UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ISMAEL IZÍDIO DE ALMEIDA

METAHEURÍSTICA HÍBRIDA UTILIZANDO GRASP REATIVO E APRENDIZAGEM POR REFORÇO: UMA APLICAÇÃO NA SEGURANÇA PÚBLICA

MOSSORÓ - RN
2014

ISMAEL IZÍDIO DE ALMEIDA

METAHEURÍSTICA HÍBRIDA UTILIZANDO GRASP REATIVO E APRENDIZAGEM POR REFORÇO: UMA APLICAÇÃO NA SEGURANÇA PÚBLICA

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação - associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semi-Árido, para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Francisco Chagas de Lima Júnior
Coorientador: Prof. Dr. Carlos Heitor Pereira Liberalino

MOSSORÓ - RN
2014

Catalogação da Publicação na Fonte.
Universidade do Estado do Rio Grande do Norte.

Almeida, Ismael Izídio de.
Metaheurística híbrida utilizando Grasp reativo e aprendizagem por reforço: uma aplicação na segurança pública. / Ismael Izídio de Almeida. – Mossoró, RN, 2014.

70 f.

Orientador(a): Prof. Dr. Francisco Chagas de Lima Júnior

Dissertação (Mestrado em Ciência da Computação). Universidade do Estado do Rio Grande do Norte. Programa de Pós-Graduação em Ciência da Computação.

1. Algoritmo Q-learning - Dissertação. 2. Criminalidade. 3. Metaheurística híbrida . I. Lima Júnior, Francisco Chagas de. II. Universidade do Estado do Rio Grande do Norte. III.Título.

UERN/ BC CDD 005.1

Bibliotecária: Elaine Paiva de Assunção – CRB - 15/492

ISMAEL IZÍDIO DE ALMEIDA

METAHEURÍSTICA HÍBRIDA UTILIZANDO GRASP REATIVO E APRENDIZAGEM POR REFORÇO: UMA APLICAÇÃO NA SEGURANÇA PÚBLICA

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA EM: **14 / 08 / 2014** .

BANCA EXAMINADORA

_____

Prof. Dr. Carlos Heitor Pereira Liberalino - UERN
Presidente

_____

Prof. Dr. Francisco Chagas de Lima Júnior - UERN
Primeiro Membro

_____

Prof. Dr. Dario José Aloise - UERN
Segundo Membro

_____

Prof. Dr. Gustavo Augusto Lima de Campos - UECE
Terceiro Membro

À minha fonte de amor incondicional: minha mãe, Maria do Socorro de Almeida Santos; e aos meus irmãos. Sem a confiança e apoio de vocês a conclusão desse trabalho não seria possível.

AGRADECIMENTOS

Primeiramente, sou grato a Deus, pelas bênçãos constantes, pelo dom da vida, pela força e coragem para realizar esse trabalho.

Agradeço também a toda a minha família pela base, incentivo, afeto, carinho e amor que têm por mim.

Minha gratidão ao meu orientador, prof. Lima Júnior, pela orientação valiosa sem a qual esse trabalho não existiria, pela confiança depositada e pela paciência empregada nas quebras de prazos.

Sou muito grato ao meu coorientador, prof. Heitor, pela tranquilidade que me passou ao sempre me receber com um sorriso sereno, pela confiança empregada e pela importante ajuda na composição deste trabalho.

Gostaria de agradecer aos meus amigos que sempre me incentivaram e acreditaram em mim. Em especial, ao meu amigo Jomar, pelos conselhos, pela força e segurança.

A todos os colegas de mestrado, pelo convívio harmônico, conversas e trocas de experiências. Meu muito obrigado!

Ao pessoal do Laboratório de Otimização e Inteligência Artificial - LOIA pelo suporte, agradável cotidiano e sufoco compartilhados. Um obrigado super especial a Marianny, Ernando e Abílio.

Um agradecimento especial aos membros do Laboratório de Engenharia de Software - LES pela recepção nesse excelente ambiente de trabalho e pelos diversos momentos agradáveis na companhia desse grupo. Especialmente aos amigos Jomar, Marlos, Anderson, Kayo, Marlon, Natan, Rodrigo, Davi. Um obrigado muito especial à querida Márcia Aratusa, pelo carinho, apoio e pela torcida para que eu fizesse parte da família LES.

Aos amigos do Programa de Educação Tutorial de Ciência da Computação - PETCC pela valiosa colaboração, em especial Aristóteles e Erick.

Agradeço pela disposição e ajuda sempre que necessário às secretarias do mestrado na UERN e na UFERSA, nas pessoas de Maninho e com especial carinho a Rosita, bem como à coordenação do programa.

À UERN e à UFERSA pela oportunidade de aperfeiçoamento acadêmico e infraestrutura

fornecida, bem como à CAPES pelo apoio financeiro.

Ao Centro Integrado de Operações de Segurança Pública (CIOSP) - Mossoró, pelo fornecimento dos dados necessários para o sucesso deste trabalho.

Por fim, quero agradecer a todos que fizeram parte da minha vida nessa etapa da vida deixando sua contribuição. A vocês todos, um caloroso abraço de agradecimento.

Não há vitória sem luta, nem vencedor solitário. O tempo pune, mas muitas vezes premia a quem é perseverante, corajoso e tem fé.

DESCONHECIDO

RESUMO

Metaheurísticas representam uma importante classe de algoritmos aproximativos para resolver problemas NP-difíceis. Uma tendência na pesquisa em otimização combinatória tem sido a exploração de metaheurísticas híbridas. Este trabalho apresenta uma versão híbrida da metaheurística GRASP Reativo que incorpora um agente de Aprendizagem por Reforço. No algoritmo híbrido proposto, um agente aprendiz, especificamente o algoritmo Q-learning, é utilizado para aprender e fornecer o melhor parâmetro α a ser utilizado na fase construtiva do GRASP Reativo, substituindo a distribuição de probabilidades utilizada no mecanismo reativo. Essa estratégia dota o método de uma memória adaptativa, que é atualizada com base na experiência adquirida ao longo das iterações. O GR-Learning híbrido foi aplicado com sucesso ao problema de localização dos *p*-Centros. O método proposto foi utilizado para determinar a melhor localização para a instalação de bases policiais na cidade de Mossoró–RN. As instâncias de teste foram elaboradas com base no histórico de localização de crimes graves nesta cidade. Os resultados obtidos com a versão híbrida foram comparados com os obtidos pelo GRASP Reativo tradicional, mostrando um melhor desempenho tanto em qualidade da solução quanto em tempo de execução pela nova abordagem. Uma análise estatística dos resultados foi feita para validar o método.

Palavras-chave: Algoritmo Q-learning, Criminalidade, Metaheurística híbrida, Problema dos *p*-Centros, Problemas de otimização combinatória.

ABSTRACT

Metaheuristics represent an important class of approximative algorithms for solving NP-hard problems. A trend in combinatorial optimization research has been the exploration of hybrid metaheuristics. This paper presents a hybrid version of Reactive GRASP metaheuristic that incorporates a Reinforcement Learning agent. In the hybrid algorithm proposal, a learner agent, specifically the Q-learning algorithm, is used to learn and provide the best α parameter to be used in the construction phase of the Reactive GRASP, replacing the probability distribution used in the reactive mechanism. This strategy give the method an adaptive memory that is updated with the experience gained over the iterations. Hybrid RG-Learning was successfully applied to the *p*-Center location problem. The proposed method was utilized here to determine the best location to installation of police bases in the city of Mossoró–RN. The test instances were prepared based on the location history of serious crimes in this city. The results obtained with the hybrid version were compared with those obtained by the traditional Reactive GRASP, showing a better performance in solution quality and in runtime by this new approach. A statistical analysis of the results was made to validate the method.

Keywords: Combinatorial optimization problems, Criminality, Hybrid metaheuristic, *p*-Center problem, Q-learning algorithm.

LISTA DE SIGLAS

GRASP *Greedy Randomized Adaptive Search Procedure*

AR Aprendizagem por Reforço

PCV Problema do Caixeiro Viajante

PDM Processo de Decisão de Markov

COEDHUCI Conselho Estadual de Direitos Humanos e Cidadania

IBGE Instituto Brasileiro de Geografia e Estatística

PPM Problema das p-Medianas

PPC Problema dos p-Centros

PLF Problema de Localização de Facilidades

PLMC Problema de Localização de Máxima Cobertura

LRC Lista Restrita de Candidatos

LC Lista de Candidatos

CIOSP Centro Integrado de Operações de Segurança Pública

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Metaheurísticas são algoritmos aproximativos que utilizam processos iterativos para guiar heurísticas subordinadas pela combinação inteligente de diferentes conceitos visando a exploração/explotação do espaço de busca, a fim de encontrar, de forma eficiente, soluções de alta qualidade (BLUM; ROLI, 2003). Elas manipulam um conjunto de soluções a cada iteração. As heurísticas subordinadas podem ser de construção, de busca local ou procedimentos de alto nível. Em todos os casos, os métodos fazem uso de conhecimento prévio das propriedades do problema. Assim, as metaheurísticas guiam heurísticas específicas, modeladas para problemas específicos, com o objetivo de encontrar soluções factíveis sem, no entanto, garantir a obtenção da solução ótima.

As metaheurísticas representam uma importante classe de métodos aproximativos, tendo grandes aplicações na resolução de problemas computacionais complexos, para os quais o uso de métodos exatos é inviável. Por meio de estratégias não determinísticas, conseguem encontrar soluções ótimas ou consideravelmente boas para problemas NP-difíceis em tempo polinomial, decidindo entre a explotação e a exploração do espaço de busca. Dentre as metaheurísticas mais conhecidas e utilizadas destacam-se o *Greedy Randomized Adaptive Search Procedure* (GRASP)(FEO; RESENDE, 1995), a Busca Tabu (GLOVER, 1986), o *Simulated Annealing* (KIRKPATRICK *et al.*, 1983), os Algoritmos Genéticos (HOLLAND, 1975) e a Otimização por Colônia de Formigas (DORIGO, 1992). Durante as últimas décadas, estas e outras metaheurísticas foram propostas e aplicadas a diversos problemas reais das várias áreas das ciências (OSMAN; LAPORTE, 1996).

A metaheurística GRASP possui diversos relatos de sucesso na literatura e de suas aplicações em problemas do mundo real. GRASP é um método fácil de implementar, usa conceitos simples, é robusto, tende a obter soluções de boa qualidade, e é eficaz, pois produz soluções com tempo computacional relativamente baixo (MESTRIA, 2011). O procedimento é composto por

duas etapas: uma **fase construtiva**, que gera uma solução inicial de forma gulosa-aleatória, e uma **fase de busca local**, responsável por melhorar esta solução. O controle entre gula e aleatoriedade na etapa de geração da solução é feito por meio de um parâmetro α. O GRASP Reativo (PRAIS; RIBEIRO, 2000) é uma versão melhorada do GRASP tradicional, onde, em vez de usar um único valor para o parâmetro α, um conjunto de valores de α previamente determinados são utilizados.

Uma característica marcante das metaheurísticas é a necessidade do ajuste dos seus parâmetros. A qualidade das soluções e o tempo de execução destes algoritmos estão diretamente relacionados à configuração adequada dos valores desses parâmetros. Mesmo assim, para problemas de grande porte, a qualidade das soluções ou o tempo de processamento podem ser melhorados. Para isso, pesquisadores têm utilizado o conceito de **metaheurísticas híbridas**, que consiste em associar duas metaheurísticas ou uma metaheurística com conceitos e técnicas de outras áreas de pesquisa.

A exploração de metaheurísticas híbridas tem sido uma tendência na pesquisa em otimização combinatória, ganhando um aumento considerável no interesse entre os pesquisadores dessa área (TALBI, 2002). Isso porque os algoritmos híbridos têm obtido melhores resultados para muitos problemas de otimização práticos e acadêmicos.

Versões híbridas da metaheurística GRASP com outras metaheurísticas e com técnicas de outras áreas têm sido propostas desde a sua concepção. Em (DELMAIRE *et al.*, 1999), (BELTRÁN *et al.*, 2004) e (RIBEIRO; URRUTIA, 2007) a hibridização ocorre com metaheurísticas clássicas. Em (RIBEIRO *et al.*, 2002), (OLIVEIRA *et al.*, 2004) e (MARINAKIS *et al.*, 2005) os métodos híbridos são obtidos utilizando-se estratégias de otimização e de pesquisa operacional. Já em (SANTOS *et al.*, 2005) e (RIBEIRO *et al.*, 2006) técnicas de mineração de dados são utilizadas no processo de hibridização. Recentemente, (LIMA JÚNIOR *et al.*, 2007), (LIMA JÚNIOR *et al.*, 2008) e (SANTOS *et al.*, 2009) propuseram versões híbridas do GRASP por meio da utilização de aprendizagem por reforço.

Aprendizagem por Reforço (AR) – do inglês, *Reinforcement Learning* – é um paradigma da Inteligência Artificial com aplicações de sucesso em vários problemas de otimização (BARTO; CRITES, 1996), (AYDIN; ÖZTEMEL, 2000), (STOCKHEIM *et al.*, 2003), (LI *et al.*, 2007a), (ZOLFPOUR-AROKHLO *et al.*, 2014). A AR possibilita a um agente aprender a partir da

interação com o ambiente no qual ele está inserido. Três fatores estão presentes no processo de aprendizagem: o conhecimento do estado do agente no ambiente, as ações efetuadas e as mudanças de estado decorrentes das ações. Dessa forma, o agente não precisa conhecer o ambiente *a priori*, pois possui a capacidade de evoluir por meio da identificação das características do ambiente com o qual está interagindo. Essa característica de aprender de forma autônoma em ambientes total ou parcialmente desconhecidos é uma das maiores vantagens da AR. De acordo com (SUTTON; BARTO, 1998), a aprendizagem por reforço não é definida por caracterizar um método de aprendizagem, e sim por caracterizar um *problema* de aprendizagem. Qualquer método que seja bem adequado a esse problema é considerado um método de AR.

O algoritmo Q-learning (WATKINS, 1989) pode ser considerado o mais importante método de AR (STOCKHEIM *et al.*, 2003), e consiste num método no qual um agente toma decisões e as avalia de acordo com os benefícios ou perdas que estas decisões lhe trouxerem. Ele é bastante utilizado em vários trabalhos científicos e é de fácil implementação devido à sua lógica simples. Hibridizações bem sucedidas foram obtidas por meio da associação do Q-learning com metaheurísticas (CHANG, 2004), (LI *et al.*, 2007b), (SANTOS *et al.*, 2014).

Em relação à utilização do GRASP e AR, (LIMA JÚNIOR *et al.*, 2008) e (SANTOS *et al.*, 2009) utilizaram o Q-learning em conjunto com o GRASP tradicional, GRASP Reativo e Algoritmo Genético. No primeiro trabalho, a fase construtiva do GRASP e GRASP Reativo é substituída por uma estratégia que gera uma solução para o Problema do Caixeiro Viajante (PCV) a partir da matriz dos Q-valores, que representa o conhecimento adquirido pelo Q-learning. O agente funciona como uma memória adaptativa, uma vez que é atualizado a cada iteração do GRASP de modo que as informações obtidas em iterações passadas não são perdidas. No trabalho de Santos *et al.* (2009), em vez de utilizar o Q-learning apenas como técnica de geração das soluções iniciais das metaheurísticas, este também é aplicado de forma cooperativa e competitiva com o Algoritmo Genético e GRASP em uma implementação paralela para o PCV.

Vale ressaltar que não foram encontrados na literatura outros trabalhos envolvendo GRASP e AR.

Neste trabalho, apresentamos uma versão híbrida da metaheurística GRASP Reativo que incorpora um agente de AR, especificamente o algoritmo Q-learning. No GRASP Reativo, a escolha do α a ser utilizado na fase construtiva é feita a partir do conjunto de valores α e é

baseada em uma distribuição de probabilidades associadas. No algoritmo híbrido proposto, o agente aprendiz é utilizado para substituir esse processo, e o parâmetro α é obtido a partir da matriz dos Q-valores, que representa o conhecimento do agente adquirido até o momento.

O presente trabalho difere dos anteriores em dois aspectos. Primeiro, não há necessidade de modelar os problemas de teste (problemas reais a serem resolvidos) como um Processo de Decisão de Markov (PDM), bastando apenas modelar o problema de aprendizagem como tal. Este problema consiste em aprender e fornecer o melhor parâmetro α a ser utilizado na fase construtiva do GRASP Reativo. Em segundo lugar, a estratégia de aprendizagem utilizada aqui usa a matriz dos Q-valores retornada pelo Q-learning para selecionar qual α será utilizado na fase de construção da solução, não havendo nenhuma alteração nesta fase. Essa estratégia é detalhada na Seção 3.2.

O método híbrido será utilizado para determinar a localização de pontos estratégicos para instalação de bases policiais na cidade de Mossoró/RN, visando o combate a uma das principais dificuldades enfrentadas pelos gestores nos últimos anos: a criminalidade. Assim, objetiva-se reduzir o tempo de resposta policial, de forma a diminuir os custos operacionais e aumentar a eficiência no controle à criminalidade.

O problema é modelado matematicamente como um Problema dos *p*-Centros (PPC), uma clássica formulação para o Problema de Localização de Facilidades (PLF). Em PLFs que envolvem serviços de emergências, é interessante que a distância do usuário mais distante de sua facilidade seja a menor possível. O PPC procura minimizar a máxima distância e pertence à classe dos problemas NP-difíceis (GAREY; JOHNSON, 1979).

1.2 MOTIVAÇÃO

A qualidade das soluções obtidas pelo GRASP depende fortemente da geração de soluções iniciais de boa qualidade (FEO; RESENDE, 1995), (PRAIS; RIBEIRO, 2000), (RESENDE; RIBEIRO, 2010). Quanto melhores e mais diversificadas forem as soluções de partida, mais eficiente será a busca local. Soluções melhores tornam a busca local mais rápida, enquanto que soluções diferenciadas possibilitam uma maior exploração do espaço de busca. Desse modo, o desempenho do método é melhorado.

Procedimentos GRASP não guardam informações do processo de busca em iterações an-

teriores. Esta característica representa uma desvantagem, pois desperdiça a oportunidade de tirar proveito de boas escolhas tomadas em instantes passados. Dessa forma, a utilização de um agente de AR supre essa deficiência dotando o método de uma memória adaptativa, uma vez que é atualizada com base na experiência adquirida ao longo das iterações. Essa abordagem permite um melhor aproveitamento das informações do processo de busca em iterações passadas (LIMA JÚNIOR, 2009).

No GRASP Reativo, a escolha do α é feita com base em uma distribuição de probabilidade. Uma vez que as probabilidades manipulam o grau de incertezas dadas as evidências, não há qualquer garantia quando se trabalha com elas, mas apenas chances de ocorrência dos eventos. Assim, a melhor decisão a ser tomada nem sempre é a escolhida. Usar o conhecimento armazenado pelo processo de aprendizagem do agente para fornecer o α mais adequado permite que, a qualquer momento, um α que gere boas soluções seja selecionado. Além disso, (LIMA JÚNIOR *et al.*, 2007, 2008) e (SANTOS *et al.*, 2009) aplicaram AR com sucesso ao GRASP obtendo melhorias significativas na performance dos algoritmos tradicional e reativo.

1.3 OBJETIVO GERAL

Em procedimentos GRASP, o ajuste do parâmetro α define a qualidade das soluções geradas. Valores altos de α favorecem a aleatoriedade, enquanto que baixos valores priorizam a escolha gulosa. A variação de parâmetros em vez de um α fixo possibilita a obtenção de soluções com maior grau de diversidade, tornando as etapas de explotação e exploração da metaheurística mais equilibradas. Essa ideia justifica o melhor desempenho da versão reativa do GRASP em relação à sua versão tradicional. Priorizar a escolha de valores α que geraram boas soluções em iterações passadas permite que soluções cada vez melhores sejam obtidas. Assim, usar um agente inteligente com habilidade para tomar decisões com base em experiências passadas e que usa estratégias de exploração/explotação para gerar seu aprendizado representa uma maior garantia de que valores mais adequadas de α serão selecionados em todas as partidas do GRASP.

Nesse contexto, este trabalho tem como objetivo utilizar aprendizagem por reforço para melhorar o desempenho da metaheurística GRASP Reativo quando aplicada ao PPC. Será utilizada uma estratégia inteligente para substituir o mecanismo reativo do GRASP, que usa conhecimento probabilístico.

1.4 OBJETIVOS ESPECÍFICOS

Como objetivos específicos deste trabalho podemos destacar:

- Modelagem do problema de aprendizagem como um PDM;
- Elaboração de uma estratégia de seleção do parâmetro α a partir do aprendizado adquirido pelo algoritmo Q-learning. Para isto, utiliza-se a matriz dos Q-valores como mecanismo de memória adaptativa;
- Implementação da metaheurística GRASP Reativo e do algoritmo híbrido proposto;
- Geração de um mapeamento da criminalidade na cidade de Mossoró/RN com base no histórico de localização de crimes graves nesta cidade;
- Criação de um conjunto de instâncias de teste para PLFs;
- Realização de experimentos computacionais com os dois algoritmos aplicados ao PPC;
- Análise do desempenho dos métodos;
- Validação dos resultados obtidos com o método proposto, por meio de uma análise estatística dos dados experimentais.

1.5 ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação encontra-se dividida em cinco capítulos. No Capítulo 2 é apresentada uma fundamentação teórica para Problemas de Localização de Facilidades (PLF), metaheurística GRASP Reativo, Processos de Decisão de Markov (PDM) e Aprendizagem por Reforço (AR), com enfoque no algoritmo Q-learning. Além disso, o capítulo apresenta informações relacionadas à criminalidade e traz a formulação matemática de alguns modelos de PLF. No Capítulo 3 o método híbrido proposto e a modelagem do problema como um PDM são descritos. O Capítulo 4 descreve o processo de criação das instâncias de teste e apresenta os resultados dos experimentos computacionais, bem como é feita uma análise estatística dos mesmos. Por fim, são apresentadas no Capítulo 5 as conclusões, e as sugestões de trabalhos futuros são apontadas. Em anexo, o Apêndice A disponibiliza a listagens de todos os resultados obtidos nos experimentos realizados.

CAPÍTULO 2

REVISÃO BIBLIOGRÁFICA

Neste capítulo são apresentados os fundamentos acerca dos temas abordados no presente trabalho. As próximas seções trazem a definição e os principais conceitos da metaheurística GRASP Reativo, definições para a Aprendizagem por Reforço (AR) e Processos de Decisão de Markov (PDM), bem como detalhes do algoritmo Q-learning, que será a técnica utilizada em conjunto com a metaheurística GRASP Reativo. Dados sobre a criminalidade e conhecimentos básicos sobre Problemas de Localização de Facilidades (PLF) também são apresentados.

2.1 CRIMINALIDADE

O crescimento dos índices de violência no Brasil e a falta de indicadores sociais de segurança pública tem sido uma das grandes dificuldades alegadas pelos gestores no que diz respeito à implementação racional e eficiente de políticas e programas de controle, prevenção e combate ao crime.

O quadro atual da violência urbana e da criminalidade no Brasil apresenta constante aumento nos índices sobre a influência de fatores culturais e sócio-econômicos, tais como: crescimento desordenado da população, falta de sistema educacional eficiente, inexistência de reais mecanismos de distribuição de renda, desiguais oportunidades no mercado de trabalho, tráfico e uso de entorpecentes, e principalmente a confiança ou quase certeza da impunidade por parte do infrator. Além disso, a quantidade insuficiente de recursos investidos em segurança pública torna difícil o trabalho da polícia. O constante sentimento de insegurança por parte do cidadão desvirtua a vida em nossas cidades, transformando a convivência e o usufruo de espaços públicos e de participação coletiva em um ambiente de medo que precisa a todo custo ser evitado.

De acordo com os números divulgados em uma prévia do Mapa da Violência 2014 (WAISELFISZ, 2014), o Rio Grande do Norte teve o maior percentual de crescimento nos casos de homicídios entre os anos de 2002 e 2012. O aumento no número de assassinatos foi de 272,4%, passando de 301 homicídios, em 2002, para 1.221, em 2012. No mesmo período, a variação na taxa de assassinatos por cada grupo de 100 mil habitantes foi a maior do país, atingindo 229,1%.

Segundo o Conselho Estadual de Direitos Humanos e Cidadania (COEDHUCI) do RN, no ano de 2013, o Rio Grande do Norte atingiu a marca de 1.636 crimes de homicídio. A situação atual dos índices de violência mostra que o Estado vem enfrentando uma total fragilização dos órgãos ligados à área da Segurança Pública, sobretudo das polícias Civil e Militar.

Esse quadro também se mantém na cidade de Mossoró, situada na região oeste do estado do Rio Grande do Norte, distando cerca de 280 quilômetros da capital Natal. Esta cidade é o segundo município mais populoso do estado, e segundo o Instituto Brasileiro de Geografia e Estatística (IBGE), é o principal município da Costa Branca Potiguar.

Mossoró tem apresentado um notável desenvolvimento econômico e na infraestrutura decorrente da instalação de indústrias, do crescimento no ramo da construção civil e do incremento na oferta do número de vagas no ensino superior. No entanto, o aparente progresso da cidade nos últimos anos vem acompanhado de um crescimento significativo nos índices de criminalidade e violência.

Ainda segundo o COEDHUCI, Mossoró foi o segundo município mais violento em 2013, registrando 179 homicídios, uma taxa alarmante de quase um homicídio a cada dois dias, estatística ultrapassada apenas pela capital do estado. Considerando-se que a criminalidade não consiste apenas em crimes de homicídios, a situação se torna ainda mais assustadora. Em uma enquete permanente realizada na página de Mossoró no portal G1 (http://g1.globo.com/rn/rio-grande-do-norte/cidade/mossoro.html), a segurança é a principal preocupação do mossoroense, atingindo o índice de 38%.

O sucesso no combate à criminalidade não depende apenas de vontade política, boas estratégias de atuação ostensiva ou até mesmo de um elevado número de policiais nas ruas, mas está condicionado ao bom aparelhamento policial, aos investimentos em inteligência policial e pesquisa científica aplicada e principalmente ao uso de modernas tecnologias da informação na obtenção e gerenciamento de dados que permitam o controle e a prevenção do delito, e que sejam capazes de garantir a precisão necessária para uma atuação investigativa eficaz e eficiente.

Neste contexto, uma outra vertente deste trabalho é a realização de um mapeamento da criminalidade na cidade de Mossoró, através do uso de um software SIG (Sistema de Informação Geográfica), com base no histórico de localização de crimes graves nesta cidade. Mapeamento é o processo de registrar informações geográficas ou de simples localização em forma de mapa.

O mapeamento da criminalidade se dará por meio apenas da localização, no mapa vetorial da referida cidade, dos locais com maior incidência dos crimes de roubos, furtos, assaltos, estupros, disparos de armas de fogo, invasões e homicídios, de modo a permitir a visualização e compreensão da distribuição espacial das ocorrências e possibilitar a criação de um conjunto de instâncias de teste que serão as entradas para as metaheurísticas implementadas neste trabalho. Não é objeto deste trabalho a identificação dos horários de maior incidência, as causas dos crimes, o perfil das vítimas ou dos criminosos, dentre outras informações.

2.2 PROBLEMAS DE LOCALIZAÇÃO DE FACILIDADES

O Problema de Localização de Facilidades – PLF (do inglês, *facility location problem*) consiste em determinar/localizar um subconjunto de locais, a partir de um conjunto de m locais candidatos, para a instalação/alocação de facilidades que atendam às demandas de n clientes, com o objetivo de otimizar um ou mais critérios definido na modelagem do problema. O termo "facilidades" é utilizado para designar postos de saúde ou de polícia, depósitos, lojas, escolas, fábricas, antenas de transmissão, pontos de ônibus, concentradores em redes de computadores, entre outros. Os "clientes", "usuários" ou "pontos de demanda" são todos os interessados nos serviços prestados ou produtos fornecidos pelas facilidades, tais como bairros, unidades de vendas, estudantes, funcionários, computadores, etc.

Localizar ou alocar facilidades é uma importante decisão estratégica a ser tomada por organizações privadas ou públicas (OWEN; DASKIN, 1998). A escolha de bons locais para a instalação ou expansão de facilidades pode levar vários fatores em consideração, como quantidade de clientes que serão servidos, custo de instalação, custo operacional, entre outros.

Os PLFs são considerados problemas NP-difícil (GAREY; JOHNSON, 1979), ou seja, são problemas complexos de natureza combinatória para os quais não se conhecem algoritmos polinomiais capazes de obter a solução exata, exigindo grande esforço computacional e tornando-os de difícil solução por meio de algoritmos exatos. Esta complexidade tem motivado a busca por heurísticas com o intuito de alcançar melhores resultados.

Na literatura, existem várias formulações ou modelos de problemas de localização. A diferença entre estes problemas está na maneira como as demandas dos clientes e a localização das facilidades são representadas. O critério a ser otimizado depende do problema que está sendo

modelado. As próximas seções apresentam três dos principais modelos para PLF.

2.2.1 Problema das *p*-Medianas

O Problema das *p*-Medianas (PPM) – *p-Median Problem* – é um problema clássico de localização de facilidades proposto inicialmente para uma única mediana por Hakimi (1964) e posteriormente generalizado para múltiplas medianas (HAKIMI, 1965).

O PPM consiste em localizar p facilidades (medianas) e associar cada cliente à facilidade mais próxima a este, de modo a minimizar a soma total das distâncias de cada cliente à sua mediana mais próxima. Modelos que minimizam a distância média ou total são os mais adequados para descrever os problemas que ocorrem no setor privado, uma vez que os custos estão diretamente relacionados com as distâncias para a satisfação dos clientes.

Segundo (MLADENOVIĆ *et al.*, 2007), o PPM pode ser formulado como um problema de programação inteira 0-1 como se segue. Sejam F um conjunto de m facilidades, C um conjunto de n clientes, D uma matriz $m \times n$ com as distâncias de percurso, e dois conjuntos de variáveis de decisão: (i) $y_j = 1$, se uma facilidade é alocada em $j \in F$, e 0, caso contrário; e (ii) $x_{ij} = 1$, se o cliente i é atendido por uma facilidade aberta em $j \in F$, e 0, caso contrário. O objetivo é minimizar o somatório das distâncias entre os usuários e as facilidades ao qual foram alocados, como apresentado em (2.1) a seguir:

$$min \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} x_{ij} \tag{2.1}$$

sujeito a

$$\sum_{j=1}^{m} x_{ij} = 1, \forall i, \tag{2.2}$$

$$x_{ij} \leq y_j, \forall i, j, \tag{2.3}$$

$$\sum_{j=1}^{m} y_j = p, \tag{2.4}$$

$$x_{ij},\ y_j \in \{0,1\}, \forall i, j. \tag{2.5}$$

A restrição (2.2) garante que a demanda de cada cliente deve ser atendida. A restrição (2.3) previne que um usuário seja atendido por outro que não seja uma facilidade alocada. Na restrição (2.4) p é definido como o número total de instalações abertas. O domínio das variáveis

é dado em (2.5).

O PPM otimiza o caso médio, ou seja, a distância média entre clientes e facilidades é baixa. Não há preocupação com as distâncias individuais de cada cliente até sua facilidade associada e como consequência disso, alguns clientes podem se localizar muito distantes de sua facilidade em relação a outros clientes. Este fator inviabiliza o uso dessa abordagem para resolver problemas que envolvam serviços de emergência.

2.2.2 Problema de Localização de Máxima Cobertura

O Problema de Localização de Máxima Cobertura (PLMC) – *Maximal Covering Location Problem* – foi formalizado inicialmente por (CHURCH; REVELLE, 1974) e representa um modelo relaxado[1] do Problema de Cobertura de Conjuntos (*Set Covering Problem*). O Problema de Cobertura de Conjuntos determina o número mínimo de instalações que são necessárias para atender a todos os clientes, para uma dada distância de cobertura (distância máxima de serviço). Modelos de cobertura são frequentemente encontrados em problemas relacionados ao setor público.

Devido a restrições de formulação, este modelo não considera a demanda individual de cada cliente. Além disso, o número de facilidades necessárias pode ser grande, incorrendo em elevados custos de instalação. Assim, o PLMC considera a instalação de um número limitado de instalações, mesmo se esse valor não for capaz de atender a demanda total.

O objetivo do PLMC é localizar p facilidades, de modo que a maior parte da demanda de clientes existente possa ser atendida, para uma dada distância de cobertura.

Dados os conjuntos F de m possíveis locais para facilidades e U de n usuários. Seja C a distância de cobertura, d_{ij} a menor distância do usuário $i \in U$ à facilidade $j \in F$ e D_i a demanda do usuário i, representando a importância em atender o usuário i. O objetivo é maximizar a quantidade de usuários cobertos, levando-se em consideração a demanda do usuário, como em (2.6) no modelo abaixo. O PLMC é formulado para programação linear inteira, segundo (CHURCH; REVELLE, 1974), como:

$$max \sum_{i \in U} D_i x_i \tag{2.6}$$

[1] Um problema é dito relaxado quando uma ou mais restrições do problema original são removidas.

sujeito a

$$\sum_{j \in N_i} y_j \geq x_i, \forall i, \tag{2.7}$$

$$\sum_{j \in F} y_j = p, \tag{2.8}$$

$$y_j,\ x_i \in \{0,1\}, \forall i, j. \tag{2.9}$$

onde $x_i = 1$, se o usuário i é coberto por uma facilidade dentro de C, e 0, caso contrário; $y_j = 1$, se a facilidade é alocada em $j \in F$, e 0, caso contrário; $N_i = j \in F | d_{ij} \leq C$ é o conjunto de locais elegíveis para prover cobertura ao usuário i. A restrição (2.7) diz que o usuário i é coberto se há pelo menos uma facilidade dentro da distância C. O número de facilidades alocadas é restrito a p na restrição (2.8). A restrição (2.9) apresenta o domínio das variáveis.

Por não garantir o atendimento de todos os clientes, o PLMC não é adequado para alocar facilidades que prestarão serviços de emergência.

2.2.3 Problema dos *p*-Centros

O Problema dos p-Centros (PPC) – *p-Center Problem* – tem sido de interesse desde a sua primeira aparição, em 1964, por Hakimi (HAKIMI, 1964) e tem sido bastante estudado em conjunto com o PPM.

O PPC consiste em localizar p facilidades (centros) e associar cada cliente à facilidade mais próxima, de maneira que a distância máxima de qualquer cliente até sua facilidade seja mínima. Desta forma, o problema busca maximizar a cobertura (clientes atendidos) dentro de uma distância determinada. Assim, um ponto de demanda é considerado coberto se está no raio de serviço de pelo menos um centro. Ao contrário do PLMC, este modelo exige que todos os clientes sejam atendidos.

Considere um conjunto F de m possíveis localizações para facilidades (centros), um conjunto U de n usuários e d_{ij} a distância entre o usuário $i \in U$ e a facilidade $j \in F$. O objetivo é minimizar a distância máxima entre centros e usuários, como apresentado em (2.10). Segundo (MLADENOVIĆ *et al.*, 2003), uma formulação do PPC para programação linear inteira é feita como segue:

$$Minimizar\ z \tag{2.10}$$

sujeito a

$$z \geq \sum_{j=1}^{m} d_{ij} x_{ij}, \forall i \tag{2.11}$$

$$\sum_{j=1}^{m} x_{ij} = 1, \forall i \tag{2.12}$$

$$x_{ij} \leq y_j, \forall i, j, \tag{2.13}$$

$$\sum_{j=1}^{m} y_j = p, \tag{2.14}$$

$$x_{ij},\ y_j \in \{0,1\}, \forall i, j. \tag{2.15}$$

onde $y_j = 1$, se um centro é alocado em $j \in F$, e 0, caso contrário; e $x_{ij} = 1$, se o usuário i é atendido por uma centro aberto em $j \in F$, e 0, caso contrário. A variável z é definida em (2.11) como a maior distância entre um usuário e seu centro aberto mais próximo. A restrição (2.12) assegura que a demanda de cada usuário deve ser atendida. A restrição (2.13) evita qualquer usuário de ser atendido por um local que não seja um centro alocado. O número total p de centros abertos é definido pela restrição (2.14). Em (2.15) é feito o domínio das variáveis.

A principal característica desse modelo é o fato de que a otimização se dá no pior caso, bem como tende a encontrar soluções onde as distâncias entre os usuários e as facilidades são mais igualitárias e as menores possíveis. Logo, este problema é adequado à instalação de facilidades que prestam serviços emergenciais, visto que, quanto mais afastados das facilidades os usuários tiverem, mais prejudicados serão, já que maior será a espera pelo atendimento.

A próxima seção apresenta a metaheurística GRASP tradicional e sua versão reativa. A modelagem para o PPC das heurísticas de construção e busca local utilizadas pela metaheurística GRASP Reativo será feita no capítulo a seguir.

2.3 METAHEURÍSTICA GRASP

A metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*) foi proposta em 1995 por Thomas Feo e Maurício Resende (FEO; RESENDE, 1995), embora sua origem remeta a trabalhos anteriores, como (LIN; KERNIGHAN, 1973), (HART; SHOGAN, 1987) e (FEO; RESENDE, 1989). Desde sua concepção, o GRASP tem sido aplicado com sucesso a diversos problemas de otimização combinatória (FESTA; RESENDE, 2009).

GRASP é um procedimento *multi-start* iterativo, ou seja, um processo que consiste em gerar

aleatoriamente soluções iniciais diferentes e aplicar em seguida uma rotina de busca local com o intuito de fazê-las convergirem para ótimos locais. Logo, em cada iteração do GRASP tem-se a execução de duas fases (FEO; RESENDE, 1995), (RESENDE; RIBEIRO, 2003), a saber:

Construtiva: constrói uma solução viável para o problema de forma gulosa-aleatória.

Busca local: procura melhorar a qualidade da solução gerada anteriormente.

O Algoritmo 2.1 ilustra as etapas do GRASP para um problema de minimização da função objetivo f.

Algoritmo 2.1 Algoritmo GRASP básico.

```
1: procedimento GRASP(α)
2:     f(s*) ← +∞
3:     enquanto não CritérioParada faça
4:         s1 ← FaseConstrutiva(α)
5:         s2 ← BuscaLocal(s1)
6:         se f(s2) < f(s*) então
7:             s* ← s2
8:         fim se
9:     fim enquanto
10:    retorne s*
11: fim procedimento
```

Observe que a melhor solução s^* encontrada até então (linha 7) é armazenada. Durante as múltiplas partidas, uma solução é construída (linha 4), melhorada (linha 5) e então avaliada e, se houver melhora, s^* é atualizada (linhas 6 a 8). Embora um número máximo de iterações seja utilizado normalmente como critério de parada, abordagens diferentes podem ser utilizadas para esse fim, tais como um número de iterações consecutivas sem melhoria da solução, localização da solução ótima (RANGEL *et al.*, 2000), estabelecimento de um tempo máximo de execução em função do tamanho da instância (KAMPKE, 2010) e definição de uma solução corte (quando a função objetivo atinge um valor pré-estabelecido) a ser atingida. Ao final da execução, s^* é retornada como solução final (linha 10).

Pode-se observar também que o Algoritmo 2.1 é de fácil implementação e que os únicos parâmetros a serem ajustados são o parâmetro de aleatoriedade α e o critério de parada.

Por ser uma metaheurística multipartida, as iterações do GRASP são totalmente independentes onde, a cada reinício, as informações da iteração anterior são perdidas. Isso representa uma desvantagem, pois o método não tira proveito de boas decisões tomadas no passado.

2.3.1 Fase Construtiva

O GRASP é um método heurístico que combina as boas características dos algoritmos puramente gulosos e dos procedimentos aleatórios na sua fase de construção de soluções factíveis.

A fase construtiva é iterativa, gulosa-aleatória e adaptativa. Ela constrói uma solução de forma iterativa adicionando um elemento por vez à solução em construção. Cada elemento é escolhido de forma aleatória a partir de uma Lista Restrita de Candidatos (LRC). O caráter adaptativo decorre do fato de a escolha do próximo elemento a compor a solução parcial ser influenciada pelas escolhas anteriores. Ou seja, quando o elemento é inserido na solução parcial, ele é removido da Lista de Candidatos (LC) possíveis, sendo esta então atualizada e os custos incrementais recalculados.

O Algoritmo 2.2 apresenta o pseudocódigo da fase de construção do GRASP.

Algoritmo 2.2 Algoritmo guloso-aleatório usado na fase construtiva.

```
1: procedimento FASECONSTRUTIVA(α)
2:     Inicializar a lista de candidatos LC
3:     s ← ∅ {Inicialmente a solução está vazia}
4:     enquanto LC ≠ ∅ faça
5:         LRC ← ConstruirLRC(LC, α)
6:         e ← SelecionarAleatorio(LRC)
7:         s ← s ∪ {e}
8:         Atualizar a lista de candidatos LC
9:     fim enquanto
10:    retorne s
11: fim procedimento
```

Considere um problema de minimização e um conjunto LC = $\{e_1, e_2, ..., e_N\}$, onde $N = |E|$ e E é o conjunto formado por todos os possíveis elementos candidatos a fazerem parte da solução em construção (uma lista de candidatos). Seja $c(e_i)$ uma função gulosa que designa o custo incremental associado à incorporação do elemento $e_i \in$ LC na solução corrente parcial. Representa-se por c^{min} e c^{max}, o menor e o maior custo incremental, respectivamente.

Relacionado à construção da LRC existe o parâmetro $\alpha \in [0,1]$ que determina o tamanho da LRC. O processo construtivo da LRC utiliza os elementos $e_i \in$ LC com os menores custos incrementais $c(e_i)$, e pode ocorrer de duas formas:

1. **Baseada na cardinalidade da LRC:** a LRC é constituída de acordo com os p elementos com os melhores custos incrementais, onde p é um parâmetro. Ou seja, seleciona-se os

primeiros p elementos a partir de uma lista de candidatos LC ordenada de forma decrescente segundo a função gulosa $c(e)$. O valor de p é calculado por:

$$p = 1 + \alpha(N-1) \tag{2.16}$$

onde N é o número total de elementos candidatos.

2. **Baseada na qualidade dos elementos:** a lista apresenta tamanho variável, já que depende da quantidade de elementos com bons valores de critério. A LRC é formada por todos os elementos $e_i \in$ LC cujos valores associados da função $c(e)$ respeitam a condição imposta pela equação 2.17. Assim, verifica-se que:

$$LRC = \{e \in LC | c(e) \leq c^{min} + \alpha(c^{max} - c^{min})\} \tag{2.17}$$

O controle entre gula e escolha aleatória é feito pelo parâmetro α, de modo que no caso $\alpha = 0$, o algoritmo é puramente guloso, pois a LRC conterá apenas o elemento c^{min}, enquanto $\alpha = 1$ corresponde ao algoritmo totalmente aleatório, já que a lista conterá todos os candidatos possíveis. Para um problema de maximização, a LRC será constituída por todos os elementos cujo valor da função de avaliação está no intervalo $[c^{max} + (1-\alpha)(c^{min} - c^{max}), c^{max}]$.

A presença da aleatoriedade na geração da solução de partida garante que novos mínimos locais possam ser alcançados, e também representa o aspecto probabilístico da metaheurística. Dessa forma, a fase construtiva é responsável pela **exploração** da busca no espaço de soluções. Sem o elemento aleatório, fase construtiva mais busca local só poderia ser aplicada uma única vez (RESENDE; RIBEIRO, 2003).

2.3.2 Busca local

A fase construtiva não garante a geração de ótimos locais, sendo, portanto, fortemente recomendada uma etapa de busca local. Esta fase representa a etapa de **explotação** da metaheurística, pois investiga a vizinhança da solução gerada até encontrar um mínimo local.

Usualmente, vizinhanças simples são utilizadas. Dado um conjunto discreto de soluções S, uma vizinhança (proximidade) de uma solução $s \in S$ é um subconjunto ou mapeamento de S, ou seja, $N(s) = \{s_1, s_2, ..., s_k\}, k \leq |S|$, onde qualquer elemento de $N(s)$ é um vizinho de s. Um

vizinho é alcançado a partir de uma solução por meio de um *movimento*. Uma solução s^+ é dita um **ótimo local** se não houver qualquer solução vizinha melhor que ela. Um **ótimo global** (ou solução ótima) s^* é a melhor solução presente no espaço de busca.

Os algoritmos de busca local apresentam três etapas durante a exploração de uma determinada vizinhança no espaço de soluções. A partida é uma solução obtida por um método construtivo. Na etapa de iteração, a solução corrente é substituída sucessivamente pela melhor encontrada por uma busca na sua vizinhança. A parada do algoritmo ocorre quando um ótimo local é encontrado. O Algoritmo 2.3 demonstra o procedimento básico de busca local para um problema de minimização.

Algoritmo 2.3 Algoritmo de busca local.

```
1: procedimento BUSCALOCAL(s)
2:     enquanto ∃s' ∈ N(s) tal que f(s') < f(s) faça
3:         Selecione s' ∈ N(s) {Busca na vizinhança da solução}
4:         se f(s') < f(s) então
5:             s ← s'
6:         fim se
7:     fim enquanto
8:     retorne s {Ótimo local}
9: fim procedimento
```

Em problemas de sequenciamento, tais como o problema do Caixeiro Viajante e o problema de Alocação de Tarefas, normalmente são utilizados movimentos baseados em trocas de posição entre elementos da sequência que forma a solução, de modo a obter um novo ordenamento. Movimentos do tipo 2-*Opt* e 3-*Opt* são os mais comuns, e significam, respectivamente, que dois e três movimentos são necessários para transformar uma sequência em outra. O método 2-*Opt* parte de uma solução corrente, testa todas as combinações obtidas com a troca de posição entre dois elementos e escolhe a combinação que retorna o melhor valor da função de custo (JOHNSON; MCGEOCH, 2003).

A busca pode ser implementada utilizando uma estratégia *first-improving*, que substitui a solução corrente quando a primeira solução aprimorante é encontrada, ou *best-improving*, que investiga toda a vizinhança e então a solução corrente move-se para o melhor vizinho encontrado. Outras estratégias de busca têm sido aplicadas ao GRASP, como o método *Variable Neighborhood Descent* (MESTRIA, 2011), Busca Tabu (ABDINNOUR-HELM; HADLEY, 2000) e *Simulated Annealing* (LIU *et al.*, 2000).

A eficiência de um procedimento de busca local depende de vários aspectos, tais como a estrutura da vizinhança, a estratégia de busca na vizinhança, a rápida avaliação da função de custo dos vizinhos, e da qualidade da solução de partida (RESENDE; RIBEIRO, 2003).

2.3.3 GRASP Reativo

O GRASP Reativo difere do tradicional apenas em relação ao parâmetro α, que tem seu valor auto-ajustado ao longo das iterações em função da qualidade das soluções obtidas anteriormente. O método foi proposto por Marcelo Prais e Celso Ribeiro (PRAIS; RIBEIRO, 2000) e foi motivado pelas discussões feitas por (FEO; RESENDE, 1995) acerca do efeito do valor do α na qualidade e na diversidade das soluções geradas. Os autores mostraram que a utilização de um único valor fixo para α frequentemente impede a obtenção de soluções de melhor qualidade, que poderiam ser geradas se outros valores de α fossem utilizados. O estudo mostra ainda que a melhor solução para o problema proposto é frequentemente encontrada utilizando-se a estratégia com variação do parâmetro α.

No GRASP Reativo, o α a ser utilizado na fase de construção da iteração corrente é selecionado de forma aleatória a partir de um conjunto $\Psi = \{\alpha_1, \alpha_2, ..., \alpha_m\}$ contendo m valores pré-determinados. A cada $\alpha_k \in \Psi$ existe uma probabilidade p_k associada, que será utilizada como base para escolha do α_k. Inicialmente, todas as probabilidades são iguais, ou seja, $p_i = 1/m, i = 1, ..., m$. As probabilidades $\{p_1, p_2, ..., p_m\}$ são recalculadas de maneira adaptativa, normalmente a cada y iterações, para favorecer valores de α que tenham produzido boas soluções. Dessa forma, estes valores terão maior probabilidade e, consequentemente, terão maiores chances de serem escolhidos nas próximas iterações.

Para atualizar a distribuição de probabilidades é utilizada a seguinte expressão:

$$p_i = \frac{q_i}{\left(\sum_{j=1}^{m} q_j\right)}, \forall i = 1, 2, ..., m \tag{2.18}$$

onde

$$q_i = \left(\frac{f(s^*)}{m_i}\right)^{\delta}, \forall i = 1, 2, ..., m. \tag{2.19}$$

Na equação 2.19, considera-se $f(s^*)$ o valor da função objetivo da melhor solução encontrada até o momento e m_i o valor médio de todas as soluções encontradas usando $\alpha = \alpha_i$, com

$i = 1, ..., m$. O parâmetro de amplificação δ tem o objetivo de penalizar as probabilidades cuja média m_i é muito maior que $f(s^*)$ amplificando a diferença entre os valores q_i.

Quanto menor for o valor de m_i, maior será o valor de q_i e, por consequência, a probabilidade recalculada p_i será maior. Dessa forma, nos blocos de iterações seguintes, o GRASP Reativo tende a usar os valores de α que encontraram, em média, as melhores soluções.

A estratégia genérica do GRASP Reativo é mostrada no Algoritmo 2.4. Na linha 9, um $\alpha_i \in \Psi$ é selecionado aleatoriamente, com probabilidade p_i, para ser usado na iteração corrente. As linhas de 15 a 19 correspondem à etapa de atualização das probabilidades p_i e são executadas a cada bloco de y iterações. Na linha 20, o algoritmo passa para a próxima iteração.

Algoritmo 2.4 Pseudocódigo do algoritmo GRASP Reativo.

```
1: procedimento GRASP REATIVO
2:     f(s*) ← +∞
3:     it ← 1
4:     Definir Ψ = {α_1, α_2, ..., α_m}
5:     para i ← 1 até m faça
6:         p_i ← 1/m; m_i ← 0
7:     fim para
8:     enquanto não CritérioParada faça
9:         α ← Selecione α_i ∈ Ψ com probabilidade p_i
10:        s_1 ← FaseConstrutiva(α)
11:        s_2 ← BuscaLocal(s_1)
12:        se f(s_2) < f(s*) então
13:            s* ← s_2
14:        fim se
15:        se it mod y = 0 então
16:            Recalcular m_i, ∀i ∈ {1,2,...,m}
17:            Recalcular q_i, ∀i ∈ {1,2,...,m} pela equação 2.19
18:            Atualizar as probabilidades p_i, ∀i ∈ {1,2,...,m} pela equação 2.18
19:        fim se
20:        it ← it + 1
21:    fim enquanto
22:    retorne s*
23: fim procedimento
```

Pela observação do Algoritmo 2.4, nota-se que o GRASP Reativo adiciona apenas uma pequena sobrecarga sobre a versão tradicional do GRASP, já que a complexidade das fases construtiva e de busca local é preservada.

A eficácia do GRASP Reativo reside no fato de que a utilização de diferentes valores de α em iterações distintas permite a construção de LRC diferentes, possibilitando a geração de

soluções diferenciadas que não seriam obtidas por meio do uso de um único valor fixo de α.

O GRASP Reativo pode ser visto como uma estratégia que usa uma memória de longo prazo. Resumidamente, o que ele faz é aprender durante o processo que valores de α resultaram em solução com melhores custos e favorecer a escolha de tais valores. Este trabalho propõe utilizar um agente inteligente para realizar esta tarefa, em substituição à distribuição de probabilidades, tornando o método mais eficiente.

2.4 APRENDIZAGEM POR REFORÇO

A aprendizagem por reforço – AR (do inglês, *Reinforcement Learning – RL*) é um formalismo da Inteligência Artificial (SUTTON; BARTO, 1998) que preocupa-se com o problema de um agente inteligente aprender, por tentativa e erro, a atingir um objetivo interagindo com um ambiente desconhecido. O objetivo do agente normalmente é escolher ações que visem maximizar os ganhos acumulados durante o aprendizado.

A Figura 2.1 (traduzida a partir da original, obtida em (SUTTON; BARTO, 1998)) representa o modo como um agente de AR interage com o ambiente durante processo de aprendizagem.

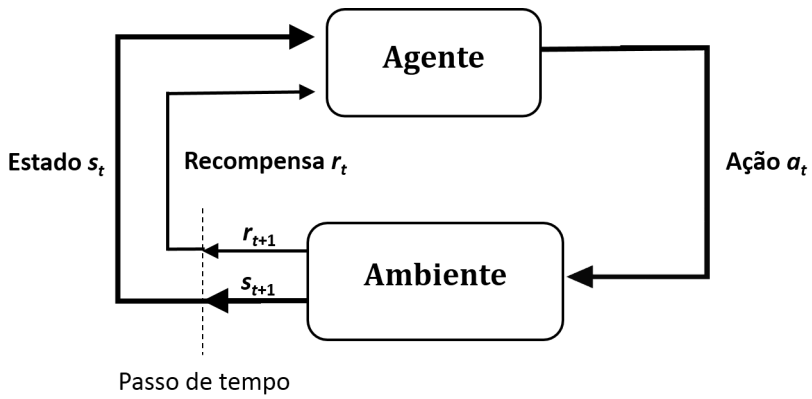

Figura 2.1: Processo de interação agente-ambiente na aprendizagem por reforço. (SUTTON; BARTO, 1998)

De acordo com a figura, a cada instante de tempo o agente percebe o ambiente e, com base nessa informação, executa uma ação. Esta ação produz dois resultados: o agente recebe uma recompensa imediata e o processo evolui para um novo estado, de acordo com uma determinada distribuição de probabilidades. No passo de tempo seguinte, o ambiente encontra-se alterado e

o agente percebe um novo estado com um outro conjunto de ações disponíveis para realizar sua escolha.

Uma das principais características da AR é o fato de não necessitar de modelos do ambiente, ou seja, sua adaptação é automática a ambientes desconhecidos e dinâmicos. Assim, o aprendizado por reforço é indicado quando não se conhece *a priori* o ambiente, ou as informações sobre as situações a serem enfrentadas pelo agente não estão disponíveis ou são imprecisas. O agente interage diretamente com o ambiente a fim de executar uma política ótima de ações que o leve a alcançar seus objetivos. Portanto, a técnica consiste em mapear estados em ações de modo a maximizar o valor numérico de retorno, sem que o agente necessite conhecer as ações que deve tomar, mas descobrir as que o levam a obter maiores valores numéricos.

Outra característica marcante da AR refere-se ao dilema da explotação versus exploração do aprendizado. O agente deve ser capaz de decidir de forma autônoma quando irá obter um novo aprendizado ou utilizar o conhecimento adquirido até o momento. A dificuldade está na tarefa de examinar uma série de possibilidades, favorecendo progressivamente aquelas que fornecem melhores resultados, mas também agir de forma a explorar ações ainda não selecionadas.

Além de agente e ambiente, pode-se considerar quatro elementos básicos de AR (SUTTON; BARTO, 1998), a saber:

- **Política:** uma política define o comportamento do agente aprendiz, isto é, como ele deve escolher suas ações em um dado instante de tempo. Representa o núcleo de um agente de AR no sentido que ela sozinha é suficiente para determinar o comportamento. Seja S o conjunto de estados do ambiente e $A(s)$ o conjunto de ações disponíveis para um estado $s \in S$. Uma política π é uma função de mapeamento de estados para ações:

$$a = \pi(s),\ \forall a \in A(s), s \in S \tag{2.20}$$

  ou seja, dado um estado s a política determina qual é a ação a que o agente deverá tomar. Geralmente, as políticas podem ser estocásticas.

- **Função de recompensa:** é quem define o objetivo em um problema de AR. Ela associa cada estado (ou par estado-ação) a um sinal numérico (uma *recompensa*), indicando a atratividade do estado. Assim, o agente tem como objetivo a maximização da quantidade

total de reforços recebidos ao longo da execução, denominado de retorno acumulado. Para que o valor do retorno não se torne muito grande um fator de desconto $\gamma\,(0 \leq \gamma \leq 1)$ é introduzido para reduzir gradativamente os valores futuros.

- **Função valor:** é obtida utilizando os reforços atual e futuro. Enquanto a função de recompensa atribui um sinal imediato às boas escolhas, a função valor indica uma estimativa do ganho total que pode ser acumulado pelo agente durante a aprendizagem, partindo de um estado s, e considerando os estados que o sucedem. Por exemplo, um estado pode sempre produzir baixas recompensas imediatas, mas ainda assim ter um alto *valor*, pois ele é regularmente frequentado por outros estados que produzem altas recompensas. Quando a função valor considera apenas o estado s, ela é denominada função valor-estado, e denotada por $V(s)$:

$$V(s) = E\{r_{t+1} + r_{t+2} + ... + r_{t+n} | s_t = s\} \tag{2.21}$$

onde E é o valor total esperado dos retornos acumulados pelo agente seguindo uma política π, a partir de um estado $s_t = s$, no instante t. No caso em que a função valor considera as ações possíveis para um dado estado, esta é denominada função valor estado-ação e é denotada por $Q(s,a)$, como segue:

$$Q(s,a) = E\{r_{t+1} + r_{t+2} + ... + r_{t+n} | s_t = s, a_t = a\} \tag{2.22}$$

O que a difere da equação 2.21 é o fato de que o ganho esperado depende da ação escolhida no instante t.

- **Modelo do ambiente:** o modelo representa o comportamento do ambiente, ou seja, dados o estado e a ação, o modelo pode antecipar o próximo estado e a próxima recompensa de acordo com a probabilidade de transição.

O domínio de um problema (ou tarefa) de AR deve ser modelado como um processo de decisão de Markov. Isso significa que os algoritmos de AR foram desenvolvidos para serem aplicados a esse tipo de processo. A próxima seção explana o formalismo em torno dos processos de decisão Markovianos.

2.4.1 Processos de decisão de Markov

Um Processo de Decisão de Markov – PDM (*Markov Decision Process – MDP*) é um processo onde as transições entre os estados que o compõem são probabilísticas, onde o estado atual do processo pode ser determinado e onde por meio de ações é possível interferir no processo periodicamente (sendo assim chamados de processos "de decisão"). Um PDM é um processo de decisão sequencial que não considera o histórico de suas decisões.

Formalmente, um processo de decisão sequencial, segundo (PUTERMAN, 2005), é definido como uma quíntupla $M = \{T, S, A, R, P\}$, onde:

- T é um conjunto discreto de instantes de tempo em que as decisões são tomadas, denominados instantes de decisão.
- S é um conjunto de estados do ambiente em que o processo pode estar. Os estados são as percepções do agente sobre o ambiente.
- A corresponde a um conjunto de ações possíveis de serem executadas em diferentes instantes de decisão. $A(s)$ são as ações disponíveis quando o processo encontra-se no estado s, de forma que $A = \bigcup_{A(s)}, \forall s \in S$ é o resultado da união de todas as ações disponíveis ao agente em qualquer estado.
- $R : S \times A \rightarrow \mathbb{R}$ representa uma função de retorno que atribui uma recompensa $r_t(s,a)$ como consequência pelo agente ter escolhido a ação $a \in A(s)$ estando no estado $s \in S$, no instante de decisão $t \in T$.
- $P : S \times A \times S \rightarrow [0,1]$ é uma função de probabilidades de transição, de tal modo que a função $p_t(s'|s,a)$ corresponde à probabilidade de o processo passar para um estado $s' \in S$ no instante seguinte, estando no estado s no instante t e o agente decidiu executar a ação $a \in A(s)$.

Em um PDM, o conjunto de ações possíveis, os retornos e as probabilidades de transição dependem somente do estado e da ação atual do sistema, e não de como o processo chegou a tal estado. Assim, a probabilidade de um estado s passar a um estado s', dado um par estado-ação (s, a), é dada por:

$$P^a_{s,s'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \tag{2.23}$$

onde Pr é o operador de probabilidade e representa a probabilidade do estado s_{t+1} ser s', sempre que a ação a_t for a e o estado s_t for s no instante de decisão t. A equação 2.23 expressa a propriedade de Markov, e é o que define se um processo pode ser considerado markoviano.

Analogamente, o valor do retorno esperado, dados o estado e a ação atuais e o estado seguinte s', será:

$$R^a_{s,s'} = E\{r_{t+1}|s_t = s, a_t = a, s_{t+1} = s'\} \tag{2.24}$$

onde E corresponde ao valor esperado do retorno r_{t+1}, sempre que o estado s_t, no instante t, passa a ser s' no instante $t+1$ escolhendo a ação a_t.

2.4.2 Algoritmo Q-learning

O Q-learning é um algoritmo baseado em *Diferença Temporal (DT)* que dispensa política (*off-police method*). Foi proposto por Watkins (1989) em sua tese de doutorado. O desenvolvimento desse algoritmo representa um dos maiores avanços na área de AR. Um método de DT não exige um modelo exato do ambiente, permite ser incremental e busca estimar valores de utilidade para cada estado do ambiente. Apresenta a vantagem de atualizar as estimativas da função valor a partir de outras estimativas já aprendidas, sem a necessidade de alcançar o estado final de um episódio[2] antes da atualização.

O Q-learning é capaz de aprender diretamente a partir da experiência, dispensando a necessidade de uma modelagem completa do ambiente para a determinação de uma política ótima de atuação. Assim, sua convergência para valores ótimos de Q independe da política utilizada.

Seja s_t o estado corrente, a_t a ação realizada no estado s_t, r_t o reforço imediato obtido após executar a ação a_t em s_t, s_{t+1} o estado seguinte, $maxQ(s_{t+1}, a)$ a máxima função valor (que encontra e seleciona a ação do estado seguinte que a maximize), e $\gamma \in [0,1]$ e $\alpha_q \in [0,1]$, respectivamente, o fator de desconto e o coeficiente de aprendizagem[3]. Para atualizar a matriz dos Q-valores, o Q-learning utiliza a seguinte expressão:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_q[r_t + \gamma maxQ(s_{t+1}, a) - Q(s_t, a_t)] \tag{2.25}$$

Observe que a função valor estado-ação $Q(s_t, a_t)$ é atualizada a partir do seu valor atual. O

[2]Um episódio é um intervalo de tempo em que o agente percorre uma sequência de estados até atingir um estado final.

[3]Usa-se o índice q para diferenciar do parâmetro α utilizado no GRASP.

fato de selecionar a ação que maximize a função valor no estado seguinte facilita a busca da função valor-ação ótima.

Como citado anteriormente, o dilema de decisão entre exploração e explotação também está presente na AR. No entanto, neste caso, o agente deve optar por aprender ou usar o conhecimento já adquirido. O algoritmo Q-learning comumente utiliza uma política de decisão ε-gulosa, se comportando como um algoritmo guloso-aleatório, para decidir entre intensificar as buscas baseado na melhor informação que o agente dispõe no momento ou explorar novas regiões de forma a obter novas informações do problema que possam ser úteis no futuro (exploração). Nessa política a "gula" e a aleatoriedade são equilibradas pelo valor do parâmetro de controle ε. A política atua de modo aleatório com probabilidade ε e de forma gulosa com probabilidade (1 - ε).

Na implementação do Q-learning, um agente aprende qual a melhor ação a ser escolhida a cada nova iteração, chamada de episódio. O Algoritmo 2.5 apresenta o pseudocódigo do Q-learning de forma simplificada com base em Watkins (1989).

Algoritmo 2.5 Estrutura genérica do algoritmo Q-learning.

```
1: procedimento Q-LEARNING(α_q, ε, γ, NumEpis)
2:     Inicializar R(s,a) {Matriz de recompensas}
3:     Inicializar Q(s,a) {Matriz dos Q-valores inicialmente zerada}
4:     repita
5:         Inicializar s {Estado inicial}
6:         repita
7:             Selecionar uma ação a de acordo com a política apropriada
8:             Receber a recompensa r(s,a) e observar o próximo estado s'
9:             Atualizar Q(s,a) de acordo com a equação 2.25
10:            s ← s'
11:        até Encontrar um estado final
12:    até NumEpis
13:    retorne Q(s,a)
14: fim procedimento
```

Este trabalho utiliza o algoritmo Q-learning como estratégia de aprendizagem dos melhores valores do parâmetro α a serem utilizados na fase de construção da metaheurística GRASP Reativo. Este método será detalhado no próximo capítulo.

2.5 CONCLUSÕES

Neste capítulo, foi apresentada uma fundamentação teórica necessária da metaheurística GRASP Reativo, dos Processos de Decisão de Markov e Aprendizagem por Reforço, com ênfase para o algoritmo Q-learning, visando uma melhor compreensão da versão proposta. Também objetivou-se dotar o leitor com uma visão geral sobre os modelos de PFL, especialmente o PPC que é o modelo utilizado nesse trabalho, e também apresentar a questão da criminalidade como um problema de Segurança Pública digno de atenção.

Não teve-se interesse em apresentar modelos para problemas multiobjetivo ou para as versões capacitadas, bem como não buscou-se um estudo aprofundado sobre os temas aqui abordados.

CAPÍTULO 3

ALGORITMO HÍBRIDO PROPOSTO

Este capítulo descreve os detalhes relativos à implementação do método proposto bem como a modelagem do problema de teste para as metaheurísticas implementadas.

Como mencionado no Capítulo 2, a metaheurística GRASP Reativo utiliza a variação de valores do parâmetro α na fase de construção para permitir uma exploração mais ampla do espaço de busca, possibilitando que a solução ótima global eventualmente seja encontrada. O valor do α a ser utilizado em cada iteração é selecionado de forma aleatória a partir de uma distribuição de probabilidades. O método híbrido apresentado neste capítulo utiliza o algoritmo Q-learning como mecanismo reativo para o GRASP.

3.1 MODELAGEM DO PPC PARA O GRASP REATIVO

O problema consiste em determinar a localização ótima para instalação de bases de polícia na cidade de Mossoró–RN. Para isso, é levado em consideração o histórico da localização das ocorrências criminais num determinado período de tempo. Este problema será modelado como um problema de localização dos p-centros (PPC). O objetivo é reduzir a distância entre as facilidades alocadas e os crimes mais distantes.

O problema é modelado como um grafo ponderado G(V, E, W), onde:

- V é o conjunto de vértices, cuja cardinalidade é n. Os vértices representarão tanto os usuários quanto as facilidades, sendo que n é o número de usuários que deverão ser atendidos;
- E é o conjunto de arestas conectando os vértices. Para o PPC, existe aresta ligando todos os vértices, de modo que o grafo é completo;
- W é o conjunto de pesos associados às arestas, e representa o conjunto de distâncias entre os vértices, de forma que d_{ij} é a distância entre os vértices i e j.

O PPC consiste em localizar p facilidades (centros) e associar cada usuário à facilidade mais próxima a este, de forma que a máxima distância entre um usuário e sua facilidade associada

seja a menor possível. Cada usuário é associado ao centro mais próximo a ele. O termo "centros" é utilizado para denotar bases policiais, bem como o termo "usuários" denota os crimes.

A formulação matemática garante o atendimento às seguintes restrições:

- Todo usuário deve está associado a um centro;
- Um usuário não pode está associado a mais de um centro;
- Um usuário não pode ser atendido por outro que não seja um centro.

Problemas que envolvem emergência precisam garantir que todos os usuários serão atendidos. Nesses problemas, a maior distância de atendimento representa uma maior garantia. É importante que essa distância seja a menor possível.

O PPC tende a encontrar soluções onde as distâncias entre os usuários e os centros são mais igualitárias e as menores possíveis, o que torna sua modelagem adequada a problemas que envolvem serviços de emergência.

A instalação de bases policiais em pontos estratégicos, de modo a realizar uma maior cobertura das zonas quentes (*hotspots*), possibilita reduzir o tempo de resposta às solicitações policiais e coibir a criminalidade. Além disso, contribui para a redução do custo operacional.

Representação da solução e estrutura de vizinhança

Uma solução do PPC obtida pela metaheurística GRASP Reativo é um vetor contendo os p centros alocados. A função objetivo é a distância do usuário mais distante do seu centro. Ao longo das iterações da metaheurística o objetivo é minimizar essa distância.

A estrutura de vizinhança é explorada neste trabalho por meio do movimento 2-*Opt*. Um movimento consiste em retirar um centro da solução e adicionar um outro que não fazia parte.

Fase Construtiva

A fase construtiva utiliza uma LRC baseada na qualidade dos elementos. A lista de candidatos (LC) contém todos os possíveis candidatos a centros, ou seja, todos os usuários que não foram selecionados como centros. No início da fase construtiva, um usuário é escolhido aleatoriamente para compor a solução parcial. Os demais serão escolhidos a partir da LRC.

Inicialmente, todos os usuários estarão associados a esse único centro. A cada iteração da fase construtiva, um centro é adicionado à solução parcial e os usuários são reassociados às facilidades selecionadas. Dessa forma, para p centros, o procedimento de reassociação dos usurários às facilidades deverá ser executado p vezes. Logo, a cada passo, a LC é readaptada e retorna novos c^{min} e c^{max} (menor e maior custo incremental, respectivamente), uma vez que os custos incrementais (distâncias aos centros associados) foram recalculados. A fase termina quando os p centros forem alocados.

Fase de Busca Local

Dada uma solução S, a busca local determina, para cada facilidade $f \notin S$, qual facilidade $g \in S$ dá a maior contribuição para a solução se f e g forem trocadas. O procedimento continua até encontrar um ótimo local.

A busca analisa todas as possíveis trocas e seleciona a que mais contribui com o processo de otimização. Ao encontrar a solução que mais apresenta melhoria para a solução atual, a solução corrente passa a ser a recentemente encontrada. O processo continua até que não seja possível realizar uma troca que reduza o valor da função objetivo da solução atual. A busca é então finalizada e a solução atual é retornada.

3.2 O MÉTODO GR-LEARNING HÍBRIDO

Um problema de AR precisa estar modelado matematicamente como um PDM. Antes de apresentar o método híbrido, é feita a modelagem do problema, bem como são descritos os detalhes da política de seleção de ações utilizada na implementação do algoritmo Q-learning.

3.3 MODELAGEM MATEMÁTICA

O problema de aprendizagem a ser resolvido pelo agente consiste em aprender e fornecer o melhor parâmetro α a ser utilizado na fase construtiva do GRASP Reativo. Seja m um número fixo de valores α permitidos e previamente determinados e seja $\Psi = \{\alpha_1, \alpha_2, \alpha_3, ..., \alpha_m\}$ um conjunto discreto contendo esses valores. O processo de aprendizagem do agente ocorrerá sobre um ambiente com uma matriz $Q_{m \times m}$. O problema pode ser modelado como um PDM $M = \{T, S, A, R, P\}$ da seguinte forma:

- $T = \{1,2,3,...,m\}$ é o conjunto de instantes de decisão e m corresponde à cardinalidade de Ψ.

- $S = \{s_1, s_2, ..., s_m\}$ é o conjunto de estados, onde cada estado s_i representa um $\alpha_i \in \Psi, i = 1,2,...m$.

- O conjunto de ações A é a união dos conjuntos de ações disponíveis a cada estado s. Ou seja, $A = \{A(s_1) \cup A(s_2) \cup ... \cup (s_m)\}$. Uma ação consiste em o processo passar de um estado atual $(s_i), \forall i \in \{1,2,...,m\}$ para outro (s_{i+1}) se houver melhoria na solução gerada utilizando-se o α representado pelo estado. Uma restrição imposta é o fato de que quando o processo está em um estado s, ele não poderá escolher uma ação que o faça permanecer no mesmo estado. Por exemplo, seja $S = \{s_1, s_2, s_3, s_4, s_5\}$, $A = \{A(s_1) = \{s_2, s_3, s_4, s_5\} \cup A(s_2) = \{s_1, s_3, s_4, s_5\} \cup A(s_3) = \{s_1, s_2, s_4, s_5\} \cup A(s_4) = \{s_1, s_2, s_3, s_5\} \cup A(s_5) = \{s_1, s_2, s_3, s_4\}\}$.

- R é a função de recompensa $r(s,a), \forall a \in A(s)$ e $s \in S$. O retorno atribui uma recompensa para as melhores ações. Para o problema em questão, a recompensa é definida como a relação entre a melhor solução x^* encontrada até o momento e a diferença entre esta e a solução gerada no estado alcançado. Ou seja:

$$R = \frac{f(x^*)}{f(x) - f(x^*)} \tag{3.1}$$

  onde $f(x^*)$ é o custo da melhor solução alcançada até a iteração atual e $f(x)$ é o valor da função de avaliação da solução gerada pela fase de construção utilizando o α contido no estado s_{i+1} alcançado pela ação a. Esta equação assume que quanto mais próxima uma solução estiver da ótima, maior será a recompensa atribuída pela ação. Há a possibilidade de $f(x^*)$ e $f(x)$ serem iguais, gerando uma indeterminação. Neste caso, faz-se $R = f(x^*)/2$.

- P é a função de probabilidades de transição $p(s'|s,a)$, que corresponde à probabilidade de se alcançar o estado s' estando no estado s e escolhendo a ação a. A natureza do problema permite que a modelagem assuma que $p(s'|s,a) = 1$, pois sempre que uma ação for escolhida a partir de um estado, o processo irá para o estado selecionado pela ação.

Uma política ótima deverá estabelecer qual a sequência de mudanças dos valores α, de modo que a soma dos retornos seja a melhor possível.

3.3.1 Política de ações do Algoritmo Q-learning

O dilema de explorar ou explotar em um problema de AR é determinado pelo agente através da política de seleção de ações. Ao longo do tempo, o Q-learning deverá convergir para uma política ótima de ações. Assim, é importante estabelecer uma política mista que pondere entre essas duas etapas.

A política de ações utilizada neste trabalho obedecerá a estratégia a seguir. Em todos os estados, todas as ações estarão disponíveis, com exceção da ação de permanecer no estado atual. Dado um estado corrente s com um valor α associado e uma função $f(x)$ que retorna o custo da solução gerada aplicando-se esse α, a escolha da próxima ação ocorrerá da seguinte forma:

1. Seleciona-se uma ação a aleatoriamente a partir de $A(s)$ e aplica-se à fase de construção do GRASP o α associado ao estado retornado por essa ação;

2. Se a solução construída apresentar o custo menor que $f(x)$, o estado retornado pela ação selecionada será o próximo estado para o qual o processo seguirá;

3. Caso contrário, retira-se a da lista de ações disponíveis e retorna-se ao passo 1.

3.3.2 Algoritmo GR-Learning implementado

O método aqui proposto mantém todas as vantagens do GRASP Reativo, mas permite um aprendizado mais eficiente de quais valores do parâmetro α favorecem a geração de soluções suficientemente diferenciadas e de boa qualidade, que são essenciais para o sucesso do procedimento de busca local. Estes fatores permitirão melhorar o desempenho da metaheurística GRASP Reativo, no que diz respeito à localização do ótimo global e tempo de execução. O método híbrido proposto será denominado GR-Learning.

No GR-Learning, o algoritmo Q-learning substituirá o mecanismo reativo do GRASP baseado em probabilidades. A cada iteração do algoritmo, um valor α é selecionado com base nas informações contidas na matriz dos Q-valores retornada pelo Q-learning. A seleção a partir dessa matriz permite que experiências bem sucedidas sejam repetidas em iterações futuras.

A estratégia estabelecida utiliza o seguinte procedimento em 4 etapas:

1. A matriz dos Q-valores é inicializada com valor zero para todos os itens;

2. Em seguida, um índice da tabela Q é sorteado aleatoriamente para iniciar o processo de atualização da mesma. Este índice passa a ser o estado s_0 para o Q-learning;

3. A partir do estado s_0, utilizando a política descrita na Seção 3.3.1, o próximo estado s_1 é obtido;

4. Em posse dos estados s_0 e s_1, o processo iterativo que atualiza a matriz Q é iniciado utilizando a equação (2.25). Após um determinado número de episódios, obtém-se a matriz Q da qual será escolhido o α utilizado na fase de construção. A escolha é feita da seguinte forma (baseada em (LIMA JÚNIOR, 2009)):

    - Seleciona-se aleatoriamente uma linha l da matriz Q;
    - A coluna que contiver o maior valor em l será o índice do α a ser utilizado na fase construtiva;

Após cada bloco de iterações y, a matriz Q é atualizada pela execução do algoritmo Q-learning, permitindo que ao longo do processo de busca a qualidade das informações coletadas seja melhorada. O Algoritmo 3.1 apresenta o pseudocódigo do método GR-Learning possibilitando uma melhor visualização do método implementado. Os parâmetros de entrada são o coeficiente de aprendizagem α_q, o fator de desconto γ e o número de episódios $NumEpis$.

No Algoritmo 3.1, é possível observar que as alterações significativas ocorrem nas linhas 6 e 13. Na linha 6, a função *AprenderAlfa* recebe a matriz dos Q-valores e seleciona um valor α a partir desta. Na primeira execução esta matriz está zerada, sendo a escolha do valor α totalmente aleatória. Na linha 13, o algoritmo Q-learning é executado por meio da chamada da função *Qlearning*, que recebe o vetor de valores α pré-determinados Ψ, a melhor solução encontrada até o momento $f(x^*)$, necessária para o cálculo da função de recompensa, e os mesmos parâmetros do algoritmo GR-Learning. O procedimento retorna a matriz dos Q-valores atualizada para ser utilizada no próximo bloco de iterações. Os procedimentos de construção e busca local não são afetados.

Algoritmo 3.1 Pseudocódigo do algoritmo GR-Learning.

```
1: procedimento GR-LEARNING(α_q, γ, NumEpis)
2:     f(x*) ← +∞
3:     it ← 1
4:     Definir Ψ = {α_1, α_2, ..., α_m}
5:     enquanto não CritérioParada faça
6:         α ← AprenderAlfa( Q )
7:         x_1 ← FaseConstrutiva(α)
8:         x_2 ← BuscaLocal(x_1)
9:         se f(x_2) < f(x*) então
10:            x* ← x_2
11:        fim se
12:        se it mod y = 0 então
13:            Q ← Qlearning(Ψ, f(x*), α_q, γ, NumEpis)
14:        fim se
15:    fim enquanto
16:    retorne x*
17: fim procedimento
```

3.4 CONCLUSÕES

Este capítulo descreveu a estratégia inteligente utilizada no método GR-Learning, de modo a prover o leitor dos detalhes inerentes à implementação deste algoritmo. Como o GR-Learning utiliza um algoritmo de AR, uma modelagem matemática do problema de aprendizagem como um PDM foi realizada. Além disso, foi feita a modelagem das heurísticas de construção e de busca local usadas no GRASP Reativo e na versão híbrida para o PPC.

O próximo capítulo traz os resultados dos experimentos computacionais obtidos com ambos os algoritmos e faz uma análise comparativa destes, a fim do apontar o método que apresenta melhor desempenho.

CAPÍTULO 4

RESULTADOS EXPERIMENTAIS

Neste capítulo, os resultados computacionais obtidos com a execução dos algoritmos GRASP Reativo e GR-Learning implementados são apresentados e comparados. Além disso, o processo de elaboração das instâncias é descrito na seção a seguir.

4.1 CRIAÇÃO DAS INSTÂNCIAS

Usando o software QGIS versão 2.0 (QGIS, 2014) para permitir o armazenamento e a manipulação dos dados georreferenciados, foi realizado um mapeamento da criminalidade na cidade de Mossoró – RN. Foi utilizado um arquivo vetorial representando o mapa das ruas de Mossoró. No mapa, foram inseridos 405 pontos onde se concentraram o maior número de ocorrências de crimes no período de 2010-2013. Os dados contendo os endereços dos crimes, com mais de 14 mil registros, foram fornecidos pelo Centro Integrado de Operações de Segurança Pública (CIOSP) de Mossoró. As informações pessoais das vítimas, como nome e CPF, foram mantidas em sigilo.

O processo de elaboração das instâncias seguiu as seguintes etapas:

- Uma filtragem sobre os dados foi feita com o intuito de determinar os endereços com maior concentração de crimes. Em seguida, recorreu-se a uma ordenação;
- Esses endereços foram então passados para o GQIS, gerando assim um conjunto de pontos. O mapeamento desses pontos foi realizado sobre um arquivo vetorial representando as ruas de Mossoró. A Figura 4.1 ilustra a inserção de pontos sobre a localização no mapa de Mossoró (os nomes dos logradouros foram ocultados).
- A latitude e a longitude de cada ponto foram então extraídas, gerando-se um arquivo de texto onde cada linha contém dois valores equivalentes à latitude e à longitude;
- A partir desse arquivo, gerou-se uma matriz contendo a distância Manhattan entre todos os pontos. Dados dois pontos $P_1 = (x_1, y_1)$ e $P_2 = (x_2, y_2)$, a distância Manhattan d_M entre

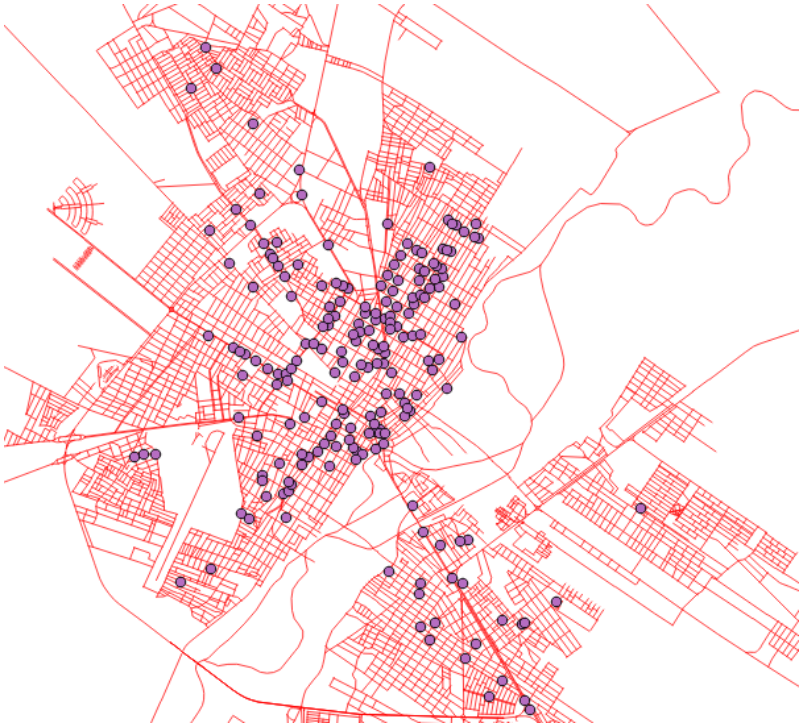

Figura 4.1: Geração da camada de pontos no QGIS durante processo de mapeamento.

esses pontos pode ser obtida pela seguinte fórmula matemática:

$$d_M = |x_1 - x_2| + |y_1 - y_2| \tag{4.1}$$

Antes de aplicar a fórmula, é necessário converter as latitudes e longitudes de graus para quilômetros. O raio da Terra é variável: nos polos é da ordem de 6357 km, enquanto que na linha do Equador é 6378 km. Considerando que a cidade de Mossoró localiza-se próximo à linha do Equador, assumiu-se o raio da Terra como sendo 6378 km. Dessa forma, uma volta na terra tem $(2 * \pi * 6378)/360$ km. Fazendo π = 3,141593, cada grau corresponderá a 111,317 km. Portanto, as latitudes e longitudes são convertidas simplesmente multiplicando o seu valor em graus por 111,317.

- Essa matriz é então usada como parâmetro de entrada para as metaheurísticas implementadas.

Ao fim do processo, foram geradas 8 instâncias, semelhantemente às instâncias geradas em (LORENA; SENNE, 2004), contendo 102, 202, 304 e 405 nodos, correspondendo aos usuários (pontos de ocorrências de crimes). O tamanho é dado pela relação (*usuários* $\times$ *centros*). Desse

modo, obteve-se as seguintes instâncias: MOSS1a (102x10), MOSS1b (102x15), MOSS2a (202x20), MOSS2b (202x30), MOSS3a (304x30), MOSS3b (304x40), MOSS4a (405x40) e MOSS4b (405x50).

4.2 RESULTADOS DOS EXPERIMENTOS COMPUTACIONAIS

Os testes foram realizados em uma máquina com processador Intel Xeon 3.40 Ghz, com 16 Gb de memória RAM, em um sistema operacional Windows 7, arquitetura de 64 bits.

Visto que os algoritmos são não determinísticos, os dados listados são a média aritmética de 30 execuções para cada instância. Os valores da função objetivo (F.O.) e tempo de processamento na íntegra estão disponíveis no Apêndice A. Os valores relativos a tempo são dados em segundos e os da função objetivo em metros.

Todos os experimentos usam o mesmo número de iterações e valores idênticos para todos os parâmetros ajustáveis. O parâmetro α do GRASP Reativo, responsável pelo controle entre gula e aleatoriedade na fase construtiva, é autoajustável e varia no intervalo [0, 1]. Em relação aos parâmetros do algoritmo Q-learning utilizado no GR-Learning, o coeficiente de aprendizagem, α_q, foi fixado em 0,9; o fator de desconto, γ, foi ajustado em 1; e o número de episódios foi definido como sendo metade do tamanho da instância (número de usuários do PPC), como listado na Tabela 4.1.

Tabela 4.1: Ajuste do número de episódios do Q-learning.

Instância	Num. Epis.
MOSS1a	50
MOSS1b	50
MOSS2a	100
MOSS2b	100
MOSS3a	150
MOSS3b	150
MOSS4a	200
MOSS4b	200

A Tabela 4.2 apresenta os resultados obtidos com os algoritmos executando apenas a fase construtiva. A última coluna mostra a diferença percentual entre os valores da função objetivo obtidos pelos métodos. É possível notar que a fase construtiva do GR-Learning obteve melhores soluções que o GRASP Reativo exigindo, no máximo, o mesmo tempo de execução. Na última

instância, houve uma melhoria de 24,9% na qualidade da solução inicial. Esta melhoria confirma a eficiência do método no processo de escolha do parâmetro α, permitindo a geração de soluções iniciais de boa qualidade e com alta taxa de diversidade. Isto influencia diretamente na etapa de busca local e consequentemente no resultado final. A Figura 4.2 ilustra estes resultados graficamente.

Tabela 4.2: Resultados da fase construtiva do GRASP Reativo e GR-Learning.

Instância	GRASP Reativo		GR-Learning		Dif.(%)
	F.O.	Tempo	F.O.	Tempo	
MOSS1a	2952,4	0,02	2538,5	0,02	14,0
MOSS1b	2450,5	0,02	2314,9	0,02	5,5
MOSS2a	2629,4	0,06	2415,6	0,05	8,1
MOSS2b	2446,9	0,26	2128,8	0,26	13,0
MOSS3a	2520,2	0,17	2353,2	0,17	6,6
MOSS3b	2398,5	0,29	2131,6	0,29	11,1
MOSS4a	2482,6	0,40	2148,9	0,40	13,4
MOSS4b	2304,0	0,59	1730,4	0,59	24,9

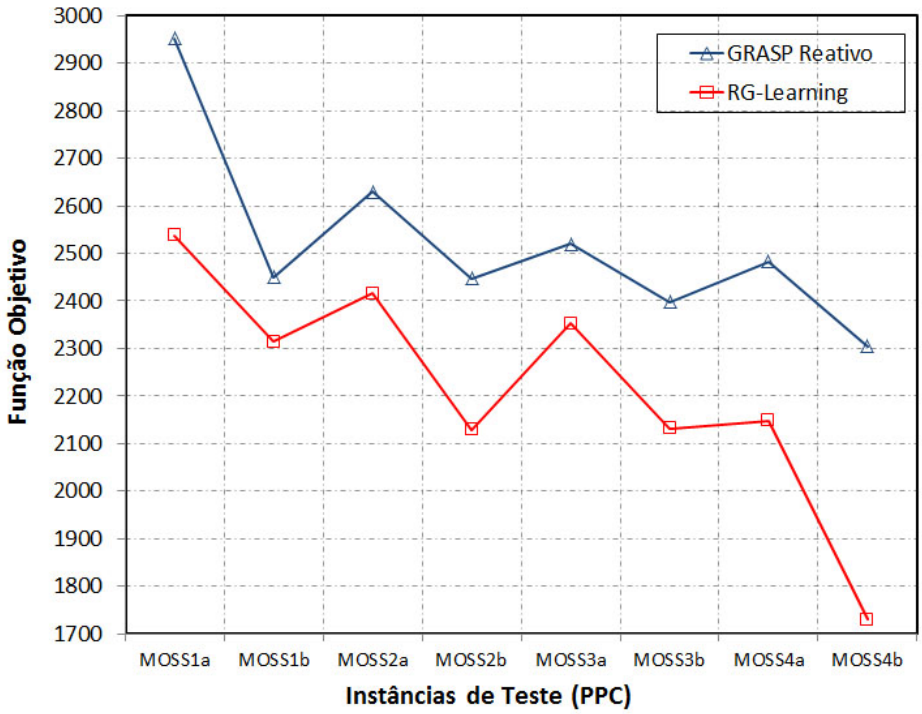

Figura 4.2: Comparação dos resultados da fase construtiva do GRASP Reativo e GR-Learning.

A Tabela 4.3 contém os resultados aos valores da função objetivo. A terceira e a quinta coluna apresentam o tempo exigido (em segundos) até que a melhor solução seja encontrada por cada algoritmo. A última coluna lista a diferença percentual entre esses tempos, representando

quão mais rápido o método é. A última linha da tabela apresenta a média das diferenças percentuais e mostra que o GR-Learning foi, em média, 21,9% mais rápido que o GRASP Reativo. A análise da tabela mostra que a estratégia proposta foi capaz de melhorar todos os resultados obtidos pelo GRASP Reativo em qualidade de solução e tempo necessário para alcançar a melhor solução. Os melhores resultados são justificados pelo fato de que iniciando a partir de soluções de alta qualidade a busca local é acelerada.

Tabela 4.3: Valores da função objetivo para o GRASP Reativo e GR-Learning.

Instância	GRASP Reativo		GR-Learning		Dif.(%)
	F.O.	Tempo	F.O.	Tempo	
MOSS1a	1383,5	1,8	1358,5	1,2	31,6
MOSS1b	1011,0	4,4	1009,5	3,0	32,0
MOSS2a	949,9	21,7	936,5	19,1	12,1
MOSS2b	689,5	51,6	688,3	39,9	22,7
MOSS3a	690,1	224,9	688,4	159,9	28,9
MOSS3b	592,5	295,0	591,9	228,4	22,6
MOSS4a	637,8	726,2	631,8	632,3	12,9
MOSS4b	539,0	1046,9	532,1	913,0	12,8
Média					**21,9**

Os resultados relativos ao tempo de execução são apresentados na Tabela 4.4. Nesta tabela, a quarta coluna contem a diferença, em porcentagem, entre os tempos de execução de ambos os métodos. Pode-se observar que, quando as instâncias são pequenas, o tempo de execução total não é melhorado. No entanto, todas as outras instâncias obtiveram melhoria considerável, onde no melhor caso, foi possível reduzir o tempo do GR-Learning em 35,6% (instância **MOSS3a**). Além disso, o método híbrido proposto reduziu em média o tempo de execução do GRASP Reativo em 14,9%. É possível notar ainda que, quanto maior é o número de centros em instâncias grandes, menor é a redução do tempo total.

A Figura 4.3 e a Figura 4.4 ilustram graficamente uma comparação entre as duas versões das metaheurísticas implementadas, considerando os valores de função objetivo e o tempo de processamento, respectivamente. Os dados apresentados nos gráficos foram submetidos a um processo de normalização, objetivando melhorar a visualização, e foram normalizados pela média dos valores obtidos por cada metaheurística, para cada uma das instância, de acordo com as equações a seguir.

$$VN_{ij} = V_{ij}/M_i \tag{4.2}$$

Tabela 4.4: Resultados dos tempos de processamento para o GRASP Reativo e GR-Learning.

Instância	GRASP Reativo	GR-Learning	Dif.(%)
MOSS1a	4,8	4,9	-1,6
MOSS1b	8,8	8,8	0,0
MOSS2a	53,4	44,8	16,1
MOSS2b	130,4	96,3	26,2
MOSS3a	496,4	319,5	35,6
MOSS3b	626,3	536,2	14,4
MOSS4a	1445,2	1182,5	18,2
MOSS4b	1975,0	1775,3	10,1
Média			**14,9**

onde

$$M_i = \left(\sum_{j=1}^{m} V_{ij} \right) / m \quad \forall i = 1, 2, ..., n \tag{4.3}$$

Na equação (4.2), considerando a instância i, VN_{ij} é o valor normalizado executando o algoritmo j, V_{ij} representa os valores tanto de função objetivo quanto de tempo de processamento executando o algoritmo j e M_i é a média dos valores. Em (4.3), m é o número de algoritmos testados e n é a quantidade de instâncias usadas no experimento.

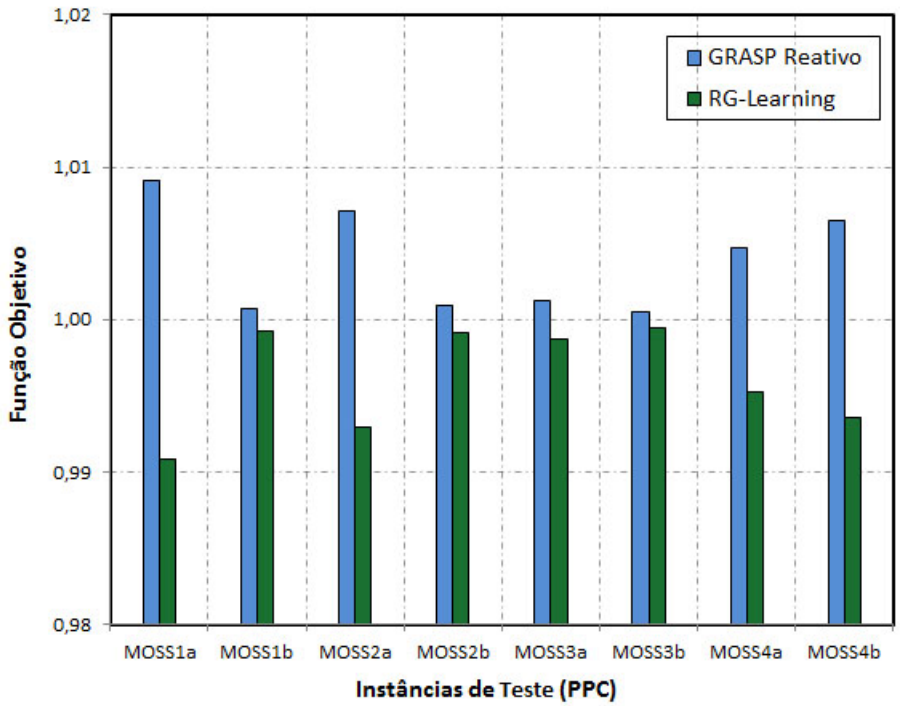

Figura 4.3: Comparativo dos resultados da função objetivo para o GRASP Reativo e GR-Learning.

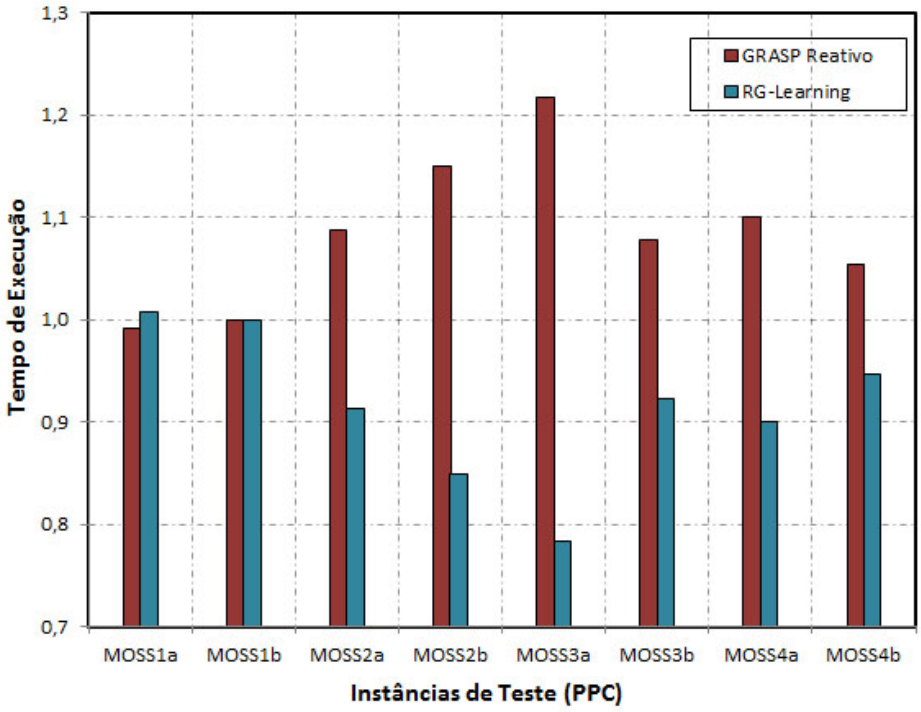

Figura 4.4: Comparação dos tempos de execução para o GRASP Reativo e GR-Learning.

4.3 ANÁLISE ESTATÍSTICA DOS RESULTADOS

Esta seção tem como objetivo validar os dados experimentais relativos aos valores da função objetivo por meio de uma análise estatística dos resultados. Isso é feito com a realização de um Teste de Hipótese Estatística.

4.3.1 Teste de Hipótese

O teste de hipótese é uma técnica para se realizar inferência estatística. Ou seja, dado um conjunto de dados amostrais, é possível inferir sobre toda a população. O objetivo dos testes de hipóteses é determinar se certas afirmações sobre uma população são suportadas pelos dados da amostra.

Neste trabalho, o teste tem o objetivo de verificar se a média das soluções obtidas pelo método híbrido é menor do que a média do GRASP Reativo, confirmando assim o melhor desempenho computacional do GR-Learning proposto. Visto que há a necessidade de tomar uma decisão, configura-se um problema de Teste de Hipótese.

O teste utilizado será o teste *t* de *Student* (recebe esse nome por utilizar a distribuição *t*

de *Student*[1]), ou simplesmente teste *t*. Este teste é o método mais utilizado para avaliação das diferenças entre as médias de dois grupos. É indicado para problemas cujas variâncias populacionais são desconhecidas e pode ser aplicado com sucesso mesmo quando as amostras são pequenas.

Serão comparadas as médias de duas distribuições normais, equivalentes a populações diferentes, sendo as amostras independentes. Assim, o teste foi elaborado sobre as variáveis aleatórias X_1 e X_2, que representam, respectivamente, os resultados dos experimentos com o GRASP Reativo e GR-Learning.

Para o teste, utilizou-se os dados amostrais obtidos com a instância **MOSS1b**, por apresentar menor variância em ambos os métodos, possibilitando generalizar o resultado do teste para as demais instâncias. Os valores das variáveis X_1 e X_2 são resultados de 30 execuções desta instância com ambos os algoritmos implementados, e podem ser consultados nas tabelas A.1 e A.3 constantes no Apêndice A.

Etapas do teste

1. **Hipóteses**

   Sejam μ_1 e μ_2 as médias da função objetivos da instância **MOSS1b**, obtidas pelas metaheurísticas GRASP Reativo e GR-Learning, respectivamente. Deseja-se testar se $\mu_1 > \mu_2$. Na hipótese nula H_0, geralmente coloca-se o contrário do que se quer provar. Assim, na definição de H_0 (4.4), assume-se que não há diferença (ou melhora) entre as médias. Ou seja:

   $$H_0 : \mu_1 = \mu_2 \tag{4.4}$$

   Uma hipótese alternativa, que contraria a hipótese nula H_0, é definida em (4.5) abaixo:

   $$H_1 : \mu_1 > \mu_2 \tag{4.5}$$

2. **Nível de significância do teste**

   Nível de significância é a probabilidade máxima de ocorrência de erro do tipo I, que consiste em rejeitar a hipótese nula, sendo ela verdadeira. Esse erro é considerado o tipo

[1] A distribuição *t* de *Student* foi introduzida em 1908 por William Gosset, que utilizou o pseudônimo de "Student" devido ao sigilo exigido no seu ambiente de trabalho.

mais grave e, portanto, o que deve ser controlado. Assim:

$$\alpha = P(\text{erro tipo I}) = P(\text{Rejeitar } H_0 | H_0 \text{ verdadeira}) \tag{4.6}$$

Para este teste, utilizou-se nível de significância α = 0,01.

3. **Estatística do teste**

   É a variável aleatória cujos valores calculados a partir da amostra permitem decidir que atitude tomar. As amostras deste teste contam apenas com 30 elementos e as variâncias populacionais são desconhecidas e não podem ser consideradas iguais. Assim, o procedimento de Welch (WELCH, 1947) estabelece que a variável estatística tem aproximadamente distribuição *t*, com valor calculado pela expressão:

$$t = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}} \tag{4.7}$$

   onde X_i, S_i^2 e n_i são, respectivamente, a média, a variância e o tamanho da amostra referentes ao i-ésimo grupo.

4. **Graus de liberdade da variável**

   O número de graus de liberdade v da variável t tem de ser estimado diretamente a partir dos dados, obedecendo à seguinte expressão:

$$v = \frac{\left(\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}\right)^2}{\frac{(S_1^2/n_1)^2}{n_1 - 1} + \frac{(S_2^2/n_2)^2}{n_2 - 1}} \tag{4.8}$$

   O valor obtido para v não é necessariamente um inteiro, assim deve-se arredondar para o inteiro imediatamente inferior.

5. **Cálculo da estatística do teste**

   Obteve-se as médias das variáveis X_1 e X_2 e suas respectivas variâncias:

   - X_1 = 1011,0;
   - X_2 = 1009,5;
   - S_1^2 = 3,75;

- S_2^2 = 1,16;
- $n_1 = n_2 = 30$.

Realizando-se a substituição dos valores das variáveis na equação 4.7, tem-se:

$$t = \frac{1011,0 - 1009,5}{\sqrt{\frac{3,75}{30} + \frac{1,16}{30}}}$$

$$t = 3,70$$

Em seguida, substituindo os valores na equação 4.8, tem-se:

$$v = \frac{\left(\frac{3,75}{30} + \frac{1,16}{30}\right)^2}{\frac{(3,75/30)^2}{30-1} + \frac{(1,16/30)^2}{30-1}}$$

$$v = 45,5932$$

$$v \cong 45$$

Em posse dos valores do nível de significância α = 0,01 e do grau de liberdade v = 36, realiza-se uma consulta ao valor tabelado t_{tab} na distribuição de *Student*, encontrando assim o valor de $t_{\alpha,v}$, que neste caso foi:

$$t_{0,01;45} = 2,690 \tag{4.9}$$

6. **Decisão sobre a hipótese**

A decisão deve ser tomada comparando-se o valor calculado t com o valor tabelado $t_{\alpha,v}$, de modo que:

$$\begin{cases} H_0 \text{ será rejeitada:} & \text{Se } t > t_{\alpha,v} \\ H_0 \text{ será aceita:} & \text{Caso contrário} \end{cases}$$

Comparando os valores, tem-se:

$$t > t_{0,01,45}$$

$$3,70 > 2,690$$

Logo, a hipótese H_0 deve ser rejeitada.

7. **Conclusão do teste**

   Já que a hipótese nula H_0 foi rejeitada, a hipótese alternativa H_1 foi aceita, significando que é possível afirmar com 99% de certeza que os resultados obtidos com o novo método GR-Learning são melhores, em média, que os resultados conseguidos com a metaheurística GRASP Reativo.

4.4 CONCLUSÕES

Este capítulo objetivou demonstrar a eficiência do algoritmo GR-Learning proposto neste trabalho quando aplicado ao PPC. Por meio dos resultados computacionais aqui apresentados, pôde-se comprovar que a abordagem híbrida da metaheurística GRASP Reativo utilizando o algoritmo Q-learning é capaz de obter melhor desempenho na obtenção de valores satisfatórios para a função objetivo quando comparada à versão que utiliza distribuição de probabilidades. Além disso, o algoritmo proposto apresentou uma redução significativa no tempo de execução.

Para comprovar a eficiência da implementação da nova metaheurística proposta e para validação dos resultados, uma análise estatística foi realizada, onde através de um Teste de Hipótese Estatística foi possível confirmar que o método híbrido tem melhor desempenho computacional que o GRASP Reativo no quesito valor da função objetivo.

O teste de hipóteses foi realizado utilizando os resultados obtidos com a instância **MOSS1b** e teve como parâmetro de interesse o valor da média da função objetivo para o problema dos *p*-centros. Esta instância foi selecionada por que apresentou a menor variação de valores nos resultados encontrados. Uma análise estatística considerando o critério tempo de processamento não se fez necessária, uma vez que o tempo médio exigido para se alcançar a melhor solução foi tabelado para cada um dos algoritmos. Essa métrica utilizada é semelhante à análise *time-to-target*, que estabelece um valor alvo e registra o instante em que o algoritmo alcança esse valor.

Adicionalmente, um conjunto de instâncias para PLF foi criado e está disponível à comunidade científica no endereço eletrônico (http://di.uern.br/loia/instancias).

CAPÍTULO 5

CONCLUSÕES

Neste trabalho, foi mostrado que o algoritmo GR-Learning apresentou desempenho superior em termos de função objetivo e tempo de processamento quando comparado ao GRASP Reativo, que é uma versão melhorada da metaheurística GRASP tradicional.

O diferencial do algoritmo GRASP Reativo em relação ao GR-Learning é apenas no processo de escolha do parâmetro α. O primeiro usa um algoritmo parcialmente guloso, que tem complexidade $O(n^2)$, enquanto que o segundo usa o algoritmo Q-learning, cuja complexidade é $O(ep * n)$, onde ep é o número de episódios e n é o tamanho do conjunto Ψ. Consequentemente, para instâncias grandes n permanece constante e ep cresce proporcionalmente ao tamanho da instância, uma vez que o espaço de busca de soluções também aumenta. Dessa forma, a atualização dos Q-valores durante a execução do GR-Learning não compromete o tempo de processamento nem processo de aprendizagem do Q-learning.

A pesquisa em metaheurística é motivada pela busca por métodos que melhorem a qualidade das soluções ou que encontrem as mesmas soluções exigindo um tempo de execução menor. Apesar de não haver redução significativa na média das soluções obtidas, o novo método foi capaz de obter soluções tão boas quanto o GRASP Reativo exigindo menor tempo de processamento, representando uma melhor eficiência.

Dotado de uma memória adaptativa baseada em aprendizado, o novo método foi capaz de utilizar o conhecimento adquirido em iterações passadas para sempre selecionar valores do parâmetro α que levem à geração de boas soluções. Este mecanismo de memória adaptativa é implementado através do uso das informações contidas na matriz dos Q-valores gerada pelo algoritmo Q-learning.

A comparação do desempenho da fase construtiva de cada algoritmo (Seção 4.2) mostrou que o GR-Learning apresentou melhoria substancial na qualidade das soluções iniciais. Partindo-se de soluções melhores, a busca local consegue ser mais eficaz no processo de melhoramento destas. Isto justifica as melhores soluções encontradas pelo GR-Learning. Além disso, a busca local também é acelerada, pois incorre em menos esforço computacional neces-

sário para convergir para um ótimo local. Esta é a principal razão para o comportamento mais rápido do GR-Learning.

Embora os valores na coluna *Tempo* na Tabela 4.3 mostram que o método híbrido pode encontrar soluções mais rapidamente que o GRASP Reativo, uma análise estatística para o critério tempo de processamento ainda pode ser realizada. No entanto, esta atividade é uma perspectiva de trabalho futuro.

A cidade de Mossoró–RN tem apresentado altos e cada vez mais crescentes índices de criminalidade. Com o intuito de contribuir para a redução desses índices, um mapeamento dos pontos com maior ocorrência de crimes foi realizado. Obviamente um mapeamento da criminalidade não se resume apenas em verificar a quantidade de crimes em determinadas áreas, no entanto, este trabalho não objetivou permitir a visualização da sistemática e origem dos crimes. A partir desse mapeamento, foi criado um conjunto de instâncias para PLF. As instâncias bem como os arquivos *shape* contendo as camadas de pontos e a camada de ruas estão disponíveis no endereço (http://di.uern.br/loia/instancias).

5.1 CONTRIBUIÇÕES DA DISSERTAÇÃO

O algoritmo GR-Learning supriu a carência de uma memória adaptativa presente no GRASP Reativo, no sentido de que a matriz dos Q-valores é atualizada incorporando-se a experiência adquirida pelo agente ao longo da sua aprendizagem. Assim, é possível utilizar informações das iterações passadas em decisões futuras, aumentando a garantia de boas decisões e reduzindo as chances de escolhas feitas ao acaso.

Assim, foi possível alcançar as seguintes contribuições:

- Desenvolvimento de uma versão híbrida da metaheurística GRASP Reativo que apresentou melhor desempenho tanto em qualidade da solução quanto em tempo de execução;
- Obtenção de um algoritmo híbrido utilizando AR que pode ser aplicado a problemas de otimização sem a necessidade de modelá-los como PDM;
- Protótipo de uma ferramenta que pode ser aplicada à Segurança Pública para auxílio à tomada de decisão, uma vez que pode ser utilizada na determinação da localização de pontos estratégicos para instalação de bases de polícia;

- Modelagem de um problema real como um PPC;
- Mapeamento da criminalidade no período de 2010-2013 na cidade de Mossoró–RN. Com isso, é possível visualizar espacialmente os locais onde a incidência de crimes é elevada e, a partir disso, estabelecer estratégias por meio de um planejamento;
- Criação e disponibilização de um conjunto de instâncias reais para PLF.

5.2 TRABALHOS FUTUROS

Como possíveis perspectivas de trabalhos futuros decorrentes desta pesquisa, têm-se:

- Aplicação da versão híbrida a outros problemas de Otimização Combinatória, bem como a instâncias de grande porte do PPC, de modo a verificar o comportamento e avaliar o desempenho do método proposto;
- Avaliação da robustez do método quando comparado com outras versões híbridas do GRASP Reativo com outras metaheurísticas, com Path-relinking e com Mineração de Dados;
- Criação de instâncias maiores, objetivando uma maior cobertura do mapeamento da criminalidade, bem como uma atualização dos pontos com maior incidência de crimes graves;
- Desenvolvimento de uma ferramenta para localização de bases policiais integrada com um WebSIG. Algumas funções primordiais são: Inserção de pontos candidatos, Otimização de bases, Geração de Relatórios;
- Realização de uma análise estatística de tempo de modo a provar que a abordagem híbrida contribui para a maior rapidez na convergência.

REFERÊNCIAS

ABDINNOUR-HELM, S.; HADLEY, S. W. Tabu search based heuristics for multi-floor facility layout. *International Journal of Production Research*, v. 38, n. 2, p. 365–383, 2000. Disponível em: <http://www.tandfonline.com/doi/abs/10.1080/002075400189464>.

AYDIN, M. E.; ÖZTEMEL, E. Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, Elsevier, v. 33, n. 2, p. 169–178, 2000.

BARTO, A.; CRITES, R. H. Improving elevator performance using reinforcement learning. *Advances in neural information processing systems*, v. 8, p. 1017–1023, 1996.

BELTRÁN, J. D.; CALDERÓN, J. E.; CABRERA, R. J.; PÉREZ, J. A. M.; MORENO-VEGA, J. M. GRASP-VNS hybrid for the strip packing problem. *Hybrid metaheuristics*, v. 2004, p. 79–90, 2004.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surveys*, n. 35, p. 268–308, 2003.

CHANG, H. S. An ant system based exploration-exploitation for reinforcement learning. *IEEE International Conference on Publication*, Systems, Man and Cybernetics, v. 4, p. 3805–3810, 2004.

CHURCH, R.; REVELLE, C. The maximal covering location problem. *Papers of the Regional Science Association*, Wiley Online Library, v. 32, n. 1, p. 101–118, 1974.

DELMAIRE, H.; DÍAS, J. A.; FERNÁNDEZ, E.; ORTEGA, M. Reactive GRASP and tabu search based heuristics for the single source capacitated plant location problem. *INFOR-Information Systems and Operational Research*, Ottawa: INFOR Journal, 1971, v. 37, n. 3, p. 194–225, 1999.

DORIGO, M. *Optimization, Learning and Natural Algorithms*. Tese (Doutorado) — Politecnico di Milano, Italy, 1992.

FEO, T. A.; RESENDE, M. G. C. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, v. 8, n. 2, p. 67–71, 1989.

FEO, T. A.; RESENDE, M. G. C. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, v. 6, p. 109–133, 1995.

FESTA, P.; RESENDE, M. G. C. An annotated bibliography of GRASP - part i: Algorithm. In: . [S.l.: s.n.], 2009. p. 1–24.

GAREY, M. R.; JOHNSON, D. S. Computers and intractability: a guide to the theory of NP-completeness. W. E. Freeman and company, 1979.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. *Computing Operations Research*, v. 13, n. 5, p. 533–549, 1986.

HAKIMI, S. L. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations research*, INFORMS, v. 12, n. 3, p. 450–459, 1964.

HAKIMI, S. L. Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research*, INFORMS, v. 13, n. 3, p. 462–475, 1965.

HART, J. P.; SHOGAN, A. W. Semi-greedy heuristics: An empirical study. *Operations Research Letters*, v. 6, p. 107–114, 1987.

HOLLAND, J. H. Adaptation in natural and artificial systems. University of Michigan Press, EUA, 1975.

JOHNSON, D. S.; MCGEOCH, L. A. The traveling salesman problem: a case study in local optimization. In: AARTS, E.; LENSTRA, J. K. (Ed.). *Local Search in Combinatorial Optimization*. [S.l.]: Princeton University Press, 2003.

KAMPKE, E. H. *Metaheurísticas para o problema de programação de tarefas em máquinas paralelas com tempos de preparação dependentes da sequência e de recursos*. Dissertação (Mestrado) — Universidade Federal de Viçosa, Viçosa, MG, 2010.

KIRKPATRICK, S.; GELLAT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science*, v. 220, n. 4598, p. 671–680, 1983.

LI, X.; SHEN, X.; JING, Y.; ZHANG, S. Simulated annealing-reinforcement learning algorithm for ABR traffic control of ATM networks. In: IEEE. *Conference on Decision and Control, 2007*. [S.l.], 2007. p. 5716 – 5721.

LI, X.; ZHOU, Y.; DIMIROVSKI, G. M.; JING, Y. Simulated annealing Q-learning algorithm for ABR traffic control of ATM networks. In: IEEE. *Conference on Decision and Control, 2007*. [S.l.], 2007. p. 5716 – 5721.

LIMA JÚNIOR, F. C. *Algoritmo Q-learning como Estratégia de Exploração e/ou Explotação para as Metaheurísticas GRASP e Algoritmo Genético*. Tese (Doutorado) — Universidade Federal do Rio Grande do Norte – UFRN, Natal, RN, 2009.

LIMA JÚNIOR, F. C.; DÓRIA NETO, A. D.; MELO, J. D. Using the Q-learning algorithm in the constructive phase of the GRASP and reactive GRASP metaheuristics. In: *IJCNN*. Hong Kong: IEEE, 2008. p. 4169–4176.

LIMA JÚNIOR, F. C.; MELO, J. D.; DÓRIA NETO, A. D. Using Q-learning algorithm for initialization of the GRASP metaheuristic and genetic algorithm. In: *IJCNN*. Florida: IEEE, 2007. p. 1243–1248.

LIN, S.; KERNIGHAN, B. W. An effective heuristic algorithm for the Traveling-Salesman Problem. *Operations Research*, v. 21, n. 2, p. 498–516, 1973.

LIU, X.; PARDALOS, P. M.; RAJASEKARAN, S.; RESENDE, M. A GRASP for frequency assignment in mobile radio networks. In: RAJASEKARAN, S.; PARDALOS, P.; F.HSU (Ed.). *Mobile Networks and Computing*. [S.l.]: American Mathematical Society, 2000, (DIMACS Series on Discrete Mathematics and Theoretical Computer Science, v. 52). p. 195–201.

LORENA, L. A. N.; SENNE, E. L. F. A column generation approach to capacitated p-median problem. *Computers and Operations Research*, v. 31, p. 863–876, 2004.

MARINAKIS, Y.; MIGDALAS, A.; PARDALOS, P. M. A hybrid geneticGRASP algorithm using lagrangean relaxation for the traveling salesman problem. *Journal of Combinatorial Optimization*, Springer, v. 10, n. 4, p. 311–326, 2005.

MESTRIA, M. *Metaheurísticas híbridas para a Resolução do problema do caixeiro Viajante com Grupamentos*. Dissertação (Mestrado) — Universidade Federal Fluminense, Niterói, RJ, 2011.

MLADENOVIĆ, N.; BRIMBERG, J.; HANSEN, P.; MORENO-PÉREZ, J. A. The p-median problem: A survey of metaheuristic approaches. *European Journal of Operational Research*, Elsevier, v. 179, n. 3, p. 927–939, 2007.

MLADENOVIĆ, N.; LABBÉ, M.; HANSEN, P. Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, Wiley Online Library, v. 42, n. 1, p. 48–64, 2003.

OLIVEIRA, C. A. S.; PARDALOS, P. M.; RESENDE, M. G. C. GRASP with path-relinking for the quadratic assignment problem. In: *Experimental and efficient algorithms*. [S.l.]: Springer, 2004. p. 356–368.

OSMAN, I. H.; LAPORTE, G. Metaheuristics: A bibliography. *Annals of Operations Research*, v. 63, p. 513–623, 1996.

OWEN, S. H.; DASKIN, M. S. Strategic facility location: A review. *European Journal of Operational Research*, Elsevier, v. 111, n. 3, p. 423–447, 1998.

PRAIS, M.; RIBEIRO, C. C. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *Journal on Computing*, v. 12, n. 3, p. 164–176, 2000.

PUTERMAN, M. L. *Markov Decision Processes Discrete Stochastic Dynamic Pro-gramming*. New York, USA: John Wiley e Sons, Inc, 2005.

QGIS. *A free and open source geographic information system.* 2014. Acesso: março-2014. Disponível em: <http://www.qgis.org/>.

RANGEL, M. C.; ABREU, N. M. M.; BOAVENTURA NETTO, P. O. GRASP para o PQA: Um limite de aceitação para soluções iniciais. *Pesquisa Operacional*, v. 20, n. 1, p. 45–58, 2000. ISSN 0101-7438. Disponível em: <http://dx.doi.org/10.1590/S0101-74382000000100006>.

RESENDE, M.; RIBEIRO, C. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In: GENDREAU, M.; POTVIN, J.-Y. (Ed.). *Handbook of Metaheuristics*. 2ª. ed. [S.l.]: Springer US, 2010, (International Series in Operations Research & Management Science, v. 146). p. 283–319.

RESENDE, M. G. C.; RIBEIRO, C. C. Greedy randomized adaptative search procedures. In: GLOVER, F.; KOCHENBERGER, G. (Ed.). *Handbook of Metaheuristics*. [S.l.]: Kluwer Academic Publishers, 2003. p. 219–249.

RIBEIRO, C. C.; UCHOA, E.; WERNECK, R. F. A hybrid GRASP with perturbations for the steiner problem in graphs. *INFORMS Journal on Computing*, INFORMS, v. 14, n. 3, p. 228–246, 2002.

RIBEIRO, C. C.; URRUTIA, S. Heuristics for the mirrored traveling tournament problem. *European Journal of Operational Research*, Elsevier, v. 179, n. 3, p. 775–787, 2007.

RIBEIRO, M. H.; PLASTINO, A.; MARTINS, S. L. Hybridization of GRASP metaheuristic with data mining techniques. *Journal of Mathematical Modelling and Algorithms*, Springer, v. 5, n. 1, p. 23–41, 2006.

SANTOS, J. P. Q.; MELO, J. D.; DÓRIA NETO, A. D.; ALOISE, D. Reactive search strategies using reinforcement learning, local search algorithms and variable neighborhood search. *Expert Systems with Applications*, Elsevier, 2014.

SANTOS, J. Q.; LIMA JÚNIOR, F. C.; MAGALHÃES, R. M.; DÓRIA NETO, A. D.; MELO, J. D. A parallel hybrid implementation using Genetic Algorithm, GRASP and Reinforcement Learning. *2009 International Joint Conference on Neural Networks (IJCNN)*, Atlanta, USA, 2009.

SANTOS, L. F.; RIBEIRO, M. H.; PLASTINO, A.; MARTINS, S. L. A hybrid GRASP with data mining for the maximum diversity problem. In: *Hybrid metaheuristics*. [S.l.]: Springer, 2005. p. 116–127.

STOCKHEIM, T.; SCHWIND, M.; KÖENIG, W. A reinforcement learning approach for supply chain management. In: *1st European Workshop on Multi-Agent Systems, Oxford, UK*. [S.l.: s.n.], 2003.

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. MA: MIT Press, 1998.

TALBI, E.-G. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, v. 8, n. 2, p. 541–564, 2002.

WAISELFISZ, J. J. Mapa da violência 2014: Os jovens do brasil. *Rio de Janeiro: Cebela, Flacso*, 2014. Disponível em: <http://www.mapadaviolencia.org.br/>.

WATKINS, C. J. C. H. *Learning from Delayed Rewards*. Tese (Doutorado) — University of Cambridge, England, 1989. Disponível em: <http://www.cs.rhul.ac.uk/ chrisw/thesis.html>.

WELCH, B. L. The generalization of "student's" problem when several different population variances are involved. *Biometrika*, JSTOR, p. 28–35, 1947.

ZOLFPOUR-AROKHLO, M.; SELAMAT, A.; MOHD HASHIM, S. Z.; AFKHAMI, H. Modeling of route planning system based on Q value-based dynamic programming with multi-agent reinforcement learning algorithms. *Engineering Applications of Artificial Intelligence*, Elsevier, 2014.

APÊNDICE A

LISTAGEM DOS RESULTADOS COMPUTACIONAIS

A.1 RESULTADOS DA METAHEURÍSTICA GRASP REATIVO

Tabela A.1: Valor da função objetivo (GRASP Reativo).

	Instâncias							
Iter.	MOSS1a	MOSS1b	MOSS2a	MOSS2b	MOSS3a	MOSS3b	MOSS4a	MOSS4b
1	1354,3	1009,5	958,7	693,5	687,5	588,9	628,4	524,4
2	1354,3	1013,2	936,5	686,9	693,5	587,9	652,9	545,7
3	1354,3	1009,5	995,6	686,9	693,5	600,1	617,6	532,9
4	1416,8	1008,7	936,5	693,5	689,1	600,1	632,0	556,7
5	1354,3	1009,5	936,5	689,1	689,1	587,9	632,0	532,9
6	1416,8	1013,2	970,2	689,1	687,5	587,9	652,9	556,7
7	1416,8	1013,2	995,6	689,1	693,5	600,1	652,9	556,7
8	1354,3	1009,5	936,5	693,5	687,6	587,9	625,0	524,4
9	1354,3	1009,5	936,5	689,1	686,9	587,9	626,2	556,7
10	1354,3	1008,7	958,7	686,9	693,5	587,9	632,0	529,5
11	1354,3	1013,2	936,5	689,1	687,5	587,9	632,2	545,7
12	1416,8	1009,5	995,6	689,1	689,1	587,9	644,2	524,4
13	1416,8	1013,2	936,5	686,9	686,9	587,9	652,9	556,7
14	1354,3	1013,2	995,6	689,1	686,9	587,9	647,4	524,4
15	1354,3	1009,5	936,5	693,5	687,5	587,9	640,2	529,5
16	1354,3	1009,5	936,5	686,9	686,9	587,9	625,0	537,0
17	1354,3	1013,2	936,5	686,9	686,9	587,9	634,2	537,0
18	1416,8	1009,5	958,7	689,1	689,1	601,1	664,2	545,7
19	1416,8	1013,2	936,5	689,1	687,5	587,9	647,8	537,0
20	1354,3	1013,2	936,5	686,9	689,1	587,9	618,8	532,9
21	1354,3	1009,5	936,5	689,1	693,5	600,1	632,0	530,0
22	1416,8	1009,5	936,5	693,5	693,5	601,1	652,9	545,7
23	1416,8	1013,2	936,5	693,5	693,5	587,9	636,0	556,7
24	1416,8	1008,7	958,7	689,1	686,9	600,1	626,2	545,7
25	1416,8	1013,2	936,5	689,1	693,5	600,1	618,8	524,4
26	1416,8	1008,7	936,5	686,9	693,5	587,9	617,6	534,9
27	1416,8	1013,2	958,7	686,9	693,5	587,9	652,9	556,7
28	1416,8	1009,5	936,5	689,1	689,1	601,1	652,9	524,4
29	1354,3	1013,2	936,5	689,1	693,5	600,1	634,2	532,8
30	1354,3	1009,5	958,7	693,5	693,5	600,1	652,9	532,9
$\overline{X}$	1383,5	1011,0	949,9	689,5	690,1	592,5	637,8	539,0
S^2	1007,68	3,75	435,93	5,99	8,47	37,40	178,46	143,86

Tabela A.2: Valores de tempo de processamento (GRASP Reativo).

	Instâncias							
Iter.	MOSS1a	MOSS1b	MOSS2a	MOSS2b	MOSS3a	MOSS3b	MOSS4a	MOSS4b
1	4,4	8,3	58,8	122,0	459,6	647,3	1701,7	1707,1
2	5,3	9,0	56,6	128,7	534,5	601,3	1650,0	2064,2
3	4,8	8,7	44,6	127,8	475,8	645,4	1402,5	2156,4
4	4,6	8,2	44,6	142,9	560,2	573,5	1391,1	2098,1
5	5,1	8,7	60,3	116,7	598,5	682,0	1390,0	1566,8
6	4,6	9,4	56,6	112,1	531,8	733,5	1516,4	1925,8
7	4,9	9,2	64,9	126,3	493,1	751,0	1288,8	1648,0
8	4,9	8,9	51,1	143,4	549,7	563,9	1401,4	1982,6
9	4,6	9,2	53,3	151,2	396,3	581,8	1281,4	2042,2
10	5,1	8,3	58,8	118,8	447,1	522,7	1436,2	2312,5
11	4,9	9,0	58,8	119,9	513,9	674,8	1619,2	2445,0
12	4,6	9,0	49,5	119,5	416,3	611,2	1381,0	2086,2
13	4,9	8,9	60,1	162,1	494,5	603,7	1488,0	1856,2
14	4,4	8,3	48,2	143,9	552,0	696,7	1219,9	1866,1
15	4,9	9,2	46,3	121,0	382,0	590,5	1530,3	1868,1
16	5,1	8,3	54,6	122,0	455,9	672,6	1619,4	1875,7
17	4,6	8,2	44,6	127,4	535,3	625,1	1498,8	2133,4
18	5,3	8,7	53,6	129,2	477,9	653,6	1350,4	1937,7
19	4,8	8,2	47,6	142,9	561,9	435,3	1580,8	1807,8
20	4,6	8,2	51,2	119,7	594,9	604,6	1379,8	1820,8
21	5,1	9,0	52,6	112,6	538,1	647,1	1424,1	1727,4
22	4,8	9,2	60,1	122,9	491,5	737,5	1315,4	2085,0
23	4,8	8,2	54,6	145,9	547,3	634,2	1445,7	2132,2
24	4,9	9,3	56,6	154,0	492,5	518,1	1439,3	2118,4
25	4,9	8,3	44,6	118,8	441,1	622,7	1423,2	1988,6
26	4,8	8,7	54,6	118,6	571,4	648,2	1613,4	1978,8
27	5,1	8,7	56,6	114,7	406,3	623,9	1480,0	2160,5
28	5,3	9,4	53,7	160,9	443,7	537,0	1418,7	1990,8
29	4,8	9,2	49,5	143,9	519,5	697,7	1339,0	2147,0
30	4,8	9,2	54,6	122,0	413,0	654,0	1328,9	1720,6
$\overline{X}$	4,8	8,8	53,4	130,4	496,5	626,4	1445,2	1975,0
S^2	0,06	0,18	29,87	215,28	3659,71	4793,71	13993,85	38921,62

A.2 RESULTADOS DA METAHEURÍSTICA GR-LEARNING

Tabela A.3: Valor da função objetivo (GR-Learning).

	Instâncias							
Iter.	MOSS1a	MOSS1b	MOSS2a	MOSS2b	MOSS3a	MOSS3b	MOSS4a	MOSS4b
1	1354,3	1009,5	936,5	686,9	686,9	587,9	644,2	531,7
2	1354,3	1009,5	936,5	689,1	686,9	587,9	632,0	541,9
3	1354,3	1009,5	936,5	686,9	687,5	591,9	625,0	532,9
4	1354,3	1008,7	936,5	689,1	689,1	601,1	655,8	545,7
5	1354,3	1008,7	936,5	689,1	686,9	587,9	655,8	524,4
6	1354,3	1008,7	936,5	686,9	693,5	587,9	618,8	545,7
7	1354,3	1009,5	936,5	686,9	690,1	571,8	617,6	532,9
8	1354,3	1008,7	936,5	689,1	686,9	600,1	632,0	530,1
9	1354,3	1009,5	936,5	686,9	686,9	587,9	617,6	524,4
10	1354,3	1009,5	936,5	686,9	693,5	587,9	632,0	524,4
11	1354,3	1009,5	936,5	686,9	686,9	600,1	618,8	525,8
12	1416,8	1009,5	936,5	693,5	693,5	600,1	617,6	530,1
13	1354,3	1008,7	936,5	686,9	686,9	601,1	632,0	545,7
14	1354,3	1009,5	936,5	686,9	687,5	591,9	652,9	532,9
15	1354,3	1009,5	936,5	686,9	686,9	601,1	618,8	524,4
16	1354,3	1008,7	936,5	686,9	687,5	587,9	625,0	534,9
17	1354,3	1009,5	936,5	689,1	686,9	587,9	655,8	524,4
18	1354,3	1013,2	936,5	693,5	687,5	593,7	618,8	532,9
19	1354,3	1009,5	936,5	686,9	689,1	587,9	652,9	513,6
20	1416,8	1009,5	936,5	693,5	687,5	587,9	632,0	524,4
21	1354,3	1008,7	936,5	686,9	686,9	587,9	632,0	524,4
22	1354,3	1009,5	936,5	686,9	687,5	600,1	632,0	524,4
23	1354,3	1008,7	936,5	689,1	686,9	587,9	647,4	529,5
24	1354,3	1008,7	936,5	686,9	693,5	587,9	625,0	545,7
25	1354,3	1008,7	936,5	686,9	687,5	599,2	626,2	529,5
26	1354,3	1013,2	936,5	686,9	693,5	587,9	632,0	548,6
27	1354,3	1009,5	936,5	693,5	686,9	600,1	628,6	534,9
28	1354,3	1009,5	936,5	686,9	687,5	588,9	625,0	532,9
29	1354,3	1008,7	936,5	689,1	686,9	599,1	618,8	545,7
30	1354,3	1009,5	936,5	686,9	687,5	587,9	632,0	524,4
$\overline{X}$	1358,5	1009,5	936,5	688,3	688,4	591,9	631,8	532,1
S^2	251,92	1,16	0,00	5,08	5,82	46,73	161,50	77,35

Tabela A.4: Valores de tempo de processamento (GR-Learning).

	Instâncias							
Iter.	MOSS1a	MOSS1b	MOSS2a	MOSS2b	MOSS3a	MOSS3b	MOSS4a	MOSS4b
1	4,7	9,5	44,2	97,1	322,9	535,8	1138,4	1796,7
2	5,1	9,2	44,9	97,5	319,6	561,5	1180,2	1732,3
3	4,7	9,1	44,2	94,9	326,6	542,8	1286,4	1892,5
4	4,8	9,2	45,9	99,8	298,1	559,2	1171,2	1804,6
5	5,0	9,1	44,9	97,1	325,9	529,9	1133,4	1678,4
6	4,9	9,1	45,9	94,5	329,3	550,8	1189,9	1803,8
7	5,0	7,2	49,5	96,6	311,2	548,8	1171,2	1819,8
8	5,0	8,8	44,2	97,8	322,2	558,0	1207,2	1808,9
9	4,8	8,8	39,7	97,1	322,8	528,8	1170,3	1737,4
10	4,7	9,2	44,2	96,1	320,9	563,4	1298,3	1786,5
11	4,7	8,8	44,2	97,5	320,3	519,1	1170,8	1706,1
12	4,9	9,0	49,5	95,6	315,9	531,6	1202,9	1763,7
13	4,9	8,0	44,2	94,0	316,5	510,4	1229,3	1756,2
14	5,2	9,2	44,6	92,9	325,1	520,9	1130,3	1769,9
15	4,8	9,2	44,2	94,5	312,4	522,2	1149,3	1777,7
16	4,5	9,0	39,4	97,5	328,6	558,2	1180,9	1743,8
17	5,1	8,4	45,9	97,1	315,9	515,1	1174,9	1828,4
18	4,6	8,8	39,7	94,5	321,8	527,8	1138,6	1732,1
19	4,4	9,1	50,6	98,5	314,5	528,8	1216,9	1758,1
20	5,2	7,2	45,9	92,1	322,0	520,0	1142,6	1835,1
21	4,7	8,8	44,9	98,0	323,1	551,8	1188,8	1789,7
22	4,8	8,8	43,9	95,8	317,6	547,9	1223,1	1795,3
23	5,2	9,0	46,0	97,6	319,4	572,1	1152,8	1690,9
24	4,6	8,8	44,9	97,4	324,6	503,8	1207,6	1785,3
25	5,5	8,1	44,2	90,9	323,9	533,5	1155,0	1739,8
26	4,5	8,2	39,7	96,3	322,9	508,6	1228,5	1815,0
27	4,7	9,0	44,9	96,1	318,9	534,7	1127,5	1701,4
28	4,9	9,1	47,9	98,0	315,0	523,4	1175,3	1793,6
29	5,2	9,0	48,9	97,5	315,6	559,4	1203,3	1871,7
30	4,8	8,0	44,2	97,8	310,2	516,9	1128,8	1746,1
$\overline{X}$	4,9	8,8	44,8	96,3	319,5	536,2	1182,5	1775,4
S^2	0,06	0,32	7,62	4,02	40,56	351,38	1842,42	2525,98