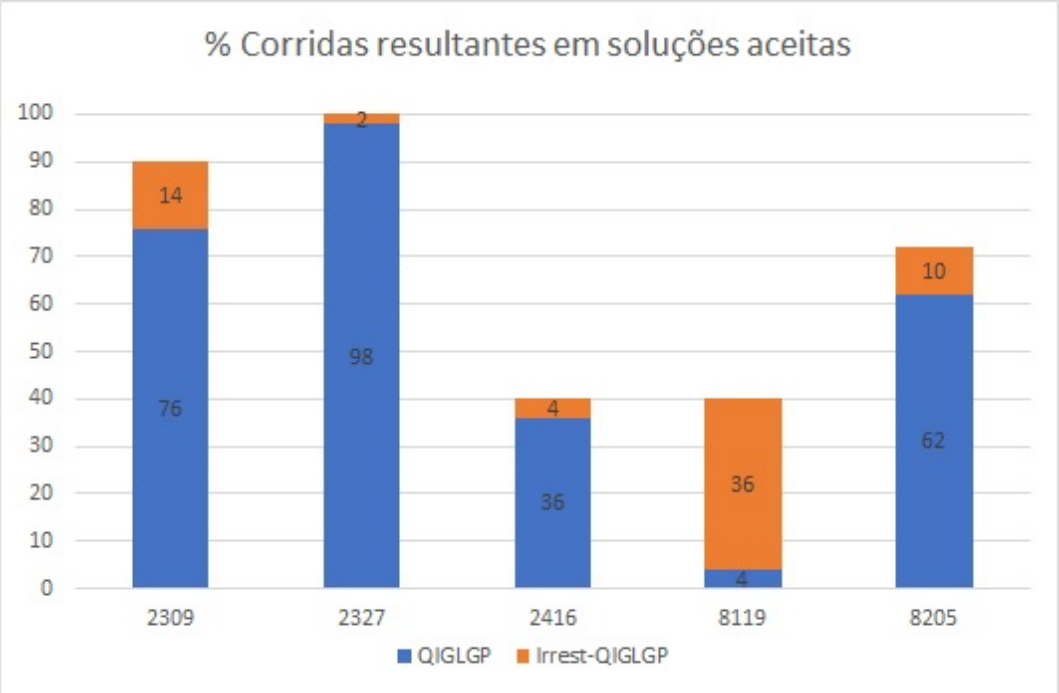

Figura 6.5: Percentual das corridas com soluções aceitas com e sem restrições.

Os modelos propostos nesta tese e apresentados nas próximas seções objetivam atingir valores de %CSA superiores ao modelo QIGLGP representado nas barras azuis da Figura 6.5 e, diferentemente deste modelo, fornecer ao programador de produção mais de uma alternativa de qualidade para que ele possa avaliar as soluções e decidir sobre sua adoção.

6.3.2
Estratégias Multiobjetivo

As seções 6.3.2.1 à 6.3.2.4 apresentam os resultados obtidos para as propostas baseadas em algoritmos multiobjetivo onde o processo evolutivo acontece em um único nível de decisão.

6.3.2.1
Non-dominated Sort QIGLGP

Nesta seção estão os resultados das propostas que investigaram modelos baseados em uma ordenação não-dominada dos indivíduos da população clássica, comparando: capacidade de representar, ou não, o não-atendimento aos objetivos críticos como uma violação da qualidade da solução; e o impacto de uma população externa influenciando a evolução. A primeira etapa foi a avaliação do modelo NSQIGLGP, que não diferencia os objetivos.

A Figura 6.6 apresenta a evolução dos hipervolumes dos cinco cenários no modelo NSQIGLGP, gerados a partir das soluções não dominadas, pertencentes à frente de Pareto ótima (*FP0*) da população. Como a proposta é avaliar se a população evolui com o algoritmo NSQIGLGP, as curvas consideram todas as réplicas, e não somente as que possam ter resultado em *schedules* viáveis. O aumento do hipervolume indica que, sim, a evolução está encontrando um conjunto de soluções não-dominadas na população e que os objetivos estão sendo minimizados. No entanto, os resultados apresentados na Tabela 6.8 mostram que a minimização não é suficiente para atingir o critério de atendimento aos objetivos críticos. Este modelo não foi capaz de encontrar soluções viáveis para o problema de *schedule* em dois cenários e também mostra que, para os outros três cenários, seu desempenho é consideravelmente pior do que o obtido pelo QIGLGP.

Tabela 6.8: Percentual de corridas com soluções aceitas no modelo NSQIGLGP.

Métrica	Modelo	Cenário				
		2309	2327	2416	8119	8205
%CSA	NSQIGLGP	2	12	0	0	18

A Figura 6.7 mostra a evolução média dos quatro objetivos. Uma comparação com a Figura 6.4 permite observar um comportamento muito diferente do gerado pelo QIGLGP. Durante toda a evolução, o objetivo de UDA está sistematicamente acima no NSQIGLGP e o decaimento característico da convergência é mais suave, mesmo no cenário mais fácil. O objetivo de Navio não tem os vales e picos da QIGLGP, mas também não há suficiente minimização de seu valor. O objetivo de movimentação do oleoduto atinge

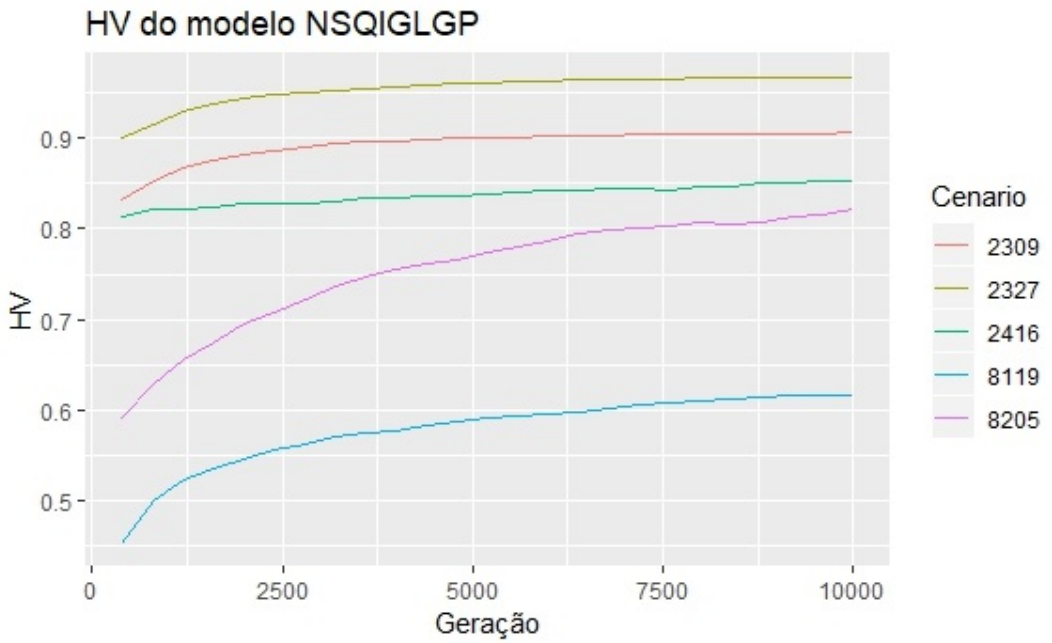

Figura 6.6: Hipervolume médio de todas as réplicas do modelo NSQIGLGP.

patamares mais baixos no começo da evolução e, ao final, é maior nos cenários 2327 e 2309 e menor no cenário 8119. A maior vantagem do modelo por dominância está no objetivo que mede as trocas de tanque. Enquanto no QIGLGP fica claro que a evolução não consegue atuar efetivamente sobre este objetivo, o NSQIGLGP tem valores baixos para ele, evidenciando que a otimização busca soluções que o minimizem. Este comportamento é alinhado com a resposta dos demais objetivos, pois se houve mais desvio em relação à carga da UDA e mais atraso (ou não entrega) no descarregamento de navios, o número de movimentações é menor e, consequentemente, o número de trocas de tanque. Com estes resultados, pode-se observar que o modelo NSQIGLGP tende a reduzir movimentações para ter bom atendimento aos objetivos não críticos, o que resulta em soluções inviáveis. Este comportamento é esperado pela não diferenciação dos objetivos, mas não atende como solução de um problema de programação. Dessa forma, a próxima seção trata da necessária diferenciação dos objetivos.

6.3.2.2 *Constrained Non-dominated Sort* QIGLGP

Como discutido na Seção 5.2.2.1, a Equação 5-5 foi proposta para medir, durante o processo evolutivo, a aptidão dos indivíduos em resolver os objetivos críticos e, na etapa de ordenação da população, influenciar a comparação por dominância entre os indivíduos. Consequentemente, a organização dos

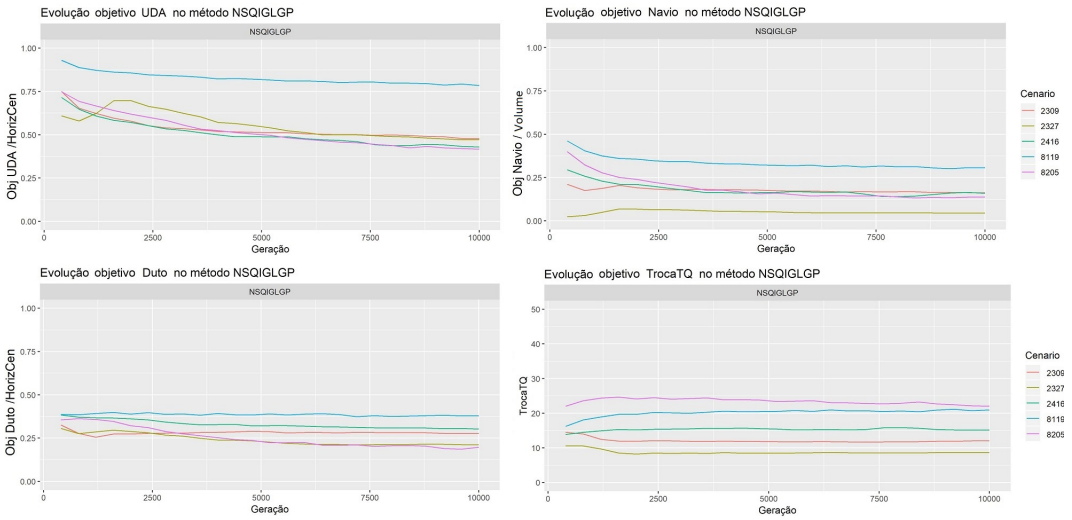

Figura 6.7: Evolução dos objetivos no modelo NSQIGLGP.

indivíduos nas frentes de Pareto é alterada, quando comparado ao que acontece com o modelo NSQIGLGP. Foram avaliadas duas propostas para a forma como o não-atendimento aos objetivos críticos influenciaria a ordenação dos indivíduos:

- "não-desempate" (ndC-NSQIGLGP). Quando dois indivíduos possuem o mesmo valor de violação para o atendimento aos objetivos, não se pode afirmar que há uma relação de dominância entre eles [85].
- "desempate" (dC-NSQIGLGP). Se dois indivíduos possuem o mesmo valor de violação para o atendimento aos objetivos críticos, então os objetivos não-críticos são comparados utilizando a dominância de Pareto, ou seja, se for possível estabelecer que um dos indivíduos não é pior em nenhum desses dois objetivos não-críticos e é melhor em pelo menos um deles, então este indivíduo domina e deve ser ordenado numa frente de Pareto de menor *rank*.

A Tabela 6.9 apresenta o percentual de corridas que resultaram em soluções aceitas para as duas metodologias. A primeira observação feita é que o desempenho de ambos os métodos é muito superior ao obtido pelo NSQIGLGP.

Tabela 6.9: Percentual de corridas com soluções aceitas nas propostas para o modelo C-NSQIGLGP.

Métrica	Modelo	Cenário				
		2309	2327	2416	8119	8205
%CSA	ndC-NSQIGLGP	98	98	76	14	62
	dC-NSQIGLGP	98	100	78	20	66

A Figura 6.8 apresenta a evolução média dos indivíduos da frente de Pareto *rank* 0 (*FP0*), que é composta apenas por indivíduos não-dominados que atendem aos objetivos críticos ou, na hipótese de não atendimento por nenhum indivíduo, é composta pelo que tiver menor violação. A *FP0* da última geração representa a frente de Pareto ótima do problema. A observação desta figura mostra o efeito da Equação 5-5, fazendo com que os modelos tendam a se comportar como a QIGLGP durante boa parte das gerações, melhor avaliando indivíduos que tenham valores mais baixos de desvio da carga programada e desvio no descarregamento de navios. Quando indivíduos sem violação começam a ser produzidos, os modelos passam a se comportar como um algoritmo multiobjetivo, ou seja, há melhora de todos os objetivos, aumentando o número de indivíduos na *FP0* e seu hipervolume.

A Figura 6.9 apresenta a evolução destas duas medidas e pode ser observado que suas curvas se acompanham, ou seja, quando o número de indivíduos sai de apenas um (indicando que a evolução conseguiu encontrar indivíduos que atendem aos objetivos críticos), o hipervolume ganha um impulso de evolução também. Nesta figura também se observa que, para os três primeiros cenários, a inflexão ocorre antes da metade da evolução e, para os dois últimos, mais à frente, em torno de 6.000 gerações. O valor final do hipervolume tem patamar maior nos mesmos três primeiros cenários, sendo que 2327 é o mais alto. Em relação ao número de indivíduos, o comportamento destoa do observado no hipervolume principalmente no cenário 8205 que se afasta do 8119 e mantém uma população não-dominada cujo tamanho é mais próximo ao dos cenários 2309 e 2416. Esta figura também apresenta o número de pontos de referência diferentes que possuem soluções da *FP0* associados. Pode-se observar que o perfil de desenvolvimento do número de pontos de referência acompanha a quantidade de indivíduos na *FP0*. O cenário que se distancia um pouco mais é o 8119, onde há, em média, 2,5 indivíduos na *FP0* e 2 pontos de referência. Estes resultados levam à observação de que os indivíduos que compõem a *FP0*, ainda que não sejam muitos, possuem diversidade pois são associados a pontos de referência diferentes.

Uma avaliação comparativa entre as duas metodologias (ndC-NSQIGLGP e dC-NSQIGLGP) permite algumas observações:

- dC-NSQIGLGP alcança maior %CSA em quatro cenários e a diferença é maior quanto pior é o resultado do cenário;
- a evolução dos objetivos é muito semelhante nas duas metodologias, assim como a evolução do hipervolume;
- em relação ao número de indivíduos na *FP0* e o número de pontos de referência associados, a metodologia dC-NSQIGLGP tem uma evolução

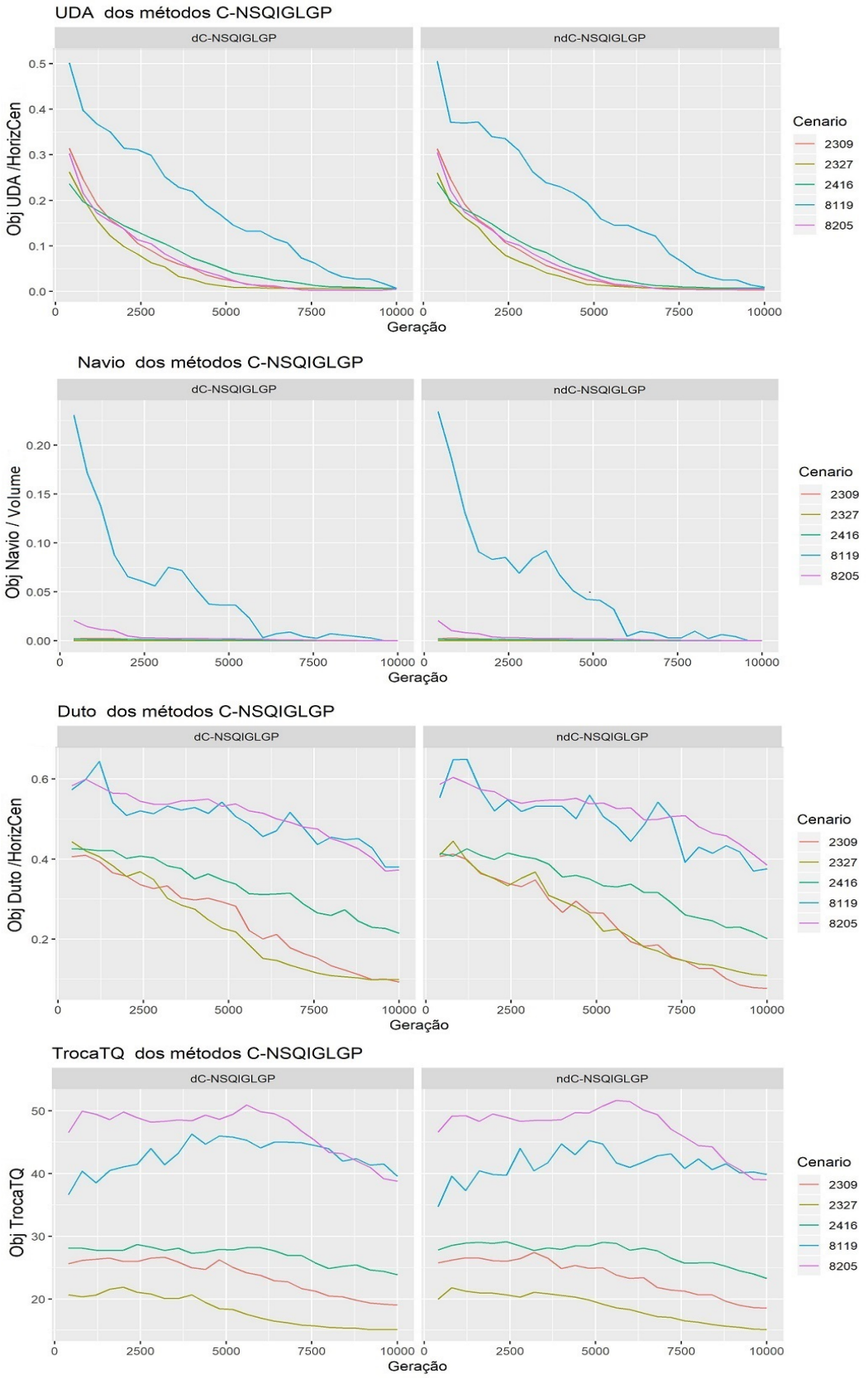

Figura 6.8: Evolução dos objetivos das alternativas para o modelo C-NSQIGLGP.

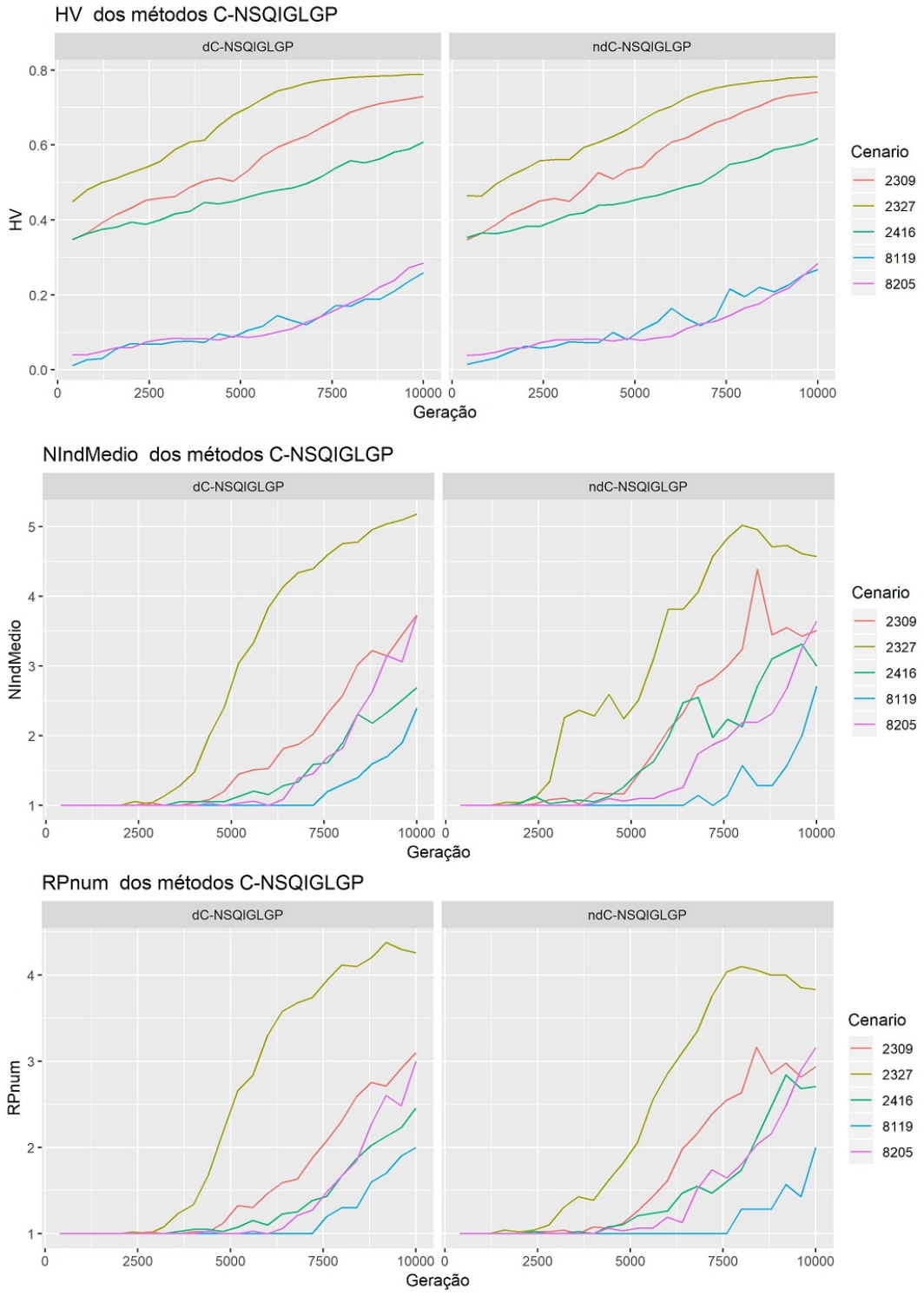

Figura 6.9: Evolução do hipervolume (no alto), quantidade de indivíduos na *FP0* (no meio) e diferentes pontos de referência presentes na FP0 (embaixo) das alternativas para o modelo C-NSQIGLGP.

mais uniforme. No entanto, na solução final, nem sempre supera a ndC-NSQIGLGP. Por exemplo, no cenário mais difícil, ndC-NSQIGLGP alcança mais soluções, mesmo que elas estejam associadas ao mesmo número de pontos de referência, o que explica sua baixa variação no hipervolume.

O melhor %CSA da metodologia dC-NSQIGLGP é importante e oferece vantagem para adoção desta abordagem para representar o modelo C-NSQIGLGP. No entanto, é necessário verificar se as indicações da Figura 6.9 representam uma perda sigificativa desta metodologia em comparação ao ndC-NSQIGLGP. Para identificar se há significância estatística que diferencie os resultados, primeiramente aplicou-se o teste de Shapiro [86], que indicou que as amostras não têm distribuição normal. Dessa forma, foi aplicado o teste de Wilcoxon [87], para variáveis independentes, de forma a testar a hipótese de que as variáveis hipervolume, número de indivíduos e número de pontos de referência na frente de Pareto ótima têm distribuição equivalente nos dois métodos. A Tabela 6.10 apresenta os p-valores do teste. Como todos são maiores de 0,05 (o que representa confiança de 95% de que são iguais), a hipótese nula de que os métodos têm soluções finais iguais não foi rejeitada. Esta indicação foi a mesma em todos os cenários e resulta na afirmação de que as frentes de Pareto ótima dos dois métodos não têm diferença estatisticamente significante.

Tabela 6.10: Teste de Wilcoxon para hipervolume, número de indivíduos e número de pontos de referência entre as metodologias ndC-NSQIGLGP e dC-NSQIGLGP.

Variável	2309	2327	2416	8119	8205
HV	0,277	0,353	0,795	0,557	0,946
NumInd	0,795	0,191	0,859	0,784	0,812
NumRPs	0,698	0,148	0,625	0,866	0,583

Dado que a metodologia dC-NSQIGLGP tem melhor desempenho na métrica %CSA e a qualidade das soluções na frente de Pareto ótima é equivalente, ela passa a representar o modelo C-NSQIGLGP.

Apesar do melhor resultado alcançado pelo C-NSQIGLGP, alguns cenários ainda representam um grande desafio para o algoritmo. Dessa forma, a próxima seção trata da avaliação do impacto de uma população externa no desempenho deste modelo.

6.3.2.3 *Archive Constrained Non-dominated Sort* QIGLGP

Na Seção 5.2.2.1 foi proposto que uma população *archive* externa (PC_A) armazenasse apenas um indivíduo associado a cada ponto de referência (RP)

que poderia ser utilizado para participar do processo evolutivo. Este indivíduo é não-dominado em relação aos que foram associados a este RP ao longo da evolução. Esta população pode interferir na evolução através da aplicação do operador quântico (aos indivíduos quânticos) seguindo as instruções de um indivíduo clássico de PC_A e não de um indivíduo clássico da população da geração em questão.

As três metodologias avaliadas para definir os momentos em que um indivíduo de PC_A orienta a evolução do algoritmo C-NSQIGLGP são:

- proporcional ao tamanho da população *archive* (tAC-NSQIGLGP);
- dependente de repetição da mesma frente de Pareto (FP) e ponto de referência (RP) na população clássica (fprpAC-NSQIGLGP);
- dependente da repetição do mesmo RP na população clássica (rpAC-NSQIGLGP);

A Tabela 6.11 apresenta o percentual de corridas que resultaram em *schedules* viáveis para as propostas com a população *archive* e também o resultado do modelo C-NSQIGLGP, apresentado na Tabela 6.9, para facilitar a comparação, onde é possível observar que não houve variação positiva ou negativa maior do que 4% em nenhum caso. Dentre todos os casos, só houve melhora do %CSA no cenário 8205 usando a metodologia fprpAC-NSQIGLGP que subiu de 66% para 70%. Esta mesma configuração manteve o desempenho nos cenários 2327 e 8119 e foi inferior em 2% nos cenários 2309 e 2416.

Tabela 6.11: Percentual de corridas com soluções aceitas nas propostas para o modelo AC-NSQIGLGP.

Métrica	Modelo	Cenário				
		2309	2327	2416	8119	8205
%CSA	tAC-NSQIGLGP	96	100	76	18	64
	fprpAC-NSQIGLGP	96	100	76	20	70
	rpAC-NSQIGLGP	94	96	74	18	62
	C-NSQIGLGP	98	100	78	20	66

Para analisar este resultado, a Figura 6.10 apresenta a probabilidade de interferência de um indivíduo de PC_A ao longo da evolução em cada uma das metodologias para os cenários 2327 (com bom desempenho e variação de 4% em apenas uma configuração) e 8205 (com desempenho médio e variação de 4% positiva e negativa, dependendo do método). Os resultados mostram que o método rpAC-NSQIGLGP é o que tem maior probabilidade de se submeter à influência de um indivíduo de PC_A e, em ambos os cenários, o desempenho do método piorou o %CSA obtido pelo C-NSQIGLGP. No cenário 2327, fprpAC-NSQIGLGP começa a ter mais atuação na metade da evolução

e tem crescimento significativo, sugerindo que os indivíduos gerados não estão contribuindo com a diversidade da população pois estão associados aos mesmos pontos de referência. Este mesmo método no cenário 8205 sai de seu patamar inicial apenas após a geração 8.000 e se mantém com probabilidade abaixo de 10%. Mesmo com o aumento no número de indivíduos na *FP0* (indicado pela inclinação da curva de tAC-NSQIGLGP), a inclinação do método fprpAC-NSQIGLGP não a acompanhou, sugerindo que os indivíduos da população estão mais dispersos pois não há aumento das repetições do par (*FP*, *RP*). Este perfil se mostrou positivo pois o %CSA do cenário 8205 aumentou.

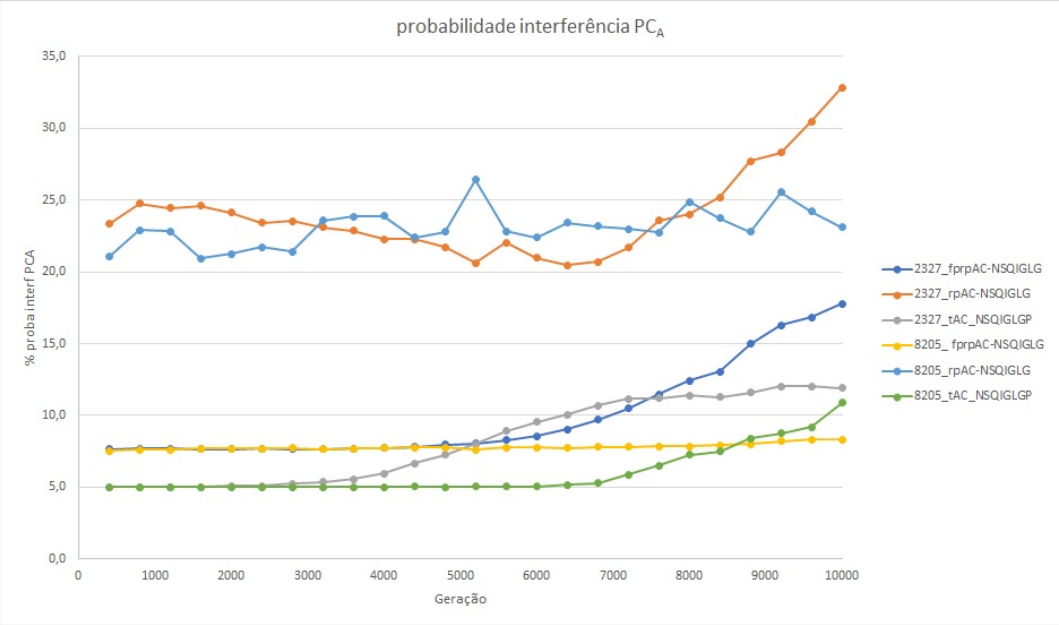

Figura 6.10: Probabilidade de que um indivíduo de PC_A interfira na evolução de acordo com as três propostas para o modelo AC-NSQIGLGP

A Figura 6.11 apresenta a variação do hipervolume e o número de indivíduos na *FP0* ao longo das gerações para as três metodologias propostas em comparação com o modelo C-NSQIGLGP. Nela pode ser observada pouca variação do comportamento do hipervolume independentemente da metodologia. A Figura 6.11 também mostra que o número de indivíduos na *FP0* apresenta semelhança entre o C-NSQIGLGP (que não utiliza população *archive*) e as propostas para AC-NSQIGLGP. As diferenças observadas no gráfico e apresentadas na Tabela 6.12 não são estatisticamente significativas, segundo o teste de Wilcoxon, para o hipervolume ou número de indivíduos na *FP0* da população final.

Foram avaliadas metodologias com maior e menor probabilidade de utilização de um indivíduo externo para orientar a atualização dos indivíduos quânticos e apenas sobre uma delas, fprpAC-NSQIGLGP, não se pode afirmar

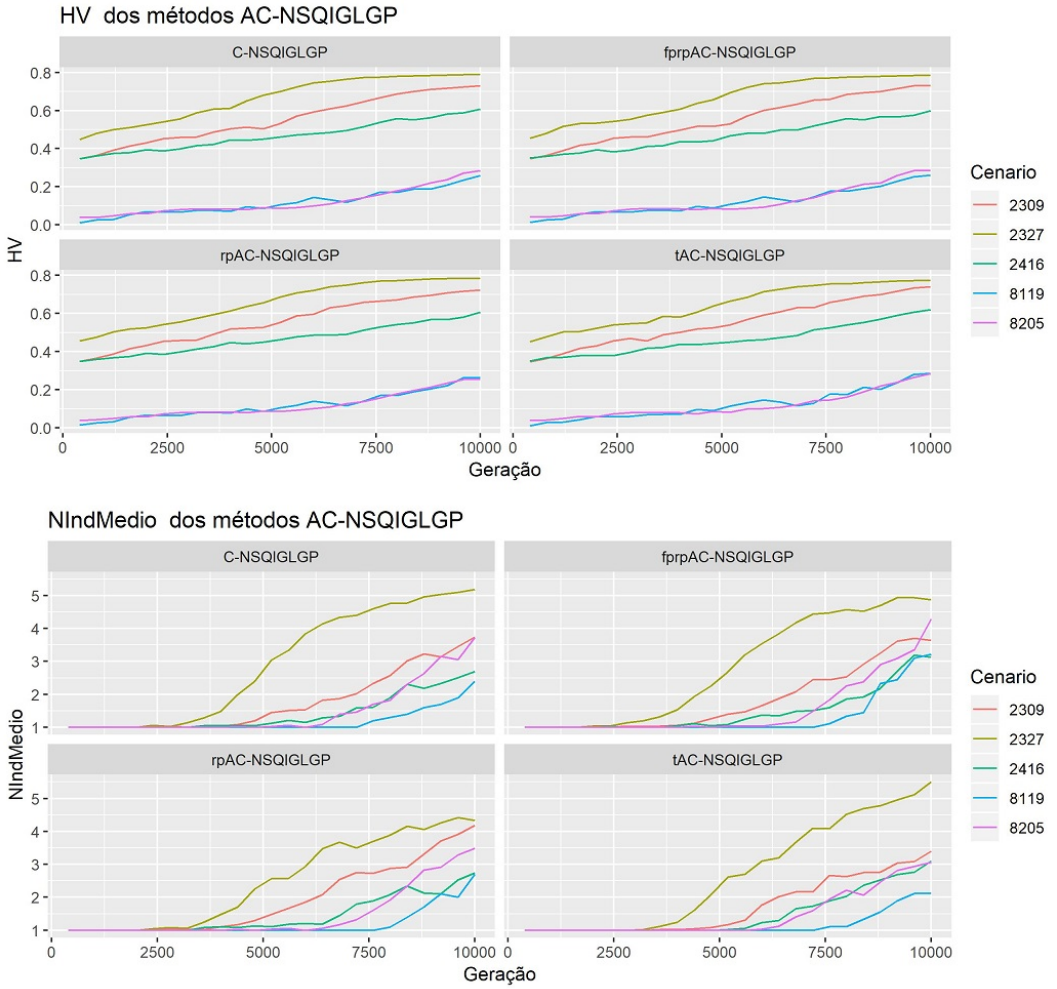

Figura 6.11: Evolução do hipervolume e quantidade de indivíduos na FP0 das alternativas para o modelo AC-NSQIGLGP.

Tabela 6.12: Hipervolume e número de indivíduos na FP0 final das metodologias propostas para o modelo AC-NSQIGLGP e do C-NSQIGLGP.

Métrica	Modelo	Cenário				
		2309	2327	2416	8119	8205
HV	C-NSQIGLGP	0,7299	0,7890	0,6076	0,2593	0,2844
	tAC-NSQIGLGP	0,7394	0,7739	0,6196	0,2856	0,2846
	fprpAC-NSQIGLGP	0,7322	0,7844	0,6000	0,2619	0,2838
	rpAC-NSQIGLGP	0,7227	0,7848	0,6060	0,2646	0,2565
NumInd	C-NSQIGLGP	3,73	5,18	2,69	2,40	3,73
	tAC-NSQIGLGP	3,40	5,50	3,11	2,11	3,06
	fprpAC-NSQIGLGP	3,64	4,88	3,14	3,22	4,29
	rpAC-NSQIGLGP	4,19	4,34	2,74	2,70	3,49

que o modelo C-NSQIGLGP supera sempre. Dessa forma, este é o algoritmo que representa o modelo AC-NSQIGLGP nas considerações finais.

6.3.2.4 *Decomposition* QIGLGP

Esta seção apresenta os resultados das proposições que investigaram modelos baseados na decomposição do problema em subproblemas, considerando ou não o atendimento aos objetivos críticos como uma violação, assim como também o impacto da população externa influenciando a evolução. A primeira etapa é a avaliação do modelo DQIGLGP, que não considera violação dos objetivos críticos.

A Tabela 6.13 apresenta o %CSA para este modelo e, para facilitar a comparação, também os resultados do QIGLGP e NSQIGLGP. Pode-se observar que, mesmo ainda sem diferenciar os objetivos, a abordagem por decomposição apresenta resultados positivos, superiores mesmo quando comparados ao QIGLGP, que apresentava os melhores até então.

Tabela 6.13: Percentual de soluções aceitas para os modelos DQIGLGP, NSQIGLGP e QIGLGP.

Métrica	Modelo	Cenário				
		2309	2327	2416	8119	8205
%CSA	DQIGLGP	86	98	76	12	90
	NSQIGLGP	2	12	0	0	18
	QIGLGP	76	98	36	4	62

A Figura 6.12 apresenta a evolução média dos objetivos na população *archive*, que representa a solução final. São considerados todos os indivíduos dessa população, que é elitista pois armazena os indivíduos não-dominados da evolução, independentemente da geração em que foram criados. Neste modelo estão incluídos indivíduos que atendem ou não aos objetivos críticos, dado que a equação de Violação ainda não é considerada. A evolução no DQIGLGP apresenta um comportamento intermediário entre o QIGLGP e o NSQIGLGP, com os pontos positivos de ambos. Pode-se observar a curva característica de convergência dos objetivos críticos, como acontece no QIGLGP, mas as grandes oscilações dos objetivos não-priorizados, característicos daquele modelo, não acontecem nesta proposta. A própria curva de convergência é mais suave nos objetivos críticos. Mesmo os objetivos não-críticos conseguem evoluir, alcançando patamares menores do que com o QIGLGP, assim como acontece com o NSQIGLGP.

O bom resultado da proposta DQIGLGP motivou a avaliação se a equação de Violação contribuiria ainda mais para o atingimento do comportamento

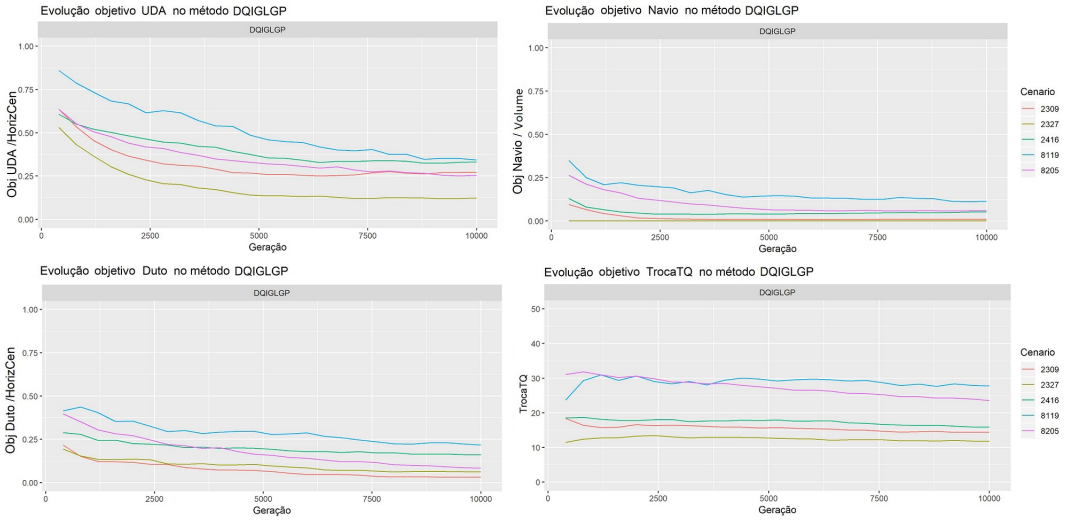

Figura 6.12: Evolução dos objetivos no modelo DQIGLGP.

desejado, ou seja, aumentando o perfil de convergência sobre os objetivos críticos inicialmente e, uma vez atingido esse estágio, a população evoluiria buscando a melhor relação de compromisso entre os quatro objetivos. A Seção 6.3.2.5 apresenta estes resultados.

6.3.2.5 *Constrained Decomposition* QIGLGP

Na Seção 5.2.2.2 foi discutido que o modelo baseado em decomposição poderia utilizar a equação de Violação 5-5 para diferenciar os objetivos em dois momentos: na aptidão do indivíduo para resolver um subproblema, conforme proposto por [47], e na atualização da população externa.

Na Seção 6.3.2.2 foi apresentado o bom desempenho desta abordagem sobre o modelo baseado em dominância de Pareto. A melhor proposta foi a que estabelecia um critério quando a Violação de dois indivíduos resultava no mesmo valor. Neste caso, o desempate era dado pela avaliação de não-dominância dos objetivos críticos. Este resultado motivou a adoção desta metodologia na abordagem por decomposição.

Dessa forma, esta seção apresenta os resultados de duas propostas para caracterizar o modelo C-DQIGLGP:

– "desempate *archive*" (daC-DQIGLGP): apenas a permanência (ou entrada) na população *archive* considera a violação dos objetivos e, em caso de empate na violação, os indivíduos são ordenados de acordo com a dominância de Pareto;

– "desempate *archive* e subproblema" (dasC-DQIGLGP): a proposta de daC-QIGLGP em relação à população externa é aplicada aqui e também é implementado que, durante a evolução, quando um indivíduo é avaliado como solução candidata de um subproblema e comparado com a solução que este subproblema já tinha, este novo indivíduo somente substituirá o existente se tiver violação menor ou, em caso de empate, menor valor da função de decomposição.

Os métodos foram avaliados considerando, ou não, a normalização dos objetivos durante o processo evolutivo, pois resultados apresentados em [38] sinalizam que a normalização pode melhorar o desempenho do modelo de decomposição em problemas com escalas de objetivos muito diferentes. A identificação das versões normalizadas se dá pela letra *m* nos nomes, por exemplo, mdaC-DQIGLGP é o método daC-DQIGLGP em que os objetivos foram normalizados.

A Tabela 6.14 apresenta os resultados da primeira métrica considerada, %CSA, comparando os dois métodos com o obtido pelo modelo DQIGLGP. Nela se observa que, em ambos os métodos, com suas versões normalizadas ou não, há um aumento significativo na capacidade do modelo baseado em decomposição de resolver o problema de programação de petróleo. À exceção do cenário 8119, que já se mostrou o mais desafiador, os outros quatro cenários atingem patamares muito próximos de 100% de CSA, quando não este nível. Mesmo o 8119, que havia atingido seu melhor resultado com 20% de CSA no modelo C-NSQIGLGP, sobe para o mínimo de 62% com as propostas de C-DQIGLGP.

Tabela 6.14: Percentual de soluções aceitas para o DQIGLGP e os métodos propostos para C-DQIGLGP

Métrica	Modelo	Cenário				
		2309	2327	2416	8119	8205
Métodos	daC-DQIGLGP	100	100	100	70	100
	dasC-DQIGLGP	100	100	100	62	100
	mdaC-DQIGLGP	100	98	96	76	100
	mdasC-DQIGLGP	100	100	100	72	100
	DQIGLGP	86	98	76	12	90

Numa comparação entre as alternativas propostas, se observa que utilizar a equação de violação durante o processo de definição da solução do subproblema (dasC-DQIGLGP e mdasC-DQIGLGP) tem desempenho pior do que seus pares que a utilizam apenas para a atualização da população *archive* (daC-DQIGLGP e mdaC-DQIGLGP) no cenário mais desafiador, que também é beneficiado pela normalização dos objetivos. No entanto, a configuração

que beneficia este cenário prejudicou os cenários 2327 e 2416, os quais tiveram perda no %CSA. Para o cenário 8119 as versões normalizadas têm desempenho melhor do que qualquer das versões não normalizadas.

Ainda que a perda do %CSA em cenários mais fáceis seja inferior ao ganho no cenário mais desafiador e que, mesmo com a perda, o patamar de desempenho dos cenários 2327 e 2416 se mantém acima de 96%, outros aspectos das soluções finais são avaliados para definir o modelo C-DQIGLGP.

A Tabela 6.15 apresenta a média e desvio padrão (entre parênteses) do hipervolume e número de indivíduos na frente de Pareto ótima. Os resultados dos quatro métodos avaliados não têm distribuição normal para nenhuma das variáveis, segundo o teste de Shapiro.

Tabela 6.15: Estatísticas para os métodos propostos para o C-DQIGLGP.

Métrica	Modelo	Cenário				
		2309	2327	2416	8119	8205
HV	daC-DQIGLGP	0,757 (0,046)	0,804 (0,024)	0,671 (0,062)	0,352 (0,056)	0,455 (0,067)
	dasC-DQIGLGP	0,751 (0,049)	0,803 (0,024)	0,689 (0,037)	0,330 (0,081)	0,469 (0,080)
	mdaC-DQIGLGP	0,757 (0,055)	0,801 (0,026)	0,685 (0,042)	0,346 (0,058)	0,478 (0,082)
	mdasC-DQIGLGP	0,756 (0,030)	0,805 (0,025)	0,674 (0,039)	0,341 (0,060)	0,463 (0,076)
Num Ind	daC-DQIGLGP	4,2 (1,4)	3,9 (1,5)	3,8 (2,0)	3,6 (1,7)	5,7 (2,0)
	dasC-DQIGLGP	3,8 (1,9)	4,4 (1,7)	4,1 (1,6)	3,4 (2,2)	6,0 (2,7)
	mdaC-DQIGLGP	3,4 (1,7)	4,0 (1,3)	3,9 (1,8)	3,4 (2,0)	6,3 (2,6)
	mdasC-DQIGLGP	3,2 (1,5)	4,0 (1,6)	3,7 (1,6)	3,1 (1,7)	6,0 (2,6)

Em relação à variável hipervolume, o teste de Wilcoxon não rejeitou a hipótese nula de que os modelos têm médias iguais, ou seja, resultam em frentes de Pareto ótima de qualidade equivalente sob esta métrica. A Figura 6.13 apresenta a evolução do hipervolume ao longo das gerações, onde se observa que as curvas têm comportamentos semelhantes e confirma, visualmente, o resultado apresentado no teste de Wilcoxon.

Em relação ao número de indivíduos, o teste de Wilcoxon identificou diferença entre os casos normalizados e não-normalizados em alguns cenários. Em geral, as versões normalizadas resultaram em números médios de indivíduos menores que suas versões não-normalizadas. Como o hipervolume foi equivalente em todos os casos, os resultados sugerem que as versões normalizadas

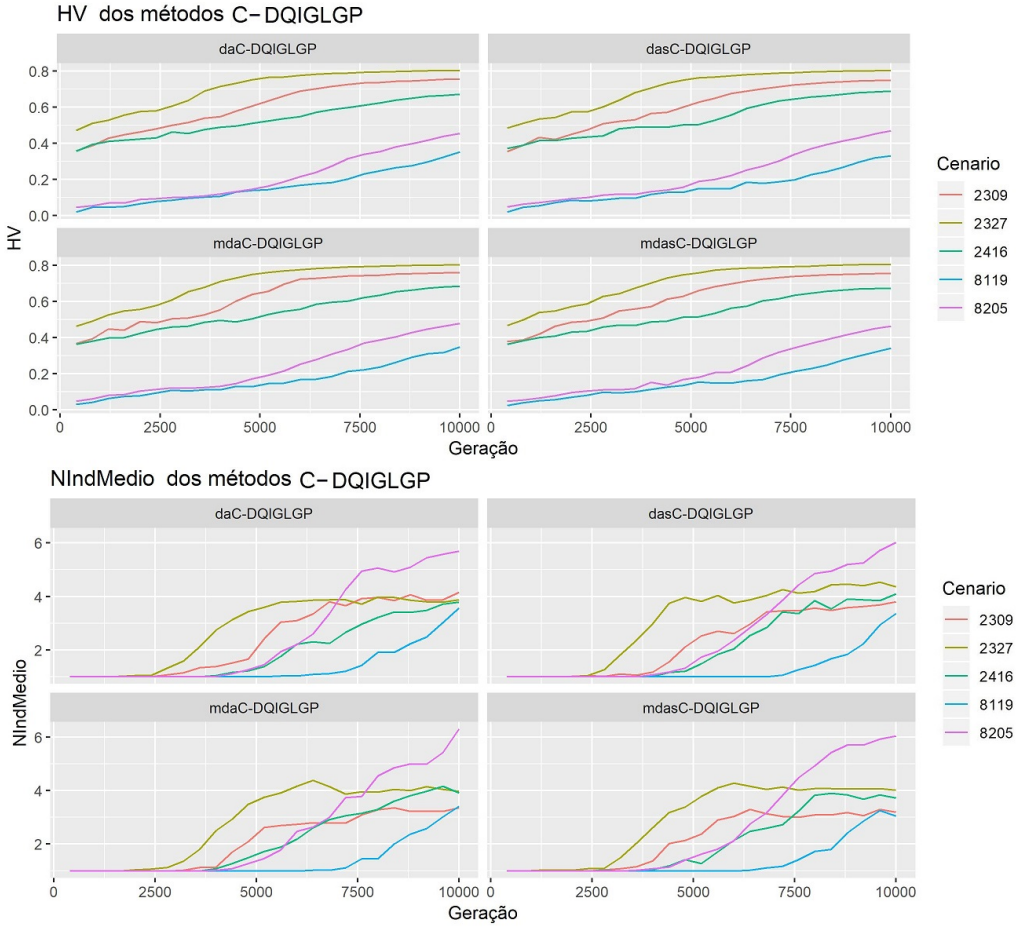

Figura 6.13: Evolução do hipervolume e número de indivíduos para os métodos propostos para o modelo C-DQIGLGP.

possuem indivíduos mais distintos entre si pois um menor número é capaz de produzir uma figura com o mesmo hipervolume.

Com estes resultados pode-se considerar que, assim como aconteceu para o algoritmo C-NSQIGLGP, a qualidade da população Pareto ótima não é degenerada em nenhuma das propostas, mas pela relação com o número de indivíduos, se considera uma vantagem para as versões normalizadas dos modelos. Também se avalia que o método dasC-DQIGLGP é sempre superado por algum de seus pares (daC-DQIGLGP ou mdasC-DQIGLGP) e, por isso, este modelo não é considerado na próxima seção que trata do impacto da população *archive* na evolução.

6.3.2.6 *Archive Constrained Decomposition* QIGLGP

Na Seção 5.2.2.2 foi proposto que a população *archive* que armazena os indivíduos não-dominados ao longo da evolução fosse utilizada para guiar a

atualização de indivíduos quânticos. Esta seção apresenta os resultados dos dois métodos avaliados para definir a probabilidade desta influência:

– probabilidade fixa (fAC-DQIGLGP): parâmetro do modelo, definido arbitrariamente em 10%.
– proporcional ao tamanho da população *archive* (tAC- DQIGLGP): dada pela equação 5-6.

Como os resultados apresentados na Seção 6.3.2.5 não convergiram de forma consistente a um único método, três deles são mantidos na avaliação da população *archive*, de forma a verificar se pode haver um efeito sinérgico positivo entre as atuações.

A Tabela 6.16 apresenta o percentual de corridas que resultaram em soluções aceitas para os métodos propostos. Os métodos combinados são identificados na tabela pelas letras minúsculas, por exemplo, fdaAC-DQIGLGP significa probabilidade fixa do método daC-DQIGLGP (desempate dos objetivos não-críticos na *archive* e sem normalização). Já fmdaAC-DQIGLGP se diferencia do anterior porque possui normalização. Finalmente, tmdaAC-DQIGLGP se diferencia do anterior porque substitui a probabilidade fixa pela proporcional ao tamanho da população *archive*.

Tabela 6.16: Percentual de soluções aceitas para os métodos propostos para o AC-DQIGLGP.

Métrica	Modelo	Cenário				
		2309	2327	2416	8119	8205
%CSA	fdaAC-DQIGLGP	90	86	74	66	100
	fmdaAC-DQIGLGP	92	78	82	70	98
	fmdasAC-DQIGLGP	100	100	92	60	96
	tdaAC-DQIGLGP	90	84	80	62	96
	tmdaAC-DQIGLGP	90	74	80	68	100
	tmdasAC-DQIGLGP	100	100	96	64	100

Uma comparação entre os dados das Tabelas 6.16 e 6.14 mostra que houve perda de desempenho em todos os cenários, nos três métodos (prefixos “daC”, “mdaC” e “mdasC”) seja com probabilidade de atualização fixa (prefixo “f”) ou variável com o tamanho da população (prefixo “t”). Este desempenho pode ser consequência de o indivíduo da população *archive* não ser uma boa solução para o subproblema ao qual foi aleatoriamente associado. Portanto, atualizar o indivíduo quântico na sua direção parece representar um retrocesso no processo evolutivo.

Este resultado em relação ao %CSA é suficiente para não recomendar nenhuma das propostas de utilização da população *archive*. Entretanto, para

manter a informação completa, a Tabela 6.17 apresenta o hipervolume e o número de indivíduos na frente de Pareto ótima. O teste de Wilcoxon destas variáveis mostrou diferença estatisticamente significante para algumas configurações. A Tabela 6.18 apresenta a quantidade de cenários em que essa diferença foi detectada para cada par de métodos. O nome dos métodos nas colunas está truncado apenas por questões de formatação, mas todos possuem o complemento "QIGLGP". De acordo com esta tabela, observa-se diferença principalmente dos métodos fmdasAC-DQIGLGP e tmdasAC-DQIGLGP em relação aos demais, o que significa dizer que o maior hipervolume conseguido por estes métodos para os cenários 2309 e 2327 é concreto, assim como a perda no valor do hipervolume para os demais cenários.

Tabela 6.17: Hipervolume e número de indivíduos para os métodos propostos para o AC-DQIGLGP.

Métrica	Modelo	Cenário				
		2309	2327	2416	8119	8205
HV	fdaAC-DQIGLGP	0,725	0,776	0,674	0,405	0,530
	fmdaAC-DQIGLGP	0,721	0,778	0,689	0,406	0,517
	fmdasAC-DQIGLGP	0,745	0,801	0,682	0,305	0,439
	tdaAC-DQIGLGP	0,723	0,767	0,683	0,402	0,531
	tmdaAC-DQIGLGP	0,717	0,767	0,682	0,381	0,545
	tmdasAC-DQIGLGP	0,771	0,811	0,678	0,332	0,425
Num Ind	fdaAC-DQIGLGP	3,7	3,5	3,8	4,0	4,9
	fmdaAC-DQIGLGP	2,8	3,4	2,9	3,9	5,2
	fmdasAC-DQIGLGP	3,7	4,1	3,8	2,3	4,9
	tdaAC-DQIGLGP	3,8	3,3	3,0	4,0	5,0
	tmdaAC-DQIGLGP	3,5	3,0	3,0	4,1	4,8
	tmdasAC-DQIGLGP	3,2	3,8	3,5	2,8	5,1

Tabela 6.18: Total de cenários em que foi identificada diferença estatisticamente significante entre os métodos.

	Hipervolume					
Modelo	fdaAC-D	fmdaAC-D	fmdasAC-D	tdaAC-D	tmdaAC-D	tmdasAC-D
fdaAC-DQIGLGP	0	0	3	0	0	4
fmdaAC-DQIGLGP	0	0	4	0	0	5
fmdasAC-DQIGLGP	3	4	0	4	4	1
tdaAC-DQIGLGP	0	0	4	0	0	4
tmdaAC-DQIGLGP	0	0	4	0	0	4
tmdasAC-DQIGLGP	4	5	1	4	4	0
Total	7	9	16	8	8	18
	Número de indivíduos					
Modelo	fdaAC-D	fmdaAC-D	fmdasAC-D	tdaAC-D	tmdaAC-D	tmdasAC-D
fdaAC-DQIGLGP	0	2	2	0	1	1
fmdaAC-DQIGLGP	2	0	4	1	0	2
fmdasAC-DQIGLGP	2	4	0	2	2	0
tdaAC-DQIGLGP	0	1	2	0	0	1
tmdaAC-DQIGLGP	1	0	2	0	0	2
tmdasAC-DQIGLGP	1	2	0	1	2	0
Total	6	9	10	4	5	6

Uma comparação entre os métodos utilizando informações das Tabelas 6.16 e 6.17 mostra que a proposta de probabilidade fixa supera a proporcional ao tamanho da população em sete casos, mas perde em outros cinco. A maior diferença não ultrapassa 4%. Em relação ao número de indivíduos, há empate pois em cinco casos a população final dos métodos de probabilidade fixa tem mais indivíduos do que o proporcional à população, mas em outros cinco casos, possui menos.

Com nenhuma dessas representações se observa benefício na proposição de um modelo AC-DQIGLGP e, por isso, ele não é recomendado.

6.3.3 Estratégia "Bilevel"

As seções 6.3.3.1 e 6.3.3.2 apresentam os resultados obtidos para as propostas baseadas em uma estrutura em que o processo evolutivo acontece em dois níveis de decisão.

6.3.3.1 *Bilevel Constrained* QIGLGP e C-NSQIGLGP

A primeira avaliação consiste em analisar o impacto de alternar a evolução entre considerar ou não as restrições de programação [88]. O problema Seguidor as ignora e o problema Líder as considera. Esta alternância segue por um número definido de ciclos. Para manter o número de gerações compatível com os demais modelos propostos nesta tese, o número de gerações total é distribuído entre os ciclos e os níveis. Foram arbitrados dez ciclos de execução e avaliadas algumas variações do número de gerações em cada nível. As avaliações consideram tanto a QIGLGP como o C-NSQIGLGP como algoritmos do núcleo evolutivo. O C-DQIGLGP não foi utilizado pois esta avaliação foi feita previamente à proposição do modelo e os resultados não sugeriram essa integração. O conjunto testado foi:

- BiQGP-SC55: representa dez ciclos em que a QIGLGP é executada no Líder e no Seguidor. A evolução ocorre através de 500 gerações no Seguidor e outras 500 gerações no Líder em cada ciclo;
- BiQGP-SC37: a diferença para o BiQGP-SC55 é que são executadas 300 gerações no Seguidor e 700 no Líder. A redução no Seguidor é devida ao fato de que a versão irrestrita do problema é mais fácil, então poderia ser resolvida mais rapidamente;
- BiCNSQGP-SC55: semelhante ao BiQGP-SC55, substituindo o QIGLGP pelo C-NSQIGLGP no núcleo evolutivo do Líder e do Seguidor.

– BiCNSQGP-SC37: semelhante ao BiQGP-SC37, mas substituindo o QIGLGP pelo C-NSQIGLGP no núcleo evolutivo do Líder e do Seguidor.

A Tabela 6.19 apresenta o percentual de corridas que resultaram em soluções de *schedule* viáveis para as quatro propostas dos cenários 2309, 2327 e 2416. Há resultados apenas para os três primeiros modelos porque, como será demonstrado, esta abordagem teve desempenho pior do que o obtido pelos algoritmos multiobjetivo de apenas um nível e, dessa forma, esta linha de pesquisa não foi continuada. Para facilitar a visualização, são repetidos os resultados obtidos nos métodos QIGLGP e C-NSQIGLGP em apenas um nível de decisão.

Tabela 6.19: Percentual de corrida que resultaram em soluções aceitas para as propostas *bilevel* das restrições

Métrica	Modelo	Cenário		
		2309	2327	2416
%CSA	BiQGP-SC55	22	92	4
	BiQGP-SC37	40	98	4
	BiCNSQGP-SC55	44	98	0
	BiCNSQGP-SC37	58	98	6
	C-NSQIGLGP	98	100	78
	QIGLGP	76	98	36

Uma comparação dos dados desta tabela mostra perda de desempenho em qualquer configuração da abordagem *bilevel* com restrições em todos os cenários. Esta perda se agrava quanto mais desafiador é o cenário. Ou seja, o cenário 2327 se mantém acima de 90% CSA, mas o cenário 2309 não ultrapassa os 58% e o 2316, apenas 6%. O algoritmo C-NSQIGLGP alcança 98% e 78% nestes casos.

Analisando somente os métodos *bilevel* com restrição, pode-se observar que as versões SC37 (menor número de gerações no Seguidor - sem restrições - do que no Líder) têm desempenho melhor do que seus pares SC55. Este comportamento pode estar associado a uma evolução insuficiente do Líder nos pares SC55 devido ao baixo número de gerações. Outra observação que pode ser feita é que os métodos que utilizam o algoritmo C-NSQIGLGP nos dois níveis do processo evolutivo têm perda menor do que seus pares que utilizam o QIGLGP. A exceção é apenas o cenário 2416 com o método BiCNSQGP-SC55. Este resultado sugere que a abordagem multiobjetivo por dominância de Pareto e com violação traz benefício sobre a priorização de objetivos.

Estes resultados sinalizam que a abordagem por otimização em dois níveis de decisão, separados pela consideração de restrições, não é uma boa estratégia. Dois questionamentos podem ser feitos a partir daí. O primeiro é

se há vantagem numa estrutura *bilevel* e o segundo é se, por exemplo, uma estratégia de divisão dos problemas baseada na forma de tratar os objetivos não seria mais adequada ao processo decisório.

Para avaliar o benefício de uma estrutura em dois níveis, são propostas duas variações sobre as versões com o núcleo QIGLGP: apenas um ciclo evolutivo e mudança na relação tamanho da população e número de gerações.

– BiQGP-L1SC37: apenas um ciclo Seguidor-Líder onde o Seguidor evolui por 3.000 gerações e o Líder por 7.000;
– BiQGP-SC614I28: são mantidos os dez ciclos Seguidor-Líder, mas o número de gerações em cada ciclo é dobrado (600 no Seguidor e 1.400 no Líder) para dar mais oportunidade ao processo evolutivo. De forma a manter o número total de avaliações, a população é reduzida para 28 indivíduos.

A Tabela 6.20 apresenta os resultados destes dois modelos. Uma comparação com os dados da Tabela 6.19 mostra que BiQGP-L1SC37 tem desempenho ligeiramente pior do que QIGLGP no cenário 2327 e significativamente superior nos cenários 2309 e 2416, mais complexos. Este modelo é a versão *bilevel* que permite uma boa evolução em cada nível e está isolada do efeito da quantidade de ciclos. Por isso, seu desempenho melhor do que o QIGLGP sugere que há benefício numa estrutura em dois níveis de evolução.

Tabela 6.20: Percentual de corridas que resultaram em soluções aceitas para variações da estrutura *bilevel*.

Métrica	Modelo	Cenário		
		2309	2327	2416
%CSA	BiQGP-L1SC37	90	96	46
	BiQGP-SC614I28	98	100	64

Ainda na Tabela 6.20 se observa que o método BiQGP-SC614I28 apresenta melhores resultados até mesmo do que o BiQGP-L1SC37, sugerindo que os ciclos Seguidor-Líder podem ser mantidos desde que se garanta um número de gerações suficiente para a evolução. A menor população destes métodos não compromete seu desempenho, o que é coerente com outros trabalhos onde a programação genética com inspiração quântica apresentou bom desempenho com populações pequenas [81].

Os resultados desta seção permitiram compreender que pode haver benefício em uma estrutura *bilevel* para resolver problemas de programação de produção desde que cada nível hierárquico consiga evoluir adequadamente. Eles também mostraram que a integração entre os dois níveis de resolução com um

algoritmo multiobjetivo é benéfica quando comparada ao uso de um algoritmo hierárquico. Entretanto, os resultados obtidos nesta seção apresentam desempenho inferior aos já apresentados pelos modelos mulitobjetivo em um nível. Por isso seus resultados não são explorados além da métrica de %CSA.

Acredita-se que as avaliações desta seção cumpram o seu papel de contribuir com o entendimento sobre a evolução *bilevel*, ainda que os métodos utilizados aqui sejam, em sua maioria, baseados em priorização. A próxima seção trata do segundo questionamento formulado anteriormente sobre a proposição de uma estrutura *bilevel* que divida os problemas com base nos objetivos.

6.3.3.2 *Bilevel Objective* C-DQIGLGP

Nesta seção são apresentados os resultados de um modelo com uma estrutura em dois níveis onde o Seguidor evolui considerando apenas os objetivos críticos e o Líder considera todos os objetivos. O algoritmo evolutivo nos dois níveis é o C-DQIGLGP, com normalização dos objetivos. Os resultados apresentados na Tabela 6.21 sobre o %CSA mostram desempenho semelhante ao algoritmo com um nível. Apenas dois cenários não atingem 100%CSA, o 2327 com 98% e o cenário 8119, com 76%, que repete o melhor resultado obtido até então.

Tabela 6.21: %CSA, hipervolume e número de indivíduos na frente de Pareto ótima do modelo BiO-CDQIGLGP.

Métrica	Modelo	Cenário				
		2309	2327	2416	8119	8205
%CSA	BiOC-DQIGLGP	100	100	98	76	100
HV	BiOC-DQIGLGP	0,746	0,782	0,667	0,365	0,489
Num Ind	BiOC-DQIGLGP	3,5	3,6	3,9	5,0	6,5

A Figura 6.14 apresenta a evolução do hipervolume e o número médio de indivíduos da frente de Pareto ótima ao longo das gerações. O primeiro quarto de gerações corresponde somente aos indivíduos do nível Seguidor. De 2.500 a 10.000 gerações corresponde às 7.500 gerações definidas para o nível Líder e, por isso, somente seus indivíduos são uilizados. Uma comparação com as curvas das propostas C-DQIGLGP (Figura 6.13) mostra que no caso *bilevel* as duas medidas possuem valores maiores no primeiro quarto da evolução do que seus pares do modelo de um nível. Esse dado sugere que o problema Líder recebe mais de um indivíduo que representa um *schedule* válido no início de sua execução, o que poderia resultar em desempenho melhor ao final do processo. No entanto, os dados da Tabela 6.21 não sugerem isso.

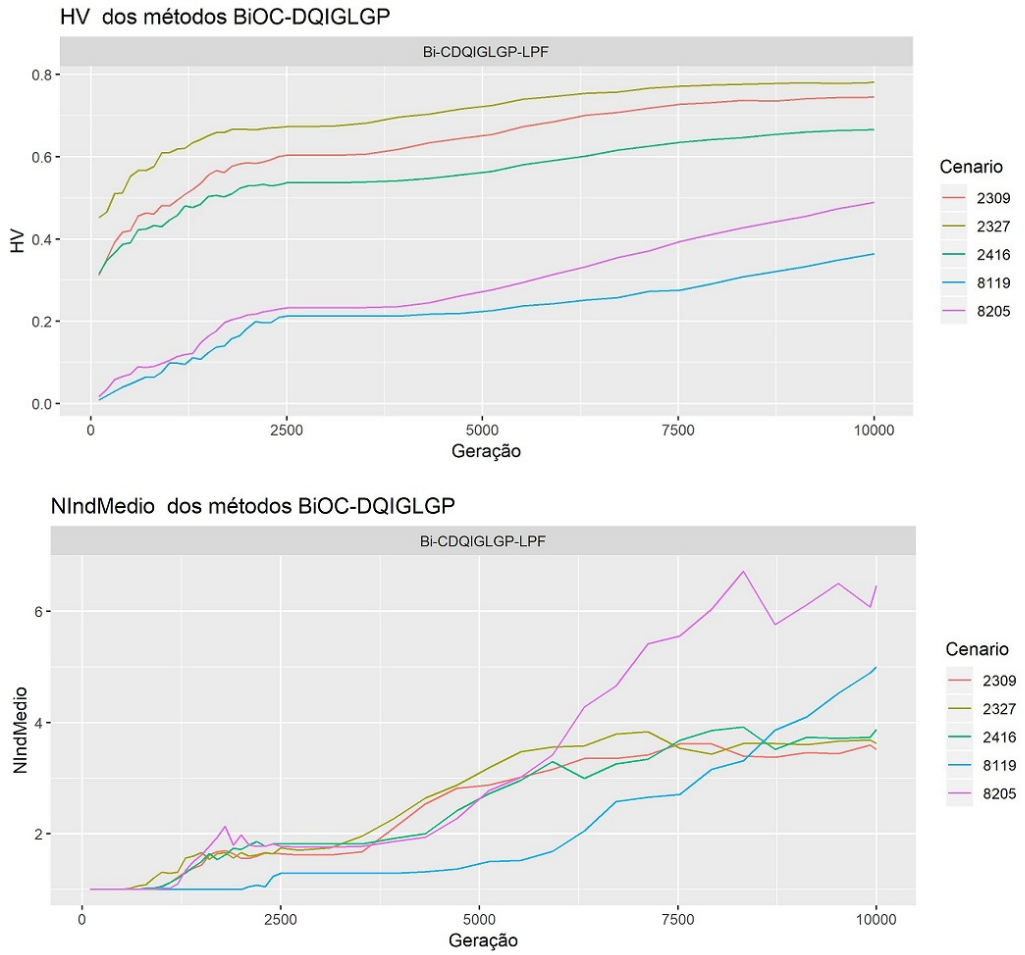

Figura 6.14: Evolução dos hipervolume e número de indivíduos para o método BiO-CDQIGLGP proposto.

Uma comparação dos dados das Tabelas 6.21 e 6.15 mostra uma pequena diferença entre os hipervolumes dos dois métodos em todos os cenários. Há uma inversão de tendência pois nos primeiros três cenários há vantagem para o modelo em um nível e nos dois últimos cenários é o modelo em dois níveis que tem maior hipervolume. Em relação ao número de indivíduos, a diferença mais significativa é no cenário desafiador que sobe de 3,4 para 5,0 indivíduos em média.

Os resultados desta seção mostram que a estrutura *bilevel* é capaz de atingir desempenho competitivo com os algoritmos multiobjetivo. A Seção 6.3.3.3 se propõe a uma avaliação comparativa do melhor resultado de cada estratégia.

6.3.3.3 Comparação dos melhores algoritmos de cada abordagem

Os resultados apresentados até o momento mostraram que há, pelo menos, uma proposta mais adequada do que o caso base (QIGLGP) para tratar o problema de programação de petróleo em cada estratégia avaliada. A estratégia E-MO, de algoritmos multiobjetivo, teve seu melhor desempenho com o princípio de decomposição do problema em subproblemas, que também apresentou melhor resultado na estrutura em dois níveis de decisão, da estratégia E-Bi. A melhor proposta de algoritmo multiobjetivo baseado em dominância também é utilizada nesta comparação. Dessa forma, esta seção recapitula alguns resultados já apresentados com o objetivo de comparar os modelos:

- QIGLGP: possui múltiplos objetivos, mas lida com eles de forma hierarquizada. A otimização do próximo objetivo só acontece quando os anteriores possuem valores mais baixos.
- C-NSQIGLGP: multiobjetivo baseado em dominância de Pareto. Na ordenação da população, durante o processo evolutivo, utiliza a Equação 5-5 para medir violação aos critérios de atendimento de uma solução de *schedule* como forma de diferenciar os objetivos. Quando há violação e empate entre dois indivíduos nesta métrica, os objetivos não-críticos são usados para avaliar a dominância entre eles.
- AC-NSQIGLGP: se diferencia do C-NSQIGLGP pela utilização de uma população externa para influenciar na atualização da população quântica quando a população clássica contém mais de um indivíduo na mesma frente de Pareto associado ao mesmo ponto de referência.
- C-DQIGLGP: multiobjetivo baseado em decomposição do problema em subproblemas usando a função de Tchebycheff, apresentada na Equação 2-5. Estão presentes as duas versões normalizadas mdaC-DQIGLGP e mdasC-DQIGLGP.
- BiOC-DQIGLGP: *bilevel* que executa um modelo C-DQIGLGP em cada nível. O Seguidor evolui apenas com os objetivos críticos e o Líder considera todos. Todos os indivíduos da população *archive* do Seguidor migram, aleatoriamente, para subproblemas do Líder.

A Tabela 6.22 mostra o %CSA, número de indivíduos e hipervolume médios da frente de Pareto ótima para estes métodos. Para a variável hipervolume o teste de Wilcoxon mostrou que a diferença entre os modelos baseados em dominância não tem significância estatística, mas eles diferem dos modelos baseados em decomposição, seja em um ou dois níveis de decisão, em quatro

dos cinco cenários. Os modelos baseados em decomposição também não possuem diferença estatística significante entre si, mas diferem do modelo em dois níveis de decisão no cenário 8119. Em relação ao número de indivíduos, o diagnóstico é equivalente. A principal diferença está no cenário 2309 onde não se observou diferença significativa entre os métodos baseados em decomposição e os baseados em dominância.

Tabela 6.22: %CSA, hipervolume e número de indivíduos comparativo para os melhores métodos propostos.

Métrica	Modelo	Cenário				
		2309	2327	2416	8119	8205
%CSA	C-NSQIGLGP	98	100	78	20	66
	fprpAC-NSQIGLGP	96	100	76	20	70
	mdaC-DQIGLGP	100	98	96	76	100
	mdasC-DQIGLGP	100	100	100	72	100
	BiOC-DQIGLGP	100	96	100	76	100
	QIGLGP	76	98	36	4	62
Num Ind	C-NSQIGLGP	3,73	5,18	2,69	2,40	3,73
	fprpAC-NSQIGLGP	3,64	4,88	3,14	3,22	4,29
	mdaC-DQIGLGP	3,36	3,98	3,92	3,42	6,30
	mdasC-DQIGLGP	3,20	4,02	3,72	3,06	6,04
	BiOC-DQIGLGP	3,52	3,63	3,88	5,00	6,46
	QIGLGP	1,00	1,00	1,00	1,00	1,00
HV	C-NSQIGLGP	0,7371	0,7946	0,6175	0,2736	0,2985
	fprpAC-NSQIGLGP	0,7393	0,7902	0,6101	0,2763	0,2977
	mdaC-DQIGLGP	0,7631	0,8059	0,6918	0,3541	0,4859
	mdasC-DQIGLGP	0,7621	0,8098	0,6811	0,3492	0,4709
	BiOC-DQIGLGP	0,7618	0,7951	0,6859	0,3908	0,5138
	QIGLGP	0,6747	0,7481	0,6060	0,2611	0,2627

A Figura 6.15 apresenta a evolução do hipervolume dos cenários nos cinco modelos considerados. Algumas observações podem ser feitas com esta visão. A primeira é que o modelo *bilevel*, em qualquer cenário, inicia com maior patamar de hipervolume, pois há um impulso inicial na geração de indivíduos que atendem aos objetivos críticos. No entanto, em três cenários essa característica não se reverteu em benefício para a população final. No cenário mais desafiador há uma vantagem, confirmada pelo teste de Wilcoxon. No cenário 2327 as curvas de todos os modelos se aproximam em torno da geração 6.000. Este comportamento é coerente com o julgamento de que este é um cenário mais fácil e todos os métodos o resolvem. Nos cenários 2416, 8119 e 8205 se observa que a evolução vai diferenciando cada vez mais o desempenho dos métodos por dominância e por decomposição. Os dois primeiros, pelo %CSA se mostraram mais desafiadores, o que sugere que os métodos por decomposição são mais adequados para resolver programações mais complexas.

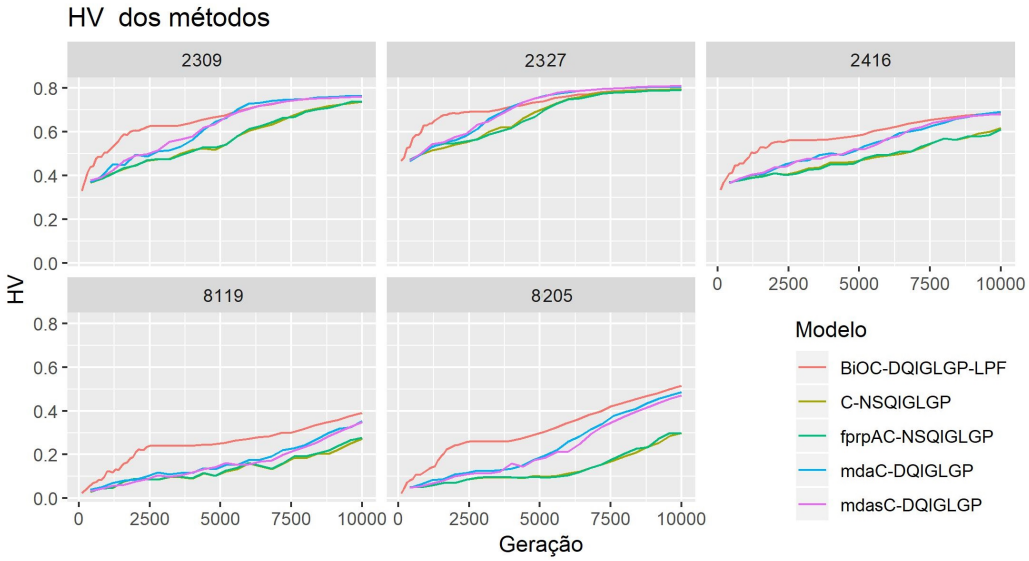

Figura 6.15: Evolução do hipervolume nos cinco cenários para os cinco modelos.

Combinando as informações da Figura 6.15 e da Tabela 6.22 se avalia que os métodos mdaC-DQIGLGP e BiOC-DQIGLGP são os mais robustos para resolver cenários de programação pois eles apresentam maior desempenho na maioria dos casos e, quando isso não acontece, ainda mantêm um patamar satisfatório de resultados. Dessa forma, valores médios dos objetivos desses dois métodos são comparados com os valores dos cenários de programação reais propostos na refinaria.

A Figura 6.16 apresenta a distribuição de todas as soluções em diferentes faixas de desvio de UDA para cada cenário em cada modelo com melhor desempenho. Por exemplo, é possível observar que, dentre todas as soluções finais geradas por ambos modelos, em 100% dos casos, o objetivo de UDA teve desvio inferior a 2% para o cenário 2309. No entanto, no cenário 8119, cerca de 2,4% de todas soluções deste cenário (considerando as múltiplas soluções encontradas em cada corrida) tiveram desvio entre 2% e 4% (barra laranja), 0,5% entre 4% e 6% (barra amarela) etc. Por este critério, o modelo C-DQIGLGP apresenta desvios menores do que o *bilevel* nos cenários 2327 e 8119, o que pode significar que ele estaria mais próximo da convergência caso fosse permitido mais gerações para evolução.

De forma análoga, a Figura 6.17 apresenta a distribuição de todas as soluções em diferentes faixas de desvio no objetivo do navio para cada cenário em cada modelo com melhor desempenho. Por exemplo, pode-se observar que os cenários 2309, 2327, 2416 e 8205 tiveram suas programações de navios sempre abaixo da tolerância de uma hora de atraso. Este critério não desqualificaria soluções nestes cenários. Já o cenário 8119 apresenta dificuldade

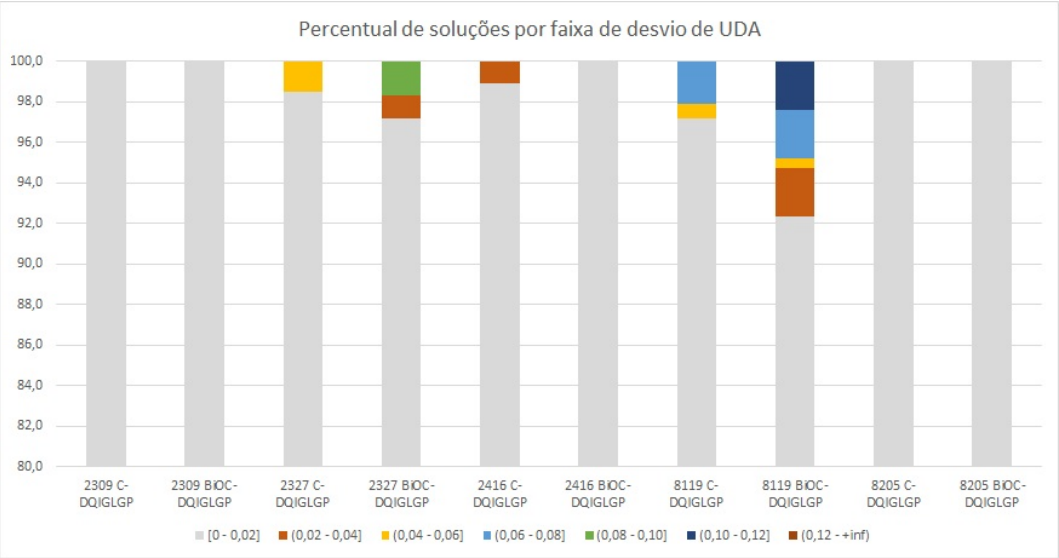

Figura 6.16: Percentual do total de soluções por faixa no objetivo de UDA

na programação de navios. As barras cinza, amarela e azul clara indicam faixas de valores até 1.000 para este objetivo, onde estão totalizadas as horas de atraso. Nenhuma solução está na faixa de milhares (barras verde e azul escuro), o que significa que em todos os casos, ao fim do cenário, não havia petróleo por ser descarregado em nenhum navio. Para este objetivo e baseado apenas neste cenário, o modelo C-DQIGLGP apresentou maior dificuldade de encontrar boas soluções do que a versão *bilevel*.

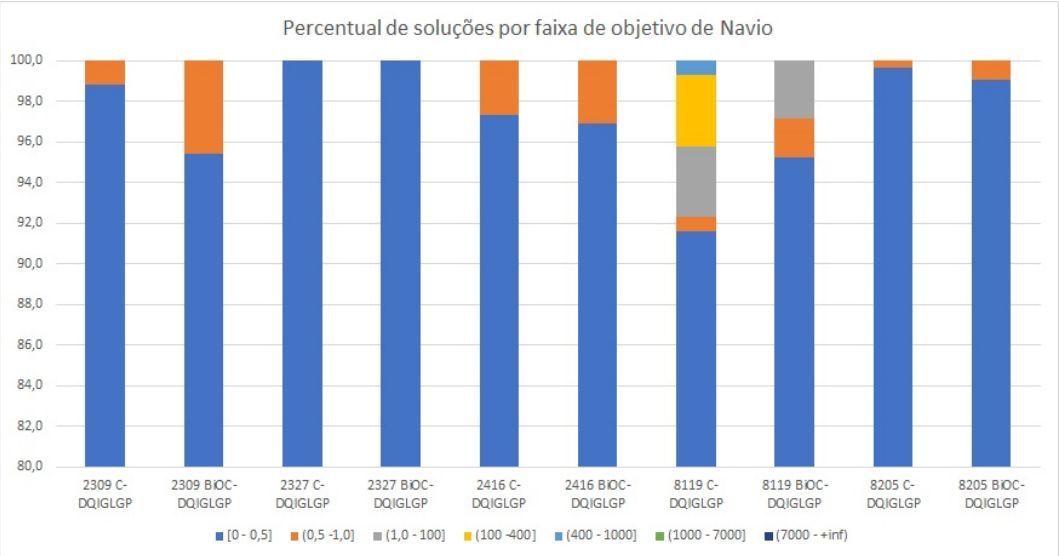

Figura 6.17: Percentual do total de soluções por faixa no objetivo de navio

A Tabela 6.23 apresenta os valores dos objetivos nos cenários do programador e nos modelos C-DQIGLGP (mdaC-DQIGLGP) e BiOC-DQIGLGP.

Pelos dados apresentados se observa que, em média, há uma perda de produção nas UDAs, mas que este valor é inferior a 0,8%, menos de 40% da tolerância aceita com base nos dados experimentais apresentados na Seção 5.1.5. A solução da refinaria, claro, não programa este desvio, assim como também não programa atraso no descarregamento de navios. Nos modelos, desvios deste objetivo estão na segunda casa decimal. Em relação ao tempo de parada do oleoduto, se observa que, em dois dos cinco cenários, os modelos movimentaram o duto por mais tempo. No cenário 2309 o programador conseguiu uma solução de operar o duto de forma contínua. As movimentações estão todas encadeadas até que ele já não programa nenhuma. Os modelos não encontraram esta solução. Em relação ao número de trocas de tanques, os modelos propostos fazem mais do que o programador especialmente nos cenários maiores, 8119 e 8205. Este comportamento deve estar relacionado com a maior operação dos dutos nestes cenários. Em relação ao não-atendimento das restrições de qualidade, se observa que o cenário 8119, com alta concentração de óleo ácido, se mostrou muito desafiador até mesmo para a experiência do programador que, sem tempo hábil para gerar muitas soluções, operou acima dos limites de qualidade por mais de 200 horas. A principal propriedade violada foi a acidez dos cortes de diesel leve e pesado. A consequência deste cenário é ter tornado mais difícil a programação das demais unidades de processo e tanques de produtos intermediários da refinaria, provavelmente demandando mais misturas para o processamento dessas frações.

Com base nos valores médios, pode-se assumir que as soluções propostas pelos modelos são compatíveis com as geradas pelo programador de produção. Mais especificamente, o cenário 8119 pode ser resolvido sem violar restrições em cerca de 70% das corridas, soluções estas que o programador não teve tempo de encontrar. O tempo de execução dos modelos é equivalente ao que o programador leva para propor o *schedule* de um cenário, sendo cerca de 40 minutos para os cenários menores e 80 minutos para os de, aproximadamente, 20 dias.

A Figura 6.18 apresenta as soluções da frente de Pareto ótima geradas pelo modelo C-DQIGLGP para o cenário 2309 e a Figura 6.19 apresenta as soluções da frente de Pareto ótima gerada pelo modelo BiOC-DQIGLGP para este mesmo cenário.

Nestas figuras, o eixo das abcissas é o horizonte de programação, o eixo das ordenadas apresenta os equipamentos do cenário, as barras são as atividades do equipamento na linha e a cor da barra indica o tipo de atividade. A laranja representa que o equipamento está enviando para algum destino, a verde clara que o equipamento está recebendo de uma origem e a verde escura

Tabela 6.23: Média e desvio padrão dos objetivos comparados à solução da refinaria.

Objetivo	Modelo	Cenário 2309	2327	2416	8119	8205
Desvio carga UDAs (%)	C-DQIGLGP	0,56 (0,43)	0,59 (0,34)	0,55 (0,37)	0,70 (0,60)	0,41 (0,24)
	BiOC-DQIGLGP	0,43 (0,41)	0,60 (0,44)	0,53 (0,39)	0,71 (0,49)	0,46 (0,28)
	programador	0	0	0	0	0
Desvio descarga Navios (h)	C-DQIGLGP	0,02 (0,06)	0,01 (0,04)	0,04 (0,11)	0,01 (0,04)	0,00 (0,03)
	BiOC-DQIGLGP	0,03 (0,08)	0,00 (0,01)	0,02 (0,06)	0,02 (0,07)	0,01 (0,04)
	programador	0	0	0	0	0
Parada de oleoduto (h)	C-DQIGLGP	16,65 (14,19)	15,48 (8,32)	37,59 (14,04)	149,50 (24,83)	120,64 (30,35)
	BiOC-DQIGLGP	15,23 (16,09)	18,39 (11,47)	33,66 (10,45)	135,21 (24,65)	110,63 (28,21)
	programador	0	21	32,5	164,7	179,0
Troca de Tanque	C-DQIGLGP	17,7 (2,0)	14,1 (1,3)	21,1 (2,0)	35,2 (4,0)	30,2 (3,4)
	BiOC-DQIGLGP	18,0 (2,5)	14,5 (1,6)	21,1 (2,2)	33,6 (4,2)	28,7 (3,1)
	programador	17	14	22	29	23
Violação de restrições (h)	C-DQIGLGP	0	0	0	0	0
	BiOC-DQIGLGP	0	0	0	0	0
	programador	0	0	0	219,4	20

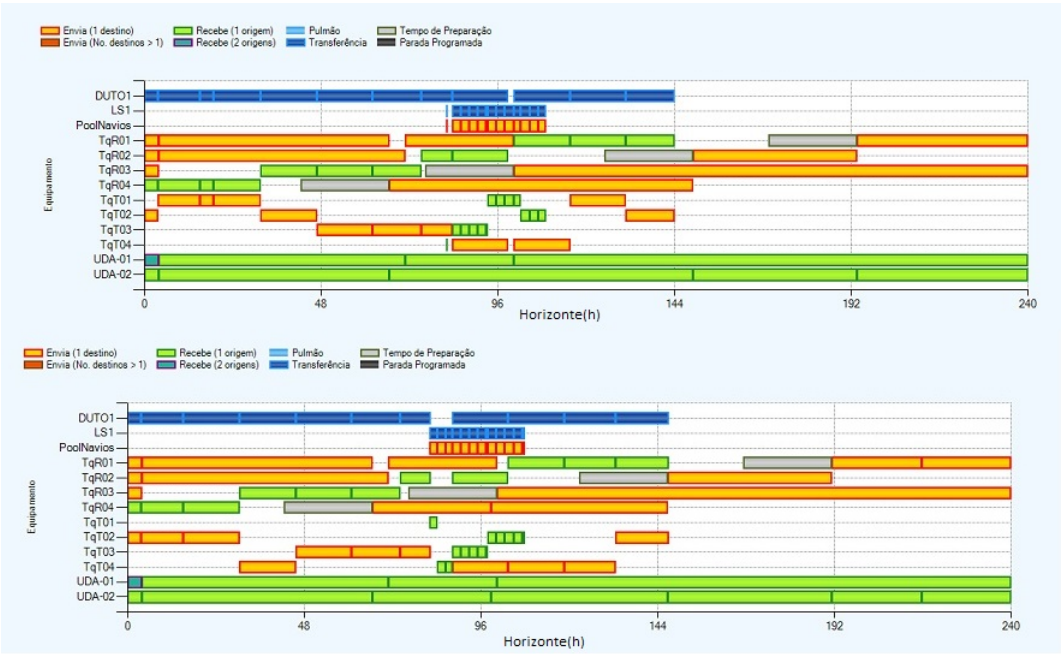

Figura 6.18: Soluções propostas para o cenário 2309 pelo modelo C-DQIGLGP.

Figura 6.19: Soluções propostas para o cenário 2309 pelo modelo BiOC-DQIGLGP.

indica duas origens. Por exemplo, UDA-01 começa com uma pequena atividade em que está recebendo de duas origens, no caso, os tanques TqR-01 e TqR-02. Na sequência, a carga é assumida pelo TqR02. A barra cinza representa o tempo mínimo de preparação dos tanques antes de ser carga de uma unidade de destilação e, por último, a barra azul mostra as movimentações de duto.

A Figura 6.18 mostra que o modelo C-DQIGLGP gerou duas soluções na frente de Pareto ótima. A programação das movimentações da refinaria é bastante semelhante, já a programação do terminal tem decisões diferentes. Por exemplo, na primeira solução TqT-01 bombeia para o oleoduto, abrindo espaço para participar do descarregamento do navio. Na segunda solução, TqT-01 recebe um volume muito menor do navio. TqT-04 e TqT-02 também tem participações diferentes no descarregamento do navio nas duas soluções. TqT-04 tem mais programações de bombeio no segundo cenário também. A Figura 6.19 mostra que o modelo BiOC-DQIGLGP resulta em três soluções na

frente de Pareto ótima e, entre elas, a programação das UDAs é diferente. Por exemplo, na primeira solução, cinco tanques compõem a sequência de carga da UDA-01. Número este que reduz para quatro tanques na segunda solução e sobe para seis tanques na terceira. Também há maior diferença nas movimentações do oleoduto, como pode ser visto pela barra azul do DUTO1.

A Figura 6.20 apresenta as soluções da frente de Pareto ótima geradas pelo modelo C-DQIGLGP para o cenário 8119 e a Figura 6.21 apresenta as soluções da frente de Pareto ótima gerada pelo modelo BiOC-DQIGLGP para este mesmo cenário. Pelas figuras, pode-se observar que ambos os modelos geraram duas soluções na frente de Pareto ótima de uma corrida deste cenário.

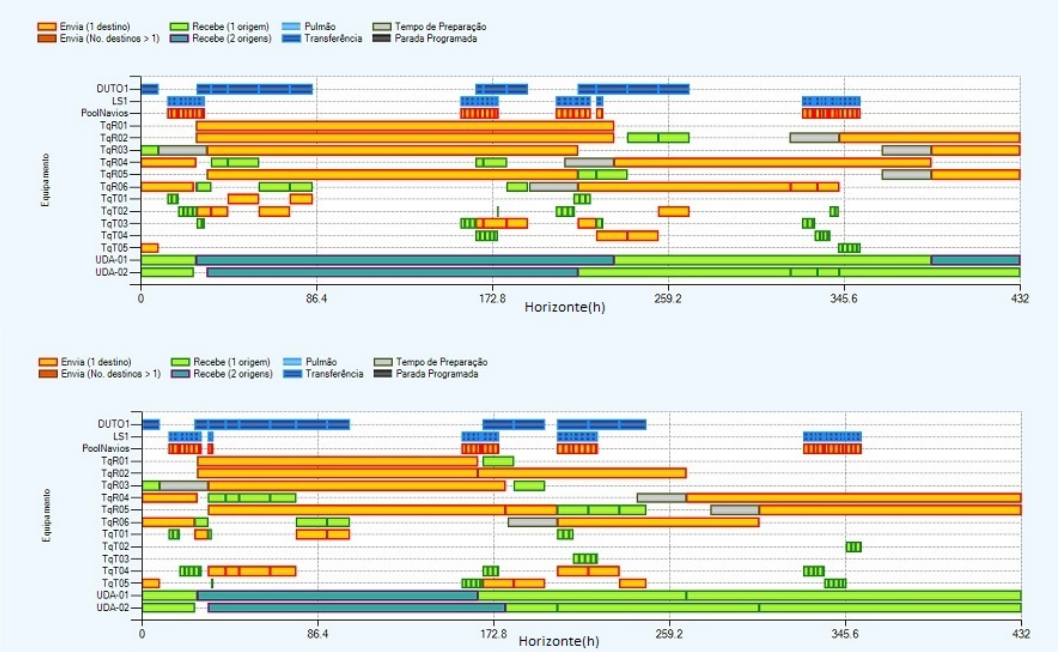

Figura 6.20: Soluções propostas para o cenário 8119 pelo modelo C-DQIGLGP.

Ambas as soluções de cada modelo são diferentes entre si em todos os tipos de atividades, seja carga de unidades, descarregamento de navios e movimentação do oleoduto. Neste exemplo, o modelo BiOC-DQIGLGP, em média, têm mais movimentações de oleoduto e, consequentemente, mais trocas de tanque. Já as alternativas do modelo C-DQIGLGP representam uma programação mais suave. Todas essas soluções poderiam ser apresentadas ao programador para sua avaliação e decisão posterior.

Os resultados apresentados nesta tese indicam que os modelos multiobjetivo baseados em decomposição são os mais adequados para o problema de programação de petróleo. Os modelos BiOC-DQIGLGP e C-DQIGLGP podem gerar soluções diferentes entre si, mas ambos se mostraram capazes de oferecer soluções de *schedule* otimizadas e viáveis para o programador.

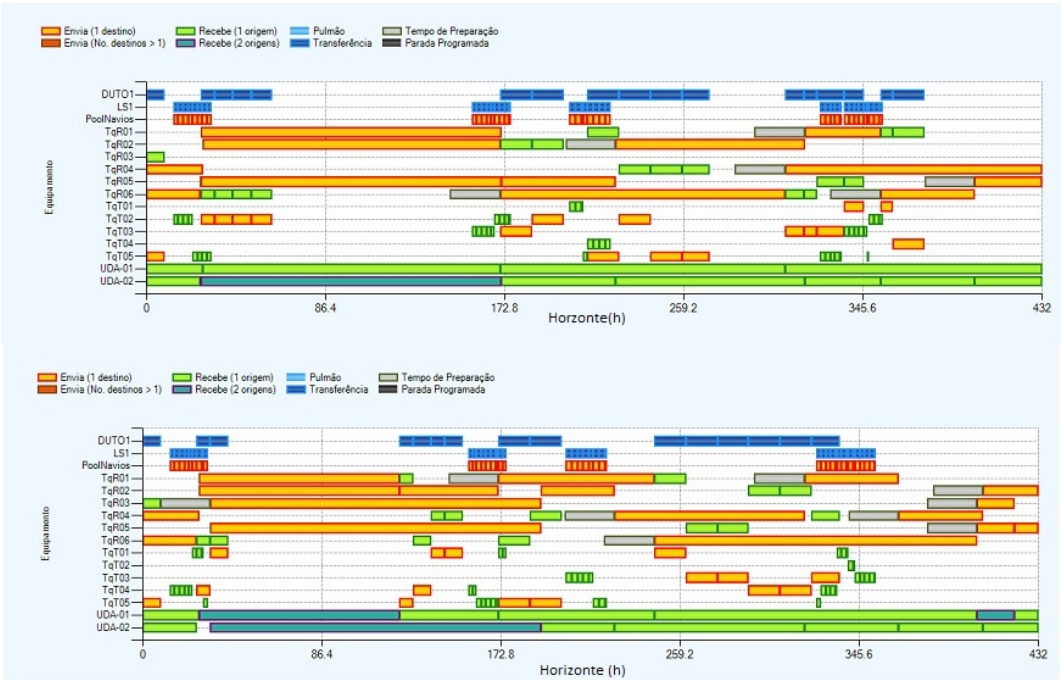

Figura 6.21: Soluções propostas para o cenário 8119 pelo modelo BiOC-DQIGLGP.

7
Conclusão e Trabalhos Futuros

Este trabalho apresentou propostas de novos algoritmos de otimização baseados em computação evolucionária multiobjetivo e programação genética com inspiração quântica para resolver problemas de programação de petróleo em refinaria, nas quais os objetivos pertencem a dois níveis diferentes de importância. Os modelos desenvolvidos são aplicáveis a diferentes topologias de planta, restrições de processo e condições iniciais da programação. Os resultados obtidos mostram que as soluções geradas pelas melhores propostas atendem aos critérios de qualidade que definem uma programação viável no ambiente operacional mesmo em condições de média e alta complexidade.

Os modelos foram baseados essencialmente em duas linhas de pesquisa: algoritmos evolucionários multiobjetivo (estratégia E-MO) e algoritmo *bilevel* com um núcleo evolucionário em cada nível (estratégia E-Bi). Para avaliação e comparação dos modelos foram adotadas duas métricas: percentual de corridas que geraram soluções aceitas (%CSA) e o hipervolume. A primeira trata da capacidade do modelo de gerar *schedules* viáveis e a segunda trata da capacidade de evoluir suas soluções e conferir diversidade à frente de Pareto ótima. Os modelos foram testados em cinco cenários de uma refinaria real definidos de forma a trazer diferentes desafios para os algoritmos.

A estratégia E-MO se desdobrou em duas propostas: ordenação dos indivíduos por dominância de Pareto (família de modelos NSQIGLGP) e por decomposição do problema em subproblemas (família de modelos DQIGLGP). Ambas as propostas foram avaliadas considerando, ou não, uma métrica de Violação (Equação 5-5) usada para diferenciar a importância dos objetivos "desvio da carga programada" e "desvio na programação de navios", associados à manutenção do processamento na refinaria, em relação aos outros dois objetivos "operação do oleoduto" e "troca de tanque", associados a uma operação mais suave. Ambas as propostas também foram avaliadas permitindo a influência de uma população externa no processo evolutivo, através de uma lógica em que o indivíduo da população externa orienta a atualização da distribuição de probabilidades dos indivíduos quânticos da geração.

Para a família de modelos NSQIGLGP, os resultados apresentados mostram que é necessário utilizar a métrica de Violação ou, em caso contrário, a

evolução simultânea dos objetivos beneficia os não-críticos a um ponto em que não garante os critérios para considerar a solução final um *schedule* viável. O impacto de permitir que uma população *archive* externa interfira na evolução não mostrou benefício na maioria dos casos. Existe apenas uma exceção que é o %CSA do cenário 8205, com a modelagem que tem menor probabilidade de que a interferência aconteça. Uma comparação dos resultados desta família de modelos conclui que o melhor para representar a abordagem por dominância de Pareto é o que utiliza apenas a equação de Violação para ordenação dos indivíduos e, em caso de empate no valor da Violação, desempata aplicando o princípio de dominância sobre os objetivos não-críticos. Este modelo, denominado C-NSQIGLGP, tem desempenho superior ao QIGLGP em todos os cenários na métrica %CSA e é capaz de gerar entre duas e cinco soluções na frente de Pareto ótima.

Para a família de modelos DQIGLGP, os resultados mostram que esta abordagem supera o QIGLGP e o NSQIGLGP na métrica %CSA, mesmo antes de considerar a medida de Violação para atualização da população externa. Esta população é a que armazena os melhores indivíduos e representa a frente de Pareto ótima do modelo. Quando a questão da violação é tratada, gerando o modelo C-DQIGLGP, os resultados mostram que quatro dos cinco cenários atingem patamares acima de 96%CSA e, mesmo o cenário 8119, salta para mais de 62%. A normalização dos objetivos só provocou alteração significativa na resposta do modelo no cenário mais complexo, melhorando seu desempenho. Também se observa que, diferente do que aconteceu com o C-NSQIGLGP, neste caso não há vantagem em adotar uma alternativa para desempate da Violação dentro do processo evolutivo (na comparação para o melhor indivíduo para um dado subproblema). Em relação à proposta de utilizar a população externa para influenciar na evolução, houve perda de desempenho nos dois métodos dessa avaliação. Este comportamento pode estar associado ao fato de que os indivíduos da população externa não necessariamente são bons para resolver o subproblema para o qual foi selecionado. Estratégias diferentes, que levem esse aspecto em consideração, podem ser propostas como trabalhos futuros.

A estratégia E-Bi se desdobra em duas propostas para diferenciar os níveis Seguidor e Líder: considerar ou não as restrições do problema; e considerar somente objetivos críticos e depois todos. O primeiro conjunto se desdobra em outras duas linhas, as famílias de modelos Bi-QIGLGP e BiC-NSQIGLGP, que utilizam como algoritmo evolutivo base o QIGLGP e o C-NSQIGLGP, respectivamente. As primeiras avaliações consideram 10 ciclos de execução do par Seguidor-Lider e o número de gerações é dividido entre os níveis e número de ciclos de forma que se mantenham totalizando 10.000. É

avaliada uma variação entre o número de gerações no Líder e no Seguidor de forma a favorecer que o Líder (que considera as restrições) evolua por mais tempo. Foi possível observar que a versão com mais gerações no Líder tinha resposta melhor e que as versões com C-NSQIGLGP como algoritmo base também apresentavam desempenho melhor. No entanto, os resultados mostram significativa perda no %CSA em todas as propostas nessas bases. Este desempenho levou a duas variações para favorecer a evolução em cada nível através do aumento do número de gerações e manutenção do total de avaliações: a execução em apenas um ciclo evolutivo, que resultou em melhora do %CSA; e a manutenção dos dez ciclos, mas com uma redução do tamanho da população e um proporcional aumento do número de gerações por nível, o que também resultou em melhora do %CSA. Os resultados da linha de considerar as restrições para diferenciar os problemas sugerem que há benefício em otimizar o problema em dois níveis, mas que separar por restrições pode não ser a proposta mais adequada.

Diante dos bons resultados obtidos pelo modelo C-DQIGLGP, a segunda linha de estudo da estratégia E-Bi propôs um modelo em que o Seguidor evolui considerando apenas os objetivos críticos e o Líder evolui com todos os problemas. Este modelo é chamado BiOC-DQIGLGP. Em ambos os níveis o algoritmo evolutivo base é o C-DQIGLGP e a migração entre indivíduos do Seguidor para o Líder acontece apenas uma vez. Por considerar apenas dois objetivos, o Seguidor pode ter uma população menor e assim, mantendo o número total de avaliações, ter mais gerações neste nível, o que faz com que o Líder receba uma população mais evoluída. Os resultados mostram o bom desempenho desta proposta e a avaliação comparativa conclui que C-DQIGLGP e BiOC-DQIGLGP são as melhores propostas para resolver os problemas de programação de petróleo apresentados. As Figuras 6.18 à 6.21 apresentam exemplos de frentes de Pareto ótimas destes modelos para os cenários 2309 e 8119.

C-DQIGLGP e BiO-DQIGLGP geram, em média, de duas a seis soluções na frente de Pareto ótima. Este valor é adequado para que o programador de produção possa avaliá-las e decida qual adotar. Seres humanos tendem a sofrer em processos de escolha em que há muitas alternativas. Nossos níveis de satisfação e felicidade degeneram quando é necessário decidir entre muitas alternativas, conforme apresentado em [89] que mostrou que as pessoas são mais confortáveis e seguras para decidir entre 6 alternativas do que entre 24.

Este estudo é uma contribuição de pesquisa neste tema e, certamente, há outros que podem ser desenvolvidos. Dessa forma, são apresentadas algumas sugestões de trabalhos futuros.

7.1 Sugestão de Trabalhos Futuros

Nesta seção são sugeridos alguns trabalhos como continuação desta pesquisa que podem melhorar o desempenho dos algoritmos desenvolvidos ou avaliar seu desempenho em outras aplicações. São sugeridos:

- elitismo no C-DQIGLGP: nos estudos aqui realizados a utilização de um indivíduo da população *archive* não tem nenhuma preocupação com sua aptidão para resolver um subproblema. Pode-se avaliar uma proposta em que se verifique qual indivíduo da população externa tem maior aptidão e utilizar este critério;
- pós-processamento ou busca local: pode ser aplicado em soluções que atendem aos objetivos críticos de forma a otimizar localmente situações como pequenas paradas de UDA, por exemplo, convertendo-as em redução de vazão e garantindo a correta alocação dos equipamentos, ou situações como a observada no terceiro gráfico da Figura 6.19 onde as últimas movimentações dos tanques TqT-03 e TqT-04 poderiam ser reordenadas de forma a evitar a troca desnecessária.
- aumentar a robustez das conclusões feitas nesta tese através do aumento da base de dados, ou seja, aplicar estes modelos a mais cenários e a outros perfis de refinarias;
- avaliar esta abordagem para programação de outras áreas da refinaria;
- tratar restrições como um novo objetivo: pela tabela 6.23, o programador da refinaria gerou soluções violando restrição, o que sugere que há alguma flexibilidade em relação a este critério.
- paralelismo: para aumentar o desempenho computacional e, através desse, permitir maior evolução das soluções sem comprometimento do tempo de execução dos modelos;
- *Bilevel*: nas propostas feitas nesta tese o problema é dividido em dois níveis, mas não há segregação das variáveis de decisão, o que é uma característica de problemas *bilevel*. Acredita-se que esta abordagem pode ser benéfica em problemas de planejamento e programação integrados onde, por exemplo, o planejamento é modelado como Líder, definindo vazões, campanhas e volumes de produção, e o *schedule* é modelado como Seguidor, recebendo assim estas variáveis e avaliando se com elas é possível criar uma solução de programação viável. As soluções são devolvidas ao Líder para um novo ciclo iterativo.

Referências bibliográficas

[1] CAMARGO, V. C. B.. **Optimization of processes in textile industry: models and solution methods**. Tese de doutorado, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Rio de Janeiro, 2012.

[2] MESQUITA, M. A.; SANTORO, M. C.. **Análise de modelos e práticas de planejamento e controle da produção na indústria farmacêutica**. Produção, 14:64–77, 2004.

[3] HARJUNKOSKI, I.;MARAVELIAS, C. T.; BONGERS, P.; CASTRO, P. M.; ENGELL, S.; GROSSMANN, I. E.; HOOKER, J.; MENDEZ, C.; SAND, G.; WASSICK, J.. **Scope for industrial applications of production scheduling models and solution methods**. Computers and Chemical Engineering, 62:161–193, 2014.

[4] TAMARA, C. R. L.. **Oil refinery scheduling optimisation**. Dissertação de mestrado, School of Engineering, Department of Process & Systems Engineering, Cranfield, England, 2003.

[5] PEREIRA, C. S.. **Otimização multiobjetivo da programação de petróleo em refinaria por programação genética em linguagem específica de domínio**. Dissertação de mestrado, Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, PUC-Rio, Rio de Janeiro, 2012.

[6] MORO, L. F. L.. **Técnicas de otimização mista-inteira para o planejamento e programação de produção em refinarias de petróleo**. Tese de doutorado, Escola Politécnica, Universidade de São Paulo, São Paulo, 2000.

[7] AGÊNCIA NACIONAL DE PETRÓLEO (ANP). **Processamento de petróleo e produção de derivados**. Site ANP, 2018. Acesso em 01 de Outubro de 2018.

[8] MAGALHÃES, M. V. O.. **Refinery scheduling**. Tese de doutorado, Department of Chemical Engineering and Chemical Technology, Imperial College, London, 2004.

[9] JOLY, M.; MIYAKE, M.. **Lessons learned from developing and implementing refinery production scheduling technologies**. Frontiers of Engineering Management, 4(3):325–337, 2017.

[10] ASPENTECH. **Tecnologia da informação para indústrias**. Site Aspentech, 2016. Acesso em: Novembro de 2016.

[11] PRINCEPS. **Tecnologia da informação para refinarias**. Site Princeps, 2016. Acesso em: Novembro de 2016.

[12] MÉNDEZ, C. A.; GROSSMANN, I.; HARJUNKOSKI, I.;KABORÉ, P.. **A simultaneous optimization approach for off-line blending and scheduling of oil-refinery operations**. Computers and Chemical Engineering, 30:614–634, 2006.

[13] PUROHIT, A.; SURYAWANSHI, T.. **Integrated product blending optimization for oil refinery operations**. IFAC Proceedings, 10:343–348, 2013.

[14] CASTILLO, P. A. C.; CASTRO, P. M.; MAHALEC, V.. **Global optimization of nonlinear blend-scheduling problems**. Engineering, 3 n.2:188–201, 2017.

[15] LOTERO, I.; TRESPALACIOS, F.; GROSSMANN, I. E.; PAPAGEORGIOU, D. J.; CHEON, M.. **An milp-minlp decomposition method for the global optimization of a source based model of the multiperiod blending problem**. Computers and Chemical Engineering, 87:13–35, 2016.

[16] XU, J.; ZHANG, S.; ZHANG, J.; WANG, S.; XU, Q.. **Simultaneous scheduling of front-end crude transfer and refinery processing**. Computers and Chemical Engineering, 96:212–236, 2017.

[17] SHAH, N. K.; IERAPETRITOU, M. G.. **Lagrangian decomposition approach to scheduling large-scale refinery operations**. Computers and Chemical Engineering, 79:1–29, 2015.

[18] MOURET, S.; GROSSMANN, I.; PESTIAUX, P.. **A new lagrangian decomposition approach applied to the integration of refinery planning and crude-oil scheduling**. Computers and Chemical Engineering, 35:2750–2766, 2011.

[19] WU, N. Q.; BAI, L. P; ZHOU, M. C.. **An efficient scheduling method for crude oil operations in refinery with crude oil type mixing**

requirements. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 46 (3):413–426, 2016.

[20] HOU, Y.; WU, N.; ZHOU, M.. **Scheduling crude oil operations in refineries with genetic algorithm**. ICNSC 2016 - 13th IEEE International Conference on Networking, Sensing and Control, p. 1–6, 2016.

[21] RAMTEKE, M.; SRINIVASAN, R.. **Large-scale refinery crude oil scheduling by integrating graph representation and genetic algorithm**. Industrial & Engineering Chemistry Research, 51:5256–5272, 2012.

[22] CRUZ, D. D. S.. **Programação da produção em refinaria usando algoritmos genéticos: Um estudo para o caso de scheduling de petróleos**. Tese de mestrado, COPPE/UFRJ, Rio de Janeiro, 2007.

[23] MASOOD, A.;MEI, Y.; CHEN, G;ZHANG, M.. **Many-objective genetic programming for job-shop scheduling**. IEEE Congress on Evolutionary Computation, 63:219–216, 2016.

[24] SHAHSAVAR, M.; NAJAFI, A.,A.; AKHAVAN NIAKI, S. T.. **Three self-adaptive multi-objective evolutionary algorithms for a triple-objective project scheduling problem**. 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), 87:4–15, 2015.

[25] LI, B.; LI, J.; TANG, K.; YAO, X.. **Many-objective evolutionary algorithms : A survey**. ACM Computing Surveys (CSUR), 48:Article 13, 2015.

[26] ALAM, T.;RAZA, Z;. **Quantum genetic algorithm based scheduler for batch of precedence constrained jobs on heterogeneous computing systems**. Journal of Manufacturing Systems, 135:126–142, 2018.

[27] DEB, K.. **Multi-objective Optimization using Evolutionary Algorithms**. John Wiley & Sons, England, 1st edition, 2001.

[28] COELLO, C., A., C.. **Evolutionary multi-objective optimization: a historical view of the field**. IEEE computational intelligence magazine, 1:28–36, 2006.

[29] SCHAFFER, J. D.. **Multiple objective optimization with vector evaluated genetic algorithms**. Proceedings of the first international conference on genetic algorithms, 1985.

[30] SRINIVAS, N.; DEB, K.. **Muiltiobjective optimization using nondominated sorting in genetic algorithms**. Evolutionary Computation, 2 (3):221–248, 1994.

[31] MURUTA, T.; ISHIBUCHI,, H.. **Moga: Multi-objective genetic algorithm**. Proceedings of the IEEE international conference on evolutionary computation, 1995.

[32] DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T.. **A fast and elitist multiobjective genetic algorithm: Nsga-ii**. IEEE Transactions on Evolutionary Computation, 6:182–197, 2002.

[33] ZITZLER, E.; LAUMANNS, M.; THIELE, L.. **Spea2: Improving the strength pareto evolutionary algorithm**. Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, 1:2–21, 2001.

[34] BERENGUER, J. A. M.. **Optmización evolutiva multiobjetivo basada en el algoritmo de kuhn-munkres**. Dissertação de mestrado, Centro de investigación y de estudios avanzados del instituto politécnico nacional, México, 2014.

[35] FLEISCHER, S.. **The measure of pareto optima : Applications to multi- objective metaheuristics**. Lecture Notes in Computer Science, 2632:519–533, 2003.

[36] ZITZLER, E.; SIMON, K.. **Indicator-based selection in multiobjective search**. 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), i:832–842, 2004.

[37] BEUME, N., NAUJOKS, B., EMMERICH, M.. **Sms-emoa: Multiobjective selection based on dominated hypervolume**. European Journal of Operational Research, 181:1653–1669, 2007.

[38] ZHANG, Q.; LI, H.. **Moea/d: A multiobjective evolutionary algorithm based on decomposition**. IEEE Transactions on Evolutionary Computation, 11:712–731, 2007.

[39] TRIVEDI, A.; SRINIVASAN, D.; SANYAL, K.; GHOSH, A.;. **A survey of multiobjective evolutionary algorithms based on decomposition**. Applied Soft Computing Journal, 21 (3):440–462, 2017.

[40] QI, Y.; MA, X.; LIU, F.; JIAO, L.; SUN, J.; WU, J.. **Moea/d with adaptative weight adjustment**. Evolutionary Computation, 22:221–248, 2014.

[41] WANG, Z.; ZHANG, Q.; GONG, M.; ZHOU, A.. **A replacement strategy for balancing convergence and diversity in moea/d**. Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014, p. 2132–2139, 2014.

[42] ZHANG, Q., LI, H.. **Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii**. IEEE Transactions on Evolutionary Computation, 13:284–302, 2009.

[43] WANG, R.; XIONG, J.; ISHIBUCHI, H.; WU, G.; ZHANG, T.;. **On the effect of reference point in moea/d for multi-objective optimization**. Applied Soft Computing Journal, 58:25–34, 2017.

[44] DEB, K.; JAIN, H.. **An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part i: Solving problems with box constraints**. IEEE Transactions on Evolutionary Computation, 18:577–601, 2014.

[45] PURSHOUSE, R. C.; FLEMING, P. J.. **On the evolutionary optimization of many conflicting objectives**. IEEE Transactions on Evolutionary Computation, 11:770–784, 2007.

[46] SATO, H.; AGUIRRE, H. E., TANAKA, K.. **Pareto partial dominance moea in many-objective optimization**. CEC, 1:1–8, 2010.

[47] JAIN, H.;DEB, K.. **An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach**. IEEE Transactions on Evolutionary Computation, 18:602–622, 2014.

[48] DAS, I.; DENNIS, J.. **Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems**. SIAM J. Optimization, 8:631–657, 1998.

[49] DEB, K.; IBRAHIM, A.; MARTIN, M. V.; RAHNAMAYAN, S.. **Elite nsga-iii: An improved evolutionary many-objective optimization algorithm**. IEEE Congress on Evolutionary Computation, 63:973–982, 2016.

[50] ALAM, T.;SEGREDO, E.; ŚANCHEZ-PI, N.; HART, E.;. **Impact of selection methods on the diversity of many-objective pareto set approximations**. Journal of Manufacturing Systems, 112:844–853, 2017.

[51] SHEIKH, S.; KOMAKI, G. M.; KAYVANFAR, V.;. **Multi objective two-stage assembly flow shop with release time**. Computers and Industrial Engineering, 124:276–292, 2018.

[52] KAYVANFAR, V.; HUSSEINI, S. M. M.; KARIMI, B.; SAJADIEH, M. S;. **Bi-objective intelligent water drops algorithm to a practical multi-echelon supply chain optimization problem**. Journal of Manufacturing Systems, 44:93–114, 2017.

[53] CHAHARDOLI, S.; HADIAN, H.; VAHEDI, R;. **Optimization of hole height and wall thickness in perforated capped-end conical absorbers under axial quasi-static loading (using nsga-iii and moea/d algorithms)**. Journal of Manufacturing Systems, 127:540–555, 2018.

[54] BHESDADIYA, R.H.;TRIVEDI, I.N.; JANGIR, P.; JANGIR, N.;. **An nsga-iii algorithm for solving multi-objective economic / environmental dispatch problem**. Cogent Engineering, 47:1, 2017.

[55] LI, K.; DEB, K.; ZHANG, Q.; KWONG, S.;. **An evolutionary many-objective optimization algorithm based on dominance and decomposition**. IEEE Transactions on Evolutionary Computation, 19 (5):694–716, 2015.

[56] SATO, H;. **Inverted pbi in moea/d and its impact on the search performance on multi and many-objective optimization**. Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, p. 645–652, 2014.

[57] MOHAMMADI, A.; OMIDVAR, M. N.; LI, X.; DEB, K.;. **Integrating user preferences and decomposition methods for many-objective optimization**. IEEE Congress on Evolutionary Computation (CEC), p. 421–428, 2014.

[58] TALBI, EG.. **Metaheuristics for Bi-level Optimization**, chapter 1. A taxononmy of Metaheuristics for Bilevel Optimization, p. 1–40. Springer, Heidelberg, 1st edition, 2013.

[59] SADIGH, A. N.; MOZAFARI, M.; KARIMI, B.. **Manufacturer-retailer supply chain coordination: A bi-level programming approach**. Advances in Engineering Software, 45:144–152, 2012.

[60] SINHA, A.; DEB, K.. **Metaheuristics for Bi-level Optimization**, chapter 9. BilevelMulti-Objective Optimization and Decision Making, p. 245–284. Springer, Heidelberg, 1st edition, 2013.

[61] CANDLER, W.; NORTON, R.. **Multilevel programming**. Technical Report 20, World Bank Development Research, 1997.

[62] KOH, A.. **Metaheuristics for Bi-level Optimization**, chapter 6. A metaheuristic framework for bilevel programming problems with multidisciplinary applications, p. 153–187. Springer, Heidelberg, 1st edition, 2013.

[63] DEB, K.; SINHA, A.. **An evolutionary approach for bilevel multi-objective problems**. Applied Soft Computing, 35:17–24, 2009.

[64] LEGILLON, F.; LIEFOOGHE, A.; TALBI, EG.. **Metaheuristics for Bi-level Optimization**, chapter 4. CoBRA: A CoevolutionaryMetaheuristic for Bi-level Optimization, p. 95–114. Springer, Heidelberg, 1st edition, 2013.

[65] TILAHUN, S. L.; KASSA, S. M.; ONG, H. C.. **A new algorithm for multilevel optimization problems using evolutionary strategy , inspired by natural adaptation**. Produção, 1:577 – 588, 2012.

[66] SAHARIDIS„ A. K. D.; CONEJO, A. J,; KOZANIDIS, G.. **Metaheuristics for Bi-level Optimization**, chapter 8. Exact Solution Methodologies for Linear and (Mixed) Integer Bilevel Programming, p. 221–245. Springer, Heidelberg, 1st edition, 2013.

[67] MARINAKIS, Y.; MARINAKI, M.. **Metaheuristics for Bi-level Optimization**, chapter 3. A Bilevel Particle Swarm Optimization Algorithm for Supply Chain Management Problems, p. 69–93. Springer, Heidelberg, 1st edition, 2013.

[68] LEGILLON, F.; LIEFOOGHE, A.; TALBI, EG.. **Cobra: A cooperative coevolutionary algorithm for bi-level optimization**. IEEE Congress on Evolutionary Computation, CEC 2012, p. 10–15, 2012.

[69] LU, Y.; WAN, Z.. **A smoothing method for solving bilevel multi-objective programming problems**. Journal of the Operations Research Society of China, 2:511–525, 2014.

[70] LV, Y.; WAN, Z.. **Solving linear bilevel multiobjective programming problem via exact penalty function approach**. Journal of Inequalities and Applications, 1:258–269, 2015.

[71] FLIEGE, J.; VICENTE, L.. **Multicriteria approach to bilevel optimization**. Journal of Optimization Theory and Applications, 131:209–225, 2006.

[72] SINHA, A.; MALO, P.; DEB, K.. **Efficient evolutionary algorithm for single-objective bilevel optimization 2 past research on bilevel optimization using evolutionary algorithms**. Journal of Global Optimization, 1:1–34, 2014.

[73] MARINAKIS, Y.. **A new bilevel formulation for the vehicle routing problem and a solution method using a genetic algorithm**. Journal of Global Optimization, 38:555–580, 2007.

[74] MARINAKIS, Y.. **An improved particle swarm optimization algorithm for the capacitated location routing problem and for the location routing problem with stochastic demands**. Applied Soft Computing, 37:680–701, 2015.

[75] ODUGUWA, V.; ROY, R.. **Bi-level optimisation using genetic algorithm**. Proceedings - 2002 IEEE International Conference on Artificial Intelligence Systems, ICAIS 2002, 1:322–327, 2002.

[76] WANG, Y.; JIAO, Y.C.; LI, H. C.. **Evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme**. IEEE Transactions on Systems, Man and Cybernetics, Part C, 35:221–232, 2005.

[77] WANG, J. Y. T.; EHRGOTT, M.. **Transport sustainability analysis with a bilevel biobjective optimisation model**. Proceedings of the 2011 Conference on Multi-Criteria Decision Making, 2011.

[78] SINHA, A.. **Progressively interactive evolutionary multiobjective optimization**. Tese de doutorado, Department of Business Technology, Aalto University, Finland, 2011.

[79] DEURSEN, A. V.; KLINT, P.. **Domain specific language design requires feature descriptions**. Journal of Computing and Information Technology, 10, n.1:1–17, 2002.

[80] ARAUJO, S. G.. **Síntese de sistemas digitais utilizando técnicas evolutivas**. Tese de doutorado, COPPE/UFRJ, Rio de Janeiro, 2004.

[81] DIAS, D. M.. **Programação genética linear com inspiração quântica**. Tese de doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2010.

[82] BRAMEIER, M.. **On Linear Genetic Programming**. Dissertation, Universidade de Dortmund, Alemanha, 2004.

[83] ZITZLER, E.; THIELE, L.. **Multiobjective optimization using evolutionary algorithms: A comparative case study**. In: PARALLEL PROBLEM SOLVING FROM NATURE V, p. 292–301, Springer, Amsterdam, 1998.

[84] LOPEZ, E. M;. **Un algoritmo evolutivo multiobjetivo basado en hipervolumen en gpus**. Dissertação de mestrado, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, México, 2014.

[85] PEREIRA, C. S.; DIAS, D. M.; VELLASCO, M. B. R.; VIANA, F. H. F.; MARTÍ, L.. **Crude oil refinery scheduling: addressing a real world multiobjective problem through genetic programming and dominance-based approach**. Proceedings of the Genetic and Evolutionary Computation Conference, p. 1821–1828, 2018.

[86] SHAPIRO, S. S.; WILK, M.B.;. **An analysis of variance test for normality (complete samples)**. Biometrika, 52 (3/4):591–611, 1965.

[87] NEUHÄUSER, M.; . **International Encyclopedia of Statistical Science**, chapter Wilcoxon-Mann–Whitney Test, p. 1656–1658. Springer, Heidelberg, 1st edition, 2011.

[88] PEREIRA, C. S.; DIAS, D. M.; VELLASCO, M. B. R. **Programação de petróleo utilizando otimização *bilevel* em programação genética com inspiração quântica**. Anais do XXI Encontro Nacional de Modelagem Computacional, 2018.

[89] OULASVIRTA, A.; HUKKINEN, J..; SCHWARTZ, B..; . **When more is less: the paradox of choice in search engine use**. Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, p. 516–523, 2009.

A
Artigos da Tese Publicados em Congresso

O método *Nondominated Sort Quantum Inspired Grammar based Linear Genetic Programming* apresentado nesta tese gerou uma publicação no *Genetic and Evolutionary Computation Conference* (GECCO) 2018, no Japão, sob o título *"Crude oil refinery scheduling: addressing a real-world multiobjective problem through genetic programming and dominance-based approaches"*. O artigo completo está publicado nos *proceedings* do congresso (doi 10.1145/3205651.3208291).

O método *Bilevel Quantum Inspired Grammar based Linear Genetic Programming* apresentado nesta tese gerou uma publicação no XXI Encontro Nacional de Modelagem Computacional (ENMC) 2018, no Brasil, sob o título "Programação de petróleo utilizando otimização *bilevel* em programação genética com inspiração quântica". O artigo completo está publicado nos anais do congresso, e pode ser acessado através do site: `http://www.essentiaeditora.iff.edu.br/index.php/enmc-ectm/article/view/12431`

Estas publicações são requisitos parciais para obtenção do título de Doutor pelo Programa de Pós-graduação em Engenharia Elétrica da PUC-Rio.