

UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

MELHORANDO HEURÍSTICAS PARA O NRP ATRAVÉS DA
VISUALIZAÇÃO DO ESPAÇO DE BUSCA

Richard Fuchshuber

Orientador
Márcio de Oliveira Barros

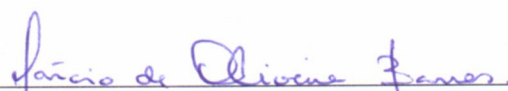
RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2015

MELHORANDO HEURÍSTICAS PARA O NRP ATRAVÉS DA
VISUALIZAÇÃO DO ESPAÇO DE BUSCA

Richard Fuchshuber

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA A
OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-
GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO
ESTADO DO RIO DE JANEIRO (UNIRIO). APROVADA PELA COMISSÃO
EXAMINADORA ABAIXO ASSINADA.

Aprovada por:



Prof. Márcio de Oliveira Barros, D.Sc. – UNIRIO



Prof. Adriana Cesário de Faria Alvim, D.Sc. – UNIRIO



Prof. Leonardo Gresta Paulino Murta, D.Sc. – UFF

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2015

Fuchshuber, Richard

F951 Melhorando heurísticas para o NRP através da visualização do espaço de busca / Richard Fuchshuber, 2015.

77 f. ; 30 cm

Orientador: Márcio de Oliveira Barros.

Dissertação (Mestrado em Informática) – Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2015.

1. Software - Desenvolvimento. 2. Programação heurística. 3. Algoritmo. 4. Next Release Problem. I. Barros, Márcio de Oliveira. II. Universidade Federal do Estado do Rio de Janeiro. Centro de Ciências Exatas e Tecnológicas. Curso de Mestrado em Informática. III. Título.

CDD - 005.1

*À pessoa que há 20 anos me
ensinou a programar em Basic, a
editar textos no Carta Certa, e
que sempre serviu de inspiração.
Vielen Dank für alles,
Großvater.*

Agradecimentos

Agradeço aos amigos e familiares pelas palavras de incentivo e pela compreensão das ausências que por vezes se fizeram necessárias. Em especial agradeço à minha esposa, Romina, pelo carinho e ajuda durante esse anos. Planos tiveram que ser adiados, o que acabou por afetar não apenas o meu dia a dia, mas o dela também. Não encontro palavras suficientes para agradecê-la por isso. Por fim, agradeço ao meu orientador, Prof. Márcio Barros, pela serenidade, foco e dedicação que foram determinantes para moldar o que hoje é este trabalho.

FUCHSHUBER, Richard. **Melhorando Heurísticas para o NRP através da Visualização do Espaço de Busca**. UNIRIO, 2015. 77 páginas. Dissertação de Mestrado. Departamento de Informática Aplicada, UNIRIO.

RESUMO

A seleção de quais requisitos incluir na próxima versão de um software requer o equilíbrio entre as necessidades dos clientes e as restrições orçamentárias. O Problema da Próxima Versão do Software (*Next Release Problem - NRP*) consiste em selecionar um subconjunto dos requisitos para serem implementados na próxima versão do software, de modo a maximizar o benefício gerado aos clientes sem exceder o orçamento disponível. Dado um grande conjunto de requisitos e clientes, uma busca exaustiva que examina todos os subconjuntos não pode ser executada em um tempo computacional aceitável. Por isso, algoritmos heurísticos são empregados para obter soluções para o NRP.

Através da visualização do espaço de busca de uma perspectiva da concentração de clientes, foi identificado um padrão gráfico que foi observado em diversas instâncias do NRP. Este trabalho apresenta estes resultados e descreve uma Busca Local Iterada (*Iterated Local Search - ILS*) que foi adaptada para utilizar este padrão no processo de busca. O algoritmo proposto é muito mais simples que a atual heurística estado-da-arte para o NRP, conhecida como BMA (*Backbone-Based Multilevel Algorithm*). Ao ser executado em dois conjuntos de instâncias amplamente utilizadas pela literatura, o BMA gerou soluções, em média, 4,1% piores que as soluções ótimas destas instâncias. O algoritmo proposto neste trabalho foi capaz de reduzir o afastamento médio em relação aos valores ótimos para 1,3%, uma melhoria de 68% em relação ao BMA.

Palavras-chave: Heurísticas, Next Release Problem, Visualização do espaço de busca

ABSTRACT

The selection of requirements to be included in the next release of a software requires balancing customers needs and budget constraints. The Next Release Problem (NRP) aims to select a subset of the requirements to be implemented in the next release that maximizes customer's benefits while the development budget is satisfied. Given a large set of requirements or clients, an exhaustive search that examines all subsets cannot be performed in feasible time. Thus, heuristic algorithms are employed to find solutions for the NRP.

Visualizing the fitness landscape from a customer concentration perspective, we observed a recurring pattern in several instances of the NRP. This work presents these findings and describes an Iterated Local Search (ILS) algorithm modified to take advantage of this pattern. The proposed algorithm is much simpler than the current state-of-art heuristic approach for the NRP, known as BMA (*Backbone-Based Multilevel Algorithm*). When executed against two commonly used large scale instance sets, BMA generated solutions that are, on average, 4.1% worse than the optimal solution for each instance. The algorithm proposed in this work reduced the distance from the optimal solutions to 1.3%, an 68% improvement when compared with BMA.

Keywords: Heuristics, Next Release Problem, Fitness Landscape Visualization

Sumário

Lista de Figuras	x
Lista de Tabelas	xi
Lista de Abreviaturas	xiii
1 Introdução	1
1.1 Objetivos	2
1.2 Estrutura da Dissertação	2
2 Heurísticas e o NRP	4
2.1 O Problema da Próxima Versão de Software	4
2.2 Algoritmos Heurísticos de Busca	7
2.2.1 Subida de Encosta	8
2.2.2 Busca Local Iterada	9
2.2.3 Algoritmos Genéticos	11
2.3 Algoritmos Heurísticos aplicados ao NRP	12
2.3.1 Formulação Mono-objetivo do NRP	12
2.3.2 Formulação Bi-objetivo do NRP	14
2.4 Considerações Finais	15
3 Solução Proposta	17
3.1 Visualização do Espaço de Busca do NRP	17
3.2 Busca Local orientada por Visualização	19
3.2.1 Número Máximo de Clientes	23
3.3 Busca Local Iterada orientada por Visualização	26
3.4 Considerações Finais	28
4 Avaliação Experimental	29
4.1 Questões de Pesquisa	29
4.2 Instâncias do Problema	30
4.3 Calibragem de Parâmetros	32
4.3.1 Solução Inicial	32
4.3.2 Força da Perturbação	33
4.3.3 Tamanho da Amostragem Aleatória	35
4.4 Análise e Discussão	35

4.4.1 Ameaças à Validade	51
4.5 Considerações Finais	55
5 Conclusão.	56
Referências Bibliográficas	58
A Avaliação dos Construtores	62

Lista de Figuras

2.1	Dependências entre requisitos e solicitações de cada cliente. Adaptado de Xuan et al.	6
3.1	Amostragem aleatória da instância nrp-1 com orçamento de 50% do custo de implementar todos os requisitos	18
3.2	Amostragem Aleatória para instâncias adaptadas por Xuan et al. [1] do trabalho de Bagnall et al. [2]	20
3.3	Amostragem Aleatória das instâncias geradas por Xuan et al. [1] . . .	21
4.1	Comparação da qualidade das soluções geradas pelo VisILS e BMA para as instâncias clássicas. Os box-plots representam as soluções obtidas pelo VisILS ao longo das 30 execuções. O quadrado verde representa o valor médio obtido pelo BMA, enquanto o losango vermelho representa a sua melhor solução.	43
4.2	Comparação da qualidade das soluções geradas pelo VisILS e BMA para as instâncias realistas. Os box-plots representam as soluções obtidas pelo VisILS ao longo das 30 execuções. O quadrado verde representa o valor médio obtido pelo BMA, enquanto o losango vermelho representa a sua melhor solução.	44
4.3	Gráficos <i>time-to-target</i> para duas instâncias realistas e duas clássicas	49
4.4	Amostragem Aleatória para instâncias adaptadas por Xuan et al. [1] do trabalho de Bagnall et al. [2] com destaque para a posição das soluções ótimas.	51
4.5	Amostragem Aleatória para instâncias realistas criadas por Xuan et al. [1] com destaque para a posição das soluções ótimas.	52

Lista de Tabelas

3.1	Qualidade da melhor solução obtida por cada configuração ao longo das 30 execuções.	24
3.2	Qualidade média das soluções obtidas por cada configuração ao longo de 30 execuções.	25
4.1	Detalhes das instâncias clássicas do NRP	31
4.2	Detalhes das instâncias realistas do NRP	32
4.3	Qualidade média das soluções obtidas por cada construtor dentre as 30 execuções de cada configuração. Valores em negrito indicam a maior média encontrada entre as configurações. Observa-se que o construtor guloso obteve os melhores resultados em todas as instâncias, sendo em média 15% melhor que o aleatório.	34
4.4	Qualidade média das soluções obtidas ao longo das 30 execuções para cada intensidade de perturbação. Valores em negrito indicam a maior média encontrada entre as configurações. Nota-se que quando menor a força da perturbação, maior o <i>fitness</i> médio das soluções obtidas.	36
4.5	Qualidade da melhor solução obtida por cada configuração dentre as 30 execuções. Valores em negrito indicam o maior valor encontrado entre as configurações. Observa-se que o VisILS obteve os melhores resultados para 29 das 39 instâncias.	38
4.6	Qualidade média obtida por cada configuração ao longo das 30 execuções. Valores em negrito indicam o maior valor médio encontrado entre as configurações. Observa-se que o VisILS obteve os melhores resultados para 32 das 39 instâncias.	39
4.7	Comparação entre as configurações ILS e VisILS. Valores em negrito indicam o maior valor médio encontrado ao longo das 30 execuções de cada instância. Observa-se que o VisILS obteve o maior valor médio para 33 da 39 instâncias, dos quais 16 resultados foram estatisticamente significativos com 95% de confiança (coluna PV). O tamanho de efeito (coluna ES) mostra que o VisILS produz resultados com maior qualidade em 65% das execuções (média).	41

4.8	Comparação entre as configurações VisILS e BMA quanto à qualidade das soluções. Valores em negrito indicam a maior média obtida para cada instância. Observa-se que o VisILS obteve a melhor média para mais instâncias (36 de 39), dos quais 35 resultados são significativamente diferentes do BMA.	45
4.9	Tempo médio de execução para cada configuração. Valores em negrito representam a menor média observada em cada instância, considerando as 30 execuções. As configurações HC e ILS obtiveram tempos quase idênticos, enquanto o VisILS foi cerca de 16 vezes mais lento. .	46
4.10	Número de avaliações de <i>fitness</i> até encontrar a melhor solução. O valor indicado representa a média das 30 execuções de cada algoritmo.	48
4.11	Distância percentual entre cada configuração e o ótimo. As distâncias foram calculadas com base no valor médio de <i>fitness</i> de cada configuração.	50
A.1	Número de vezes que cada cliente apareceu na melhor solução encontrada, dentre as 30 execuções do HC, para a instância nrp1-30. Observa-se que as soluções com a maior razão lucro/custo (L/C) apareceram mais vezes.	63
A.2	Correlação, para cada instância, entre o número de vezes que um cliente aparece na solução e o seu lucro, custo e razão lucro/custo (L/C).	64

Lista de Abreviaturas

BMA	Backbone-based Multilevel Algorithm
GA	Genetic Algorithm
HC	Hill Climbing
ILS	Iterated Local Search
NRP	Next Release Problem
RP	Release Planning
SBSE	Search-Based Software Engineering
TTT	Gráfico time-to-target
VisHC	Visual Hill Climbing
VisILS	Visual Iterated Local Search

1. Introdução

Hoje em dia, muitas empresas de software estão envolvidas com o desenvolvimento, manutenção e melhoria de sistemas grandes e complexos, utilizados por muitos clientes [3]. Frequentemente, a equipe do projeto é responsável tanto pelas atividades de manutenção quanto pelo desenvolvimento de novos requisitos, que devem ser tratados concomitantemente. Muitas vezes, o esforço requerido para desenvolver um software é suficientemente grande para justificar sua divisão em diversas versões (*releases*), de modo a atender a restrições financeiras ou à data de entrega de alguns dos requisitos [4]. Neste cenário, o planejamento de quais requisitos implementar em cada versão do software é um aspecto determinante para o sucesso de um projeto, já que quanto maior e mais complexo o projeto, mais difícil e suscetível a erros o planejamento da versão será [5].

Com o intuito de maximizar a satisfação dos clientes, os requisitos mais importantes devem ser priorizados. O Problema da Próxima Versão do Software (*Next Release Problem* - NRP) [2] consiste na seleção dos requisitos a incluir na próxima versão de um software de forma a maximizar benefícios, como a satisfação do cliente ou o lucro alcançado, enquanto restrições, como o orçamento ou tempo disponível, são satisfeitas. No entanto, identificar o melhor subconjunto dentre um conjunto de requisitos candidatos é uma tarefa complexa [4], visto que cada cliente possui suas necessidades e geralmente é impossível atender às demandas de todos simultaneamente, devido às restrições de recursos ou de tempo.

O NRP é similar ao Problema da Mochila, que é NP-Difícil [6]. Como consequência, uma busca exaustiva não consegue resolver de forma eficiente instâncias grandes do problema, compostas por centenas ou milhares de requisitos e clientes, fazendo com que heurísticas sejam frequentemente utilizadas para se obter boas soluções. A Engenharia de Software Baseada em Buscas [7] (*Search-Based Software Engineering* - SBSE) é uma área de pesquisa que lida com a resolução de problemas complexos de Engenharia de Software como o NRP, modelando-os como problemas de otimização e usando estratégias de busca heurística de forma a obter boas soluções para os problemas em questão. Estudos sugerem que os resultados gerados por técnicas de SBSE são comparáveis aos obtidos por especialistas humanos [8].

1.1 Objetivos

Este trabalho introduz uma nova técnica de visualização do espaço de busca do NRP que revelou um padrão gráfico presente em todas instâncias do problema que foram analisadas. Ele tem como objetivo determinar se este padrão pode contribuir para um melhor entendimento do NRP e de suas características, facilitando assim a criação de heurísticas mais eficientes para encontrar soluções para este problema. Este objetivo levou à criação de versões modificadas dos algoritmos Subida de Encosta (*Hill Climbing - HC*) e Busca Local Iterada (*Iterated Local Search - ILS*), que utilizam o padrão identificado como guia para o processo de busca, restringindo a sua atuação a regiões do espaço de busca que se mostraram mais promissoras.

Avaliações experimentais mostraram que as heurísticas que utilizam o padrão gráfico foram capazes de superar suas versões originais, que não fazem uso deste padrão. Apesar de sua simplicidade, o ILS modificado (denominado *VisILS*) obteve resultados superiores aos obtidos pelo *Backbone-based Multilevel Algorithm* (BMA), uma heurística que representa o estado-da-arte na busca de soluções para o NRP e que superou diversos outros algoritmos heurísticos aplicados a este problema em instâncias frequentemente utilizadas na literatura. Os resultados obtidos pelo BMA apresentaram um afastamento médio de 4,1%, em termos de *fitness*, das soluções ótimas de cada instância utilizada neste trabalho. O ILS modificado foi capaz de reduzir este afastamento médio para 1,3%, uma melhoria de 68% em relação ao BMA. No contexto de um grande projeto de software, esta melhoria pode representar uma economia de centenas de milhares de reais.

Visa-se, ainda, chamar a atenção dos pesquisadores da área de SBSE para o fato de que a visualização do espaço de busca pode ajudar a melhorar o desempenho de procedimentos de busca heurística para problemas complexos que possuam um grande espaço de busca. Espera-se que os bons resultados apresentados neste trabalho possam servir de incentivo para a aplicação de abstrações similares a outros domínios da Engenharia de Software.

1.2 Estrutura da Dissertação

O presente trabalho está organizado em cinco capítulos, sendo o primeiro deles esta introdução. O próximo capítulo apresenta o NRP em termos formais e introduz algumas das heurísticas utilizadas para resolver este problema. O mesmo capítulo apresenta, ainda, uma revisão bibliográfica sobre trabalhos existentes na literatura que tratam do NRP sob diversos prismas.

O terceiro capítulo trata da técnica de visualização do espaço de busca do NRP que será utilizada neste trabalho para criar uma nova heurística para o problema.

Apresenta-se a técnica de visualização em si, bem como a variante do ILS desenvolvida para explorar esta visualização. Apresenta-se, ainda, resultados de experimentos iniciais que foram utilizados para validar e guiar a proposta.

No capítulo quatro, o mais extenso desta Dissertação, as questões de pesquisa que nortearam este trabalho são apresentadas e um estudo experimental é definido e executado. Em seguida, os resultados do estudo são analisados. Discute-se, ainda, as ameaças à validade do experimento. Por fim, o quinto capítulo apresenta as considerações finais deste trabalho.

2. Heurísticas e o NRP

Este capítulo introduz informações necessárias para o entendimento da proposta apresentada no capítulo seguinte. Primeiramente, a formulação para o NRP é apresentada. Em seguida, são descritos três algoritmos de busca heurística frequentemente empregados na resolução de problemas de otimização: Subida de Encosta, Busca Local Iterada e Algoritmos Genéticos. Por fim, a literatura existente sobre o NRP é comentada, mostrando diferentes formulações e perspectivas sob as quais o problema foi abordado.

2.1 O Problema da Próxima Versão de Software

O problema da próxima versão de software (*Next Release Problem – NRP*) [2] trata da seleção de requisitos em um contexto onde diversos grupos de clientes tem interesses distintos no que tange aos requisitos a serem desenvolvidos para uma versão de um software. Cada requisito tem um custo de desenvolvimento associado e existe um orçamento fixo que pode ser utilizado na produção da próxima versão de um software. Neste contexto, uma empresa tem que decidir quais requisitos incluir nesta versão, de forma que o custo total de implementação não ultrapasse o orçamento disponível. Por outro lado, a empresa deve considerar a recompensa obtida ao implementar cada requisito, dado o lucro que se espera obter dos clientes cujos ensejos são satisfeitos pela versão sendo produzida. O NRP consiste em obter o subconjunto (mais próximo ao) ótimo de requisitos que maximizem o lucro esperado e cujo custo total não exceda o orçamento disponível.

Existem diversas definições para este problema e suas nuances dependem dos objetivos sendo otimizados, do uso de diferentes aspectos para caracterizar cada requisito e dos tipos de relacionamento existentes entre estes requisitos. A formulação mono-objetivo tem a maximização do lucro como seu único objetivo e estabelece uma restrição orçamentária a ser utilizada na implementação dos requisitos a serem incluídos na próxima versão do software. Também existe uma formulação bi-objetivo, que substitui a restrição orçamentária por um segundo objetivo: minimizar o custo associado à versão. A Seção 2.3 neste capítulo discutirá trabalhos anteriores que utilizaram perspectivas distintas ao abordar o NRP.

Este trabalho adota a formulação mono-objetivo proposta por Bagnall et al. [2], que é apresentada a seguir. Seja R o conjunto de requisitos de um software. Cada cliente i tem um conjunto de requisitos $R_i \subseteq R$ e um lucro $w_i \in Z^+$, que indica o benefício que será gerado caso o cliente tenha seus requisitos incluídos na próxima versão. Associado ao conjunto R , existe um grafo acíclico direcionado $G = (R, E)$ onde $(r, r') \in E$ se, e somente se, r é um pré-requisito de r' . G também é transitivo, dado que $(r, r') \in E \ \& \ (r', r'') \in E \rightarrow (r, r'') \in E$. Caso a empresa decida implementar todos os requisitos do cliente i , além de desenvolver R_i ela também deve desenvolver seus pré-requisitos:

$$parents(R_i) = \{r \in R | (r, r') \in E, r' \in R_i\} \quad (2.1)$$

Como G é transitivo, o conjunto $\hat{R}_i = R_i \cup parents(R_i)$ inclui todos os requisitos para os quais existe um caminho em G de tamanho ≥ 0 terminando em um vértice em R_i . \hat{R}_i compreende todos os requisitos que devem ser implementados para atender o cliente i . Cada $r \in R$ tem um custo associado $cost(r) \in Z^+$, que é o custo de desenvolver este requisito. Se $R' \subseteq R$, então podemos definir

$$cost(R') = \sum \{cost(r) | r \in R'\} \quad (2.2)$$

Assumindo que a empresa possui n clientes, deve-se encontrar um subconjunto de clientes $S \subseteq \{1, 2, \dots, n\}$ cujos requisitos serão incluídos na próxima versão. O custo para atender os clientes em S é

$$cost(S) = cost\left(\bigcup_{i \in S} \hat{R}_i\right) \quad (2.3)$$

Portanto, o NRP consiste em identificar um subconjunto $S \subseteq \{1, 2, \dots, n\}$ tal que

$$\begin{aligned} \sum_{i \in S} w_i \text{ seja maximizado} \\ \text{sujeito à } cost\left(\bigcup_{i \in S} \hat{R}_i\right) \leq B, \end{aligned} \quad (2.4)$$

onde $B \in Z^+$ é o orçamento disponível.

Dado que $\hat{A} \cup \hat{B} = \widehat{A \cup B}$, uma formulação alternativa para a Equação 2.3 é

$$\begin{aligned} cost(\hat{X}) = cost(X) + cost(parents(X) \setminus X), \\ \text{onde } X = \bigcup_{i \in S} R_i \end{aligned} \quad (2.5)$$

No caso especial onde nenhum requisito tem pré-requisitos, $E = \emptyset$, e o problema é dito básico. Neste caso, $R_i = \hat{R}_i$ para qualquer subconjunto e o objetivo passa a ser encontrar um subconjunto S tal que

$$\sum_{i \in S} w_i \text{ seja maximizado}$$

$$\text{sujeito à } \text{cost} \left(\bigcup_{i \in S} R_i \right) \leq B \quad (2.6)$$

Qualquer instância do NRP pode ser formulada como a versão básica se for realizado o pré-processamento de cada R_i , trocando-o por $R_i \cup \text{parents}(R_i)$. Este trabalho utiliza dois conjuntos de instâncias em seus experimentos. O primeiro deles, derivado do trabalho de Bagnall et al.[2] utiliza a formulação mais genérica, apresentada na Equação 2.4, enquanto o segundo conjunto, criado por Xuan et al.[1] utiliza a formulação básica da Equação 2.6.

Xuan et al. [1] apresentam um exemplo de instância do NRP com sete clientes e oito requisitos, reproduzido a seguir. A Figura 2.1 apresenta o grafo de dependência e os requisitos solicitados por cada cliente.

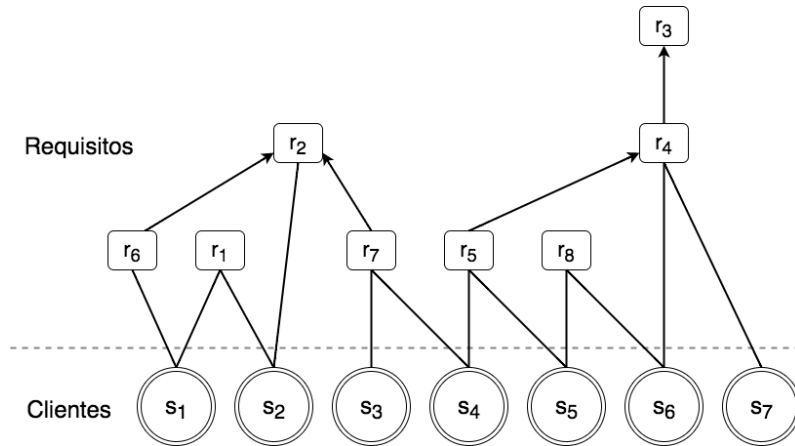


Figura 2.1: Dependências entre requisitos e solicitações de cada cliente. Adaptado de Xuan et al.

As setas de cima para baixo indicam as dependências entre requisitos (por exemplo, $r_2 \rightarrow r_6$ indica que o requisito r_2 precede o requisito r_6) e as linhas indicam os requisitos solicitados por cada cliente. Para o conjunto de requisitos $R = \{r_1, r_2, \dots, r_8\}$, sejam os custos c_1, c_2, \dots, c_8 desses requisitos iguais a 2, 5, 4, 3, 8, 1, 5 e 2, respectivamente. Para o conjunto de clientes $S = \{s_1, s_2, \dots, s_7\}$, sejam os lucros w_1, w_2, \dots, w_7 obtidos ao satisfazer cada um desses clientes iguais a 7, 2, 6, 5, 4, 3 e 1. Utilizando s_1 como exemplo, os requisitos solicitados por s_1 são $R_1 = \{r_1, r_2, r_6\}$, o custo em implementar todos esses requisitos é $\text{cost}(R_1) = 8$, e o lucro obtido ao satisfazer s_1 é $w_1 = 7$.

Dada uma restrição orçamentária $B = 26$, o lucro e o custo de uma solução X_1 que implementa os requisitos solicitados pelos clientes s_1, s_3 e s_7 são, respectivamente 14 e 20. De forma similar, o lucro e o custo de uma solução X_2 que implementa os requisitos solicitados pelos clientes s_1, s_5, s_6 e s_7 são 15 e 25. Claramente, a solução X_2 é melhor que X_1 . Já a solução X_3 que implementa os requisitos solicitados pelos clientes s_2, s_4 e s_6 é inviável, visto que seu custo de 29 unidades de custo é superior ao limite orçamentário $B = 26$.

Bagnall et al. [2] mostrou que o NRP é uma variação do problema da mochila 0/1 [9], cujo problema de decisão associado é conhecidamente NP-Completo. Este resultado justifica a utilização de algoritmos heurísticos de busca, apresentados a seguir, na resolução do NRP.

2.2 Algoritmos Heurísticos de Busca

As soluções para o NRP podem ser representadas como vetores contendo um elemento para cada cliente. Cada elemento do vetor pode ser preenchido com o valor um, que indica que todos os requisitos solicitados por aquele cliente serão incluídos na próxima versão do software, ou com zero, indicando que ao menos um dos requisitos solicitados pelo cliente não será implementado naquela versão.

Em casos envolvendo centenas ou milhares de requisitos e clientes, o conjunto de possíveis soluções para o NRP não pode ser enumerado em um tempo aceitável pelos computadores atuais. De fato, este é o caso de muitas aplicações na ciência e indústria [10], onde é necessário aceitar soluções “boas o suficiente”, que podem ser obtidas em um tempo computacional aceitável através do uso de algoritmos conhecidos como heurísticas. Este tipo de algoritmo percorre o espaço de potenciais soluções, isto é, o espaço de busca do problema, usando um procedimento sistemático que inteligentemente deixa regiões desinteressantes inexploradas, concentrando os esforços em regiões do espaço de busca que apresentem soluções mais interessantes em relação ao objetivo da otimização.

Abordagens baseadas em busca aplicam procedimentos de busca heurística para encontrar boas soluções para uma instância particular de um problema. Neste sentido, diversas estratégias de busca podem ser exploradas. Este trabalho analisa procedimentos de busca local e busca local estendida, comparando os resultados obtidos por esses procedimentos com os obtidos por estratégias de busca globais, como Algoritmos Genéticos e o *Backbone-based Multilevel Algorithm* (BMA) [1].

2.2.1 Subida de Encosta

Um procedimento de busca local consiste em mover-se de uma solução para outra em sua vizinhança através de regras bem definidas [11]. Uma estratégia de busca local geralmente inicia de uma solução arbitrária e, a cada etapa, uma nova solução é escolhida da vizinhança da solução corrente. Esta vizinhança é o conjunto de soluções que podem ser alcançadas ao se realizar pequenas modificações na solução corrente, usualmente através da mudança de um único elemento da solução.

O algoritmo de Subida de Encosta (*Hill Climbing* – HC) [10] é uma busca local que se move de uma solução inicial até um ótimo local realizando apenas movimentos que melhoram a solução. No HC, a busca inicia de uma solução arbitrária do espaço de busca e considera apenas os vizinhos desta solução. Ao encontrar um vizinho de maior qualidade, este se torna a solução corrente e o processo de exame da vizinhança é repetido usando esta nova solução como ponto inicial. Caso nenhum vizinho de maior qualidade seja encontrado, um ótimo local foi encontrado e a busca termina. Uma função objetivo é utilizada a fim de comparar a qualidade de diferentes soluções.

Algoritmo 2.1 Pseudo-código da Subida de Encosta

```
1: função HILLCLIMBING(instância)
2:    $bf = -\infty$ 
3:   repita
4:      $s_0 = \text{gerarSolução}(\textit{instância})$ 
5:      $s'_0 = \text{BuscaLocal}(s_0, \textit{instância})$ 
6:     se funçãoObjetivo(instância,  $s'_0$ ) >  $bf$  então
7:        $s_* = s'_0$ 
8:        $bf = \text{funçãoObjetivo}(\textit{instância}, s'_0)$ 
9:     fim se
10:  até critério de parada
11:  retorne  $s_*$ 
12: fim função
```

O Algoritmo 2.1 apresenta o pseudo-código do HC. Este pseudo-código, assim como outros códigos mostrados neste trabalho, usa um conjunto de funções que permite ao algoritmo acessar informações sobre a instância do problema sendo otimizada. Estas funções são: *contarClientes*(*instância*), que retorna o número de clientes presentes em uma instância; *gerarSolução*(*instância*), que cria e retorna uma solução inicial para a busca, utilizando um gerador de soluções previamente selecionado (veja a Seção 4.3.1); *obterClientes*(*solução*), que retorna o número de clientes atendidos pela solução recebida como parâmetro; e *funçãoObjetivo*(*instância*, *solução*), que calcula o valor da função objetivo (o *fitness*) de uma solução para uma instância, de acordo com a equação 2.4. Esta última função pode ainda retornar um valor negativo proporcional ao custo necessário para desenvolver os requisitos selecionados caso este custo exceda o orçamento disponível.

O Algoritmo 2.2 detalha a função *BuscaLocal()*, que é utilizada pelo HC. Esta função itera sobre os clientes presentes em uma determinada instância, adicionando ou removendo um cliente por vez através da função *inverteCliente()*. A função *OrdemClientesAleatória()* é responsável por gerar uma ordem aleatória para a visitação dos clientes, introduzindo um componente aleatório no algoritmo de modo que este possa explorar caminhos diferentes no espaço de busca a cada execução. A busca local implementada adota a estratégia de *first improvement*, na qual o primeiro vizinho encontrado cuja qualidade seja melhor que a solução corrente é utilizado como ponto de partida para as próximas iterações da busca local. Já a vizinhança é definida como todas as soluções que diferem da solução corrente por apenas um cliente.

Um problema da busca local é que ela pode ficar presa em uma solução localmente ótima que pode ser muito pior, em termos de *fitness*, que o ótimo global do espaço de busca. Buscas locais com reinícios, como o HC descrito anteriormente, tratam este problema através da execução do procedimento de busca diversas vezes, partindo de soluções iniciais distintas e retornando a melhor solução encontrada dentre todos os reinícios.

Algoritmo 2.2 Pseudo-código da função de Busca Local

```

1: função BUSCALOCAL( $s_0$ , instância)
2:    $s_* = s_0$ 
3:    $bf = \text{funçãoObjetivo}(\text{instância}, s_*)$ 
4:   repita
5:      $achouMelhor = \text{falso}$ 
6:      $ordemClientes = \text{OrdemClientesAleatória}(\text{instância})$ 
7:     para cada cliente em  $ordemClientes$  faça
8:       se critério de parada então
9:         retorne nil ▷ Busca exaurida
10:      fim se
11:       $s' = \text{inverteCliente}(s_*, \text{cliente})$ 
12:      se  $\text{funçãoObjetivo}(\text{instância}, s') > bf$  então
13:         $s_* = s'$ 
14:         $bf = \text{funçãoObjetivo}(\text{instância}, s_*)$ 
15:         $achouMelhor = \text{verdadeiro}$ 
16:      encerre o laço
17:    fim se
18:  fim para
19:  até  $achouMelhor$ 
20:  retorne  $s_*$ 
21: fim função

```

2.2.2 Busca Local Iterada

A qualidade da solução final obtida por um algoritmo de busca local é dependente da solução inicial. Em uma busca local com reinícios, a solução inicial tipicamente é escolhida arbitrariamente e não está relacionada com os ótimos locais encontrados

em iterações anteriores. A Busca Local Iterada (*Iterated Local Search – ILS*) melhora a busca local com reinícios ao aplicar um operador de perturbação ao ótimo local obtido e considerar esta solução perturbada como ponto de partida para a próxima iteração [10].

O Algoritmo 2.3 descreve o algoritmo do ILS. Primeiramente, uma solução inicial é gerada e otimizada por uma busca local. Depois disso, a cada iteração, uma perturbação é aplicada ao ótimo local obtido, de forma a gerar um novo ponto de partida. Finalmente, a busca local é aplicada à solução perturbada e a melhor solução obtida por esta busca é aceita como a nova solução corrente, segundo algum critério. O algoritmo é encerrado após atingir um critério de parada.

A perturbação deve alterar partes da solução, enquanto mantém outras partes intocadas. Ela é aplicada como uma tentativa de mover a solução de um ótimo local para outra região próxima, na expectativa de que ao preservar partes da solução seja possível manter características de uma boa solução e ao mesmo tempo livrar a busca da atração do ótimo local.

Algoritmo 2.3 Pseudo-código da Busca Local Iterada

```
1: função ILS(instância, forçaDaPerturbação)
2:    $s_0 = \text{gerarSolução}(\textit{instância})$ 
3:    $s_* = \text{BuscaLocal}(s_0, \textit{instância})$ 
4:   repita
5:      $s' = \text{Perturbar}(s_*, \textit{forçaDaPerturbação})$ 
6:      $s'_* = \text{BuscaLocal}(s', \textit{instância})$ 
7:      $s_* = \text{Aceitar}(s_*, s'_*)$ 
8:   até critério de parada
9:   retorne  $s_*$ 
10: fim função
```

Para a maioria dos problemas reais de otimização, o operador de perturbação é mais eficiente que realizar reinícios aleatórios independentes [10]. A perturbação deve ser forte o suficiente para impedir ciclos na busca, isto é, o retorno ao mesmo ótimo local a partir da solução perturbada, mas ao mesmo tempo não tão forte a ponto de transformar o procedimento em um reinício aleatório. O critério de aceitação determina se a solução s'_* deve ser aceita ou não como a nova solução corrente e é utilizado para equilibrar a busca entre os processos de intensificação e diversificação.

Diversificação é o processo que faz a busca explorar novas regiões do espaço de busca, enquanto a intensificação concentra a busca em regiões já exploradas que apresentaram soluções de boa qualidade. Tipicamente, um algoritmo heurístico aceita maior diversificação em suas primeiras iterações, visando encontrar novas regiões promissoras do espaço de busca, mas passa a privilegiar a intensificação a medida que se aproxima do critério de parada estabelecido para o término da sua execução.

2.2.3 Algoritmos Genéticos

Os Algoritmos Genéticos [12] (*Genetic Algorithm – GA*) são heurísticas que empregam conceitos biológicos como herança, mutação e *crossover* para imitar o processo de seleção natural, fazendo uso deste mecanismo como meio para se obter boas soluções para um problema. Ao contrário dos algoritmos apresentados anteriormente, que utilizam apenas uma solução como ponto de partida, os GAs utilizam diversas soluções simultaneamente. Estas soluções constituem um grupo chamado de população.

O Algoritmo 2.4 apresenta o pseudo-código de um GA. No início do algoritmo, uma população de indivíduos é gerada, normalmente de forma aleatória. Cada indivíduo desta população representa uma possível solução para o problema em questão e uma função objetivo é utilizada para atribuir um valor de *fitness* a cada um dos indivíduos. Estes indivíduos representam a primeira geração de soluções do problema. Em seguida, o algoritmo entra em um ciclo de repetição cujo passo representa uma geração em que a população de soluções é evoluída.

A cada passo do ciclo, indivíduos são selecionados de acordo com o seu *fitness* para fornecerem material genético para a próxima geração – isto é, para serem os pais – sendo a probabilidade de seleção diretamente proporcional ao valor do *fitness*. Em seguida, os indivíduos selecionados se reproduzem par-a-par através do operador de *crossover*, que seleciona parte do material genético de cada pai para formar os filhos que comporão a próxima geração. Estes filhos são ainda manipulados por um operador de mutação, utilizado para introduzir diversidade na nova população. Ao final da reprodução os pais são totalmente ou parcialmente substituídos pelos seus filhos, dando origem a uma nova geração. O processo é então repetido, até atingir um critério de parada previamente estabelecido.

Algoritmo 2.4 Pseudo-código do Algoritmo Genético

```
1: função GA(instância, tamanhoPopulação)
2:   população =  $\emptyset$ 
3:   para  $i = 1$  até tamanhoPopulação faça
4:     população.adicionar(gerarSolução(instância))
5:   fim para
6:   funçãoObjetivo(instância, população)
7:   repita
8:     pais = Selecionar(população)
9:     filhos = Reproduzir(pais)
10:    filhos = Mutação(filhos)
11:    funçãoObjetivo(instância, filhos)
12:    população = CriarPróximaGeração(filhos, pais)
13:  até critério de parada
14:  retorne melhor solução na população
15: fim função
```

Conforme já foi mencionado, cada indivíduo da população representa uma das soluções possíveis no espaço de busca do problema. A forma de representação de um indivíduo depende do tipo do problema e do que se deseja manipular, sendo a representação binária, com um vetor de zeros e uns, a mais simples e mais frequentemente utilizada. O GA já foi aplicado com sucesso a diversos problemas reais e complexos, sendo um dos algoritmos baseados em população mais estudados na literatura [10].

2.3 Algoritmos Heurísticos aplicados ao NRP

Esta seção apresenta heurísticas e outras abordagens previamente utilizadas na literatura para tratar o NRP. Os trabalhos sobre o NRP podem ser divididos em duas categorias baseadas no número de objetivos que compõem a formulação do problema: mono-objetivo e bi-objetivo. Em formulações mono-objetivo, o orçamento disponível para uma versão do software é pré-definido e o objetivo é obter o maior lucro possível através da implementação dos requisitos sem ultrapassar o limite orçamentário proposto. Nas formulações bi-objetivo, o limite do custo de implementação não é definido *a priori*, mas tratado como um segundo objetivo, a ser minimizado.

2.3.1 Formulação Mono-objetivo do NRP

A maioria dos trabalhos relacionados ao NRP utiliza a formulação mono-objetivo, que foi proposta por Bagnall et al. [2] em 2001. Em seu trabalho, o NRP foi modelado, instâncias do problema foram criadas e resolvidas por quatro algoritmos distintos: programação linear, GRASP, HC e *Simulated Annealing* (SA). Neste trabalho, os autores concluíram que algoritmos de otimização exatos são suficientes para instâncias com poucos clientes e requisitos. Contudo, para instâncias grandes, o SA mostrou-se mais eficiente. As instâncias utilizadas nesta análise tinham entre 100 e 500 clientes e entre 140 e 3250 requisitos. Freitas et al. [13] utilizaram o método exato Simplex para obter soluções para o NRP que superaram os resultados de um GA e de um SA para instâncias consideravelmente menores, com até 100 clientes e 200 requisitos.

Tonella et al. [14] desenvolveram um GA interativo que utiliza entradas dos usuários durante o processo de otimização para guiar o procedimento de busca. Os autores compararam sua proposta com uma busca aleatória e com um GA tradicional, que não utiliza entradas do usuário. Para a comparação dos algoritmos, foi utilizada uma instância real do problema com 49 requisitos. Os autores relataram que o GA interativo superou os resultados dos dois outros algoritmos, mas destaca-

ram que a utilização de apenas uma instância do problema dificulta a generalização dos resultados. Souza et al. [15] desenvolveram um algoritmo de Colônia de Formigas (*Ant Colony Optimization – ACO*), que se inspira na comunicação indireta (via trilhas de feromônios) realizada por formigas durante a busca por comida. O algoritmo proposto foi comparado com um SA e um GA, utilizando 72 instâncias sintéticas, e obteve resultados superiores na grande maioria dos casos, apesar de apresentar um tempo de execução até 90 vezes maior que os demais algoritmos em alguns casos.

Jiang et al. [16] avaliaram o uso de um algoritmo híbrido que combina um HC com o ACO. Esse algoritmo híbrido superou os algoritmos GRASP, HC, SA e ACO ao ser executado em instâncias sintéticas derivadas do trabalho de Bagnall et al [2]. Ngo-The e Ruhe [17] propuseram um processo de otimização composto por duas etapas, que utiliza Programação Inteira para reduzir o espaço de busca do problema antes de executar um GA. Os autores compararam sua abordagem híbrida com um algoritmo guloso simples, obtendo resultados superiores em todas as 600 instâncias sintéticas utilizadas. Del Sagrado et al. [18] avaliaram os algoritmos GRASP, GA e ACO utilizando duas instâncias, uma com 20 requisitos e 5 clientes e outra com 100 requisitos e 5 clientes, para as quais o GRASP obteve os melhores resultados e o menor tempo de execução.

Xuan et al. [1] usaram uma estrutura chamada de *backbone* para diminuir a escala do problema no seu algoritmo conhecido como *Backbone-Based Multilevel Algorithm* (BMA). O BMA identifica e fixa partes de boas soluções durante o processo de otimização, de forma a obter soluções de alta qualidade para instâncias grandes em um tempo computacional aceitável. Após reduzir a escala do problema ao construir o *backbone*, o algoritmo prossegue refinando a solução. Este algoritmo superou tanto um GA quando um SA com reinícios, ou *Multi-start Strategy-based SA* (MSSA). O MSSA é uma variação do SA em que o algoritmo clássico do SA é executado múltiplas vezes, de forma independente, e a melhor solução dentre as execuções é escolhida como solução final. A fim de verificar o desempenho de cada algoritmo, os autores criaram dois conjuntos de instâncias. O primeiro conjunto foi criado baseado nas instâncias utilizadas no trabalho de Bagnall et al. [2], enquanto o segundo conjunto foi gerado a partir de repositórios de bugs de três projetos reais de software livre. Estes mesmos conjuntos de instâncias serão utilizados no estudo experimental relatado no Capítulo 4, bem como em avaliações preliminares apresentadas na Seção 3.2.

2.3.2 Formulação Bi-objetivo do NRP

Zhang et al. [3] propuseram a formulação bi-objetivo do NRP e apresentaram três versões de GA, das quais a versão baseada no algoritmo NSGA-II se destacou. O NSGA-II[19] é uma variação do GA simples adaptado para tratar de mais de um objetivo. Este algoritmo foi capaz de gerar soluções comparáveis com as obtidas por abordagens mono-objetivo anteriormente usadas pela literatura. O algoritmo gerou ainda um conjunto de soluções não-dominadas que visam ajudar o tomador de decisões, que pode escolher dentre as soluções dependendo das prioridades existentes no momento. Uma solução é dita não-dominada se não é possível melhorar um de seus objetivos sem deteriorar ao menos um outro objetivo. As soluções não-dominadas formam um conjunto chamado de Fronteira de Pareto, que representa os compromissos existentes entre os diversos objetivos conflitantes de um problema com mais de um objetivo.

Durillo et al. [20] compararam o NSGA-II com o MOCcell[21], outro GA adaptado para resolver problemas com mais de um objetivo. Foram utilizados dois indicadores para avaliar a qualidade dos resultados: *spread* [22] e *hipervolume* [23]. Segundo os autores, estes são, respectivamente, indicadores de diversidade e de convergência. Neste contexto, um algoritmo que apresente alta diversidade gera soluções que se espalham amplamente pelo espaço de soluções ao invés de se concentrarem em uma pequena região deste espaço. Convergência, por outro lado, implica em que o algoritmo tenha gerado soluções próximas às soluções ótimas. Como usualmente estas soluções ótimas não são conhecidas *a priori*, elas são computadas a partir das melhores soluções encontradas por todos os algoritmos aplicados ao problema que está sendo examinado. A qualidade de um algoritmo é inversamente proporcional ao *spread* do seu conjunto de soluções não-dominadas e diretamente proporcional ao seu *hipervolume* deste conjunto. Neste estudo, o MOCcell produziu resultados com menor *spread*, enquanto o NSGA-II obteve o maior *hipervolume* para o NRP.

O trabalho de Saliu and Ruhe [24] estuda o Problema de Planejamento de *Releases* (RP) com o foco na interação entre requisitos e restrições de implementação. Enquanto o NRP considera apenas uma *release* de software, o RP planeja uma série de versões. Portanto, o NRP é um caso especial do RP. Em seu trabalho, os autores propõem uma técnica para detectar o acoplamento entre requisitos com base em uma análise de impacto de mudanças. Os requisitos com grande acoplamento são mantidos na mesma versão porque, segundo os autores, isto diminui o esforço cognitivo necessário para implementá-los.

Recentemente, Zhang et al. [25] publicaram um estudo comparando o resultado de diversas heurísticas quando aplicadas ao RP bi-objetivo, utilizando 10 conjuntos de instâncias diferentes. Neste trabalho, os autores também investigam o uso

de hiper-heurísticas [26] para resolver o RP. As hiper-heurísticas encapsulam informações específicas do problema sendo resolvido em um conjunto de heurísticas, chamadas de operadores de busca. Esta abordagem difere das heurísticas por percorrer o espaço de busca dos operadores de busca, ao invés de percorrer o espaço de busca do problema. Ela busca automatizar o processo de selecionar, combinar, gerar ou adaptar diversas heurísticas mais simples, de forma a resolver um problema de forma eficiente. No trabalho de Zhang et al. [25], uma hiper-heurística baseada no NSGA-II obteve os melhores resultados em termos de qualidade das soluções e tempo de execução.

Finkelstein et al. [27] propõem 3 medidas de equilíbrio (*fairness*) para serem utilizadas ao escolher os requisitos para a próxima versão de forma a atender de forma igualitária cada um dos clientes. A primeira medida procura maximizar o número de requisitos implementados para cada cliente, enquanto minimiza o desvio padrão entre esses valores. A segunda medida é similar à primeira, mas ao invés de utilizar o número de requisitos implementados, considera o valor do benefício gerado para cada cliente. A terceira medida é mais complexa, sendo formada por 4 objetivos: minimizar o desvio padrão dos custos de implementação dos requisitos de cada cliente, minimizar o desvio padrão dos benefícios gerados para cada cliente, maximizar o benefício gerado para cada cliente e, por fim, minimizar o custo geral da próxima versão. Tais medidas permitiram aos autores observar pontos de tensão nas instâncias utilizadas no estudo. Os autores observaram, ainda, que quanto mais requisitos são selecionados, menos os clientes são atendidos de forma igualitária.

Em sua revisão sistemática, Pitangueira et al. [28] listaram trinta e dois algoritmos que foram aplicados ao NRP. Como frequentemente observado em abordagens baseadas em busca para resolução de problemas de Engenharia de Software, a maioria dos autores utilizou GA e diversas variações de buscas locais para resolver as formulações mono e bi-objetivo do NRP.

2.4 Considerações Finais

Este capítulo apresentou a formulação do NRP, o algoritmo de busca local de Subida de Encosta e os Algoritmos Genéticos. Apresentou também o algoritmo de Busca Local Iterada (ILS), uma extensão da busca local convencional. Em seguida, passou-se para uma revisão da literatura que abordou a utilização de algoritmos heurísticos para resolver o NRP, onde foi possível perceber que os dois primeiros algoritmos supracitados foram amplamente aplicados para encontrar soluções para este problema. No entanto, não foi encontrada na literatura nenhuma aplicação do ILS para resolver o NRP.

No próximo capítulo apresentaremos um padrão visual que foi identificado em todas as instâncias frequentemente utilizadas nos estudos do NRP e utilizaremos este padrão para orientar um algoritmo de ILS para encontrar soluções para o problema. Até onde se sabe, este trabalho é o primeiro a explorar mecanismos de visualização para aprimorar um algoritmo de busca local e a utilizar o ILS para encontrar soluções para o NRP.

3. Solução Proposta

Este capítulo apresenta o padrão gráfico encontrado durante o estudo do espaço de busca do NRP, bem como resultados de estudos experimentais preliminares que levaram à criação da Busca Local Iterada orientada por Visualização (VisILS), uma modificação do ILS que utiliza características do padrão gráfico observado para diminuir o tamanho do espaço de busca. Por fim, o algoritmo do VisILS é apresentado.

3.1 Visualização do Espaço de Busca do NRP

Visualização é o processo de transformar informação em uma representação visual, fazendo com que usuários possam observar essa informação [29]. A exposição resultante permite identificar visualmente características e comportamentos que podem estar ocultos nos dados originais e que podem ser úteis na sua exploração e análise.

A utilidade da visualização está relacionada com as propriedades físicas e com a especialização do cérebro humano: nosso cérebro é constituído por duas unidades de processamento, cada uma localizada em um dos hemisférios. Enquanto o hemisfério esquerdo é responsável pelo raciocínio verbal, analítico, racional, temporal e sequencial, o hemisfério direito ocupa-se do raciocínio não-verbal, sintético, intuitivo, não-temporal e paralelo. Técnicas de visualização exploram a capacidade do cérebro ao integrar os dois hemisférios [30].

No contexto do SBSE, a visualização do espaço de busca de certos problemas pode ser utilizada para desenvolver algoritmos de otimização heurística que se aproveitam de características desses espaços de busca [31]. Harman [31] destacou a visualização do espaço de busca como um dos problemas abertos na área de otimização em Engenharia de Software. Em seu trabalho, Lu et al. [32] afirmam que a caracterização do espaço de busca é uma ferramenta importante para permitir o desenvolvimento da área de SBSE, dado que esta caracterização captura a natureza do relacionamento entre o problema e os algoritmos empregados para resolvê-lo.

O gráfico apresentado na Figura 3.1 é resultado da investigação realizada sobre o espaço de busca do NRP. Este gráfico foi construído a partir de soluções do NRP geradas de forma aleatória. Para cada número possível de clientes, de um até o

número total de clientes presentes na instância (n), foram geradas 100 soluções aleatórias.

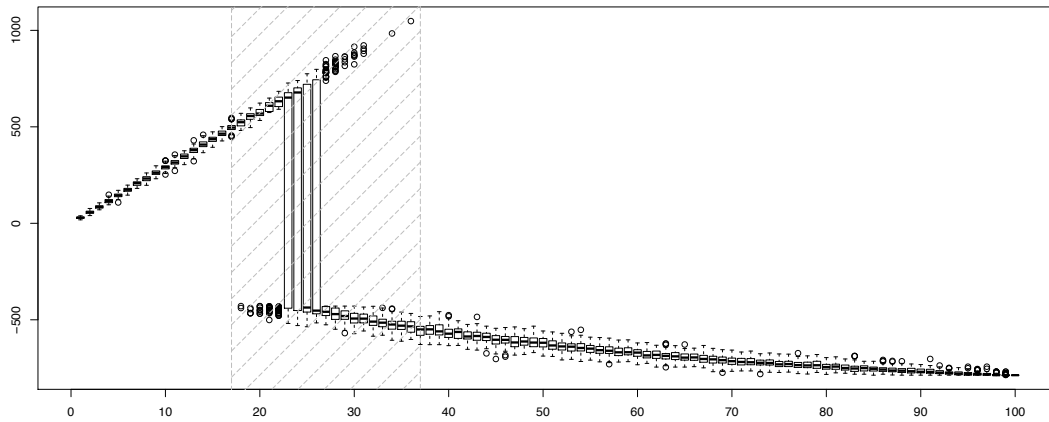


Figura 3.1: Amostragem aleatória da instância nrp-1 com orçamento de 50% do custo de implementar todos os requisitos

O Algoritmo 3.1 apresenta o pseudo-código do algoritmo utilizado para gerar este gráfico. O *loop* externo itera sobre o número possível de clientes atendidos pela solução (de 1 a n), enquanto o *loop* interno cria um número pré-definido de soluções, cada uma com um determinado número de clientes atendidos. A linha 6 é responsável pela criação das soluções. Cada solução é criada pela função *gerarSoluçãoCom()*, uma variação da função *gerarSolução()* que cria soluções com um determinado número de clientes. Um fator aleatório é utilizado para selecionar um conjunto diferente de clientes para cada solução gerada durante o processo. Cada solução criada é armazenada em uma lista, que é retornada pelo processo de amostragem aleatória ao seu final.

Algoritmo 3.1 Pseudo-código para o algoritmo de Amostragem Aleatória

```

1: função AMOSTRAGEMALEATÓRIA(instância, tamanho)
2:   rsols =  $\emptyset$ 
3:   clientes = contarClientes(instância)
4:   para nCliente = 1 até clientes faça
5:     para i = 1 até tamanho faça
6:       s = gerarSoluçãoCom(nCliente, instância)
7:       rsols.adicionar(s)
8:     fim para
9:   fim para
10:  retorne rsols
11: fim função

```

Cada coluna no gráfico exibido na Figura 3.1 é um box-plot representando a qualidade em termos de *fitness* das soluções criadas pelo processo de amostragem aleatória para um determinado número de clientes. O eixo x representa o número

de clientes atendidos por uma solução (isto é, o número de clientes cujos requisitos foram incluídos na solução), enquanto o eixo y representa a qualidade da mesma. É possível observar um aumento na qualidade das soluções quando o número de clientes atendidos aumenta, até um ponto onde apenas soluções que ultrapassam o orçamento disponível, portanto inviáveis, são encontradas. Estas soluções inviáveis são representadas no gráfico por valores negativos de *fitness*. Existe, ainda, uma área de transição, demarcada por um retângulo rachurado, onde uma mistura de soluções inviáveis e viáveis pode ser observada.

O mesmo padrão gráfico foi observado em todas as instâncias do NRP utilizadas por Bagnall et al. [2] e por Xuan et al. [1], conforme mostrado nas Figuras 3.2 e 3.3. O presente trabalho explora esta propriedade do espaço de busca do NRP para melhorar os resultados obtidos por algoritmos de busca heurística. Através do uso desta informação, espera-se melhorar os resultados obtidos por algoritmos de busca simples, como buscas locais e suas variantes, aproximando-os dos resultados obtidos por heurísticas mais complexas.

3.2 Busca Local orientada por Visualização

A fim de avaliar a utilidade desta descoberta sobre o espaço de busca do NRP no sentido de melhorar o comportamento de um algoritmo de busca heurística, foram executados experimentos preliminares para comparar o desempenho de um HC clássico (conforme descrito na Seção 2.2.1) com uma versão modificada deste algoritmo que se utiliza do padrão gráfico para remover do processo de otimização partes do espaço de busca que não contém soluções interessantes. A comparação entre os dois algoritmos foi realizada com base na qualidade da melhor solução encontrada por eles. O HC modificado foi chamado de Subida de Encosta orientada por Visualização (*Visual Hill Climbing - VisHC*). O HC foi escolhido para esta primeira avaliação por ser um algoritmo de fácil implementação, fator importante ao se tratar de um estudo com o propósito de direcionar o foco da pesquisa, bem como de validar a utilidade do padrão de visualização recém-descoberto.

O VisHC difere do HC clássico apresentado na Seção 2.2.1 de duas formas. Primeiramente, o algoritmo de amostragem aleatória é executado no início do VisHC, onde diversas soluções com N clientes atendidos são geradas, com N variando de um até o número total de clientes presentes na instância. A solução de maior *fitness* é salva como melhor solução encontrada até o momento e o número de clientes atendidos por esta solução (N^*) é armazenado para ser utilizado por etapas seguintes do algoritmo. As funções que geram uma nova solução (*gerarSoluçãoCom*) e que realizam a busca local (*BuscaLocalCom*) também diferem das utilizadas pelo HC, pois não visitam soluções que estejam abaixo do limite (N^*) definido pela etapa

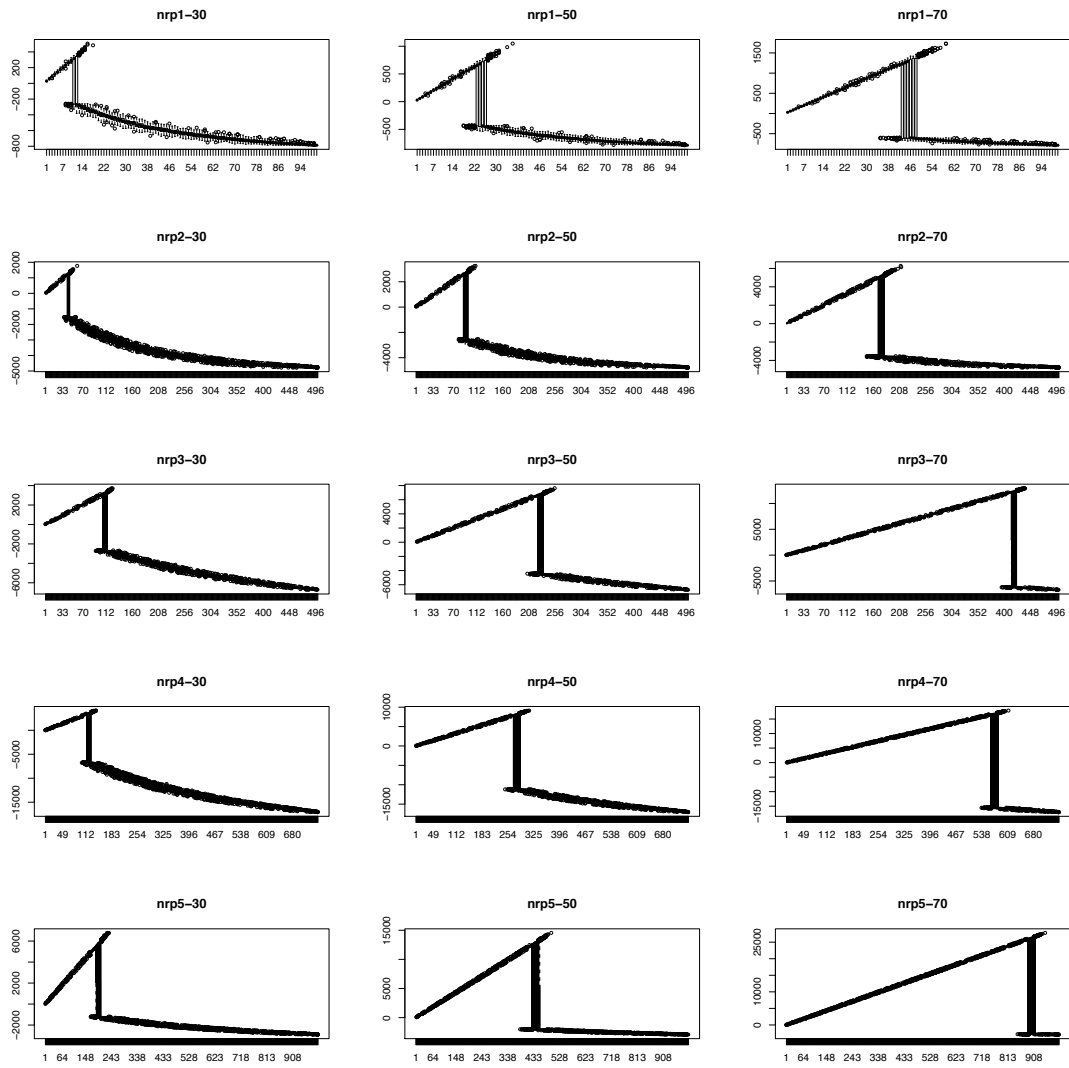


Figura 3.2: Amostragem Aleatória para instâncias adaptadas por Xuan et al. [1] do trabalho de Bagnall et al. [2]

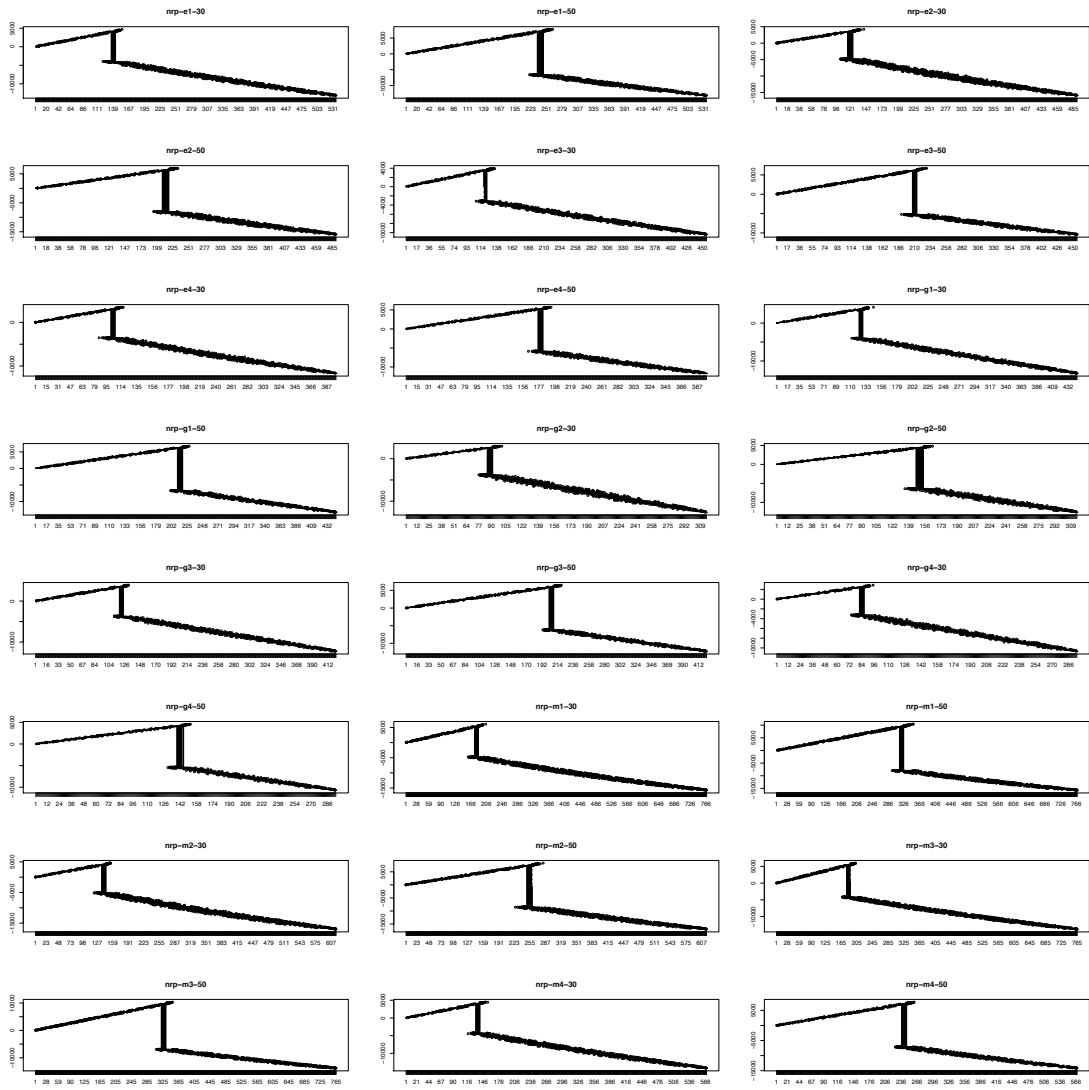


Figura 3.3: Amostragem Aleatória das instâncias geradas por Xuan et al. [1]

de amostragem aleatória. Esses conceitos serão apresentados com mais detalhes na próxima seção, que trata do VisILS, outra heurística adaptada para fazer uso do padrão gráfico descrito anteriormente. O pseudo-código do VisHC é apresentado no Algoritmo 3.2.

Algoritmo 3.2 Pseudo-código da Subida de Encosta orientada por Visualização

```

1: função VisHC(instância, tamanho)
2:   rsols = AmostragemAleatória(instância, tamanho)
3:    $N^* = 0$ 
4:    $bf = -\infty$ 
5:   para cada s em rsols faça
6:     se funçãoObjetivo(instância, s) > bf então
7:        $N^* = \text{contarClientes}(s)$ 
8:        $bf = \text{funçãoObjetivo}(instância, s)$ 
9:     fim se
10:  fim para
11:  repita
12:     $s_0 = \text{gerarSoluçãoCom}(N^*, instância)$ 
13:     $s'_0 = \text{BuscaLocalCom}(N^*, s_0, instância)$ 
14:    se funçãoObjetivo(instância,  $s'_0$ ) > bf então
15:       $s_* = s'_0$ 
16:       $bf = \text{funçãoObjetivo}(instância, s'_0)$ 
17:    fim se
18:  até critério de parada
19:  retorne  $s_*$ 
20: fim função

```

Neste experimento, cada algoritmo foi executado 30 vezes para cada instância. Cada execução resultou em uma única solução, cujo *fitness* foi capturado. Foram utilizados dois conjuntos de instâncias, anteriormente utilizadas por Bagnall et al. [2] e Xuan et al. [1]. O Capítulo 4 apresenta em mais detalhes as instâncias e a configuração dos parâmetros utilizados para configurar os algoritmos no experimento (dado que estes parâmetros foram utilizados na avaliação principal que compõe este trabalho).

O VisHC apresentou resultados melhores que o HC em 25 das 39 instâncias utilizadas, com uma melhoria de qualidade média de pouco menos de 1%. Os resultados obtidos pelo VisHC são, em média, 33% piores do que os resultados obtidos pelo BMA [1], uma diferença grande porém esperada, dada a simplicidade dos algoritmos de busca local. Apesar de ainda muito distante dos resultados obtidos por heurísticas mais avançadas, como o BMA, esta avaliação preliminar mostrou que o VisHC foi capaz de melhorar os resultados de um HC clássico. Tal resultado, promissor ainda que modesto, levou à criação do VisILS, algoritmo baseado no ILS que será apresentado na Seção 3.3.

3.2.1 Número Máximo de Clientes

Conforme descrito anteriormente, o VisHC possui um valor determinado pela fase de amostragem aleatória que indica o número mínimo de clientes (N^*) que devem estar presentes em uma solução para que a mesma possa ser considerada pelo algoritmo de busca. Esta seção apresenta um estudo realizado sobre um conceito similar, que indica o número máximo de clientes que devem estar presentes nas soluções visitadas pelo algoritmo de busca. Com a adição deste novo limite, o algoritmo passa apenas a visitar soluções cujo número de clientes satisfeitos esteja no intervalo $[N^*, \min(N^* + N * \alpha, N)]$, onde N^* é o número mínimo de clientes, N é o número de clientes na instância e $\alpha \leq 1$ é um parâmetro do algoritmo. Por exemplo, se a melhor solução encontrada durante a fase de amostragem aleatória possuir 15 clientes atendidos e a instância tiver 100 clientes no total, o VisHC visitará apenas soluções cujo número de clientes atendidos esteja dentro do intervalo $[15, \min(15 + 100 * \alpha, 100)]$.

A fim de determinar o melhor valor para o parâmetro α , foi realizado um experimento comparando quatro configurações, que diferem apenas pelo valor atribuído ao α . São elas: VisHC, que é o algoritmo descrito anteriormente e que não impõe limite quanto ao número máximo de clientes atendidos por uma solução ($\alpha = 1.0$), VisHC-10 que possui $\alpha = 0.1$, VisHC-20 onde $\alpha = 0.2$ e, finalmente, VisHC-40, que utiliza $\alpha = 0.4$. As Tabelas 3.1 e 3.2 apresentam os resultados de cada uma das configurações. O HC também foi incluído para enriquecer a comparação. A Tabela 3.1 mostra a qualidade (*fitness*) da melhor solução encontrada por cada configuração, enquanto a Tabela 3.2 apresenta a qualidade média obtida por cada configuração ao longo de 30 execuções. Em ambas as tabelas, valores em negrito indicam o maior valor encontrado dentre todas as configurações.

Observando a Tabela 3.1, que apresenta o *fitness* da melhor solução obtida por cada configuração entre as 30 execuções, percebe-se que o HC foi superado por ao menos uma variação do VisHC em todas as instâncias, exceto uma. A configuração VisHC-10, a mais restritiva, obteve a melhor solução em 14 das 39 instâncias, sendo a maioria delas do conjunto de instâncias realistas (nrp-e1-30 em diante). Já a configuração VisHC, que não define um limite para o número máximo de clientes na solução, obteve a melhor solução em 12 das 39 instâncias, a maioria no conjunto de instâncias sintéticas. As configurações VisHC-20 e VisHC-40 obtiveram, respectivamente, 7 e 5 vezes o melhor resultado.

Já a Tabela 3.2 exibe a média das soluções obtidas por cada configuração ao longo de 30 execuções. Observa-se que o HC obteve a melhor média em apenas dois casos, sendo superado por ao menos uma variação do VisHC nas demais instâncias. A configuração VisHC-10 obteve a melhor média em 25 das 39 instâncias e, assim como aconteceu com a melhor solução, esses resultados concentraram-se nas instâncias

Tabela 3.1: Qualidade da melhor solução obtida por cada configuração ao longo das 30 execuções.

Instância	HC	VisHC-40	VisHC-20	VisHC-10	VisHC
nrp1-30	1072	1081	1088	896	1178
nrp2-30	3463	3331	3301	3166	4310
nrp3-30	5026	4980	4985	4993	5129
nrp4-30	5951	5946	5927	5991	5847
nrp5-30	14116	14355	13002	10198	14853
nrp1-50	1732	1714	1651	1434	1774
nrp2-50	6239	6322	6180	5089	7418
nrp3-50	9030	9026	9187	9044	9086
nrp4-50	11684	11841	12028	11724	11935
nrp5-50	21905	22479	20873	17891	22837
nrp1-70	2455	2467	2446	2244	2495
nrp2-70	9796	10478	9477	8240	10898
nrp3-70	13693	13838	13751	13761	13729
nrp4-70	19390	19444	19443	19501	19572
nrp5-70	28678	28737	28703	28707	28681
nrp-e1-30	5260	5197	5179	5270	5194
nrp-e2-30	4722	4814	4715	4815	4720
nrp-e3-30	4416	4460	4600	4456	4385
nrp-e4-30	3819	3892	3868	3897	3752
nrp-g1-30	4329	4445	4378	4431	4260
nrp-g2-30	3322	3278	3303	3340	3238
nrp-g3-30	4257	4212	4274	4248	4153
nrp-g4-30	3048	3064	3111	3168	3003
nrp-m1-30	6503	6380	6532	6493	6583
nrp-m2-30	5279	5246	5372	5298	5047
nrp-m3-30	6388	6674	6619	6480	6445
nrp-m4-30	4915	4889	4947	5113	4809
nrp-e1-50	8426	8411	8394	8378	8459
nrp-e2-50	7662	7712	7709	7774	7665
nrp-e3-50	7202	7043	7222	7155	7090
nrp-e4-50	6236	6150	6330	6279	6292
nrp-g1-50	7094	7042	7109	7114	6958
nrp-g2-50	5200	5282	5234	5249	5151
nrp-g3-50	6806	6646	6758	6838	6746
nrp-g4-50	4944	4873	4887	5010	4865
nrp-m1-50	10953	10948	11007	11124	11076
nrp-m2-50	8843	8685	8847	8976	8924
nrp-m3-50	11069	10906	11062	11111	10916
nrp-m4-50	8402	8211	8317	8391	8245

Tabela 3.2: Qualidade média das soluções obtidas por cada configuração ao longo de 30 execuções.

Instância	HC	VisHC-40	VisHC-20	VisHC-10	VisHC
nrp1-30	1024.5	1029.0	1020.2	836.4	1127.9
nrp2-30	3237.6	3164.4	3106.3	3057.2	4116.7
nrp3-30	4710.8	4779.0	4814.3	4866.5	4736.0
nrp4-30	5509.2	5599.6	5677.3	5759.9	5562.0
nrp5-30	13797.3	13969.7	12785.1	9903.8	14627.3
nrp1-50	1629.5	1658.5	1578.5	1320.4	1715.6
nrp2-50	5995.4	6094.4	5852.7	4830.0	7190.3
nrp3-50	8893.5	8883.9	8958.1	8935.9	8858.0
nrp4-50	11423.4	11400.5	11490.3	11386.6	11263.6
nrp5-50	21597.6	22097.7	20424.6	17603.6	22380.2
nrp1-70	2413.6	2416.4	2343.6	2107.5	2400.0
nrp2-70	9623.6	10100.3	9093.8	7685.0	10851.2
nrp3-70	13515.9	13661.8	13643.0	13651.4	13637.1
nrp4-70	19152.1	19280.7	19237.8	19382.4	19287.3
nrp5-70	28556.7	28638.4	28632.1	28622.3	28626.6
nrp-e1-30	4895.3	4903.1	4988.2	5097.3	4856.2
nrp-e2-30	4477.0	4486.5	4515.4	4617.3	4420.8
nrp-e3-30	4191.4	4227.8	4291.5	4340.4	4127.5
nrp-e4-30	3638.9	3635.3	3739.1	3796.7	3580.0
nrp-g1-30	4184.7	4199.7	4255.5	4305.1	4159.2
nrp-g2-30	3098.7	3143.6	3155.5	3227.6	3081.4
nrp-g3-30	4021.8	4061.8	4114.5	4164.9	4015.3
nrp-g4-30	2918.1	2942.0	2997.5	3032.7	2904.3
nrp-m1-30	6143.0	6124.4	6266.8	6346.9	6124.4
nrp-m2-30	4954.8	5016.0	5051.2	5114.0	4828.6
nrp-m3-30	6184.5	6214.9	6334.2	6347.4	6101.8
nrp-m4-30	4663.1	4680.1	4744.5	4864.9	4634.9
nrp-e1-50	8232.1	8102.1	8184.6	8262.9	8025.0
nrp-e2-50	7499.4	7297.8	7385.9	7562.3	7264.2
nrp-e3-50	7026.2	6858.6	6977.2	7028.8	6853.7
nrp-e4-50	6127.9	5977.1	6092.5	6130.2	5967.2
nrp-g1-50	6964.8	6827.9	6917.2	6954.4	6797.4
nrp-g2-50	5110.2	5009.1	5076.0	5115.8	4961.2
nrp-g3-50	6705.1	6555.3	6611.5	6683.4	6530.9
nrp-g4-50	4817.1	4690.5	4773.3	4825.8	4711.3
nrp-m1-50	10791.4	10629.6	10695.9	10895.7	10547.5
nrp-m2-50	8617.8	8455.0	8563.6	8713.8	8379.8
nrp-m3-50	10786.7	10562.0	10703.9	10824.3	10553.3
nrp-m4-50	8126.3	7950.1	8085.8	8148.5	7951.7

realistas. A configuração VisHC obteve a melhor média em 7 instâncias, todas pertencentes ao grupo de instâncias sintéticas. Já as configurações VisHC-20 e VisHC-40 obtiveram, respectivamente, 2 e 3 vezes as melhores médias.

Conforme pode ser observado nas Tabelas 3.1 e 3.2, a configuração sem restrição quanto ao número máximo de clientes atendidos pela solução (VisHC) obteve bons resultados para o grupo de instâncias sintéticas, enquanto que o seu desempenho no grupo de instâncias realistas foi aquém do esperado. Já a configuração mais restritiva (VisHC-10) apresentou o comportamento contrário: obteve resultados ruins no grupo de instâncias sintéticas e os melhores resultados nas instâncias realistas. Não foi possível identificar a melhor configuração para o parâmetro α , que define o número máximo de clientes que podem estar presentes na solução, para que a mesma seja considerada pelo algoritmo. Por isto, o algoritmo VisILS apresentado a seguir não utiliza este conceito. Novos estudos serão necessários para se determinar o melhor valor para tal limite que, conforme os resultados mostram, pode afetar consideravelmente a qualidade das soluções geradas. A execução deste estudo é um ponto de possível extensão do presente trabalho.

3.3 Busca Local Iterada orientada por Visualização

Esta seção introduz uma versão modificada de um ILS que utiliza o padrão gráfico apresentado previamente para reduzir o espaço de busca de instâncias do NRP, concentrando a busca em regiões mais promissoras e descartando regiões cujas soluções apresentam baixa qualidade. O ILS foi escolhido para este estudo por ser uma extensão natural do HC que apresenta resultados competitivos para diversos problemas da literatura. Soma-se a isto o fato do ILS nunca ter sido aplicado ao NRP.

A Busca Local Iterada orientada por Visualização (*Visual Iterated Local Search* - *VisILS*) foi construída utilizando como base o algoritmo ILS apresentado na Seção 2.2.2. O Algoritmo 3.3 apresenta o pseudo-código do VisILS. Inicialmente, uma fase de amostragem aleatória é executada. Nesta fase, diversas soluções aleatórias com N clientes atendidos são geradas, com N variando de um até o número total de clientes presentes na instância. A solução de maior *fitness* é salva como a melhor solução e o número de clientes atendidos por esta solução (N^*) é armazenado.

O N^* funciona como uma restrição para o VisILS: o algoritmo somente pode visitar soluções cujo número de clientes atendidos esteja compreendido no intervalo $[N^*, n]$, onde n é o número de clientes presentes na instância. Por exemplo, caso a melhor solução encontrada durante a amostragem aleatória atenda a 25 clientes e a instância tenha 100 clientes no total, o algoritmo irá considerar apenas soluções

Algoritmo 3.3 Pseudo-código da Busca Local Iterada orientada por Visualização

```
1: função VisILS(instância, intensidadePerturbação, tamanho)
2:   rsols = AmostragemAleatória(instância, tamanho)
3:    $N^* = 0$ 
4:    $bf = -\infty$ 
5:   para cada s em rsols faça
6:     se funçãoObjetivo(instância, s) > bf então
7:        $N^* = \text{contarClientes}(s)$ 
8:        $bf = \text{funçãoObjetivo}(instância, s)$ 
9:     fim se
10:  fim para
11:   $s_0 = \text{gerarSoluçãoCom}(N^*, instância)$ 
12:   $s_* = \text{BuscaLocalCom}(N^*, s_0, instância)$ 
13:  repita
14:     $s' = \text{PerturbarCom}(N^*, s_*, intensidadePerturbação)$ 
15:     $s'_* = \text{BuscaLocalCom}(N^*, s', instância)$ 
16:     $s_* = \text{Aceitar}(s_*, s'_*)$ 
17:  até critério de parada
18:  retorne  $s_*$ 
19: fim função
```

cujo número de clientes atendidos esteja no intervalo $[25, 100]$. Esta restrição faz com que o VisILS se concentre em uma região onde, durante a fase de amostragem aleatória, soluções de maior qualidade foram encontradas.

É importante destacar que a função *BuscaLocalCom()* utilizada pelo VisILS difere levemente do Algoritmo 2.2. Esta variante recebe um parâmetro adicional, N^* , que é determinado durante a fase de amostragem aleatória e que é utilizado para garantir que a busca local apenas visite soluções cujo número de clientes atendidos seja igual ou maior ao valor armazenado em N^* . Mais precisamente, a função *InverteCliente()* verifica se um cliente pode ser adicionado ou removido da solução sem violar o limite imposto pelo N^* . A mesma lógica se aplica às funções *gerarSoluçãoCom()* e *PerturbarCom()*.

Após a fase de amostragem aleatória, uma solução inicial é criada e o *loop* principal se inicia. Este *loop* otimiza a solução inicial usando uma busca local. A busca local procura por soluções de melhor qualidade através da adição ou remoção de um único cliente da solução corrente por vez, sempre respeitando a restrição de apenas visitar soluções cujo número de clientes atendidos seja igual ou maior que o limite definido durante a fase de amostragem aleatória.

Sempre que uma solução de maior qualidade é encontrada, a mesma é armazenada como a nova melhor solução encontrada e a sua vizinhança é explorada em busca de soluções ainda melhores. Se nenhuma alteração nos clientes selecionados for capaz de melhorar o *fitness*, uma perturbação é aplicada à solução corrente. O operador de perturbação foi construído de forma a somente retornar soluções que respeitem o limite definido pela fase de amostragem aleatória. O critério de aceitação foi configurado para apenas aceitar soluções cuja qualidade seja superior

à qualidade da solução corrente. Por fim, a busca é encerrada após atingir um determinado número de avaliações de *fitness*.

Uma questão que pode ser levantada é a que não existem garantias de que a solução ótima estará localizada após a melhor solução encontrada pela fase de amostragem aleatória, responsável pela definição de N^* , o número mínimo de clientes que devem ser atendidos por uma solução para que a mesma seja considerada pelo algoritmo do VisILS. Contudo, o formato do espaço de busca observado pela perspectiva do número de clientes em cada solução indica claramente uma tendência crescente de soluções viáveis cada vez melhores, do ponto de vista do *fitness*, até um ponto em que nenhuma solução é viável. Assumindo que esta tendência seja monótona não-decrescente, o que é sugerido pelos gráficos apresentados nas Figuras 3.2 e 3.3, a solução de maior qualidade deve ser a melhor solução encontrada durante a fase de amostragem aleatória ou alguma solução à sua direita. Deste modo, apesar de não existir certeza de que a solução ótima está à direita da melhor solução obtida pela amostragem aleatória, isto será assumido como verdade a fim de limitar o espaço de busca a ser percorrido pelo procedimento de busca.

3.4 Considerações Finais

Este capítulo apresentou o padrão gráfico encontrado nas instâncias do NRP analisadas e descreveu duas heurísticas que foram adaptadas para utilizar este conhecimento de forma a diminuir o espaço de busca do problema. Comentou-se, ainda, sobre um estudo experimental preliminar utilizando o VisHC, que obteve soluções em média 1% melhores que as obtidas pelo HC clássico. O resultado deste estudo serviu para verificar a utilidade do padrão de visualização no sentido de reduzir o espaço de busca a ser percorrido pelos algoritmos heurísticos e deu origem ao VisILS, que também foi apresentado no presente capítulo. A tentativa de definição de um número máximo de clientes atendidos também foi discutida. O próximo capítulo apresenta um estudo mais completo do VisILS, comparando seus resultados com o BMA, uma heurística que representa o estado-da-arte em buscas heurísticas aplicadas ao NRP.

4. Avaliação Experimental

Visando avaliar se o algoritmo VisILS, apresentado no capítulo anterior, é capaz de aumentar a eficiência e eficácia da busca por soluções do NRP, instâncias do problema serão otimizadas por três processos de busca, que serão então comparados quanto a qualidade das soluções e quanto ao número de avaliações de *fitness* até que a melhor solução seja encontrada. Este capítulo trata do projeto e execução do experimento, apresentando as questões de pesquisa, as instâncias selecionadas, a calibragem dos parâmetros e os resultados experimentais.

4.1 Questões de Pesquisa

São cinco as Questões de Pesquisa (QP) que orientam este estudo:

QP1. O VisILS supera um algoritmo de busca local quanto à qualidade das soluções?

A resposta a esta questão não é vista como uma contribuição relevante deste trabalho, dado que a comparação entre ILS e algoritmos de busca local mais simples é vasta na literatura [10, 33, 34]. Esta questão é utilizada apenas como uma validação, para garantir que os resultados de trabalhos anteriores podem ser aplicados ao NRP e às instâncias selecionadas. Espera-se que o VisILS supere o algoritmo de busca local.

QP2. O VisILS supera um ILS que não utiliza a informação sobre o espaço de busca do NRP em relação à qualidade das soluções?

Esta questão verifica se o procedimento de amostragem aleatória descrito no Capítulo 3 melhora uma heurística existente. Espera-se que a diminuição no espaço de busca a ser percorrido pelo VisILS faça com que o mesmo apresente resultados superiores quando comparado ao algoritmo ILS clássico.

QP3. O VisILS supera uma heurística que representa o estado-da-arte para o NRP mono-objetivo em termos de qualidade das soluções?

Até onde se sabe, o algoritmo BMA[1] é a heurística que produziu as melhores soluções conhecidas para as instâncias utilizadas neste trabalho. Dada a redução no espaço de busca ocasionada pelo padrão identificado através da análise do espaço de busca do problema, espera-se que o VisILS supere o BMA, principalmente nas maiores instâncias.

QP4. O VisILS converge para a melhor solução em menos iterações que o ILS?

Dado que o VisILS é mais complexo que o ILS clássico, ele provavelmente levará mais tempo para executar. Por outro lado, o VisILS pode encontrar a melhor solução consumindo menos avaliações de *fitness*.

QP5. Quão distantes são os resultados do VisILS e a solução ótima de cada instância?

Muito recentemente, Veerapen et al.[35] apresentaram uma abordagem de programação linear inteira que, pela primeira vez, foi capaz de resolver de forma exata todas as instâncias do NRP utilizadas neste trabalho. Esta questão de pesquisa investiga a qualidade das soluções geradas pelo VisILS em relação aos ótimos, agora conhecidos.

4.2 Instâncias do Problema

Este trabalho utiliza dois conjuntos de instâncias que foram previamente empregadas e avaliadas por diversos algoritmos e autores. O conjunto de instâncias clássicas consiste em 15 instâncias geradas por Xuan et al. [1] baseado no trabalho de Bagnall et al.[2]. São, portanto, instâncias sintéticas. O outro conjunto é composto por 24 instâncias realistas que foram extraídas por Xuan et al. [1] de repositórios de erros (*bugs*) de três projetos de software livre.

O conjunto de instâncias clássicas do NRP é formado por cinco grupos, cada qual contendo três instâncias. Em cada grupo, as instâncias têm diferentes restrições orçamentárias, cada uma igual a um percentual (30%, 50% ou 70%) multiplicado pelo custo de implementar todos os requisitos existentes na instância. Por exemplo, se o custo de implementar todos os requisitos é de 10.000 unidades, as três instâncias do grupo terão restrições de $10.000 \times 0.3 = 3.000$, $10.000 \times 0.5 = 5.000$ e $10.000 \times 0.7 = 7.000$. A Tabela 4.1 descreve os cinco grupos de instâncias clássicas. A primeira coluna contém o nome do grupo, enquanto a segunda coluna indica o número de requisitos presentes em cada instância. Todos os requisitos foram classificados em três níveis, separados por barras (“/”). Um requisito do segundo nível

Tabela 4.1: Detalhes das instâncias clássicas do NRP

Instância	Reqs/Nível	Custo Req	Depend	Reqs/Cli	Clientes	Lucro
nrp-1	20/40/80	1~5/2~8/5~10	8/2/0	1~5	100	10~50
nrp-2	20/40/80/ 160/320	1~5/2~7/3~9/ 4~10/5~15	8/6/4/2/0	1~5	500	10~50
nrp-3	250/500/750	1~5/2~8/5~10	8/2/0	1~5	500	10~50
nrp-4	250/500/750/ 1000/750	1~5/2~7/3~9/ 4~10/5~15	8/6/4/2/0	1~5	750	10~50
nrp-5	500/500/500	1~3/2/3~5	4/4/0	1	1000	10~50

pode ter como dependência (pré-requisito) requisitos do primeiro nível, enquanto requisitos do terceiro nível podem depender de requisitos do primeiro e segundo níveis. A terceira coluna apresenta um intervalo que indica os valores mínimo e máximo (separados por *til*) dos custos associados a cada requisito de cada nível. A quarta coluna mostra o número máximo de dependentes (filhos) que um requisito pode possuir, em cada nível. As colunas cinco e seis indicam, respectivamente, o número de requisitos solicitado por cada cliente e o número total de clientes presentes na instância. Finalmente, a última coluna contém um intervalo que indica os valores mínimo e máximo do lucro obtido ao se atender um cliente. O número de requisitos varia de 140 a 3.250.

Usando o grupo nrp-1 como exemplo, existem três níveis de requisitos, com 20, 40 e 80 requisitos respectivamente. Os custos dos requisitos integrantes do primeiro nível variam de 1 a 5 unidades monetárias. Já o custo dos requisitos de nível dois varia de 2 a 8 unidades, enquanto os requisitos do terceiro nível custam de 5 a 10 unidades monetárias. Um requisito do primeiro nível tem no máximo oito dependentes, enquanto um requisito do segundo nível tem no máximo dois dependentes. Existem 100 clientes nesta instância, cada um requisitando de 1 a 5 requisitos. Além disso, cada cliente atendido gera um lucro de 10 a 50 unidades monetárias.

O segundo conjunto de instâncias é composto por 12 grupos, com duas instâncias em cada grupo. Os percentuais de custo utilizados para estes grupos foram 30% e 50%, os mesmos utilizados por Xuan et al. [1] em seu estudo. Segundo Xuan et al., o percentual de 70% não foi utilizado pois um valor tão alto torna as instâncias muito fáceis de serem resolvidas. Neste conjunto de instâncias, o número de requisitos varia de 2.246 a 4.368 e o número de clientes de 100 a 1.000. A fim de construir essas instâncias, Xuan et al.[1] usaram repositórios de erros (*issue tracking system*) de três projetos de software livre, a saber, Eclipse, Mozilla e Gnome. Um registro de erro e um usuário foram mapeados, respectivamente, para um requisito e um cliente de uma instância do NRP. Além disso, cada comentário de usuário no registro de erros foi traduzido para uma solicitação de um requisito, enquanto a gravidade de cada bug foi mapeada como o custo do requisito. De forma similar às instâncias clássicas

Tabela 4.2: Detalhes das instâncias realistas do NRP

Instância	Reqs	Custo Req	Reqs/Cli	Clientes	Lucro
nrp-e1	3502	1~7	4~20	536	10~50
nrp-e2	4254	1~7	5~30	491	10~50
nrp-e3	2844	1~7	4~15	456	10~50
nrp-e4	3186	1~7	5~20	399	10~50
nrp-m1	4060	1~7	4~20	768	10~50
nrp-m2	4368	1~7	5~30	617	10~50
nrp-m3	3566	1~7	4~15	765	10~50
nrp-m4	3643	1~7	5~20	568	10~50
nrp-g1	2690	1~7	4~20	445	10~50
nrp-g2	2650	1~7	5~30	315	10~50
nrp-g3	2512	1~7	4~15	423	10~50
nrp-g4	2246	1~7	5~20	294	10~50

do NRP, o lucro obtido ao atender cada cliente foi gerado aleatoriamente, dentro de um determinado intervalo. A Tabela 4.2 apresenta os detalhes das instâncias realistas, de forma similar ao mostrado na Tabela 4.1. As únicas diferenças são a coluna *Reqs*, que mostra o número total de requisitos presente em cada instância, e a ausência da coluna *Depend*, dado que estas instâncias não possuem dependências entre os requisitos.

Independentemente de ser parte do primeiro ou do segundo conjunto, o nome de uma instância é formado pelo nome do grupo e pelo percentual de custo. Por exemplo, a instância nrp-1-30 é uma instância do grupo nrp-1 com um percentual de custo igual a 30%. As instâncias de um mesmo grupo compartilham as mesmas características, diferindo apenas no orçamento disponível para a implementação dos seus requisitos.

4.3 Calibragem de Parâmetros

As heurísticas possuem componentes e parâmetros que, ao serem ajustados, podem alterar drasticamente o desempenho do algoritmo ao abordar um determinado problema. Por isso, os melhores componentes e valores de parâmetros precisam ser determinados por um processo sistemático de calibragem. Esta seção descreve como foram determinados o componente de criação da solução inicial, o parâmetro que dita a força da perturbação aplicada pelo VisILS e a quantidade de soluções geradas para cada número de clientes atendidos na fase de amostragem aleatória para os experimentos apresentados neste trabalho.

4.3.1 Solução Inicial

Os algoritmos utilizados neste experimento necessitam de uma solução inicial, isto é, uma seleção de clientes a partir de onde o processo de busca é iniciado. Uma

estratégia de construção bastante utilizada é a aleatória, que como o próprio nome indica, faz uma seleção aleatória de clientes e os inclui como parte da solução inicial, de modo que cada cliente possui exatamente a mesma probabilidade de ser selecionado. Com o objetivo de identificar outras possíveis estratégias de construção, as melhores soluções geradas pelo HC foram analisadas em mais detalhes. Nesta análise, foi observado que os clientes com a maior razão lucro/custo apareceram com mais frequência entre as melhores soluções. A partir desta observação, foi criado o construtor guloso apresentado a seguir. Mais detalhes sobre esta avaliação podem ser obtidos no Anexo A.

O construtor guloso seleciona os clientes com base na razão entre o lucro esperado com o atendimento deste cliente e o custo de implementar os requisitos demandados pelo mesmo. No construtor guloso, clientes que apresentem as maiores razões têm maior probabilidade de serem incluídos na solução inicial. Por exemplo, considere os clientes A, B e C, todos candidatos a pertencerem à solução inicial, com razões sete, dois e um, respectivamente. Neste caso, o cliente A tem a maior chance de ser selecionado, com uma probabilidade de 70%. Já o cliente B será selecionado com uma probabilidade de 20%, enquanto o cliente C tem apenas 10% de chance de ser selecionado. Deste modo, é importante destacar que o construtor guloso também apresenta um componente aleatório. Bagnall et al. [2] usou um construtor similar em seu trabalho, para gerar soluções iniciais para um algoritmo GRASP.

Um experimento foi realizado a fim de comparar o desempenho das duas estratégias de construção da solução inicial. Como a busca local apresentada no Algoritmo 2.2 forma a base para todos os algoritmos avaliados neste trabalho, a estratégia de criação da solução inicial foi selecionada comparando o desempenho de cada uma das estratégias quando aplicada ao algoritmo HC. A Tabela 4.3 apresenta os resultados de cada construtor. A primeira coluna indica o nome da instância, seguida do *fitness* médio ao longo de 30 execuções do HC com os construtores aleatório e guloso, respectivamente. A última coluna apresenta a diferença percentual entre os resultados. Nota-se claramente a superioridade do construtor guloso, visto que o mesmo alcançou o melhor resultado em todas as instâncias, sendo em média 15% melhor que o HC com o construtor aleatório. Por isso, o construtor guloso será utilizado no restante deste trabalho como base para os algoritmos HC, ILS e VisILS, inclusive na fase de amostragem aleatória.

4.3.2 Força da Perturbação

A fim de determinar a força da mudança a ser aplicada pelo operador de perturbação dos algoritmos ILS e VisILS, outro experimento foi realizado, tendo o ILS como base. Neste experimento, o número de componentes da solução, isto é, o número de clientes

Tabela 4.3: Qualidade média das soluções obtidas por cada construtor dentre as 30 execuções de cada configuração. Valores em negrito indicam a maior média encontrada entre as configurações. Observa-se que o construtor guloso obteve os melhores resultados em todas as instâncias, sendo em média 15% melhor que o aleatório.

Instância	Aleatório	Guloso	%
nrp1-30	1024.4	1135.6	10.85%
nrp2-30	3237.6	4278.8	32.16%
nrp3-30	4710.7	5648.6	19.91%
nrp4-30	5509.2	7584.3	37.67%
nrp5-30	13797.3	15733.3	14.03%
nrp1-50	1629.5	1718.3	5.45%
nrp2-50	5995.3	7345.0	22.51%
nrp3-50	8893.5	9573.8	7.65%
nrp4-50	11423.4	13174.4	15.33%
nrp5-50	21597.6	22714.0	5.17%
nrp1-70	2413.6	2434.1	0.85%
nrp2-70	9623.5	10840.0	12.64%
nrp3-70	13515.9	13758.3	1.79%
nrp4-70	19152.1	19788.5	3.32%
nrp5-70	28556.7	28681.5	0.44%
nrp-e1-30	4895.3	5888.7	20.29%
nrp-e2-30	4476.9	5541.0	23.77%
nrp-e3-30	4191.3	5038.2	20.20%
nrp-e4-30	3638.9	4426.8	21.65%
nrp-g1-30	4184.7	4797.2	14.64%
nrp-g2-30	3098.6	3666.1	18.31%
nrp-g3-30	4021.8	4679.0	16.34%
nrp-g4-30	2918.1	3401.73	16.57%
nrp-m1-30	6142.9	7549.0	22.89%
nrp-m2-30	4954.8	6172.0	24.57%
nrp-m3-30	6184.5	7259.3	17.38%
nrp-m4-30	4663.1	5565.1	19.34%
nrp-e1-50	8232.0	9344.9	13.52%
nrp-e2-50	7499.4	8751.7	16.70%
nrp-e3-50	7026.2	7919.3	12.71%
nrp-e4-50	6127.8	6955.6	13.51%
nrp-g1-50	6964.8	7639.5	9.69%
nrp-g2-50	5110.1	5756.6	12.65%
nrp-g3-50	6705.1	7358.2	9.74%
nrp-g4-50	4817.0	5286.4	9.74%
nrp-m1-50	10791.3	12459.6	15.46%
nrp-m2-50	8617.8	10206.2	18.43%
nrp-m3-50	10786.7	12140.0	12.55%
nrp-m4-50	8126.2	9228.6	13.57%

alterado pelo operador de perturbação foi testado com cinco configurações distintas: (i) mudança de dois clientes por vez, (ii) mudança de três clientes, (iii) mudança de 1% do total de clientes presentes na instância, (iv) mudança de 3% dos clientes e, finalmente, (v) mudança de 5% dos clientes.

A Tabela 4.4 mostra os resultados obtidos por cada uma das configurações. A configuração (i) superou as demais em 22 das 39 instâncias. A configuração (ii) obteve a melhor média em 11 instâncias, seguida pela configuração (iii) que alcançou a melhor média 4 vezes. As configurações (iv) e (v) conseguiram a melhor média em apenas duas instâncias cada. Observou-se que quanto menor a força da perturbação, maior será o *fitness* médio das soluções obtidas ao final do processo de busca. A configuração (i) apresentou resultados, em média, 0.1% superiores aos resultados da configuração (ii), 1.8% melhores que a configuração (iii), 8% melhores que a configuração (iv) e 12% melhores que a configuração (v). Por isso, a configuração (i) será utilizada no restante deste trabalho.

4.3.3 Tamanho da Amostragem Aleatória

O VisILS possui uma fase de amostragem aleatória em que diversas soluções são geradas com um número variável de clientes atendidos, que começa em um e vai até o número total de clientes presentes na instância. Esta etapa é responsável por mapear o espaço de busca, visando identificar as regiões mais promissoras, conforme foi discutido na Seção 3.3. A quantidade de soluções geradas para cada número de clientes também foi determinada experimentalmente.

Através da execução da fase de amostragem aleatória diversas vezes, variando o número de soluções geradas (10, 100, 200, 300, ..., 1.000) para cada número de clientes presentes na solução, foi observado que gerar mais de 10 soluções para um mesmo número de clientes não alterava significativamente o *fitness* médio das soluções obtidas. Decerto, foi constatada uma ligeira melhora na qualidade média das soluções conforme este número aumentava. Porém, este incremento foi menor que 1%. Como a execução do construtor consome tempo computacional considerável, o VisILS foi configurado para gerar apenas 10 soluções para cada número de clientes atendidos. A utilização de menos soluções durante esta fase também tem como vantagem um consumo menor de avaliações de *fitness*, permitindo assim a execução de mais iterações do algoritmo de busca.

4.4 Análise e Discussão

Esta seção apresenta o estudo que foi realizado com o intuito de obter respostas para as questões de pesquisa apresentadas na Seção 4.1. Para isto, cada uma das

Tabela 4.4: Qualidade média das soluções obtidas ao longo das 30 execuções para cada intensidade de perturbação. Valores em negrito indicam a maior média encontrada entre as configurações. Nota-se que quando menor a força da perturbação, maior o *fitness* médio das soluções obtidas.

Instância	ILS-i	ILS-ii	ILS-iii	ILS-iv	ILS-v
nrp1-30	1185.0	1197.1	1128.3	1191.0	1198.9
nrp1-50	4680.7	4671.1	4534.8	4212.4	4072.0
nrp1-70	7334.3	7327.9	7213.7	6409.7	6023.0
nrp2-30	10327.3	10283.8	9168.2	7451.5	7128.0
nrp2-50	16811.5	16916.6	16570.7	15871.6	15664.1
nrp2-70	1808.2	1818.2	1739.1	1822.5	1818.0
nrp3-30	7581.8	7677.2	7676.0	7368.7	7171.0
nrp3-50	11023.1	11053.9	11036.9	10634.3	10327.9
nrp3-70	15668.8	15696.4	15413.9	14253.7	13741.3
nrp4-30	24343.6	24398.8	24322.5	23799.9	23522.4
nrp4-50	2506.6	2507.0	2482.2	2507.0	2507.0
nrp4-70	11056.2	11107.9	11115.4	11098.3	10950.0
nrp5-30	14147.6	14165.1	14169.9	14084.9	13995.5
nrp5-50	20757.3	20807.9	20781.9	20395.5	20175.1
nrp5-70	28838.9	28853.8	28865.5	28823.6	28785.9
nrpe1-30	7814.9	7804.6	7708.8	6948.9	6485.8
nrpe1-50	7376.3	7357.5	7257.9	6583.1	6097.4
nrpe2-30	6608.2	6587.2	6501.2	5965.0	5582.9
nrpe2-50	5761.4	5758.7	5726.8	5298.3	4949.4
nrpe3-30	6073.6	6070.4	6026.5	5634.0	5334.8
nrpe3-50	4557.0	4556.3	4555.2	4393.0	4188.0
nrpe4-30	5884.4	5882.8	5854.3	5473.7	5207.6
nrpe4-50	4196.4	4194.4	4189.3	4048.3	3903.7
nrpg1-30	10575.4	10508.8	9803.5	8298.2	7798.0
nrpg1-50	8590.7	8547.7	8262.9	7049.9	6558.8
nrpg2-30	10204.0	10140.8	9503.7	8127.5	7580.7
nrpg2-50	7690.8	7651.1	7402.7	6547.9	6051.5
nrpg3-30	10982.6	10974.0	10896.0	10316.3	9922.3
nrpg3-50	10322.1	10322.2	10254.4	9679.3	9328.0
nrpg4-30	9317.0	9309.8	9242.3	8805.5	8478.2
nrpg4-50	8137.9	8132.2	8106.3	7783.9	7514.0
nrpm1-30	8846.2	8840.8	8813.4	8472.8	8191.9
nrpm1-50	6532.3	6531.9	6533.8	6398.4	6217.2
nrpm2-30	8455.1	8455.5	8435.5	8134.4	7920.6
nrpm2-50	6041.9	6043.1	6042.7	5908.5	5761.9
nrpm3-30	15360.6	15314.9	14794.2	13686.7	13142.6
nrpm3-50	12473.9	12490.1	12245.3	11335.5	10865.7
nrpm4-30	14942.4	14909.5	14441.9	13440.8	12883.5
nrpm4-50	11289.9	11276.4	11081.7	10409.3	10010.8

instâncias do NRP descritas na Seção 4.2 foi otimizada por três configurações, doravante referenciadas como HC, ILS e VisILS. A configuração HC representa o algoritmo descrito na Seção 2.2.1, enquanto o ILS refere-se ao algoritmo descrito na Seção 2.2.2. A configuração VisILS é o algoritmo proposto na Seção 3.3.

Cada configuração foi executada 30 vezes para cada instância. Cada execução resultou em uma única solução cujo *fitness* (o lucro obtido ao atender os clientes) foi coletado. Em cada rodada, a busca foi encerrada ao atingir a quantidade de 10.000.000 de avaliações de *fitness*. Este número foi o mesmo utilizado por Xuan et al.[1] para configurar um GA que serviu de base para comparação com o BMA, algoritmo proposto pelos autores. Todas as configurações foram executadas em um mesmo computador, equipado com um processador Intel i7 de 3.4Ghz, 8GB de memória RAM, executando o sistema operacional Windows 8.1 Professional e softwares de apoio necessários à execução da implementação em Java.¹

As Tabelas 4.5 e 4.6 apresentam a qualidade das soluções obtidas pelas configurações sob análise. A Tabela 4.5 mostra o *fitness* da melhor solução encontrada por cada configuração, enquanto a Tabela 4.6 apresenta o *fitness* médio obtido por cada configuração ao longo das 30 execuções. As configurações GA e BMA foram incluídas para servirem de comparação, dado que foram utilizadas por Xuan et al. [1] para resolver as mesmas instâncias do problema. O BMA é a heurística que produziu os melhores resultados até então para as instâncias utilizadas neste trabalho. Valores maiores de *fitness* indicam que a configuração produz resultados mais efetivos que as outras configurações. Na Tabela 4.5, é possível observar que o VisILS superou as outras configurações em 29 das 39 instâncias, quando a comparação considera o melhor *fitness* encontrado dentre as execuções. A configuração ILS alcançou a melhor solução 13 vezes, a configuração BMA duas e o GA apenas uma vez. Em relação às médias, exibidas pela Tabela 4.6, o VisILS superou as demais configurações em 32 das 39 instâncias, o ILS em 5 instâncias e o BMA em apenas 3 casos. A partir destes resultados, é possível observar claramente que o VisILS é capaz de superar um algoritmo de busca local, o que responde à primeira questão de pesquisa (QP1).

A Tabela 4.7 exhibe uma comparação entre a qualidade média das configurações ILS e VisILS. A primeira coluna indica o nome das instâncias. As quatro colunas após o nome apresentam, para ambas as configurações, o *fitness* médio (μ) e o seu respectivo desvio-padrão (σ). De modo a responder adequadamente à segunda questão de pesquisa (QP2), esses valores foram submetidos ao teste de inferência não-paramétrico de Mann-Whitney [36], apresentado na coluna PV. *P-values* próximos a zero indicam uma forte confiança de que os resultados sendo comparados são estatisticamente diferentes. Medidas de tamanho de efeito (*effect-size*) também

¹O código-fonte está disponível em <https://github.com/richardf/VisualNRP>

Tabela 4.5: Qualidade da melhor solução obtida por cada configuração dentre as 30 execuções. Valores em negrito indicam o maior valor encontrado entre as configurações. Observa-se que o VisILS obteve os melhores resultados para 29 das 39 instâncias.

Instância	HC	ILS	VisILS	GA	BMA
nrp1-30	1180	1204	1204	1187	1201
nrp2-30	4414	4873	4830	2794	4726
nrp3-30	5878	7409	7401	5851	7123
nrp4-30	8057	10450	10470	6675	9818
nrp5-30	16090	17381	17854	10689	17200
nrp1-50	1764	1836	1836	1820	1824
nrp2-50	7533	7778	7842	5363	7566
nrp3-50	9729	11083	11100	9639	10897
nrp4-50	13459	15775	15839	12781	15025
nrp5-50	22928	24482	24589	18950	24240
nrp1-70	2477	2507	2507	2507	2507
nrp2-70	10892	11187	11202	9018	10987
nrp3-70	13859	14178	14195	12454	14180
nrp4-70	20035	20873	20892	17327	20853
nrp5-70	28735	28883	28904	22174	28909
nrp-e1-30	6091	7867	7875	6662	7572
nrp-e2-30	5736	7417	7417	6275	7169
nrp-e3-30	5188	6643	6636	5795	6461
nrp-e4-30	4590	5789	5787	5065	5692
nrp-g1-30	4955	6099	6103	5494	5938
nrp-g2-30	3759	4574	4568	4256	4526
nrp-g3-30	4834	5905	5914	5351	5802
nrp-g4-30	3483	4214	4216	3951	4190
nrp-m1-30	7749	10664	10642	8268	10008
nrp-m2-30	6403	8643	8657	6928	8272
nrp-m3-30	7474	10260	10252	8091	9559
nrp-m4-30	5722	7716	7731	6413	7408
nrp-e1-50	9559	11024	11034	9801	10664
nrp-e2-50	8980	10348	10354	9203	10098
nrp-e3-50	8111	9337	9339	8491	9175
nrp-e4-50	7134	8159	8163	7487	8043
nrp-g1-50	7813	8867	8880	8223	8714
nrp-g2-50	5884	6544	6550	6219	6502
nrp-g3-50	7537	8492	8486	7903	8402
nrp-g4-50	5401	6055	6057	5751	6030
nrp-m1-50	12647	15454	15469	13287	14588
nrp-m2-50	10436	12511	12539	10873	11975
nrp-m3-50	12390	15021	15019	12969	14138
nrp-m4-50	9367	11330	11338	9970	10893

Tabela 4.6: Qualidade média obtida por cada configuração ao longo das 30 execuções. Valores em negrito indicam o maior valor médio encontrado entre as configurações. Observa-se que o VisILS obteve os melhores resultados para 32 das 39 instâncias.

Instância	HC	ILS	VisILS	GA	BMA
nrp1-30	1135.7	1185.0	1183.6	1178.1	1188.3
nrp2-30	4278.8	4680.7	4731.0	2737.0	4605.0
nrp3-30	5648.6	7334.3	7343.3	5719.0	7086.0
nrp4-30	7584.3	10327.3	10342.6	6595.7	9710.0
nrp5-30	15733.3	16811.5	17119.7	10507.0	17026.9
nrp1-50	1718.3	1808.2	1809.1	1806.1	1796.0
nrp2-50	7345.1	7581.8	7683.4	5276.4	7414.0
nrp3-50	9573.9	11023.1	11040.0	9574.2	10787.2
nrp4-50	13174.4	15668.8	15724.2	12595.4	14815.5
nrp5-50	22714.0	24343.6	24417.0	18732.9	24087.5
nrp1-70	2434.1	2506.6	2507.0	2505.4	2507.0
nrp2-70	10840.0	11056.2	11115.9	8881.1	10924.7
nrp3-70	13758.3	14147.6	14183.2	12360.7	14159.2
nrp4-70	19788.5	20757.3	20854.8	17189.9	20819.7
nrp5-70	28681.5	28838.9	28887.5	22026.5	28894.2
nrp-e1-30	5888.7	7814.9	7832.0	6553.4	7528.2
nrp-e2-30	5541.1	7376.3	7379.9	6219.6	7109.9
nrp-e3-30	5038.2	6608.2	6609.1	5693.1	6413.0
nrp-e4-30	4426.8	5761.4	5764.7	5023.8	5636.2
nrp-g1-30	4797.3	6073.6	6078.5	5437.0	5911.3
nrp-g2-30	3666.1	4557.0	4554.8	4195.5	4486.2
nrp-g3-30	4679.0	5884.4	5892.3	5296.6	5736.5
nrp-g4-30	3401.7	4196.4	4195.8	3909.9	4159.0
nrp-m1-30	7549.0	10575.4	10586.3	8188.3	9889.6
nrp-m2-30	6172.1	8590.7	8609.3	6863.9	8147.5
nrp-m3-30	7259.3	10204.0	10193.6	8016.1	9499.7
nrp-m4-30	5565.2	7690.8	7691.8	6341.3	7319.3
nrp-e1-50	9344.9	10982.6	10995.7	9756.3	10589.2
nrp-e2-50	8751.7	10322.1	10331.4	9172.9	9999.8
nrp-e3-50	7919.3	9317.0	9316.3	8391.1	9100.1
nrp-e4-50	6955.6	8137.9	8137.1	7418.9	7968.0
nrp-g1-50	7639.5	8846.2	8851.8	8151.7	8660.0
nrp-g2-50	5756.6	6532.3	6536.3	6138.4	6470.2
nrp-g3-50	7358.3	8455.1	8459.3	7849.8	8326.8
nrp-g4-50	5286.4	6041.9	6044.5	5721.3	5986.5
nrp-m1-50	12459.7	15360.6	15383.2	13030.8	14437.7
nrp-m2-50	10206.2	12473.9	12500.4	10776.5	11883.5
nrp-m3-50	12140.0	14942.4	14957.3	12853.4	14036.6
nrp-m4-50	9228.7	11289.9	11299.3	9923.2	10790.7

foram calculadas e apresentadas na última coluna (ES). As medidas de efeito, como a medida não-paramétrica \hat{A}_{12} de Vargha e Delaney's [36] usada na Tabela 4.7, avaliam a magnitude da melhoria observada em uma comparação par a par. Dada uma medida M para observações coletadas após a aplicação dos tratamentos A e B , \hat{A}_{12} mede a probabilidade do tratamento A gerar resultados melhores que o tratamento B . Se ambos tratamentos são equivalentes, então $\hat{A}_{12} = 0.5$; caso contrário, \hat{A}_{12} indica a frequência de melhoria. Por exemplo, $\hat{A}_{12} = 0.8$ indica que o tratamento A gera resultados melhores em 80% das comparações com B .

Ao observar a Tabela 4.7, nota-se que a configuração VisILS superou o ILS em 33 das 39 instâncias, dos quais 16 resultados foram estatisticamente significativos com pelo menos 95% de confiança. Por outro lado, o ILS superou o VisILS em apenas 6 instâncias, mas em nenhum dos casos o resultado foi estatisticamente significativo. A medida de tamanho de efeito mostra que, em média, 65% das execuções da configuração VisILS resultarão em uma solução de maior qualidade que uma execução equivalente do ILS. Estes resultados fornecem evidências para responder à QP2, ao mostrar que o VisILS supera o ILS quanto à qualidade das soluções geradas.

A terceira questão de pesquisa (QP3) indaga se o VisILS é capaz de superar o BMA, uma heurística que representa o estado-da-arte para o NRP, ou seja, que possui os melhores resultados calculados por algum procedimento de busca heurístico para as instâncias avaliadas neste trabalho. As Tabelas 4.5 e 4.6 evidenciam que, surpreendentemente, tanto o ILS quanto o VisILS superaram o BMA. Ao comparar os melhores resultados de cada configuração, o ILS obteve resultados melhores que o BMA em todas exceto duas instâncias (nrp5-70 e nrp3-70), enquanto o VisILS foi superado pelo BMA uma única vez, na instância nrp5-70. Resultados semelhantes foram observados ao se comparar as médias de cada configuração, como mostrado na Tabela 4.6. Em relação às médias, a configuração ILS foi superada pelo BMA em 6 das 39 instâncias, enquanto o VisILS obteve resultados piores em apenas duas ocasiões. Considerando todas as instâncias, as soluções produzidas pelo ILS e VisILS foram, em média, 2.6% e 3% melhores que as soluções geradas pelo BMA.

As Figuras 4.1 e 4.2 apresentam estes resultados de forma gráfica. Cada box-plot representa o *fitness* das soluções obtidas pelo VisILS ao longo das 30 execuções, enquanto o quadrado verde representa o *fitness* médio obtido pelo BMA e o losango vermelho indica a melhor solução encontrada pelo BMA. Estas figuras mostram a distância relativa entre os resultados obtidos pelos algoritmos VisILS e BMA. Para isso, os valores do *fitness* foram normalizados utilizando a seguinte equação: $z_i = (x_i - \min(x)) / (\max(x) - \min(x))$, onde x representa o conjunto com 32 valores de *fitness* resultantes dos algoritmos VisILS e BMA e z_i indica o valor de *fitness* normalizado para o i -ésimo elemento do conjunto x . A normalização permite com-

Tabela 4.7: Comparação entre as configurações ILS e VisILS. Valores em negrito indicam o maior valor médio encontrado ao longo das 30 execuções de cada instância. Observa-se que o VisILS obteve o maior valor médio para 33 da 39 instâncias, dos quais 16 resultados foram estatisticamente significativos com 95% de confiança (coluna PV). O tamanho de efeito (coluna ES) mostra que o VisILS produz resultados com maior qualidade em 65% das execuções (média).

Instância	μ_{ILS}	σ_{ILS}	μ_{VisILS}	σ_{VisILS}	PV	ES
nrp1-30	1185.0	19.1	1183.6	23.3	0.81	52%
nrp2-30	4680.7	98.9	4731.0	63.1	0.02	68%
nrp3-30	7334.3	48.7	7343.3	31.9	0.54	55%
nrp4-30	10327.3	83.7	10342.6	87.6	0.52	55%
nrp5-30	16811.5	258.2	17119.7	414.3	<0.01	73%
nrp1-50	1808.2	18.2	1809.1	18.8	0.67	53%
nrp2-50	7581.8	127.2	7683.4	104.3	<0.01	73%
nrp3-50	11023.1	43.1	11040.0	38.7	0.20	60%
nrp4-50	15668.8	60.8	15724.2	58.8	<0.01	75%
nrp5-50	24343.6	97.6	24417.0	76.4	<0.01	72%
nrp1-70	2506.6	1.5	2507.0	0.0	0.16	53%
nrp2-70	11056.2	66.7	11115.9	53.9	<0.01	75%
nrp3-70	14147.6	15.5	14183.2	7.2	<0.01	99%
nrp4-70	20757.3	59.3	20854.8	20.4	<0.01	94%
nrp5-70	28838.9	31.4	28887.5	9.5	<0.01	95%
nrp-e1-30	7814.9	32.0	7832.0	28.6	0.02	68%
nrp-e2-30	7376.3	25.4	7379.9	23.2	0.57	54%
nrp-e3-30	6608.2	20.9	6609.1	18.8	0.89	51%
nrp-e4-30	5761.4	16.5	5764.7	13.0	0.48	55%
nrp-g1-30	6073.6	14.9	6078.5	15.1	0.17	60%
nrp-g2-30	4557.0	9.7	4554.8	10.7	0.65	54%
nrp-g3-30	5884.4	12.2	5892.3	11.1	0.02	68%
nrp-g4-30	4196.4	10.4	4195.8	10.2	0.80	52%
nrp-m1-30	10575.4	49.5	10586.3	36.9	0.33	57%
nrp-m2-30	8590.7	27.6	8609.3	22.4	0.02	68%
nrp-m3-30	10204.0	38.3	10193.6	29.5	0.19	60%
nrp-m4-30	7690.8	15.7	7691.8	19.2	0.91	51%
nrp-e1-50	10982.6	22.2	10995.7	25.2	0.03	67%
nrp-e2-50	10322.1	15.4	10331.4	15.1	0.02	67%
nrp-e3-50	9317.0	14.7	9316.3	13.3	0.77	52%
nrp-e4-50	8137.9	13.7	8137.1	14.0	0.91	51%
nrp-g1-50	8846.2	13.4	8851.8	13.0	0.15	61%
nrp-g2-50	6532.3	10.8	6536.3	7.3	0.28	58%
nrp-g3-50	8455.1	14.9	8459.3	16.3	0.26	58%
nrp-g4-50	6041.9	6.5	6044.5	6.4	0.18	60%
nrp-m1-50	15360.6	39.5	15383.2	34.6	0.01	69%
nrp-m2-50	12473.9	25.0	12500.4	24.7	<0.01	78%
nrp-m3-50	14942.4	45.2	14957.3	36.4	0.15	61%
nrp-m4-50	11289.9	20.9	11299.3	17.1	0.11	62%

parar as diferentes instâncias do NRP, cujos intervalos de valores para o *fitness* são muito diferentes. Como os valores individuais do *fitness* de cada execução do BMA não estão disponíveis, apenas o melhor resultado e a média foram utilizados. No que tange às instâncias clássicas, os valores médios obtidos pelo BMA ficaram abaixo do primeiro quartil do box-plot do VisILS em 10 das 15 instâncias, enquanto o melhor resultado ficou abaixo em 7 instâncias. Já nas instâncias realistas, tanto a média quanto a melhor solução do BMA ficaram abaixo do primeiro quartil em todas as instâncias, indicando que o VisILS funciona ainda melhor em instâncias do NRP que não apresentam dependências entre os requisitos.

Com o objetivo de avaliar a significância estatística desses resultados, os mesmos foram submetidos ao teste de inferência não-paramétrico de Mann-Whitney, cujo resultado é apresentado na Tabela 4.8. Nesta tabela, observa-se que o VisILS superou o BMA em 36 das 39 instâncias, dos quais 35 resultados foram estatisticamente significativos para um nível de confiança de 95%. O BMA obteve resultados melhores que o VisILS apenas duas vezes, porém em nenhum dos casos o resultado tem significância estatística. Tais resultados fornecem evidências de que o VisILS gera soluções cuja qualidade supera as soluções obtidas pelo BMA, o que responde à terceira questão de pesquisa (QP3).

A Tabela 4.9 apresenta o tempo médio de execução (μ) e o desvio-padrão (σ) de cada configuração. As configurações HC e ILS obtiveram praticamente o mesmo tempo de execução, sendo este último ligeiramente mais rápido. O VisILS precisa de mais tempo para executar o mesmo número de avaliações de *fitness*, sendo, em média, 16 vezes mais lento que o ILS. Através da análise da execução do algoritmo, verificou-se que a fase de amostragem aleatória e a busca local modificada são as responsáveis pelo maior tempo de execução do VisILS. Em relação à fase de amostragem, ela consiste em gerar diversas soluções com o intuito de identificar pontos interessantes no espaço de busca e, conseqüentemente, envolve muitas chamadas ao construtor de soluções, o que explica o aumento no tempo de execução. O VisILS também adiciona alguma complexidade à função de busca local, visto que um esforço adicional deve ser aplicado de forma a garantir que o algoritmo visite apenas soluções que respeitem o limite quanto ao número de clientes presentes na solução que foi definido durante a fase de amostragem aleatória.

Gráficos *time-to-target* (TTT) [37] podem ser utilizados para analisar o comportamento de algoritmos com componentes aleatórios. Um gráfico TTT é gerado executando-se o algoritmo diversas vezes e medindo o tempo necessário até que ele atinja uma solução igual ou melhor que uma solução alvo. Os gráficos da Figura 4.3 foram gerados por 200 execuções das configurações ILS e VisILS para as instâncias *nrp2-30* (alvo de 4.473), *nrp-e4-50* (alvo de 8.100), *nrp-g2-50* (alvo de 6487.47) e *nrp1-30* (alvo de 1143.8). Os alvos foram definidos como um percentual (de 95% a

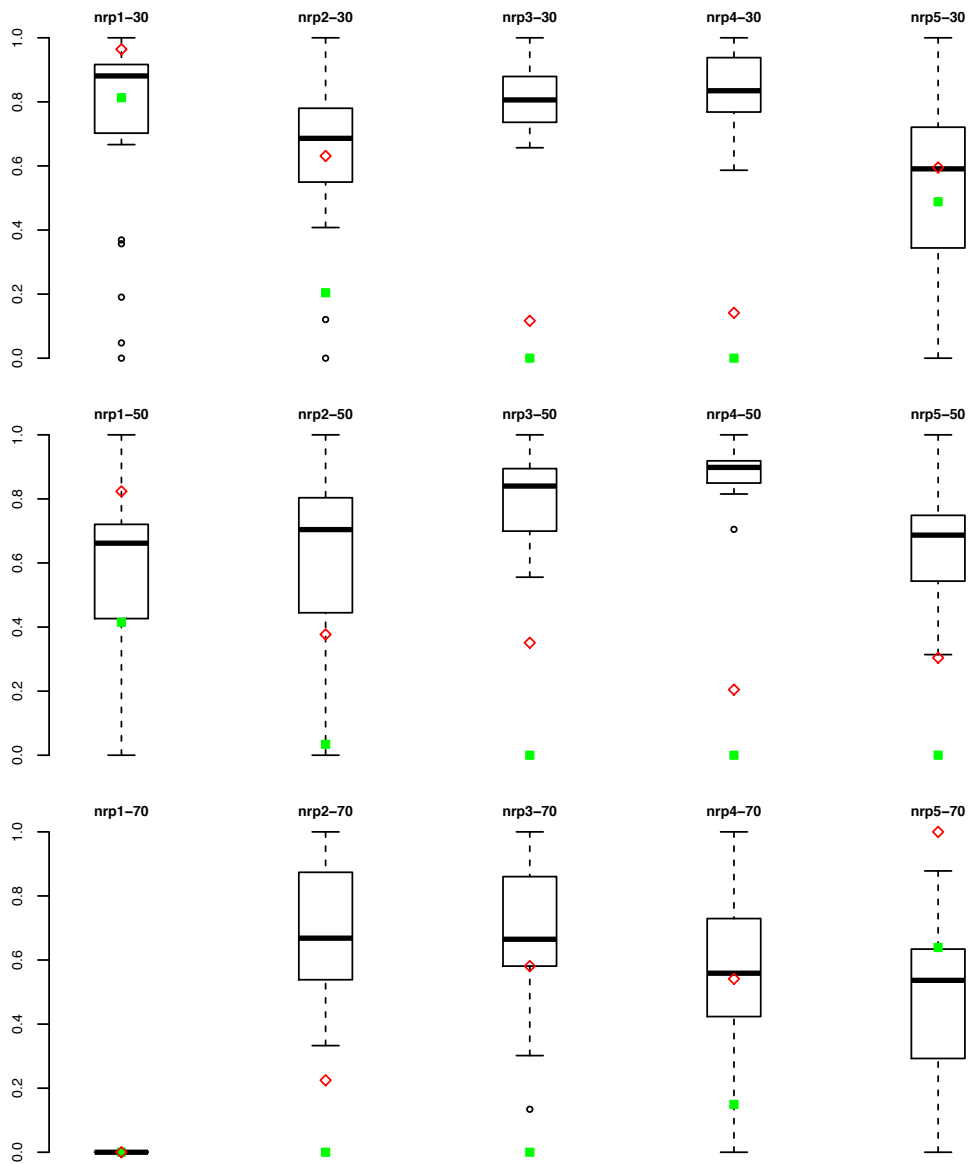


Figura 4.1: Comparação da qualidade das soluções geradas pelo VisILS e BMA para as instâncias clássicas. Os box-plots representam as soluções obtidas pelo VisILS ao longo das 30 execuções. O quadrado verde representa o valor médio obtido pelo BMA, enquanto o losango vermelho representa a sua melhor solução.

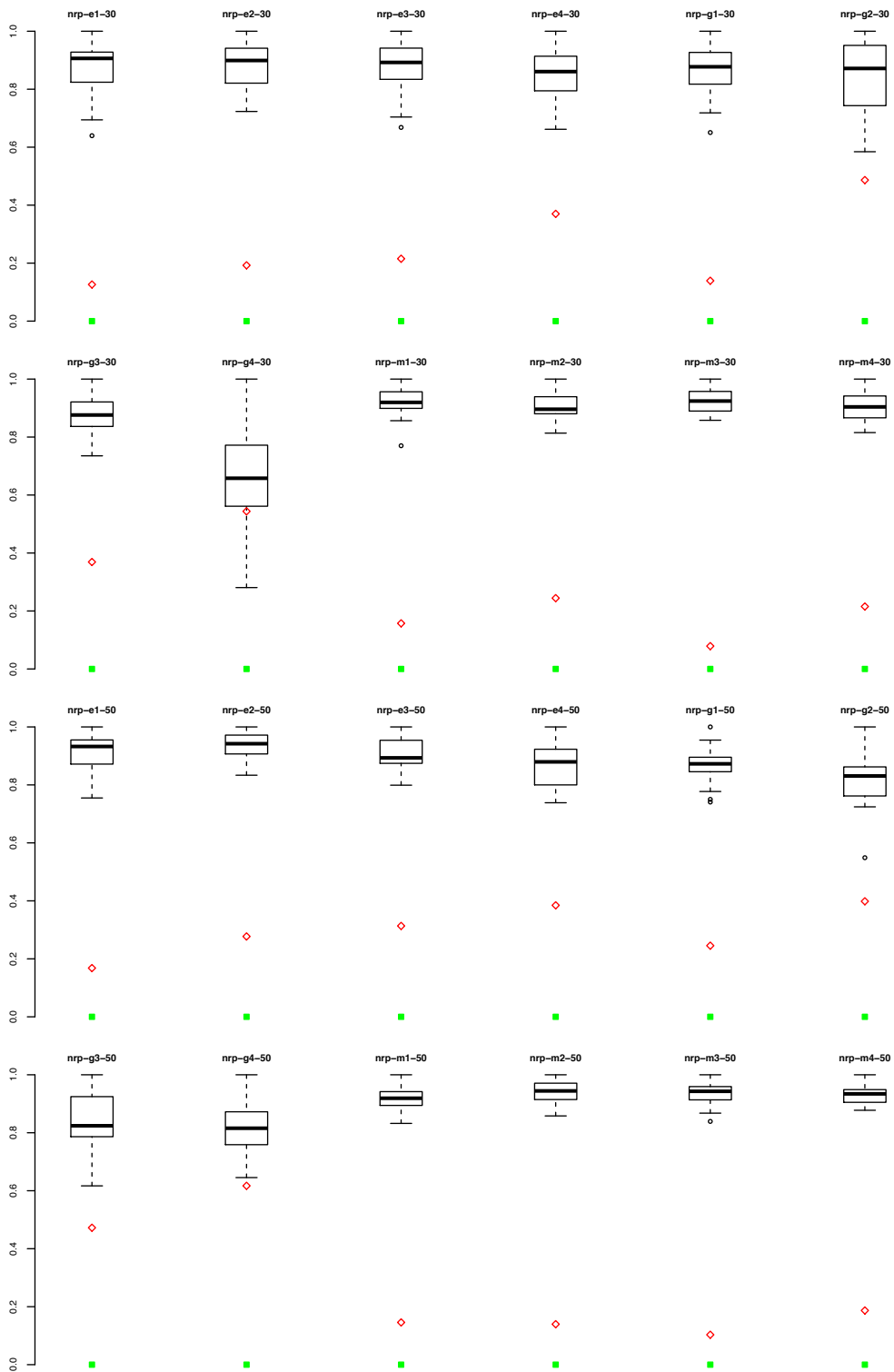


Figura 4.2: Comparação da qualidade das soluções geradas pelo VisILS e BMA para as instâncias realistas. Os box-plots representam as soluções obtidas pelo VisILS ao longo das 30 execuções. O quadrado verde representa o valor médio obtido pelo BMA, enquanto o losango vermelho representa a sua melhor solução.

Tabela 4.8: Comparação entre as configurações VisILS e BMA quanto à qualidade das soluções. Valores em negrito indicam a maior média obtida para cada instância. Observa-se que o VisILS obteve a melhor média para mais instâncias (36 de 39), dos quais 35 resultados são significativamente diferentes do BMA.

Instância	VisILS	BMA	PV
nrp1-30	1183.6	1188.3	0.32
nrp2-30	4731.0	4605.6	<0.01
nrp3-30	7343.3	7086.3	<0.01
nrp4-30	10342.6	9710.5	<0.01
nrp5-30	17119.7	17026.9	0.12
nrp1-50	1809.1	1796.2	<0.01
nrp2-50	7683.4	7414.1	<0.01
nrp3-50	11040.0	10787.2	<0.01
nrp4-50	15724.2	14815.5	<0.01
nrp5-50	24417.0	24087.5	<0.01
nrp1-70	2507.0	2507.0	-
nrp2-70	11115.9	10924.7	<0.01
nrp3-70	14183.2	14159.2	<0.01
nrp4-70	20854.8	20819.7	<0.01
nrp5-70	28887.5	28894.2	0.99
nrp-e1-30	7832.0	7528.2	<0.01
nrp-e2-30	7379.9	7109.9	<0.01
nrp-e3-30	6609.1	6413.0	<0.01
nrp-e4-30	5764.7	5636.2	<0.01
nrp-g1-30	6078.5	5911.3	<0.01
nrp-g2-30	4554.8	4486.2	<0.01
nrp-g3-30	5892.3	5736.5	<0.01
nrp-g4-30	4195.8	4159.0	<0.01
nrp-m1-30	10586.3	9889.6	<0.01
nrp-m2-30	8609.3	8147.5	<0.01
nrp-m3-30	10193.6	9499.7	<0.01
nrp-m4-30	7691.8	7319.3	<0.01
nrp-e1-50	10995.7	10589.2	<0.01
nrp-e2-50	10331.4	9999.8	<0.01
nrp-e3-50	9316.3	9100.1	<0.01
nrp-e4-50	8137.1	7968.0	<0.01
nrp-g1-50	8851.8	8660.0	<0.01
nrp-g2-50	6536.3	6470.2	<0.01
nrp-g3-50	8459.3	8326.8	<0.01
nrp-g4-50	6044.5	5986.5	<0.01
nrp-m1-50	15383.2	14437.7	<0.01
nrp-m2-50	12500.4	11883.5	<0.01
nrp-m3-50	14957.3	14036.6	<0.01
nrp-m4-50	11299.3	10790.7	<0.01

Tabela 4.9: Tempo médio de execução para cada configuração. Valores em negrito representam a menor média observada em cada instância, considerando as 30 execuções. As configurações HC e ILS obtiveram tempos quase idênticos, enquanto o VisILS foi cerca de 16 vezes mais lento.

Instância	μ_{HC}	σ_{HC}	μ_{ILS}	σ_{ILS}	μ_{VisILS}	σ_{VisILS}
nrp1-30	0.8	0.04	0.5	0.02	1.7	0.04
nrp2-30	0.9	0.02	0.7	0.02	9.9	0.05
nrp3-30	1.0	0.03	0.9	0.01	10.1	0.02
nrp4-30	1.6	0.05	1.9	0.04	22.6	0.04
nrp5-30	0.8	0.03	0.5	0.02	39.0	0.06
nrp1-50	0.8	0.03	0.4	0.02	1.6	0.03
nrp2-50	0.9	0.02	0.6	0.01	9.9	0.03
nrp3-50	1.0	0.03	0.7	0.02	10.1	0.02
nrp4-50	1.6	0.04	1.5	0.03	22.1	0.06
nrp5-50	0.7	0.03	0.6	0.03	39.8	0.06
nrp1-70	0.7	0.04	0.4	0.02	1.6	0.04
nrp2-70	0.8	0.02	0.6	0.01	9.8	0.03
nrp3-70	0.7	0.02	0.7	0.01	10.0	0.02
nrp4-70	1.2	0.03	1.3	0.02	22.1	0.08
nrp5-70	0.6	0.03	0.5	0.02	39.1	0.05
nrp-e1-30	1.0	0.04	0.9	0.03	10.7	0.28
nrp-e2-30	1.1	0.02	1.1	0.01	9.2	0.03
nrp-e3-30	0.9	0.10	0.8	0.08	8.0	0.04
nrp-e4-30	1.0	0.02	0.9	0.03	6.7	0.01
nrp-g1-30	0.9	0.01	0.7	0.03	7.6	0.01
nrp-g2-30	1.0	0.02	0.8	0.02	5.0	0.01
nrp-g3-30	0.8	0.02	0.7	0.02	7.1	0.02
nrp-g4-30	0.9	0.02	0.8	0.03	4.5	0.04
nrp-m1-30	1.0	0.02	1.0	0.02	19.9	0.04
nrp-m2-30	1.1	0.02	1.3	0.02	13.6	0.02
nrp-m3-30	0.9	0.02	1.0	0.03	19.7	0.30
nrp-m4-30	1.0	0.02	1.1	0.03	12.2	0.02
nrp-e1-50	1.1	0.06	1.0	0.03	11.1	0.05
nrp-e2-50	1.3	0.06	1.3	0.02	10.0	0.03
nrp-e3-50	1.0	0.04	0.8	0.02	8.5	0.02
nrp-e4-50	1.1	0.02	1.0	0.01	7.2	0.02
nrp-g1-50	1.0	0.03	0.7	0.03	8.0	0.02
nrp-g2-50	1.1	0.05	0.8	0.01	5.3	0.02
nrp-g3-50	1.0	0.03	0.7	0.02	7.5	0.02
nrp-g4-50	1.1	0.04	0.8	0.03	4.8	0.01
nrp-m1-50	1.1	0.05	1.2	0.03	21.2	0.07
nrp-m2-50	1.3	0.04	1.5	0.03	14.5	0.04
nrp-m3-50	1.0	0.03	1.1	0.04	21.3	0.05
nrp-m4-50	1.2	0.03	1.3	0.03	12.4	0.04

99%) em relação ao valor de *fitness* das soluções ótimas de cada instância. O i -ésimo tempo de execução ordenado t_i é associado a uma probabilidade $p_i = (i - 1/2)/200$ e os pontos $z_i = (t_i, p_i)$, para $i = 1, \dots, 200$ são plotados. Cada ponto indica a probabilidade (eixo vertical) do algoritmo atingir a solução alvo no tempo indicado (eixo horizontal).

A partir destes gráficos, é possível observar que o VisILS necessita de mais tempo para convergir para as soluções-alvo quando comparado com o ILS. Isto deve-se ao fato de os primeiros estágios da execução do VisILS serem compostos pela amostragem aleatória. Tal etapa não tem como objetivo gerar soluções de boa qualidade, e sim fornecer um mapa da distribuição das soluções pelo espaço de busca, e por isso a busca local não é executada durante a amostragem aleatória. Já o ILS começa a refinar uma solução inicial logo no início de sua execução, através da utilização da busca local e das perturbações, fazendo com que o mesmo alcance a melhor solução em menos tempo. Por exemplo, na instância *nrp2-30* (último gráfico da Figura 4.3) a probabilidade do ILS atingir a solução alvo em 2 segundos (2000ms) é de quase 100%, enquanto a probabilidade do VisILS é de, aproximadamente, 45% para o mesmo intervalo de tempo. O mesmo comportamento foi observado em todas as instâncias analisadas neste trabalho.

O número de avaliações de *fitness* necessárias para se encontrar a melhor solução em cada execução também foi registrado e pode ser visto na Tabela 4.10. Esta tabela apresenta o número médio de avaliações de *fitness* necessárias para cada instância, considerando as 30 execuções do ILS e do VisILS. A última coluna indica a diferença percentual entre esses números. O VisILS obteve a melhor solução em menos avaliações que o ILS em 20 das 39 instâncias. Exceto para cinco instâncias em que a diferença superou 10%, a diferença média entre o número de avaliações necessárias para se atingir a melhor solução nas duas configurações foi menor que 3%, sendo ora favorável ao ILS, ora ao VisILS. Estes resultados não são fortes o suficiente para rejeitar ou confirmar a quarta questão de pesquisa (QP4) e experimentos com outras instâncias podem vir a fornecer mais informações sobre esta questão. Para as instâncias utilizadas neste trabalho, salvos cinco casos, o número de avaliações de *fitness* executadas pelo VisILS e ILS foram muito próximas.

Muito recentemente, Veerapen et al.[35] apresentaram uma abordagem de programação linear inteira que, pela primeira vez, foi capaz de resolver de forma exata todas as instâncias do NRP utilizadas neste trabalho. A quinta questão de pesquisa (QP5) investiga a distância entre as soluções obtidas pelo VisILS e os ótimos de cada instância. A Tabela 4.11 apresenta o valor do *fitness* da solução ótima de cada instância e a distância percentual entre este valor e o resultado médio de cada configuração. A distância média observada para o HC foi de 16.7%, enquanto que para as configurações BMA e ILS o valor foi de 4.1% e 1.5%, respectivamente. Já o

Tabela 4.10: Número de avaliações de *fitness* até encontrar a melhor solução. O valor indicado representa a média das 30 execuções de cada algoritmo.

Instância	ILS	VisILS	%
nrp1-30	1.306.453	943.581	27.8%
nrp2-30	8.128.381	7.270.856	10.5%
nrp3-30	8.627.929	8.802.363	2.0%
nrp4-30	9.328.465	9.185.611	1.5%
nrp5-30	9.554.119	9.281.799	2.9%
nrp1-50	1.023.951	1.204.968	17.7%
nrp2-50	7.255.301	7.886.566	8.7%
nrp3-50	8.281.707	8.451.587	2.1%
nrp4-50	9.385.539	9.100.602	3.0%
nrp5-50	9.153.117	9.227.235	0.8%
nrp1-70	296.098	256.777	13.3%
nrp2-70	7.138.002	7.641.417	7.1%
nrp3-70	7.307.643	6.807.558	6.8%
nrp4-70	8.505.941	8.199.949	3.6%
nrp5-70	6.075.873	8.103.313	33.4%
nrp-e1-30	9.327.184	9.458.732	1.4%
nrp-e2-30	8.770.472	8.971.423	2.3%
nrp-e3-30	8.916.818	8.946.318	0.3%
nrp-e4-30	8.374.913	7.724.642	7.8%
nrp-g1-30	8.977.075	9.241.196	2.9%
nrp-g2-30	8.233.602	8.271.029	0.5%
nrp-g3-30	8.573.306	9.009.741	5.1%
nrp-g4-30	7.688.464	8.233.745	7.1%
nrp-m1-30	9.583.788	9.482.917	1.1%
nrp-m2-30	9.526.699	9.267.785	2.7%
nrp-m3-30	9.703.177	9.558.580	1.5%
nrp-m4-30	8.989.305	9.291.427	3.4%
nrp-e1-50	8.953.353	9.043.181	1.0%
nrp-e2-50	8.787.643	8.445.040	3.9%
nrp-e3-50	9.081.272	9.009.436	0.8%
nrp-e4-50	8.660.779	8.762.243	1.2%
nrp-g1-50	9.082.402	9.288.148	2.3%
nrp-g2-50	8.092.586	8.018.192	0.9%
nrp-g3-50	9.179.104	8.923.562	2.8%
nrp-g4-50	8.611.634	8.143.155	5.4%
nrp-m1-50	9.539.924	9.412.805	1.3%
nrp-m2-50	8.924.608	9.248.080	3.6%
nrp-m3-50	9.573.833	9.565.424	0.1%
nrp-m4-50	9.324.982	9.103.732	2.4%

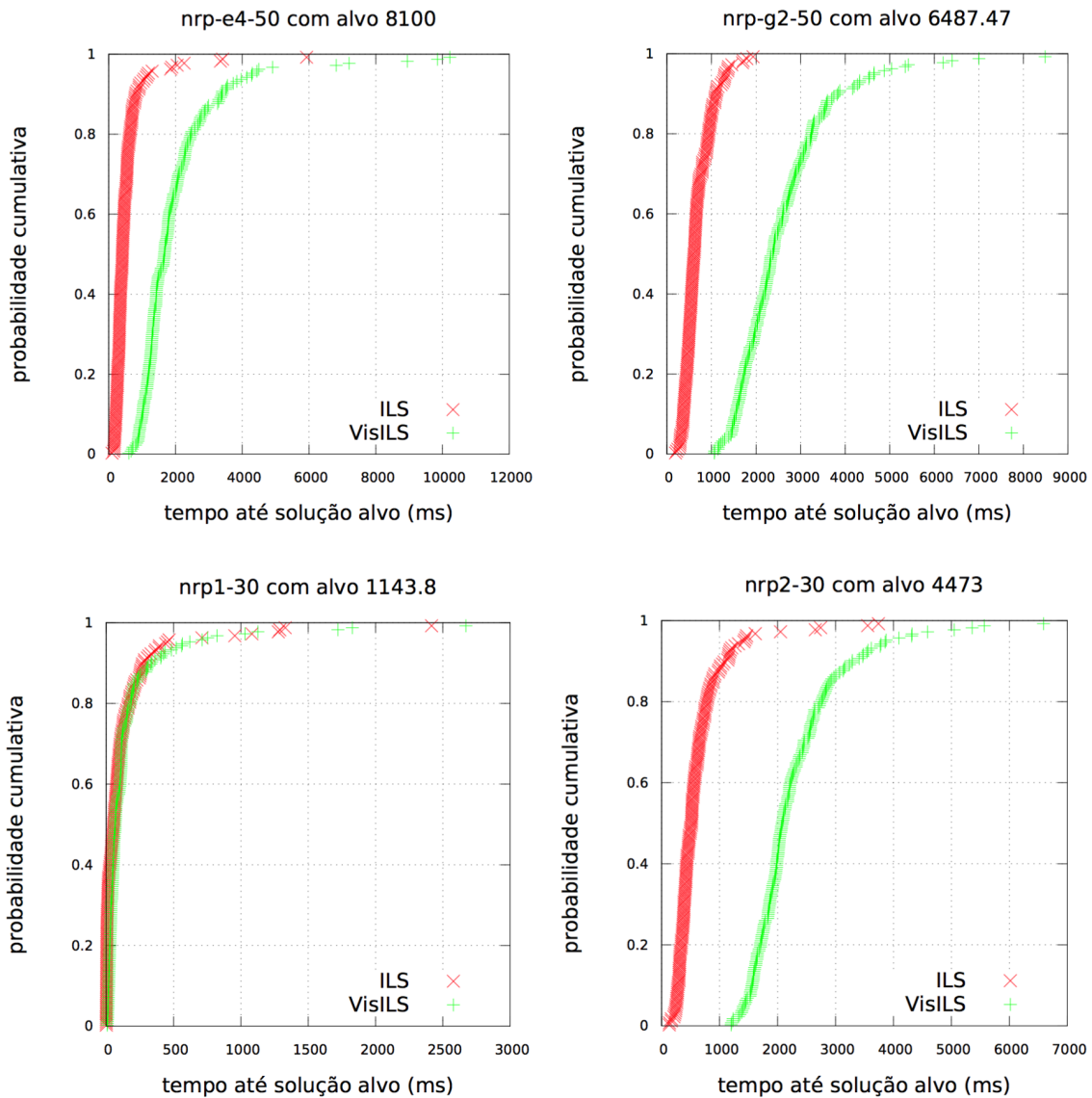


Figura 4.3: Gráficos *time-to-target* para duas instâncias realistas e duas clássicas

VisILS obteve um afastamento médio de 1.3%, uma melhora de 68% em relação ao BMA. A distância entre os resultados do VisILS e os ótimos foi inferior a 1% em 21 das 39 instâncias.

Se por um lado a divulgação dos ótimos para as instâncias torna irrelevante as soluções em si alcançadas pelo VisILS, por outro nos permitiu verificar a localização de tais soluções no espaço de busca. Conforme foi exposto na Seção 3.3, não existem garantias de que a solução ótima estará localizada após a melhor solução encontrada pela fase de amostragem aleatória que define o número mínimo de clientes atendidos pela solução para que a mesma seja considerada pelo algoritmo do VisILS. Mas de posse das soluções ótimas, foi possível gerar os gráficos exibidos nas Figuras 4.4 e 4.5. Estas figuras são similares às exibidas na Seção 3.1, mas mostram visualmente o processo de amostragem aleatória com destaque para a posição das soluções ótimas obtidas por Veerapen et al.[35], indicadas por uma linha vertical vermelha. Con-

Tabela 4.11: Distância percentual entre cada configuração e o ótimo. As distâncias foram calculadas com base no valor médio de *fitness* de cada configuração.

Instância	Ótimo	HC	ILS	VisILS	BMA
nrp1-30	1204	5.7%	1.6%	1.7%	1.3%
nrp2-30	4970	13.9%	5.8%	4.8%	7.3%
nrp3-30	7488	24.6%	2.1%	1.9%	5.4%
nrp4-30	10690	29.1%	3.4%	3.3%	9.2%
nrp5-30	18510	15.0%	9.2%	7.5%	8.0%
nrp1-50	1840	6.6%	1.7%	1.7%	2.4%
nrp2-50	8065	8.9%	6.0%	4.7%	8.1%
nrp3-50	11159	14.2%	1.2%	1.1%	3.3%
nrp4-50	15985	17.6%	2.0%	1.6%	7.3%
nrp5-50	24701	8.0%	1.4%	1.1%	2.5%
nrp1-70	2507	2.9%	0.0%	0.0%	0.0%
nrp2-70	11316	4.2%	2.3%	1.8%	3.5%
nrp3-70	14196	3.1%	0.3%	0.1%	0.3%
nrp4-70	20913	5.4%	0.7%	0.3%	0.4%
nrp5-70	28912	0.8%	0.3%	0.1%	0.1%
nrp-e1-30	7919	25.6%	1.3%	1.1%	4.9%
nrp-e2-30	7446	25.6%	0.9%	0.9%	4.5%
nrp-e3-30	6666	24.4%	0.9%	0.9%	3.8%
nrp-e4-30	5891	24.9%	2.2%	2.1%	4.3%
nrp-g1-30	6130	21.7%	0.9%	0.8%	3.6%
nrp-g2-30	4580	20.0%	0.5%	0.5%	2.0%
nrp-g3-30	5932	21.1%	0.8%	0.7%	3.3%
nrp-g4-30	4218	19.4%	0.5%	0.5%	1.4%
nrp-m1-30	10770	29.9%	1.8%	1.7%	8.2%
nrp-m2-30	8707	29.1%	1.3%	1.1%	6.4%
nrp-m3-30	10391	30.1%	1.8%	1.9%	8.6%
nrp-m4-30	7777	28.4%	1.1%	1.1%	5.9%
nrp-e1-50	11070	15.6%	0.8%	0.7%	4.3%
nrp-e2-50	10381	15.7%	0.6%	0.5%	3.7%
nrp-e3-50	9362	15.4%	0.5%	0.5%	2.8%
nrp-e4-50	8174	14.9%	0.4%	0.5%	2.5%
nrp-g1-50	8897	14.1%	0.6%	0.5%	2.7%
nrp-g2-50	6553	12.2%	0.3%	0.3%	1.3%
nrp-g3-50	8501	13.4%	0.5%	0.5%	2.0%
nrp-g4-50	6063	12.8%	0.3%	0.3%	1.3%
nrp-m1-50	15540	19.8%	1.2%	1.0%	7.1%
nrp-m2-50	12585	18.9%	0.9%	0.7%	5.6%
nrp-m3-50	15096	19.6%	1.0%	0.9%	7.0%
nrp-m4-50	11369	18.8%	0.7%	0.6%	5.1%

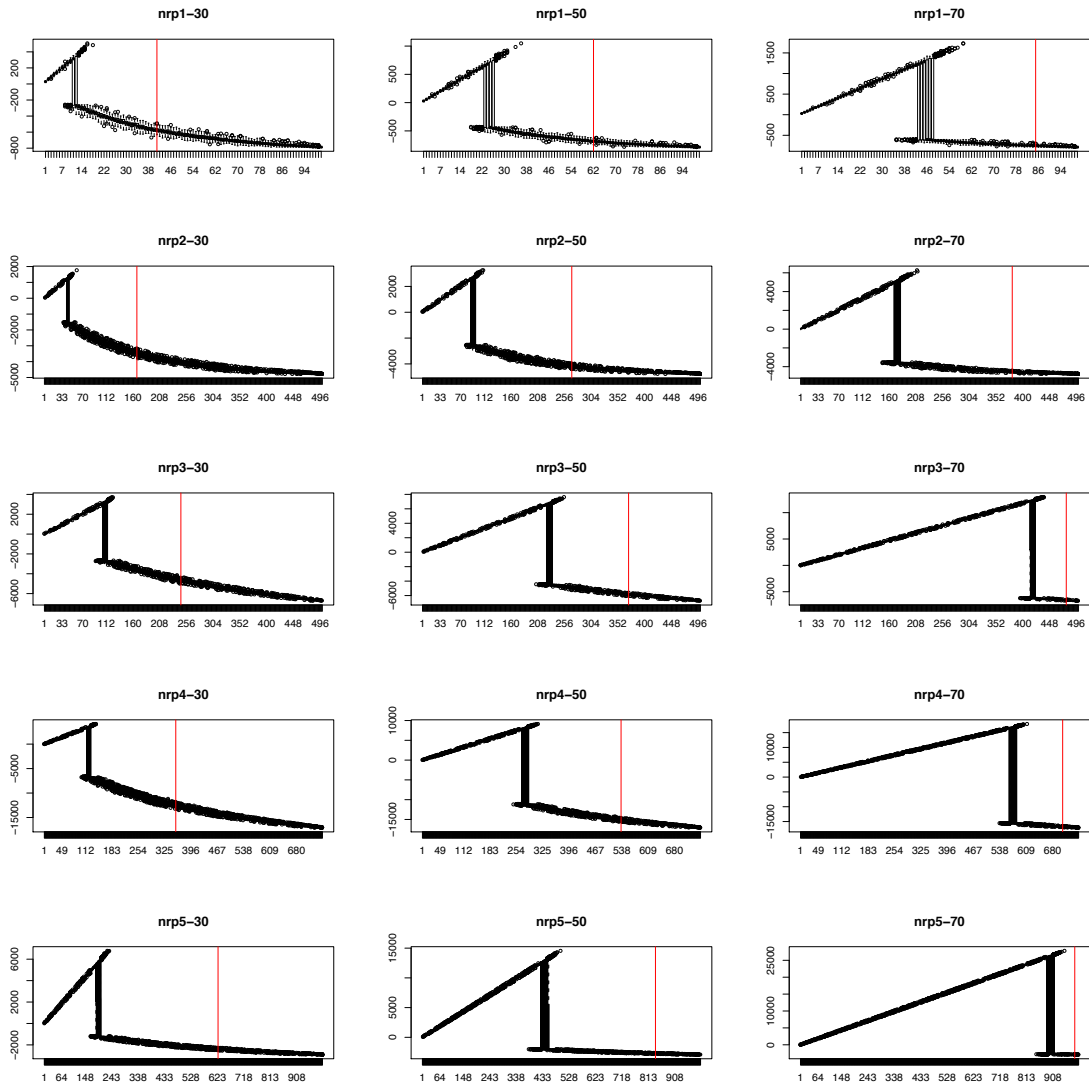


Figura 4.4: Amostragem Aleatória para instâncias adaptadas por Xuan et al. [1] do trabalho de Bagnall et al. [2] com destaque para a posição das soluções ótimas.

forme pode ser observado, em todos os casos o ótimo está situado após o limite definido pela amostragem aleatória, o que corrobora a validade da técnica de visualização apresentada e permite um futuro ajuste em seus limites.

4.4.1 Ameaças à Validade

Segundo Wohlin et al. [38], é importante preocupar-se com a validade de um estudo experimental no seu planejamento, para que os seus resultados sejam válidos no contexto assumido. Neste sentido, os autores sugerem avaliar quatro tipos de ameaças à validade: validade externa, validade interna, validade de construção e validade de conclusão. Em seu trabalho, Barros e Dias-Neto [39] enumeram as principais

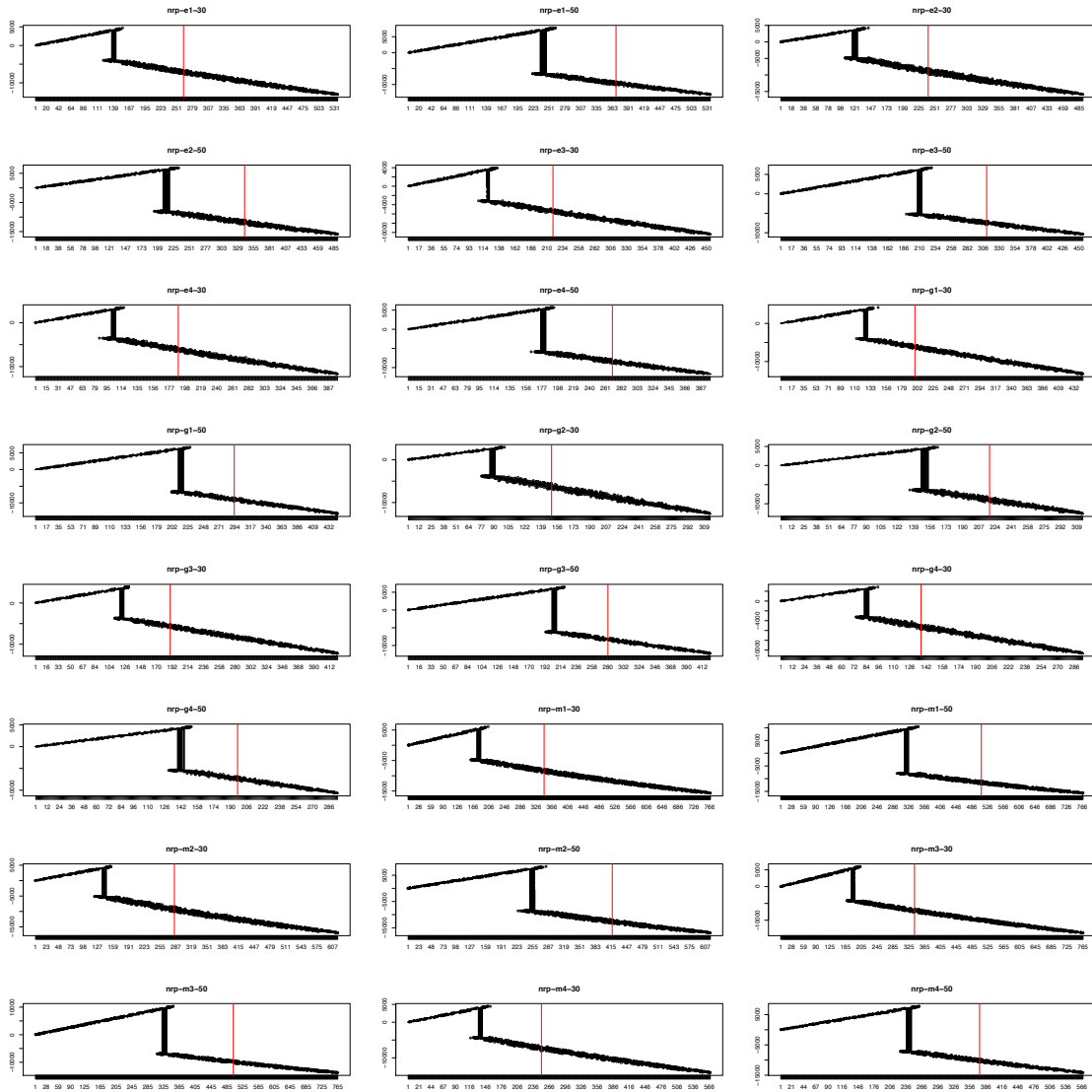


Figura 4.5: Amostragem Aleatória para instâncias realistas criadas por Xuan et al. [1] com destaque para a posição das soluções ótimas.

ameaças que podem comprometer um estudo com algoritmos de busca no contexto da Engenharia de Software.

Destas, a validade externa, isto é, a capacidade de generalizar os resultados obtidos para outras instâncias ou para diferentes modelos do NRP é a que mais se destaca no âmbito do presente trabalho. Primeiramente, foi assumida a formulação clássica mono-objetivo do NRP, como utilizada por Bagnall et al. [2] e Xuan et al. [1]. Todas as comparações e avaliações foram executadas usando esta formulação do problema. Como a Seção 2.3 mostra, existem outras formulações para o problema que levam em conta outros fatores além do custo dos requisitos, dependências do tipo pré-requisito entre os requisitos e lucro a ser obtido de cada cliente. Diversos tipos de dependência podem existir entre os requisitos (*E*, *OU*, *mudança de valor*, *mudança de custo*, dentre outras), assim como riscos de implementação e grupos de clientes podem ser considerados. Os resultados obtidos neste estudo são, portanto, limitados à formulação mono-objetivo do NRP utilizada neste trabalho.

O padrão gráfico identificado no espaço de busca do problema, que é utilizado pelo algoritmo VisILS para restringir o espaço de busca a ser percorrido, foi encontrado nas instâncias usadas por Bagnall et al. [2] e Xuan et al. [1]. Apesar de representarem um grupo relevante de instâncias (usado em diversos trabalhos anteriores), todas são caracterizadas por um grande número de clientes. Existem algumas instâncias conhecidas onde apenas alguns (menos de 20) clientes estão envolvidos. Nestas instâncias, a identificação do padrão relatado pode ser inviável dada a escassez de pontos (baixa resolução) no eixo x (eixo dos clientes) nos gráficos utilizados para demonstrar o processo de amostragem aleatória. Por isso, os resultados apresentados são limitados às instâncias em análise, embora a existência do mesmo padrão em instâncias que diferem consideravelmente entre si quanto ao número de clientes, número de dependências entre requisitos, custos, lucros e quanto à cardinalidade da relação "requisito-por-cliente" leva a crer que este padrão pode aparecer em outras instâncias que apresentem um grande número de clientes.

Ainda sobre a validade externa, os resultados estão restritos a métodos de busca local e busca local estendida, embora não exista limitação que impeça a utilização dos resultados da fase de amostragem aleatória em estratégias de busca global, como os algoritmos genéticos, ou até para melhorar heurísticas como o BMA. Deste modo, é possível que os resultados apresentados neste trabalho sejam superados por algoritmos mais complexos que por ventura venham a utilizar a mesma abordagem visual que foi utilizada como base para o algoritmo VisILS. Ainda assim, acredita-se que uma das contribuições deste trabalho é a de ressaltar a importância da visualização do espaço de busca como forma de melhorar o desempenho de algoritmos heurísticos, demonstrando um uso prático de um padrão visual que obteve êxito em melhorar os resultados de algoritmos de busca simples.

No que tange à validade interna, os principais pontos de atenção destacados por Barros e Dias-Neto [39] são a omissão ou não consideração da parametrização dos algoritmos, a falta de detalhamento sobre a implementação dos mesmos, a falta de clareza na descrição das instâncias e a ausência de instâncias reais do problema. A fim de atenuar tais problemas, a Seção 4.3 apresentou todos os parâmetros utilizados nos algoritmos que fizeram parte do estudo experimental, bem como o estudo realizado para chegar aos valores ideais para estes parâmetros. O Capítulo 2 descreveu todos os algoritmos utilizados e o código fonte de cada um também foi tornado público, estando disponível para avaliação e utilização por qualquer interessado. As instâncias utilizadas foram descritas na Seção 4.2. Enquanto um grupo de instâncias foi gerado aleatoriamente, o outro foi extraído de repositórios de sistemas de registro de erros (*issue tracking systems*) e portanto são mais realistas, dado que os itens presentes nesses repositórios podem ser vistos como um tipo de requisito. Apesar disto, a diferença entre repositórios de erros e requisitos reais pode introduzir um viés nos resultados deste trabalho. Não existem instâncias realistas com um grande número de clientes disponíveis na literatura, dado que essas informações geralmente são obtidas de empresas sob acordo de confidencialidade.

Sobre a validade de construção, um dos principais problemas é a falta de medidas para avaliar o custo de execução e a qualidade dos resultados. Segundo Barros e Dias-Neto [39], o número de avaliações da função de *fitness* é uma das medidas mais aceitas e usadas para avaliar o custo de execução de algoritmos. Neste trabalho, todos os algoritmos utilizaram um mesmo número (previamente determinado) de avaliações de *fitness*, que ao ser atingido indica o fim da execução. A qualidade dos resultados deve ser avaliada utilizando medidas relevantes ao problema, isto é, uma melhoria na medida deve estar relacionada com uma melhoria na qualidade da solução. Nos experimentos, o lucro obtido ao atender os clientes selecionados é utilizado como medida para avaliar a qualidade da solução, seguindo o uso feito por diversos outros trabalhos da literatura. Este lucro pode assumir diversos significados, conforme o caso, como retorno financeiro ou uma medida de satisfação dos clientes. Outro ponto importante sobre a validade de construção é a modelagem utilizada para representar o problema do mundo real. O ambiente no qual um software é desenvolvido é complexo, envolvendo diversas questões técnicas, financeiras e sociais. Portanto, qualquer modelo que descreva um problema relacionado a desenvolvimento de software é uma simplificação do mundo real. Este trabalho não discute a validade da modelagem do NRP utilizada, visto que trata-se de uma formalização apresentada e validada por trabalhos anteriores. Apesar disto, é importante ter em mente as limitações e simplificações existentes no modelo.

Finalmente, a validade de conclusão envolve a correta análise e interpretação estatística dos resultados. Os algoritmos heurísticos apresentados neste trabalho

possuem componentes aleatórios, o que faz com que os resultados sejam diferentes a cada nova execução, podendo determinados resultados individuais ser muito bons ou ruins. Para atenuar este efeito, os algoritmos foram executados diversas vezes. A execução repetida dos algoritmos também produz uma amostra maior de resultados, permitindo a aplicação de técnicas de estatística descritiva e inferencial, o que confere maior peso aos resultados apresentados.

4.5 Considerações Finais

Este capítulo apresentou as questões de pesquisa que orientaram o presente estudo, e buscou respondê-las mediante a execução de um experimento, retratado no mesmo capítulo. Foram detalhados o ambiente em que o experimento foi realizado, as instâncias utilizadas e o processo de calibragem de parâmetros. Os resultados demonstraram que a abordagem para diminuir o espaço de busca adotada pelo VisILS fez com que o mesmo superasse o ILS, um algoritmo que não utiliza o padrão gráfico descoberto neste trabalho. Mais ainda, o VisILS superou os resultados obtidos pelo BMA, uma heurística estado-da-arte para o NRP.

A recente descoberta das soluções ótimas para as instâncias utilizadas no estudo permitiu que se verificasse que em todos os casos o ótimo está situado após o limite definido pela etapa de amostragem aleatória do VisILS, o que corrobora a validade da técnica de visualização apresentada e permite um futuro ajuste em seus limites. Os resultados do VisILS são, em média 1.3% piores que os ótimos.

5. Conclusão

Este trabalho apresentou uma técnica de visualização do espaço de busca do NRP que revelou um padrão gráfico presente em dois grupos de instâncias frequentemente utilizadas em estudos experimentais reportados na literatura. Acredita-se que tal comportamento gráfico pode ser utilizado para guiar algoritmos heurísticos durante a busca, fazendo-os concentrar-se nas áreas mais promissoras do espaço de busca. A fim de testar esta hipótese, foram desenvolvidas variações de dois algoritmos de busca local, HC e ILS, onde uma fase de amostragem aleatória é executada para diminuir o espaço de busca do problema.

Os experimentos descritos nesta Dissertação mostraram que tanto o HC modificado, chamado de VisHC, quanto o ILS modificado, chamado de VisILS, foram capazes de gerar soluções de qualidade superior às geradas por suas versões clássicas. Além disso, os mesmos experimentos mostraram que o VisILS supera os resultados do BMA, uma heurística estado-da-arte para o NRP, muito mais complexa. Este trabalho foi o primeiro a aplicar o ILS ao NRP e, surpreendentemente, um algoritmo tão simples quanto o ILS também foi capaz de alcançar soluções superiores às obtidas pelo BMA.

O presente trabalho também mostrou que a visualização do espaço de busca pode ajudar a melhorar o desempenho de procedimentos de busca heurística para problemas complexos que possuam um grande espaço de busca. A técnica proposta converte um espaço de soluções discreto de maior ordem em uma visualização bidimensional que pode ser facilmente inspecionada tanto por humanos quanto por um programa de computador. Acredita-se que os bons resultados apresentados neste trabalho possam servir de incentivo para a aplicação de abstrações similares a outros domínios da Engenharia de Software, levando a maior utilização de técnicas de visualização do espaço de busca na área de SBSE. É possível que os resultados apresentados neste trabalho sejam superados por algoritmos mais complexos que por ventura venham a utilizar a mesma abordagem visual que foi utilizada como base para os algoritmos VisHC e VisILS.

Atualmente, o VisILS impõe um limite, determinado durante a fase de amostragem aleatória e utilizado para impedir que o algoritmo visite soluções cujo número de clientes atendidos seja menor que o limite definido. Conforme foi relatado na

Seção 3.2.1, as tentativas de se definir um limite para o número máximo de clientes atendidos mostraram-se infrutíferas, visto que não foi possível determinar um valor para este limite que apresentasse resultados consistentes em estudos experimentais preliminares. Ainda assim, futuras extensões deste trabalho podem voltar a visitar este tópico, já que isto pode vir a reduzir ainda mais o espaço de busca e, potencialmente, gerar soluções ainda melhores.

A aplicação do padrão gráfico a outras heurísticas, como algoritmos genéticos, é tida como outro possível ponto de extensão deste trabalho, assim como a aplicação da abordagem ao problema de *Release Planning* (RP), que é uma variação do NRP em que diversas *releases* são planejadas. O advento dos métodos ágeis também trouxe novas perspectivas a problemas como o NRP e RP, pois os mesmos precisam ser modificados a fim de suportar a incerteza inerente aos processos ágeis de desenvolvimento. A utilização de modelagens do NRP e RP que considerem tais fatores também é uma possível extensão para este trabalho.

Referências Bibliográficas

- [1] XUAN, J., JIANG, H., REN, Z., et al. “Solving the large scale next release problem with a backbone-based multilevel algorithm”, *Software Engineering, IEEE Transactions on*, v. 38, n. 5, pp. 1195–1212, 2012.
- [2] BAGNALL, A. J., RAYWARD-SMITH, V. J., WHITTLEY, I. M. “The next release problem”, *Information and software technology*, v. 43, n. 14, pp. 883–890, 2001.
- [3] ZHANG, Y., HARMAN, M., MANSOURI, S. A. “The multi-objective next release problem”. In: *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pp. 1129–1137. ACM, 2007.
- [4] DEL SAGRADO, J., DEL AGUILA, I. M., ORELLANA, F. J. “Ant colony optimization for the next release problem: A comparative study”. In: *Search Based Software Engineering (SSBSE), 2010 Second International Symposium on*, pp. 67–76. IEEE, 2010.
- [5] COLARES, F., SOUZA, J., CARMO, R., et al. “A new approach to the software release planning”. In: *Software Engineering, 2009. SBES'09. XXIII Brazilian Symposium on*, pp. 207–215. IEEE, 2009.
- [6] PAPADIMITRIOU, C. H., STEIGLITZ, K. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.
- [7] HARMAN, M., JONES, B. F. “Search-based software engineering”, *Information and Software Technology*, v. 43, n. 14, pp. 833–839, 2001.
- [8] TEIXEIRA DE SOUZA, J., MAIA, C. L., GOMES DE FREITAS, F., et al. “The human competitiveness of search based software engineering”. In: *Search Based Software Engineering (SSBSE), 2010 Second International Symposium on*, pp. 143–152. IEEE, 2010.
- [9] MARTELLO, S., TOTH, P. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

- [10] TALBI, E.-G. *Metaheuristics: from design to implementation*, v. 74. John Wiley & Sons, 2009.
- [11] PIRLOT, M. “General local search methods”, *European journal of operational research*, v. 92, n. 3, pp. 493–511, 1996.
- [12] MITCHELL, M. *An introduction to genetic algorithms*. MIT press, 1998.
- [13] FREITAS, F. G., COUTINHO, D. P., SOUZA, J. T. “Software next release planning approach through exact optimization”, *International Journal of Computer Applications*, v. 22, n. 8, pp. 1–8, 2011.
- [14] TONELLA, P., SUSI, A., PALMA, F. “Interactive requirements prioritization using a genetic algorithm”, *Information and software technology*, v. 55, n. 1, pp. 173–187, 2013.
- [15] DE SOUZA, J. T., MAIA, C. L. B., DO NASCIMENTO FERREIRA, T., et al. “An ant colony optimization approach to the software release planning with dependent requirements”. In: *Search Based Software Engineering*, Springer, pp. 142–157, 2011.
- [16] JIANG, H., ZHANG, J., XUAN, J., et al. “A hybrid ACO algorithm for the next release problem”. In: *Software Engineering and Data Mining (SEDM), 2010 2nd International Conference on*, pp. 166–171. IEEE, 2010.
- [17] NGO-THE, A., RUHE, G. “Optimized resource allocation for software release planning”, *Software Engineering, IEEE Transactions on*, v. 35, n. 1, pp. 109–123, 2009.
- [18] DEL SAGRADO, J., ÁAGUILA, I. M., ORELLANA, F. J. “Requirements interaction in the next release problem”. In: *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, pp. 241–242. ACM, 2011.
- [19] DEB, K., PRATAP, A., AGARWAL, S., et al. “A fast and elitist multiobjective genetic algorithm: NSGA-II”, *Evolutionary Computation, IEEE Transactions on*, v. 6, n. 2, pp. 182–197, 2002.
- [20] DURILLO, J. J., ZHANG, Y., ALBA, E., et al. “A study of the multi-objective next release problem”. In: *Search Based Software Engineering, 2009 1st International Symposium on*, pp. 49–58. IEEE, 2009.
- [21] NEBRO, A. J., DURILLO, J. J., LUNA, F., et al. “Mocell: A cellular genetic algorithm for multiobjective optimization”, *International Journal of Intelligent Systems*, v. 24, n. 7, pp. 726–746, 2009.

- [22] DEB, K. *Multi-objective optimization using evolutionary algorithms*, v. 16. John Wiley & Sons, 2001.
- [23] ZITZLER, E., THIELE, L. “Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach”, *evolutionary computation, IEEE transactions on*, v. 3, n. 4, pp. 257–271, 1999.
- [24] SALIU, M. O., RUHE, G. “Bi-objective release planning for evolving software systems”. In: *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pp. 105–114. ACM, 2007.
- [25] ZHANG, Y., HARMAN, M., OCHOA, G., et al. “An Empirical Study of Meta- and Hyper-Heuristic Search for Multi-Objective Release Planning”, *RN*, v. 14, pp. 07, 2014.
- [26] BURKE, E. K., GENDREAU, M., HYDE, M., et al. “Hyper-heuristics: A survey of the state of the art”, *Journal of the Operational Research Society*, v. 64, n. 12, pp. 1695–1724, 2013.
- [27] FINKELSTEIN, A., HARMAN, M., MANSOURI, S. A., et al. “A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making”, *Requirements Engineering*, v. 14, n. 4, pp. 231–245, 2009.
- [28] PITANGUEIRA, A. M., MACIEL, R. S. P., DE OLIVEIRA BARROS, M., et al. “A systematic review of software requirements selection and prioritization using SBSE approaches”. In: *Search Based Software Engineering*, Springer, pp. 188–208, 2013.
- [29] GERSHON, N. D. “From perception to visualization”, *Computer graphics*, v. 26, n. 2, pp. 414–415, 1992.
- [30] DIEHL, S. *Software visualization: visualizing the structure, behaviour, and evolution of software*. Springer Science & Business Media, 2007.
- [31] HARMAN, M. “The current state and future of search based software engineering”. In: *2007 Future of Software Engineering*, pp. 342–357. IEEE Computer Society, 2007.
- [32] LU, G., BAHSOON, R., YAO, X. “Applying elementary landscape analysis to search-based software engineering”. In: *Search Based Software Engineering (SSBSE), 2010 Second International Symposium on*, pp. 3–8. IEEE, 2010.

- [33] LOURENÇO, H. R., MARTIN, O. C., STÜTZLE, T. *Iterated local search*. Springer, 2003.
- [34] MARCHIORI, E. “Genetic, iterated and multistart local search for the maximum clique problem”. In: *Applications of Evolutionary Computing*, Springer, pp. 112–121, 2002.
- [35] VEERAPEN, N., OCHOA, G., HARMAN, M., et al. “An Integer Linear Programming approach to the single and bi-objective Next Release Problem”, *Information and Software Technology*, v. 65, pp. 1–13, 2015.
- [36] ARCURI, A., BRIAND, L. “A practical guide for using statistical tests to assess randomized algorithms in software engineering”. In: *Software Engineering (ICSE), 2011 33rd International Conference on*, pp. 1–10. IEEE, 2011.
- [37] AIEX, R. M., RESENDE, M. G., RIBEIRO, C. C. “TTT plots: a perl program to create time-to-target plots”, *Optimization Letters*, v. 1, n. 4, pp. 355–366, 2007.
- [38] WOHLIN, C., RUNESON, P., HÖST, M., et al. *Experimentation in software engineering*. Springer Science & Business Media, 2012.
- [39] BARROS, M., DIAS-NETO, A. “Threats to validity in search-based software engineering empirical studies”, *Relatórios Técnicos do DIA/UNIRIO*, , n. 0006, 2011.

A. Avaliação dos Construtores

Conforme dito na Seção 4.3.1, os algoritmos utilizados neste experimento necessitam de uma solução inicial, isto é, uma seleção de clientes, a partir de onde o processo de busca é iniciado. Com o objetivo de identificar outras possíveis estratégias de construção, distintas da construção aleatória, as melhores soluções geradas pelo HC foram analisadas em mais detalhes.

A Tabela A.1 apresenta o resultado desta análise para uma das instâncias. Esta tabela indica o número de vezes que cada cliente (primeira coluna) apareceu na melhor solução obtida pelas execuções do HC (segunda coluna). A terceira coluna mostra o valor do lucro obtido ao atender cada cliente, enquanto a quarta coluna indica o seu custo. A última coluna mostra o valor da razão entre o lucro e o custo. A Tabela A.1 está ordenada pelo número de ocorrências de cada cliente nas melhores soluções, do maior valor (30) para o menor. Devido ao tamanho, a Tabela A.1 não está exibindo todo o seu conteúdo, mas é possível observar que os clientes com a maior razão lucro/custo aparecem mais vezes nas melhores soluções.

De fato, uma análise de correlação de Spearman indicou uma correlação de até 70% entre a razão lucro/custo e o número de vezes que o cliente aparece nas soluções. O mesmo cálculo foi realizado para o lucro e o custo, sendo inexistente a correlação com o lucro e negativa com o custo. A Tabela A.2 apresenta os valores dos testes de correlação para cada instância. O racional por trás desta avaliação é a de que as soluções retornadas por um algoritmo de busca, no caso o HC, representam a melhor solução encontrada durante o processo de varredura do espaço de soluções do problema. São, portanto, boas soluções, e a repetição de certos clientes nas melhores soluções obtidas pode ser um indicativo de que este cliente é peça importante na criação de boas soluções para aquela instância. Como existe uma correlação entre estes clientes e a razão lucro/custo, esta razão pode ser utilizada por um algoritmo construtivo para gerar boas soluções iniciais.

Tabela A.1: Número de vezes que cada cliente apareceu na melhor solução encontrada, dentre as 30 execuções do HC, para a instância nrp1-30. Observa-se que as soluções com a maior razão lucro/custo (L/C) apareceram mais vezes.

Cliente	Ocorrências	Lucro	Custo	L/C
38	30	23	1	23.00
43	30	36	6	6.00
66	30	23	16	1.44
78	30	27	3	9.00
80	30	32	26	1.23
86	30	32	17	1.88
96	30	27	20	1.35
64	28	25	10	2.50
83	28	28	8	3.50
88	28	25	10	2.50
98	28	33	21	1.57
17	27	28	11	2.55
5	26	33	8	4.13
6	26	34	20	1.70
18	26	31	6	5.17
31	26	27	21	1.29
85	26	23	17	1.35
4	25	22	25	0.88
8	25	26	11	2.36
52	25	36	16	2.25
72	24	32	32	1.00
61	23	26	13	2.00
60	21	25	29	0.86
65	21	33	51	0.65
3	20	33	27	1.22
		...		
87	2	22	25	0.88
89	2	22	69	0.32
95	2	35	63	0.56
2	1	29	55	0.53
7	1	27	41	0.66
23	1	26	35	0.74
41	1	27	28	0.96
45	1	29	54	0.54
49	1	19	24	0.79
79	1	25	61	0.41
94	1	21	27	0.78
14	0	30	52	0.58
19	0	20	47	0.43
21	0	35	45	0.78
24	0	34	46	0.74
26	0	34	57	0.60
30	0	29	60	0.48
35	0	28	55	0.51
42	0	31	32	0.97
50	0	25	34	0.74
51	0	31	41	0.76
55	0	24	34	0.71
56	0	29	57	0.51
67	0	31	63	0.49
97	0	27	58	0.47

Tabela A.2: Correlação, para cada instância, entre o número de vezes que um cliente aparece na solução e o seu lucro, custo e razão lucro/custo (L/C).

Instância	Lucro	Custo	L/C
nrp1-30	0.005	-0.654	0.653
nrp1-50	0.035	-0.586	0.598
nrp1-70	0.051	-0.426	0.430
nrp2-30	-0.007	-0.637	0.630
nrp2-50	0.007	-0.600	0.595
nrp2-70	0.007	-0.550	0.546
nrp3-30	0.060	-0.506	0.504
nrp3-50	0.083	-0.489	0.495
nrp3-70	0.098	-0.334	0.343
nrp4-30	0.018	-0.561	0.550
nrp4-50	0.055	-0.539	0.534
nrp4-70	0.029	-0.397	0.395
nrp5-30	0.018	-0.370	0.369
nrp5-50	0.039	-0.367	0.370
nrp5-70	0.058	-0.272	0.280
nrp-e1-30	0.064	-0.471	0.474
nrp-e1-50	0.071	-0.514	0.514
nrp-e2-30	0.093	-0.502	0.506
nrp-e2-50	0.139	-0.498	0.515
nrp-e3-30	0.211	-0.489	0.527
nrp-e3-50	0.217	-0.576	0.608
nrp-e4-30	0.109	-0.568	0.565
nrp-e4-50	0.221	-0.592	0.625
nrp-g1-30	0.178	-0.494	0.545
nrp-g1-50	0.223	-0.562	0.620
nrp-g2-30	0.152	-0.587	0.612
nrp-g2-50	0.201	-0.663	0.700
nrp-g3-30	0.240	-0.459	0.530
nrp-g3-50	0.287	-0.600	0.678
nrp-g4-30	0.121	-0.601	0.624
nrp-g4-50	0.216	-0.656	0.707
nrp-m1-30	0.126	-0.366	0.384
nrp-m1-50	0.150	-0.452	0.476
nrp-m2-30	0.139	-0.455	0.472
nrp-m2-50	0.147	-0.467	0.491
nrp-m3-30	0.104	-0.399	0.408
nrp-m3-50	0.151	-0.397	0.432
nrp-m4-30	0.117	-0.482	0.485
nrp-m4-50	0.191	-0.523	0.558