UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE – UERN

FACULDADE DE CIÊNCIAS EXATAS E NATURAIS – FANAT

DEPARTAMENTO DE INFORMÁTICA – DI

Endersson Danilo da Silva Tavares

Meta-heurística GRASP com Busca Local Iterada aplicada ao Problema de Reparação da Malha Viária após Grandes Desastres

MOSSORÓ - RN

2020

Endersson Danilo da Silva Tavares

Meta-heurística GRASP com Busca Local Iterada aplicada ao Problema de Reparação da Malha Viária apósGrandes Desastres

Monografia apresentada à Universidade do Estado do Rio Grande do Norte como um dos pré-requisitos para obtenção do grau de bacharel em Ciência da Computação, sob a orientação do Profº. Dr. Dario José Aloise e a coorientação da Profª. Dra. Cicília Raquel Maia Leite.

MOSSORÓ - RN

2020

Endersson Danilo da Silva Tavares

Meta-heurística GRASP com Busca Local Iterada aplicada ao Problema de Reparação da Malha Viária após Grandes Desastres

Monografia apresentada como pré-requisito para obtenção do título de Bacharel em Ciência da Computação da Universidade do Estado do Rio Grande do Norte – UERN, submetida à aprovação da banca examinadora composta pelos seguintes membros:

Aprovado em: __/__/____

Banca Examinadora

_____

Prof. Dr. Dario José Aloise (UERN)
Presidente

_____

Prof. Dr. Sebastião Emidio Alves Filho (UERN)
Membro

_____

Prof. Dr. Francisco Chagas de Lima Júnior (UERN)
Membro

À Deus, aos meus pais Alberto e Roseneide e a minha noiva Lindiohara.

AGRADECIMENTOS

Agradeço primeiramente a Deus que me proporcionou a possibilidade de começar, permanecer e terminar este curso, e de conhecer novas pessoas que me deram a mão, não apenas na parte acadêmica, mas na vida. Não somente isto, mas tudo em minha vida tem sido proporcionado por Ele.

Agradeço a minha família que me apoiou e incentivou a ir até o fim, principalmente a meu pai Francisco Alberto e a minha mãe Roseneide.

A minha noiva Lindiohara Cosmo dos Santos, que nos momentos mais difíceis não apenas da graduação, mas, da vida, esteve ao meu lado me apoiando e sempre incentivando a permanecer e vencer.

Aos amigos e irmãos que fiz durante a graduação, Antônio Anderson, Hitalo Vinicius, Thiago Jobson, João Ferreira, João André, e a vários outros que me apoiaram, sem os quais não teria chegado aqui.

Agradeço ao meu orientador o professor Dario José Aloise que me ajudou dentro e fora do curso, me impulsionou até aqui, agradeço a professora Cicília Raquel que me ajudou e orientou nessa caminhada.

Por fim a palavra e o sentimento que eu carrego comigo durante e depois desse curso, é “gratidão” por tudo e todos que de maneira direta e indiretamente contribuiu para que eu chegasse até aqui, obrigado a todos e mais uma vez obrigado Deus.

“Porque desde a antiguidade não se ouviu, nem com ouvidos se percebeu, nem com os olhos se viu um Deus além de ti que trabalha para aquele que nele espera.”

(Isaías 64:4).

RESUMO

Mesmo com o avanço da tecnologia, muitos desastres naturais como tsunamis e terremotos são imprevisíveis e podem ocorrer a qualquer momento e em qualquer lugar. Quando acontecem, em questão de pouco tempo deixam um rastro de destruição de consequências catastróficas, que ocasionam prejuízos financeiros e vítimas humanas, afetando de maneira significativa a vida de milhares de pessoas, com perdas e devastação à vida e à propriedade. O que também implica de maneira negativa os sistemas de emergência e, dependendo da intensidade e gravidade, deteriora os processos normais da população, como alimentação, abrigo e acessibilidade. A acessibilidade da rede viária se torna uma questão importante para as equipes de socorro que precisam fornecer suprimentos e alívio aos feridos, principalmente nas primeiras horas após o desastre. Sobre esse pretexto justifica-se a utilização e/ou o desenvolvimento de meta-heurísticas eficiente para criar uma programação de trabalho para as máquinas que fazem a reparação das vias a fim de proporcionar alívio e atendimento às vítimas. Este trabalho tem esse objetivo e para isso apresenta uma melhoria na meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) com Busca Local Iterada, em inglês *Iterated Local Search* (ILS), criando assim uma meta-heurística hibrida para o problema em apreço. Testes computacionais em instâncias da literatura mostram a eficiência da meta-heurística proposta.

Palavras-chave: Desastres naturais, acessibilidade, rede viária, otimização, meta-heurística.

ABSTRACT

Even with the advancement of technology, many natural disasters such as tsunamis and earthquakes are unpredictable and can occur anytime and anywhere. When they happen, in a short time they leave a trail of destruction with catastrophic consequences, which cause financial losses and human victims, significantly affecting the lives of thousands of people, with losses and devastation to life and property. This also negatively implies emergency systems and, depending on the intensity and severity, deteriorates the normal processes of the population, such as food, shelter and accessibility. The accessibility of the road network becomes an important issue for relief teams who need to provide supplies and relief to the injured, especially in the first hours after the disaster. On this pretext, the use and / or development of efficient meta-heuristics is justified to create a work schedule for the machines that repair the roads in order to provide relief and care for the victims. This work has this objective and for that it presents a Greedy Randomized Adaptive Search Procedure (GRASP) meta-heuristic with Iterated Local Search, in English Iterated Local Search (ILS), thus creating a hybrid meta-heuristic for the problem under consideration. Computational tests in instances of the literature show the efficiency of the proposed meta-heuristic.

Keywords: Natural disasters, accessibility, road network, optimization, meta-heuristics.

LISTA DE SIGLAS

CM	Caminho Mínimo
WSP	*Work-troops Scheduling Problem*
WT	Work-troops
ILS	Iterated Local Search (Busca Local Iterada)
GRASP	Greedy Randomized Adaptive Search Procedure
LRC	Lista Restrita de Candidatos

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1. INTRODUÇÃO

Os desastres naturais de grandes proporções, como tsunamis e terremotos são devastadores, e podem ocorrer a qualquer momento e em qualquer lugar. Quando acontecem, em questão de pouco tempo deixam um rastro de consequências catastróficas, que ocasionam prejuízos financeiros e vítimas humanas, afetando de maneira significativa a vida de milhares de pessoas, com perdas e devastação à vida e à propriedade. Como ocorreu em Porto Príncipe em 2010. (SAKURABA et al., 2016; BRILHAM, 2010).

Em 12 de janeiro de 2010, um terremoto de magnitude 7,0 na escala Richter atingiu o Haiti, provocando uma série de feridos, desabrigados e mortes. Conforme o Serviço Geológico dos Estados Unidos, o terremoto ocorreu a cerca de 10 quilômetros de profundidade, a 22 quilômetros de Porto Príncipe. Esse primeiro terremoto antecedeu outros dois de magnitudes 5,9 e 5,5. Esse fato promoveu grande destruição na região da capital haitiana, estima-se que metade das construções foram destruídas, 250 mil pessoas foram feridas, 1,5 milhão de habitantes ficaram desabrigados e o número de mortos ultrapassou 200 mil. (FRANCISCO, 2019).

Figura 1:Uma das áreas do Haiti atingida pelo sismo

Fonte: https://pt.wikipedia.org/wiki/Sismo_do_Haiti_de_2010

Figura 2: Rua bloqueada devido ao sismo.

Fonte: http://g1.globo.com/mundo/noticia/2013/01/veja-fotos-do-haiti-logo-apos-o-tremor-de-2010-e-tres-anos-apos-o-desastre.html

Fatos como este afetam de maneira negativa os serviços de emergência e, dependendo da intensidade e gravidade, deterioram os processos normais da população, como alimentação, abrigo e acessibilidade (Shankar, 2011). Fazendo com que a população atingida se junte em acampamentos espontâneos pela cidade. A acessibilidade da rede viária danificada e bloqueada com escombros, como demonstra a Figura 2, se torna uma questão importante para as equipes de socorro que precisam fornecer suprimentos e alívio aos feridos, principalmente nas primeiras horas após o desastre. Sobre esse pretexto justifica-se a utilização e/ou o desenvolvimento de meta-heurística eficiente para criar uma programação de trabalho para as máquinas que fazem a reparação das vias a fim de proporcionar alívio e atendimento às vítimas.

Este trabalho tem por objetivo apresentar uma melhoria na meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP), introduzindo na fase de busca local, a meta-heurística Busca Local Iterada (em inglês: *Iterated Local Search* - ILS), criando assim uma meta-heurística híbrida, mais robusta, para otimizar a reparação da malha viária após grandes desastres.

O documento encontra-se organizado da seguinte maneira: no capítulo 2 é apresentada a fundamentação teórica, contendo os temas que serviram de base

para o desenvolvimento da meta-heurística; no capítulo 3 é apresentado a definição do problema, as heurísticas e meta-heurística usadas para o problema e o método híbrido do GRASP utilizando a Busca Local Iterada; no capitulo 4 são apresentados os resultados alcançados pelo método hibrido desenvolvido neste trabalho; e por fim, o capitulo 5 traz as considerações finais.

2. FUNDADMENTAÇÃO TEÓRICA

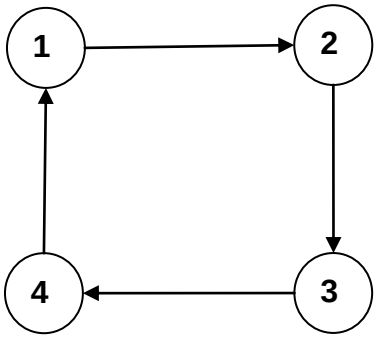
Este capítulo busca apresentar ao leitor (a) os conceitos teóricos utilizados e que são importantes para o entendimento do trabalho.

2.1. Teoria dos Grafos

Um grafo consiste em um conjunto de objetos (vértices ou nós) que se relacionam entre si; a este relacionamento, damos o nome de aresta. Na teoria dos grafos, a definição formal de um grafo é dada por $G = (V, E)$, onde $V = \{v_1, v_2, \dots, v_n\}$ representa o conjunto de vértices e $E = \{e_1, e_2, \dots, e_n\}$ representa o conjuntos de arestas (SOUZA, 2018).

Os grafos podem ter características distintas, tais como arestas direcionais, que passam a ser representada por uma seta ($\rightarrow$), caracterizando assim um grafo orientado, também denominado grafo dirigido, grafo direcionado ou dígrafo) (SOUZA, 2018). A Figura 3 apresenta um exemplo de um grafo $G = (V, A)$ com 4 vértices e 4 arestas direcionais.

Figura 3: Exemplo de um grafo dirigido G = (V, A).



Fonte: Autoria própria.

2.2. Otimização Combinatória

"Otimização combinatória é um ramo da ciência da computação e da matemática aplicada que estuda problemas de otimização em conjuntos finitos. Em um problema de otimização combinatória tem-se uma função objetivo e um conjunto de restrições, ambos relacionados às variáveis de decisão. Os valores possíveis das variáveis de decisão são delimitados pelas restrições impostas sobre estas variáveis, formando um conjunto discreto (finito ou não) de soluções factíveis a um problema. Segundo Nazareth (2004, p. 1)."

WIKIPÉDIA

Muitos problemas práticos de otimização, bem como aqueles de importância teórica, consistem na busca por uma "melhor" configuração de um conjunto de variáveis para alcançar determinadas metas. Tais problemas são divididos em duas categorias: aqueles para os quais as soluções são codificadas com variáveis de valor real, e aqueles cujas soluções são codificadas com variáveis discretas. Estes últimos, caracteriza a classe de problemas de Otimização Combinatória (BLUM; ROLI, 2003). O processo de pesquisa pela melhor solução ou solução ótima de um problema de Otimização Combinatória pode se dar através da busca entre todas as possíveis combinações de valores para melhorar as variáveis do problema, calculando a função objetivo e verificando se atende às restrições do problema. A grande questão é que, dependendo do tamanho do problema, essa busca torna-se inviável, pois em situações em que o tempo e/ou custo de processamento é importante, não se consegue chegar em uma solução em tempo razoável.

A teoria da complexidade explica a noção de tempo razoável de processamento, demonstrando que muitos problemas não possuem algoritmos que realizem a busca completa no espaço de solução em tempo polinomial. A busca exaustiva de problemas de otimização combinatória torna-se impossível quando aumenta o tamanho do problema (número de variáveis envolvidas), dada a quantidade de combinações possíveis entre seus elementos, sendo estes

problemas classificados como NP-difíceis (CUNHA; TAKAHASHI; ANTUNES, 2012). Para problemas NP-difíceis não são conhecidos algoritmos em tempo polinomial, assim, o processamento em busca de um resultado pode exigir grande quantidade de recursos computacionais, levando a tempos de computação muito elevados para fins práticos. Para tanto, existem técnicas especializadas para explorar o espaço de solução em busca de bons resultados, vasculhando-o de forma inteligente, sem comprometer a eficiência computacional, mas ao preço de não garantirem encontrar a solução ótima. A utilização de tais métodos, ditos aproximados tem recebido cada vez mais atenção. Com eles, sacrificamos a garantia de encontrar as melhores soluções para o bem de obter boas soluções em uma quantidade significativamente reduzida de tempo. Os métodos aproximados para solução dessa classe de problemas incluem: algoritmos heurísticos e meta-heurísticas (BLUM; ROLI, 2003).

2.3. Meta-heurísticas

Para problemas NP-difíceis não são conhecidos algoritmos em tempo polinomial, assim, o processamento em busca de um resultado pode exigir grande quantidade de recursos computacionais, levando a tempos de computação muito elevados para fins práticos. Mas ao preço de não garantirem encontrar a solução ótima. Com eles, sacrificamos a garantia de encontrar as soluções ótimas para o bem de obter boas soluções em uma quantidade significativamente reduzida de tempo. Os métodos não-exatos para solução dessa classe de problemas incluem: algoritmos heurísticos e meta-heurísticas (BLUM; ROLI, 2003).

Meta-heurísticas são estratégias gerais que "orientam" o processo de busca para evitar a retenção do resultado em máximos ou mínimos locais. O objetivo é explorar eficientemente o espaço de busca para encontrar soluções ótimas ou próximas das ótimas. Técnicas que abordam algoritmos meta-heurísticos vão desde procedimentos de busca locais simples até processos de aprendizagem complexos (BLUM; ROLI, 2003).

As origens das meta-heurísticas encontram-se na Inteligência Artificial e Pesquisa Operacional. O termo se refere a algoritmos aproximados para a otimização que não são especificamente expressos para um problema particular.

Exemplos de meta-heurísticas são Recozimento Simulado (KIRKPATRICK; GELATT; VECCHI, 1983), Busca Tabu (GLOVER, 1986), Algoritmo Genético (HOLLAND, 1992), Otimização por Colônia de Formigas (DORIGO; CARO, 1999), GRASP (FEO; RESENDE, 1995) e Busca Local Iterativa (LOURENÇO; MARTIN; STÜTZLE, 2002). Cada uma dessas meta-heurísticas tem seu próprio contexto histórico, sendo algumas inspiradas em processos naturais, tais como na evolução biológica, enquanto outras são extensões de algoritmos menos sofisticados como de heurísticas gulosas e da busca local.

Blum e Roli (2003) afirmam que as meta-heurísticas são estratégias de alto nível para explorar espaços de busca usando diferentes métodos. De grande importância, é o que se dá no equilíbrio dinâmico entre diversificação e intensificação. O termo diversificação geralmente se refere à exploração do espaço de busca, enquanto o termo intensificação refere-se à explotação da experiência de pesquisa acumulada. O equilíbrio entre diversificação e intensificação como mencionado é importante, de um lado, para identificar rapidamente regiões no espaço de busca com soluções de alta qualidade e, por outro lado, não desperdiçar muito tempo em regiões do espaço de busca que já foram exploradas ou que não oferecem soluções de alta qualidade (FIRMINO CEDMA, 2017).

2.4. Meta-heurísticas híbridas

Quando temos meta-heurísticas que não seguem os paradigmas tradicionais, isto é, que combinam diferentes componentes algoritmos até mesmo provenientes de outras áreas de pesquisa, temos o que chamamos hibridização de meta-heurísticas, ou meta-heurísticas hibridas (NETO, 2016). A hibridização de meta-heurística é importante pois um algoritmo híbrido pode combinar os benefícios de diferentes algoritmos, muitas vezes apresentando um desempenho mais eficiente do que algoritmos baseados em meta-heurísticas simples (GENDREAU; POTVIN,2010).

O objetivo por trás das hibridizações de diferentes conceitos algorítmicos é normalmente a obtenção de desempenho de sistemas que exploram e unem vantagens das estratégias puras individuais (RAIDL, 2006). As meta-heurísticas

hibridas se dão através da combinação de duas ou mais estratégias de soluções de problemas, tais como diferentes meta-heurísticas, meta-heurísticas com técnicas de inteligência artificial ou de pesquisa operacional, dentre outras (RAIDL, 2006). Somente quando se tornou claro que meta-heurísticas puras tinham atingido os seus limites, investigadores voltaram-se para a combinação de diferentes algoritmos (BLUM; ROLI, 2003).

Blum e Roli afirmam que existem vários tipos de hibridização que têm sido bem-sucedidos para muitas aplicações.

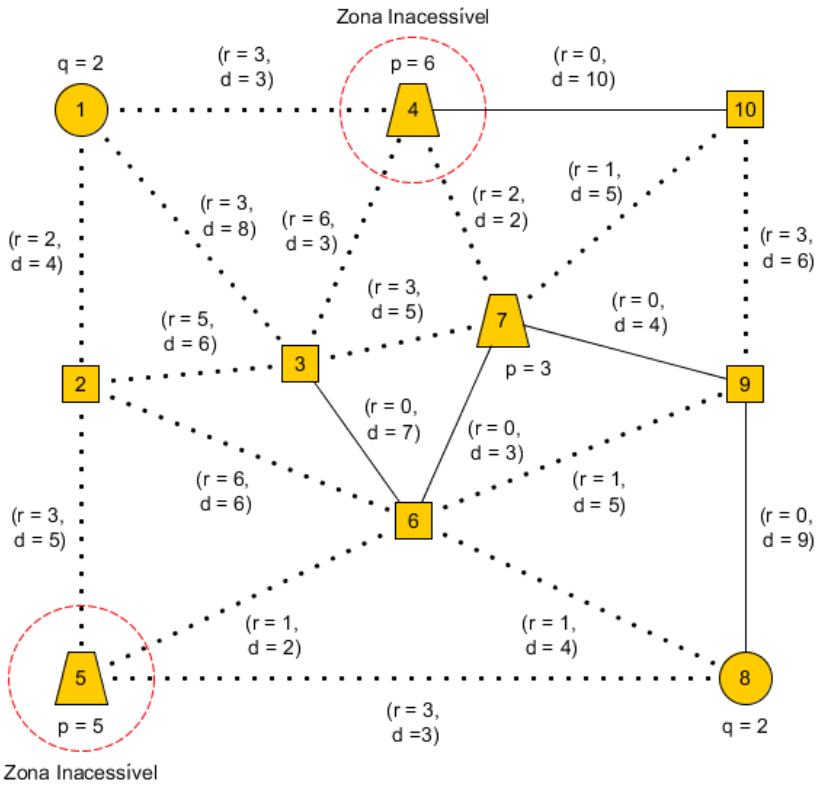
3. DESCRIÇÃO DO PROBLEMA E MODELAGEM

Este capítulo apresenta formalmente o problema tratado neste trabalho. Também é apresentado o método híbrido composto de duas meta-heurísticas de sucesso da literatura.

3.1. Work-troops Scheduling Problem

O WSP (*Work-troops Scheduling Problem*) (SAKURABA et al., 2016; SAKURABA; SANTOS; PRINS, 2016) visa calcular o planejamento para as máquinas que fazem a reparação das vias bloqueadas, diminuindo as distâncias e melhorando a acessibilidade das origens até os acampamentos o mais rápido possível. A Figura 4 ilustra um exemplo de um grafo G = (V,E), onde V = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, E = {[1, 2], [1, 3], [1, 4], [2, 3], [2, 5], [2, 6], [3, 4], [3, 6], [3, 7], [4, 7], [4, 10], [5, 6], [5, 8], [6, 7], [6, 8], [6, 9], [7, 9], [7, 10], [8, 9], [9, 10]} e B = E \{[3, 6], [4, 10], [6, 7], [7, 9], [8, 9]}, onde as origens são representadas por nós em formato de círculo (1 e 8); acampamentos, por losangos (4, 5 e 7) e nós de trasbordo, por quadrados. Vias bloqueadas $r_{ij} > 0$ são representados por arestas pontilhadas e não permitem passagem, as demais arestas $r_{ij} = 0$ são transitáveis a custo d_{ij} . A largura $u_{ij} = 1$ das vias está oculta. A rede urbana representada por G, neste exemplo, mostra que a acessibilidade até os acampamentos 4 e 5 está comprometida e não é possível alcançá-los a partir das origens 1 ou 8, pois não existe um caminho composto totalmente de arestas disponíveis entre origens e acampamentos.

Figura 4:Exemplo de instância do WSP.



A formulação matemática considera o aspecto dinâmico do problema, a evolução do estado da rede viária conforme as vias são reparadas, recalculado os caminhos mínimos para os acampamentos em cada período de tempo.

O modelo do WSP tem dois componentes principais: um fluxo da raiz artificial para cada acampamento e um fluxo de máquinas das origens até a extremidade dos arcos. Toma-se $B' \subset A$ o subconjunto de todos os arcos obtidos por substituir arestas bloqueadas $[i,j] \in B$ por arcos (i,j) e (j,i) e considera-se que o fluxo de máquinas não pode passar por um arco $(i,j) \in B'$, e seu valor d_{ij} é definido como um inteiro excessivamente grande M.

O modelo do WSP funciona da seguinte forma: a reparação do grafo é discretizada em períodos de tempo $t = 1, \dots, T$, e ao final de cada período de tempo t, a variável x_{ij}^t denota se existe fluxo pelo arco (i,j), enquantof_{ij}^t representa a quantidade de fluxo em (i,j). Além disso, a variável $z_{i\,j}^t$ especifica se a aresta $[i,j]$ está disponível ou não $(\sum_{[i,j]\in B} z_{ij}^0 = 0)$, i.e., se permite a passagem de máquinas. A variável s_i^t guarda a distância entre a raiz artificial 0 e o

vértice i no período de tempo t. Durante cada período, o fluxo de máquinas no arco (i,j) é determinado pela variável y_{ij}^t , enquanto w_{ij}^trepresenta o fluxo de máquinas entrando no arco (i,j) para repará-lo. A Tabela 1 apresenta uma referência rápida para as variáveis e constantes da formulação matemática do WSP, proposta por Sakuraba et al. (2016):

Tabela 1: Tabela de referência para as variáveis e constantes do modelo do WSP.

Variáveis

- s_i^t Distância da raiz artificial até i no período t
- $f_{i\,j}^t$ Quantidade de fluxo passando por (i,j) no período t
- $x_{i\,j}^t$ Variável binária que denota se existe, ou não, fluxo passando por (i,j) no período t
- $y_{i\,j}^t$ Fluxo de WT passando por (i,j) no período t
- w_{ij} Fluxo de WT reparando (i,j) no período t
- z_{ij}^t Denota se $[i,j]$ está disponível (reparada) no período t

Constantes

- p_i População existente em i
- $r_{i\,j}$ Quantidade de trabalho necessário para reparar $[i,j]$
- dij Comprimento físico de $[i,j]$
- uij Largura física de $[i,j]$
- qi Quantidade de máquinas inicialmente na origem i
- 0 (zero) Raiz artificial
- M Distância excessivamente alta

Conjuntos

- D Conjunto de acampamentos
- B Conjunto de arestas bloqueadas

A formulação matemática do WSP de (1.11) a (1.18) foi proposta por Sakuraba et al. (2016):

$$Min\ Z\ = \sum_{i \in D} \ \sum_{t=i}^{T} \ p_i s_i^t \ s.a. \tag{1.11}$$

$$\sum_{i:(i,j)\in A'} \ f_{ij}^t - \sum_{i:(j,i)\in A'} \ f_{j\,i}^t = \{D, se\ i = 0\ 0, se\ \forall \in V \backslash D\ 1, se\ \forall j\ \in D\ \ t = 1, \dots, T \tag{1.12}$$

$$x_{ij}^t \geq \frac{f_{ij}^t}{|D|} \qquad \forall (i,j) \in A^{'}, t = 1, \dots, T \tag{1.13}$$

$$x_{ij}^t \leq \ f_{ij}^t \qquad \forall (i,j) \in A^{'}, t = 1, \cdots, T \tag{1.14}$$

$$\sum_{j:(i,j)\in A} \ y_{ij}^t + \sum_{k:(i,k)\in B'} \ w_{ik}^t \leq q_i \qquad \forall i\ \in O, t = 1, \cdots, T \tag{1.15}$$

$$\sum_{i:(i,j)\in A} \ y_{ij}^t - \sum_{i:(j,i)\in A} \ y_{ji}^t = \sum_{k:(j,k)\in B'} \ w_{jk}^t \qquad \forall j\ \in O, t = 1, \dots, T \tag{1.16}$$

$$\sum_{[i,j]\in B} \ z_{ij}^t = \ |B| \tag{1.17}$$

$$r_{ij} z_{ij}^t \leq \sum_{t'=1}^{t} \ \left(w_{ij}^{t'} + w_{ji}^{t'}\right) \qquad \forall\ [i,j] \in E, t = 1 \dots., T \tag{1.18}$$

$$y_{ij}^t + y_{ji}^t \leq \sum_{i \in E} \ q_i z_{ij}^{t-1} \qquad \forall [i,j] \in E, t = 1, \dots, T \tag{1.19}$$

$$w_{ij}^{t} \leq u_{ij} \qquad \forall (i,j) \in A, t = 1, \dots, T \qquad (1.20)$$

$$s_{j}^{t} \geq M\left(x_{ij}^{t} - z_{ij}^{t}\right) \qquad \forall (i,j) \in A', t = 1, \dots, T \qquad (1.21)$$

$$s_{i}^{t} - s_{j}^{t} + \left(M + d_{ij}\right)x_{ij} \leq M \qquad \forall (i,j) \in A', t = 1, \dots, T \qquad (1.22)$$

$$f_{ij}^{t} \geq 0 \qquad \forall (i,j) \in A', t = 1, \dots, T \qquad (1.23)$$

$$s_{i}^{t} \geq 0 \qquad \forall i \in V', t = 1, \dots, T \qquad (1.24)$$

$$w_{ij}^{t} \geq 0 \qquad \forall (i,j) \in A, t = 1, \dots, T \qquad (1.25)$$

$$y_{ij}^{t} \geq 0 \qquad \forall (i,j) \in A, t = 1, \dots, T \qquad (1.26)$$

$$x_{ij}^{t} \in \{1, 0\} \qquad \forall (i,j) \in A', t = 1, \dots, T \qquad (1.27)$$

$$z_{ij}^{t} \in \{1, 0\} \qquad \forall [i,j] \in E, t = 1, \dots, T \qquad (1.28)$$

A função objetivo (1.11) é a soma ponderada das distâncias da raiz artificial multiplicada pela população de cada um dos acampamentos durante todos os períodos. As equações (1.12) garantem que exista um fluxo unitário da raiz artificial para cada acampamento, enquanto as inequações (1.13) e (1.14) relacionam as variáveis $f_{i\,j}^{t}$ e $x_{i\,j}^{t}$. As restrições (1.15) e (1.16) limitam o fluxo de máquinas ao máximo de qi máquinas inicialmente nas origens, alocando-as para arcos bloqueados. A equação (1.17) garante que todas as arestas inicialmente bloqueadas sejam reparadas até o último período T, enquanto que as inequações (2.18) calculam a disponibilidade de uma aresta de acordo com a quantidade de trabalho recebido das máquinas. As restrições (1.19) forçam as máquinas a

transitarem somente por arestas disponíveis. As inequações (1.20) limitam o número de máquinas que podem trabalhar em cada extremidade das vias de acordo com a sua largura física. As restrições (1.21) fixam a distância até um vértice a M se todos os caminhos estiverem bloqueados. A distância da raiz artificial até cada vértice é calcula pelas inequações (1.22), de maneira semelhante ao algoritmo de caminho mínimo de Dijkstra, acumulando-se as distâncias entre vértices. As variáveis são definidas em (1.23) – (1.28). Esta formulação contém $O(|A'|T)$ variáveis e restrições.

4. HEURÍSTICAS E META-HEURÍSTICAS PARA O PROBLEMA

Neste capitulo, apresentamos as heurísticas da literatura utilizados e as meta-heurísticas desenvolvidas.

4.1.1. Heurística de Ranque

A heurística de Ranque (SAKURABA et al., 2016) elabora uma lista ranqueada utilizando um critério guloso de avaliação das vias bloqueadas. O ranque de uma via é calculado pela quantidade de vezes que ela aparece nos caminhos mínimos entre origens e os acampamentos. Para isto, a heurística computa os caminhos mínimos ótimos entre todos os pares (o, d), com $o \in O$ e $d \in D$ e conta a ocorrência de cada rua bloqueada presente nestes caminhos; depois, as arestas são ordenadas pelo seu ranque em ordem decrescente.

Nesta heurística, a reparação de toda a rede é realizada em períodos de tempo discretizados (assim como no modelo matemático). O primeiro período é reservado para a reparação de caminhos mínimos que só possuam uma única via bloqueada. (Isto ocorreu devido a análises realizadas nos resultados produzidos pelo modelo matemático que indicaram que os caminhos com poucas vias bloqueadas são priorizados e reparados no primeiro período de tempo. A heurística então “imita” este comportamento.) Em seguida, a alocação das máquinas às arestas é feita a partir da cabeça da lista ranqueada, configurando um comportamento puramente guloso. Em caso de arestas bloqueadas com um

mesmo ranque, o critério de desempate é feito pela aresta de menor r_{ij}, pois exige menos tempo de trabalho.

O Algoritmo 1 apresenta o pseudocódigo da heurística de Ranque. As linhas 2 - 10 calculam o ranque das vias bloqueadas nos caminhos mínimos (linha 5) entre todos os pares de origens e acampamentos. Os caminhos que possuam somente uma única aresta bloqueada são selecionados nas linhas 6 e 7 e adicionados na lista LP (uma lista especial que armazena arestas que serão priorizadas). A instrução 11 ordena as arestas bloqueadas em ordem decrescente de ranque. As linhas 12 e 13 verificam se existe algum caminho mínimo com somente uma via bloqueada e $u_{ij} \geq r_{ij}$ para repará-lo no primeiro período. Os passos 15 - 22 reparam as vias bloqueadas a partir do inicio da lista ranqueada. A linha 20 atualiza os caminhos mínimos (CM) do grafo quando uma via bloqueada é totalmente reparada.

A heurística de Ranque possui complexidade de tempo $O\,(lg|V|)$ para classificar todas as arestas pelo ranque e $O(|E|lg|V|)$ para reparar todas as vias, onde E é o conjunto de arestas no grafo; V , o conjunto de vértices e $B \subset E$, o conjunto de arestas bloqueadas. O termo $|E|lg|V|$ é a complexidade assintótica do algoritmo de caminhos mínimos.

Algoritmo 1: Heurística de Ranque

1: **procedimento** HeurísticaRanque($G = (V, E), O, D, B$)
2: **para todo** $o \in O$ **faça**
3: Calcule os caminhos mínimos (CM) para os vértices usando r_{ij} como custo
4: **para todo** $d \in D$ **faça**
5: Adicione 1 ao ranque de cada $[i, j] \in B$ no SP de d
6: **se** CM até d possui somente uma via bloqueada **então**
7: Adicione esta via bloqueada à lista LP
8: **fim se**

```
 9:  fim para
10:  fim para
11:  Ordene as vias bloqueadas em ordem decrescente de ranque
12:  se existe aresta em LP com u_ij ≥ r_ij então
13:      Aloque as máquinas para estas arestas imediatamente
14:  fim se
15:  para t ← 1 até T faça
16:      para toda máquina disponível faça
17:          Aloque a máquina para a primeira via bloqueada a partir da cabeça da lista
18:      fim para
19:      se alguma [i, j] ∈ B tornou-se disponível então
20:          Atualize os caminhos mínimos utilizando a nova aresta disponível
21:      fim se
22:  fim para
23:  retorne A programação das máquinas e valor da Função Objetivo
24: fim procedimento
```

4.1.2. Greedy Randomized Adaptive Search Procedure (GRASP)

O GRASP é um método iterativo e multipartida que combina as boas características dos algoritmos puramente gulosos e dos procedimentos aleatórios na sua fase de construção de soluções. Em seguida, um procedimento de busca local é utilizado para refinar a solução inicial (FEO; RESENDE, 1989). O papel da primeira fase é de criar soluções utilizando dois paradigmas distintos: gula e aleatoriedade. A segunda fase de uma iteração é realizada por um algoritmo de busca local, que faz melhoria da solução criada pela fase de construção. Usualmente, várias iterações são realizadas e a melhor solução gerada dentre todas é guardada e exibida como resultado final (FESTA; RESENDE, 2002; RESENDE; FESTA, 2008).

O critério de parada do GRASP é um parâmetro que pode ser definido como um número máximo de iterações, um tempo máximo de execução ou um número máximo de iterações sem melhoria da melhor solução encontrada. Este

último leva em consideração quantas fases de construção e busca local ocorreram sem que uma melhor solução tenha sido encontrada. Quase sempre, o tempo de execução não varia muito de iteração para iteração e o tempo total de processamento é previsível, aumentando linearmente conforme o número de iterações (RESENDE; RIBEIRO, 2003).

A busca local demanda um maior tempo de execução se comparado à fase de construção do GRASP. Segundo RESENDE; RIBEIRO, 2003; BINATO et al., 2000, é nesse momento que uma boa fase de construção se mostra importante, para que soluções de qualidade possam ser utilizadas como ponto de partida na fase de busca local, e implique em menos tempo de processamento.

Considerando um problema de minimização, uma iteração do GRASP inicia com uma solução vazia e, durante a sua construção, os elementos (candidatos) disponíveis para serem adicionados à solução sendo construída são mantidos em uma lista de candidatos C ($C = [c_1, c_2, \dots, c_n]$) em uma ordem decrescente de classificação definida por uma função gulosa $g(c)$, i.e., $g(c1) \geq g(c2) \geq \dots \geq g(c_n)$ (RESENDE; RIBEIRO, 2003).

Durante uma iteração, a solução é construída candidato a candidato, com aqueles que já foram utilizados removidos da lista de candidatos. Com isso, em um dado momento da construção, C contém somente os candidatos que estão prontamente disponíveis para serem inseridos à solução parcial, mas que ainda não foram (MOUZON; YILDIRIM, 2008).

O α é um parâmetro que pertence ao intervalo [0, 1] e é responsável por restringir a lista de candidatos, criando a lista restrita de candidatos (LRC). Baseando-se na cardinalidade. Sejam $G_{min} = min\{c \in C\}$ e $G_{max} = max\{c \in C\}$ o menor e maior custo incremental dos candidatos de acordo com a função gulosa $g(c)$, onde C é a lista de todos os possíveis candidatos a compor a solução. De acordo com Lima Júnior F. C. (2009), tem-se:

- Estratégia de construção da LRC com base na cardinalidade: ordena-se C em ordem decrescente de benefício segundo a função g(c), de modo que para o primeiro candidato $g(c) = G_{min}$, e para o último, $g(c) = G_{max}$. A

partir desse ponto, incluem-se os p primeiros elementos de C na LRC, sendo o valor de p calculado por:

$$p = 1 + \alpha(N - 1) \qquad (3.2)$$

O □ (alfa) influencia o aspecto probabilístico do GRASP. Normalmente realizam-se experimentos de calibração para analisar qual valor de □ apresenta melhores soluções para o problema empregado (FESTA; RESENDE, 2011; Lima Júnior F. C., 2009).

Antes de se incorporar um novo elemento à solução parcial, repete-se o processo de gerar a lista restrita de candidatos e escolhe-se um elemento aleatoriamente a partir desta. Uma vez que o candidato é adicionado à solução, a lista de candidatos é atualizada e os candidatos são reavaliados pela função gulosa g. O processo se repete até que se tenha uma solução completa (RESENDE; RIBEIRO, 2003).

O Algoritmo 3 apresenta o pseudocódigo da meta-heurística GRASP adaptado de Martí et al. (2015) e Resende e Ribeiro (2003). Os parâmetros α e $maxIts$ são passados como entrada (linha 1), em seguida, cria-se uma variável, inicialmente vazia, para armazenar a melhor solução (linha 2). As linhas 3 - 7 realizam o processo de iterações. Cada iteração começa com um chamado à heurística construtiva (linha 4), seguida pela fase de busca local (linha 5). Após as duas fases, a melhor solução é atualizada, se a solução retornada pela busca local for melhor que a melhor solução encontrada até o momento (linha 6). O procedimento finaliza quando o critério de parada é alcançado, retornando a melhor solução dentre todas as iterações (linha 8).

Algoritmo 2: Pseudocódigo da meta-heurística GRASP.

```
1: procedimento GRASP(α, maxIts)
2:    s+ ← □ ;
3:    para it ← 1 até maxIts faça
4:       s ← ConstroiSolução(α)
5:       s' ← BuscaLocal(s)
```

6: $s^+ \leftarrow \text{Min}\,(s', s^+)$
7: **fim para**
8: **retorne** s^+
9: **fim procedimento**

O Algoritmo 4 apresenta um pseudocódigo para uma heurística de construção genérica mencionada na linha 4 do Algoritmo 3, também adaptado de Festa e Resende (2011). O pseudocódigo utiliza a estratégia de *LRC* baseada na cardinalidade. Os elementos que serão usados para gerar a solução são passados por parâmetro, assim como o parâmetro α que controla a gula e a aleatoriedade da solução (linha 1). A construção começa com uma solução inicialmente vazia (linha 2), e então se dá até que esteja completa (linha 8). A linha 5 seleciona os p melhores elementos da C para compor a *LRC* de acordo com a equação 3.1, e então um candidato aleatório é sorteado para ser adicionado à solução sendo construída (linha 6). Após este processo, os candidatos são novamente avaliados de acordo com uma função gulosa (7). Quando se obtém uma solução completa, a heurística retorna a solução construída (linha 9).

Algoritmo 3: Heurística de construção do GRASP.

1: **procedimento** ConstroiSoluçao($candidatos$, α)
2: $s = \square$
3: $C =$ candidatos avaliados de acordo com uma função gulosa $g(c)$
4: **enquanto** s não for uma solução completa **faça**
5: *LRC* = p melhores elementos de C
6: selecione um elemento da *LRC* aleatoriamente e adicione a s
7: $C =$ candidatos reavaliados de acordo com uma função gulosa $g(c)$
8: **fim enquanto**
9: **retorne** s
10: **fim procedimento**

Recomenda-se que as soluções construídas passem por uma fase chamada de busca local. Esta fase representa a etapa de melhoria, pois investiga a vizinhança da solução construída até encontrar um mínimo local (RESENDE; RIBEIRO, 2003).

O processo de busca local é influenciado pela qualidade da solução construída pela heurística de construção, usualmente é importante iniciar o procedimento a partir de soluções que já possuam uma boa qualidade. Geralmente, soluções que foram construídas com um método guloso possuem qualidade melhor do que aquelas que são totalmente aleatórias. Isto significa que o procedimento de busca local consumirá possivelmente menos tempo para convergir para um ótimo local (LOURENÇO; MARTIN; STÜTZLE, 2003).

O Algoritmo 5, obtido de Resende e Ribeiro (2003), apresenta o pseudocódigo para o procedimento de busca local. Inicialmente, uma solução inicial e um modelo de vizinhança são passados por parâmetro (linha 1). O laço das linhas 2 - 5 realiza o processo de refinamento da solução inicial. Na linha 3, a vizinhança é examinada utilizando o critério de primeiro aprimorante ou melhor aprimorante. Na linha 4, a variável s que servirá de solução base para a próxima iteração é atualizada. A busca para se nenhuma solução melhor é encontrada. A melhor solução é retornada na linha 6.

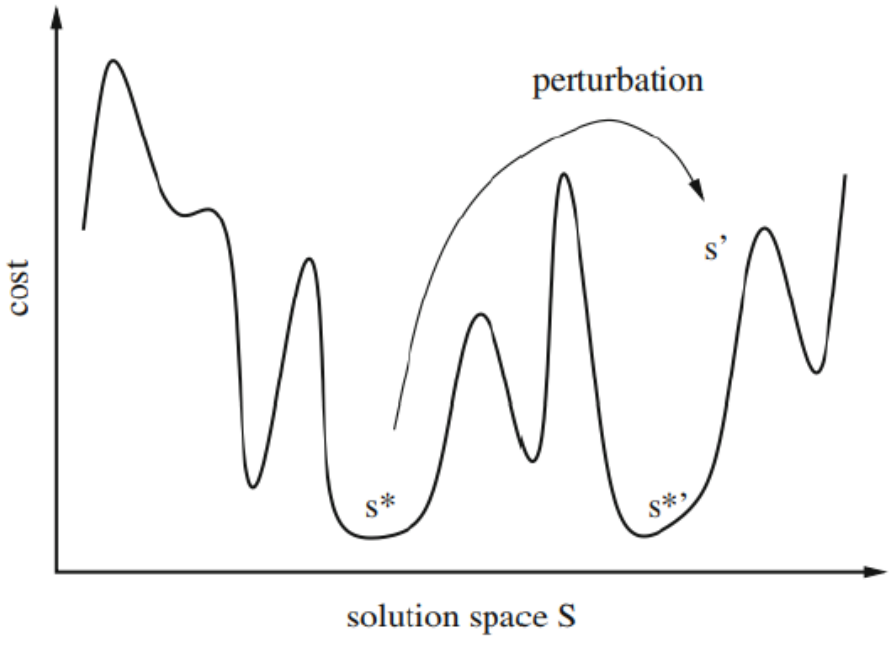
Algoritmo 4: Procedimento de busca local.

1: **procedimento** BuscaLocal(s, N)
2: **enquanto** s não é um ótimo local **faça**
3: Encontre $s' \in N(s)$, tal que $f(s') < f(s)$
4: $s = s'$
5: **fim enquanto**
6: **retorne** s
7: **fim procedimento**

4.1.3. Busca Local Iterada

A meta-heurística ILS (do inglês: *Iterated Local Search*) tem como ideia geral guardar subsequências de ótimos locais, além de utilizar perturbações em partes da solução para sair de ótimos locais. Uma perturbação é um procedimento que modifica parcialmente uma solução de forma aleatória, permitindo que uma nova vizinhança seja explorada pela busca local. A ILS possui três componentes principais: a construção de uma solução viável, a perturbação e a busca local. Inicialmente, uma solução é gerada e a busca local é utilizada. Em seguida, iterações sucessivas são aplicadas alternando perturbação e busca local (LOURENÇO; MARTIN; STÜTZLE, 2003). A Figura 5 ilustra este processo: partindo de um mínimo local $s*$, aplica-se uma perturbação probabilística que leva $s*$ *em* s'; após refinar s' com uma busca local, encontra-se um novo ótimo local s'*, que pode ser melhor que s*. Este processo pode ser realizado diversas vezes.

Figura 5: Representação do procedimento ILS.



Fonte: Lourenço, Martin e Stützle (2003).

A ideia da ILS é de construir uma caminhada aleatória/probabilística através do espaço de soluções existentes, que pode ou não ser finito. Cada "passo" é determinado por um procedimento de busca local que é capaz de encontrar um ótimo local na vizinhança de uma solução e uma função de perturbação que é aplicado às soluções para "saltar" para um novo ótimo local já otimizado LOURENÇO; MARTIN; STÜTZLE, 2003).

Um pseudocódigo da meta-heurística ILS é representada pelo Algoritmo 2, onde quatro procedimentos são especificados: (linha 2) um procedimento ou função para criar uma solução inicial, onde pode-se utilizar uma construção puramente aleatória ou uma heurística gulosa; (linhas 3 e 6) um procedimento de busca local para refinar as soluções de acordo com um ou mais modelos de vizinhança; (linha 5) uma função de perturbação, que aceita uma solução como entrada e aplica uma modificação parcial e aleatória; e (linha 7) um critério pré-definido de aceitação, que determina se uma solução s*' é aceita como um melhor ótimo local. Um exemplo de um critério de aceitação simples é verificar se o custo de s*' é o melhor até o momento (PENNA; SUBRAMANIAN; OCHI, 2013).

Algoritmo 5: Busca Local Iterada.

1: **procedimento** ILS
2: $s_0 \leftarrow$ ConstroiSolução
3: $s^* \leftarrow$ BuscaLocal(s_0)
4: **para** $it \leftarrow 1, maxIts$ **faça**
5: $s' =$ Perturbação(s^*)
6: $s^{*'} =$ BuscaLocal(s')
7: $s^* =$ CriterioDeAceitação(s^*, s')
8: **fim para**
9: **retorne** s^*
10: **fim procedimento**

O laço de repetição nas linhas 4 - 8 é executado até um número máximo de iterações, onde cada iteração faz uso de uma função de perturbação, da busca local e do critério de aceitação.

A função de perturbação na linha 5 é utilizada pela ILS em conjunto com o critério de aceitação na linha 7 para guiar as iterações pelo espaço de soluções possíveis. A força da perturbação (em inglês, *pertubation strength*) é medida pelo número ou percentual da solução que é alterada. Se esta força é muito alta, então o procedimento é semelhante a um reinício aleatório, pois grande parte da solução é modificada. Por outro lado, em perturbações sutis, o procedimento é capaz de gerar soluções que pertencem à outras vizinhanças, mas que preservam parte das características do ótimo local anterior (BESTEN;STÜTZLE; DORIGO, 2001).

O critério de aceitação na linha 7 pode utilizar diferentes estratégias para decidir se aceita ou não a solução $s^{*'}$ como um melhor ótimo local. Uma forma simples de implementar este critério é pela aceitação somente de ótimos locais com melhor função objetivo, como na função:

$$\text{Melhor}(s^*, s^{*'}) = \{s^{*'}, se\ f\left(s^{*'}\right) < f\ (s^*)\ s^*, caso\ contrário \quad (3.1)$$

Onde s^* é um ótimo local, possivelmente alcançado na iteração anterior, e $s^{*'}$ é um outro ótimo local alcançado a partir de uma perturbação aplicada à s^*, seguida de uma busca local.

A ILS necessita de uma heurística para gerar uma solução inicial viável. Nenhuma condição particular é imposta para essa geração inicial. Desse modo, a construção da solução inicial é vista como uma caixa-preta e pode ser realizada por uma heurística específica para o problema ou por um procedimento puramente aleatório, onde a primeira é preferível sobre a segunda (LOURENÇO; MARTIN; STÜTZLE, 2003).

4.2. Meta-heurística híbrida GRASP-ILS

Este capitulo apresenta o método híbrido desenvolvido neste trabalho consistindo da meta-heurística GRASP para construir as soluções e da meta-heurística ILS para refiná-las, realizando o papel da busca local.

Na fase construtiva do GRASP foi utilizada a heurística de Ranque descrita no capitulo 3.1.1. Esta escolha se deu porque essa heurística pode ser diretamente adaptada para ser usada como fase de construção do GRASP, incluindo o conceito de lista restrita de candidatos (LRC).

O pseudocódigo para a heurística de Ranque como fase construtiva do GRASP é descrito no Algoritmo 6. As linhas 2 - 7 calculam o ranque das arestas de acordo com a ocorrência nos caminhos mínimos (CM). A linha 8 cria a lista de candidatos C ordenando as vias bloqueadas em ordem decrescente de ranque. Na linha 10, para cada período de reparação, os candidatos da *LRC* são escolhidos de acordo com a equação 3.1. O laço das linhas 11 - 13 sorteia as arestas bloqueadas aleatoriamente da *LRC* e as alocam para as máquinas. O sorteio é realizado entre as arestas que são acessíveis. A instrução na linha 16 atualiza os caminhos mínimos quando uma aresta é totalmente reparada. Na linha 17, a aresta completamente reparada é adicionada à solução s, que é uma representação da solução e servirá como entrada para o procedimento de busca local. A linha 18 retorna a solução s e o valor da função objetivo.

Algoritmo 6: Fase construtiva do GRASP.

1: **procedimento** ConstróiSolução($G = (V, E), O, D, B, \alpha$)
2: **para todo** $i \in O$ **faça**
3: Calcule os caminhos mínimos (CM) para os vértices usando r_{ij} como custo
4: **para todo** $j \in D$ **faça**
5: Adicione 1 ao ranque de cada $[i, j] \in B$ no CM
6: **fim para**
7: **fim para**
8: $C =$ arestas bloqueadas em ordem decrescente de ranque
9: **para** $t \leftarrow 1$ até T **faça**
10: LRC $= p$ melhores elementos de C
11: **para toda** máquina disponível **faça**

```
12:          Aloque a máquina aleatoriamente para uma aresta acessível em LRC
13:      fim para
14:      se alguma [i, j] ∈ B tornou-se disponível então
15:          Atualize os CM utilizando a nova aresta disponível
16:          Adicione [i, j] à solução s
17:      fim se
18:  fim para
19:  retorne s e valor da Função Objetivo
20: fim procedimento
```

4.3. Busca local utilizando ILS

O método híbrido GRASP-ILS utiliza a meta-heurística ILS para refinar as soluções, fazendo o papel de busca local. Em cada iteração, o GRASP é responsável por construir uma nova solução de acordo com o critério guloso-aleatório da lista restrita de candidatos; depois, a solução construída é passada para a ILS.

O Algoritmo 7 apresenta o pseudocódigo do método híbrido GRASP-ILS. Inicialmente os dados do problema são passados por parâmetro. Na linha 2, a solução inicial é construída pelo GRASP; a solução recém construída passa então por uma busca local simples que faz leves alterações na solução; nas linhas 4-11, acontecem as iterações da ILS, composta de perturbação aleatória da solução (linha 5), aplicação da busca local simples na solução perturbada (linha 6) e utilização do critério de aceitação, que aceita a nova solução s'' se ela for melhor que s^*.

]

Algoritmo 7: Fase de Busca Local do GRASP-ILS.

```
 1: Procedimento GRASP-ILS()
 2: so ← Grasp(MaxIterGrasp, MaxIterFiltro, s);
 3: s ←BuscaLocal(so);
 4: enquanto i < maxIts faça
 6:     s' ← pertuba(s);
 6:     s'' ← BuscaLocal(so);
 7:     se f(s'') < f(s*) faça
 8:         s* ← s'';
 9:     fim se
10: fim enquanto
11: retorna s*;
12: fim procedimento
```

A finalização de cada iteração GRASP-ILS é determinada após a verificação de melhoria do resultado obtidos da metodologia ILS. A execução da hibridização chega ao fim, após ser atingido o máximo de iterações.

5. RESULTADOS COMPUTACIONAIS

Neste capítulo são apresentados os resultados dos experimentos computacionais realizados com a heurística de Ranque reproduzida de Sakuraba

et al. (2016), e a meta-heurística híbrida GRASP-ILS desenvolvida neste trabalho. Todos os algoritmos foram implementados em Python 3.7 e os experimentos foram realizados em um desktop Windows 7, 64*bits*, com processador Intel Core i7 a 3.4GHz de *clock* e 16GB de memória RAM. O objetivo dos experimentos é de avaliar o desempenho e eficiência de cada método, compará-los, e testá-los com um conjunto de instâncias descrito no capitulo 4.1.

5.1. Conjunto de instâncias

Este trabalho utiliza um conjunto de 8 instâncias consistindo em um subconjunto das instâncias publicadas em Sakuraba *et al*. (2016). As vias do grafo possuem largura de rua 2. Estas instâncias foram originadas de um grafo com 10 vértices, 20 arestas, 2 origens e 3 acampamentos. Cada instância possui uma configuração aleatória e possuem de 5 a 20 vias bloqueadas. Utilizamos $10 \leq T \leq 13$ (limite máximo de períodos) para todas as instâncias simuladas. Usamos a notação n[.]r[.]B[.] nas instâncias, onde n é o número de vértices, r é a quantidade de períodos necessários para reparar as vias bloqueadas e B é o número de vias bloqueadas.

A Tabela 2 apresenta os resultados computacionais entre a heurística de Ranque e o GRASP-ILS. Para cada instância, executou-se o GRASP-ILS 20 vezes e os valores da melhor solução encontradas são mostrados, ao lado do desvio padrão (σ) e da diferença relativa (em porcentagem). A coluna CPLEX indica os resultados ótimos conhecidos. Valores em negrito representam as melhores avaliações encontradas para uma instância.

Tabela 2: Resultados computacionais entre heurística de Ranque e GRASP-ILS para o conjunto de instâncias.

Instância	CPLEX	Ranque		GRASP-ILS		
		Melhor	Dif (%)	Média	Melhor	□
n10r40B5	**520**	600	-4,9%	**520**	**520**	0.0
n10r40B10	**574**	695	-17,4%	**574**	**574**	0.0
n10r40B15	**714**	813	-12,4%	**714**	**714**	0.0
n10r40B20	**2208**	**2208**	0,0%	**2208**	**2208**	0.0
n10r45B5	**638**	**638**	0,0%	**638**	**638**	0.0
n10r45B10	**712**	791	-10,8%	**712**	**712**	0.0
n10r45B15	**1312**	1448	-9,4%	**1312**	**1312**	0.0
n10r45B20	**2686**	**2686**	0,0%	**2686**	**2686**	0.0

A avaliação da função objetivo do WSP envolve uma soma ponderada das distâncias entre origens e acampamentos. É possível ver que o GRASP-ILS atingiu o resultado ótimo para todas as 8 instâncias. O desvio padrão é 0, indicando que todas as execuções atingiram o resultado ótimo. Isso possivelmente ocorreu porque as instâncias possuem tamanhos pequenos e/ou são de dificuldade baixa e, por ser uma meta-heurística híbrida, o GRASP-ILS procura por soluções mais diversificadas do que a heurística de Ranque. A heurística de Ranque foi capaz de atingir o resultado ótimo em 3 instâncias; para os resultados não-ótimos, teve proximidade entre 4,9% e 17,4%.

5.2. Simulação de um cenário real

Utilizamos o GRASP-ILS para testar um cenário de simulação real. A Figura 6 apresenta o cenário da instância n10r40B15, onde são mostrados os estados inicial e final da rede danificada. Os círculos e os losangos indicam as origens (nós 1 e 8) e os acampamentos (nós 4, 5 e 7), respectivamente. As ruas bloqueadas por escombros são representadas por arestas pontilhadas. Na Figura 6), 15 arestas bloqueadas diminuem a acessibilidade da rede, fazendo com que dois acampamentos fiquem inacessíveis, enquanto a distância para o acampamento 7 é de 13. Após a reparação das vias bloqueadas na Figura 7), a acessibilidade para

os acampamentos 4 e 5 é restaurada, enquanto os caminhos mínimos (arestas pretas) são reduzidos para 3, 3 e 5.

Figura 6:Estado inicial do grafo

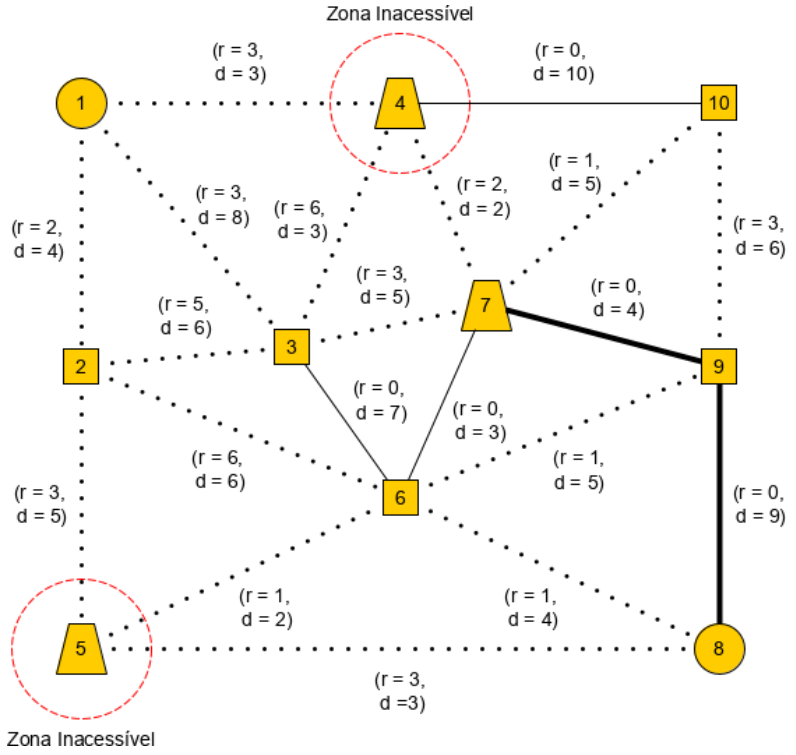
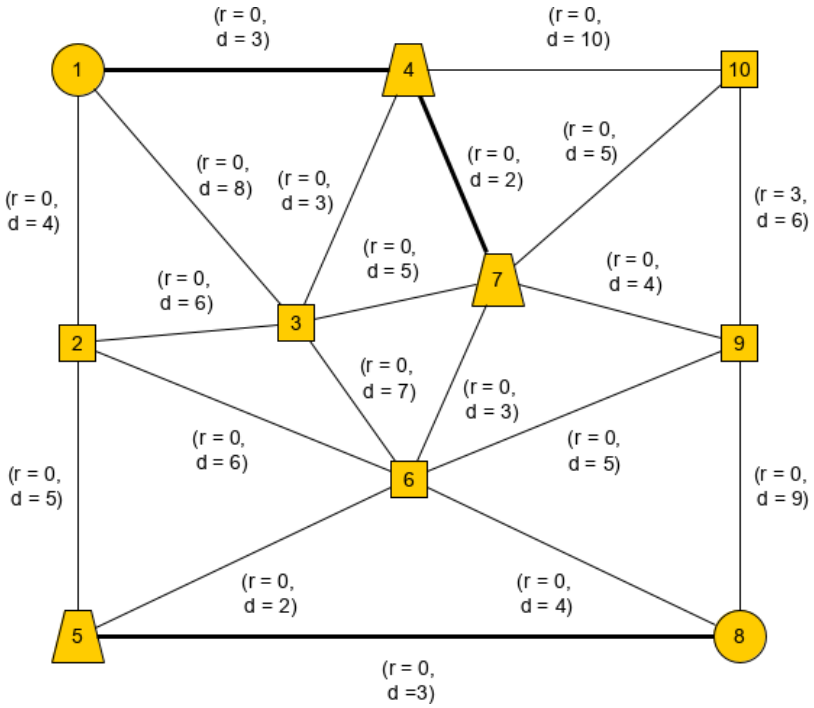


Figura 7: Estado final após a reparação.



6. CONSIDERAÇÕES FINAIS

Esta monografia tratou do problema de otimização associado à reparação das vias bloqueadas em uma rede viária danificada por um desastre natural; e também apresenta a continuidade do projeto PIBIC-UERN desenvolvido entre agosto de 2018 a agosto de 2019 onde foi inicialmente desenvolvido um método GRASP tradicional para o problema em apreço.

A escolha da meta-heurística GRASP e o ILS se deram pelo fato de estas possuírem características que casavam perfeitamente com as características do problema apresentado. Por exemplo, a construção da solução que o GRASP apresenta e a diversificação do ILS.

Uma grande dificuldade para criar algoritmos para o problema é que a avaliação das soluções é realizada por um algoritmo de caminhos mínimos que adiciona pelo menos O(nlgn), devido a característica dinâmica do problema que faz que com a rede viária mude de estado conforme as vias são reparadas. É preciso sempre garantir que exista uma matriz de caminhos mínimos com as distâncias atualizadas e em sincronia com o estado atual do grafo. Uma falha neste requisito pode gerar situações inválidas que não poderiam acontecer de acordo com as restrições do problema.

A meta-heurística Busca Local Iterada (ILS) foi desenvolvida para funcionar como a busca local para o GRASP tradicional. A ILS foi criada e adicionada ao GRASP para contornar os ótimos locais com a ajuda de uma função de perturbação que modifica até 30% da solução. O método híbrido desenvolvido utiliza o GRASP para construir uma solução de forma gulosa e aleatória e a ILS como busca local para explorar o espaço de soluções.

A principal contribuição deste trabalho pode ser resumida como: (i) uma meta-heurística híbrida que gera soluções de melhor qualidade do que a heurística de Ranque da literatura; o método híbrido desenvolvido foi capaz de encontrar os melhores resultados quando comparados com a heurística de Ranque. Os resultados alcançados pelo GRASP-ILS mostram-se melhores em 5 das 8

instâncias. Nas 3 instâncias restantes, tanto o GRASP-ILS quanto a heurística de Ranque encontraram soluções com o mesmo valor da função objetivo. As 8 instâncias testadas são de tamanho pequeno (10 nós e 20 arestas), sendo necessário testes com instâncias maiores para que seja possível avaliar melhor o desempenho de tempo do GRASP-ILS.

Existem diversas perspectivas futuras para este trabalho. Por exemplo, testar instâncias maiores, novas vizinhanças podem ser desenvolvidas para melhorar ainda mais os resultados, fazer uma comparação dos resultados alcançados pelas meta-heurísticas separadas, ou seja os resultados do GRASP, da ILS e comparar com o GRASP-ILS, utilizando a mesma instância. Além disto, a questão da escala real pode ser analisada. Os grafos reais de cidades possuem milhares de nós e de arestas. Isto necessita uma otimização das estruturas de dados e uma atenção particular à complexidade de implementação dos algoritmos, esse foi o ponto que levou a escolha da heurística de Ranque como fase constritiva para o GRASP. Enfim, o problema pode ainda ser estendido para integrar outras características como o roteamento da transferência de destroços até as áreas de descarga, a integração de drones para coordenar as operações, outra possibilidade é aplicar esta meta-heurística hibrida em cenários de enchentes que deixam vias bloqueadas e intransitáveis, tendo em vista que o Brasil não tem desastres do tipo tsunamis e terremotos.

REFERÊNCIAS

BESTEN, M.; STÜTZLE, T.; DORIGO, M. Design of iterated local search algorithms. In: BOERS, E. J. W. (Ed.). Applications of Evolutionary Computing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. p. 441–451.

BINATO, S. et al. A grasp for job shop scheduling. AT&T Labs Research Technical Report: 00.6.1, 2000.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv., ACM, New York, NY, USA, v. 35, n. 3, p. 268–308, set. 2003. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/937503.937505>.

BRILHAM, R. Lessons from the haiti earthquake. Nature, Nature Publishing Group, v. 463, 2010.

CUNHA, A. G.; TAKAHASHI, R.; ANTUNES, C. H. Manual da Computação Evolutiva. University of Chicago: Coimbra: Imprensa da Universidade de Coimbra, 2012.

DORIGO, M.; CARO, G. D. New ideas in optimization. In: CORNE, D. et al. (Ed.). Maidenhead, UK, England: McGraw-Hill Ltd., UK, 1999. cap. The Ant Colony Optimization Meta-heuristic, p. 11–32. ISBN 0-07-709506-5.

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. Grasp: An annotated bibliography. In: . [S.l.: s.n.], 2002.

FESTA, P.; RESENDE, M. G. C. GRASP: basic components and enhancements. Telecommunication Systems, v. 46, n. 3, p. 253–271, 2011.

Firmino, Cedma Ranielly, *Meta-Heurística Para o Problema de Localização de Seções Eleitorais e Alocação de Eleitores.* 2017. 60*f. Dissertação* – Universidade do Estado do Rio Grande do Norte, Mossoró, 2017.

FRANCISCO, Wagner de Cerqueria e. "*O Terremoto no Haiti* "; *Brasil Escola*. Disponível em: https://brasilescola.uol.com.br/geografia/o-terremoto-no-haiti.htm. Acesso em 13 de setembro de 2019.

GENDREAU, M.; POTVIN, J.-Y. *handbook of metaheuristics*. 2. ed. New York Dordrecht Heidelberg London: Springer, 2010.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research, v. 13, n. 5, p. 533 – 549, 1986. ISSN 0305-0548. Applications of Integer Programming.

HOLLAND, J. H. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. Cambridge, MA, USA: MIT Press, 1992. ISBN 0262082136.

https://www.passeidireto.com/arquivo/47351145/matematica-computacional-aula-impressa-4

http://dsc.inf.furb.br/tcc/index.php?cd=9&tcc=1574

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

LIMA, F. C. J. 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, 2009.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Handbook of metaheuristics. In:. [S.l.]: Kluwer Academic Publisher, 2003. (International Series in Operations Research & Management Science), cap. Iterated Local Search, p. 321–353.

LOURENçO, H. R.; MARTIN, O. C.; STüTZLE, T. Iterated local search. In: Handbook of Metaheuristics, volume 57 of International Series in Operations Research and Management Science. [S.l.]: Kluwer Academic Publishers, 2002. p. 321–353.

MARTÍ, R. et al. Multiobjective GRASP with Path Relinking. European Journal of Operational Research, v. 240, n. 1, p. 54 – 71, 2015.

MOUZON, G.; YILDIRIM, M. B. A framework to minimize total energy consumption and total tardiness on a single machine. International Journal of Sustainable Engineering, 2008.

Neto, Antonio Alves, *Meta-Heurística de Otimização Tradicionais e Híbridas Utilizadas Para Construção de Comitês de Classificação. 2016. 196f. Tese doutorado* – Universidade do Estado do Rio Grande do Norte, Natal, 2016.

PENNA, P. H. V.; SUBRAMANIAN, A.; OCHI, L. S. An Iterated Local Search heuristic for the heterogeneous fleet vehicle routing problem. Journal of Heuristics, v. 19, n. 2, p. 201–232, 2013.

RAIDL, G. R. A unified view on hybrid metaheuristics. In: Hybrid Metaheuristics, Third International Workshop, HM 2006, Gran Canaria, Spain, October 13-15, 2006, Proceedings. [s.n.], 2006. p. 1–12. Disponível em: <https://doi.org/10.1007/11890584_1>.

RESENDE, M. G. C.; FESTA, P. An annotated bibliography of grasppart i: Algorithms. AT&T Labs Research Technical Report, 2008.

RESENDE, M. G. C.; RIBEIRO, C. C. Greedy randomized adaptive search procedures. In: . Handbook of Metaheuristics. [S.l.]: Kluwer Academic Publisher, 2003. cap. 8, p. 219–249.

SAKURABA, C. S. et al. Road network emergency accessibility planning after a major earthquake. EURO Journal on Computational Optimization, v. 4, n. 3, p. 381–402, 2016.

SAKURABA, C. S.; SANTOS, A. C.; PRINS, C. Work-troop scheduling for road network accessibility after a major earthquake. Electronic Notes in Discrete Mathematics, v. 52, p. 317 – 324, 2016.

SHANKAR, G. Post Disaster Management, Poverty and Food. mar. 2011. Mar., 2011. Disponível em: <https://earthzine.org/post-disaster-management-poverty-and-food/>. Acesso em Outubro 20, 2019.

SOUZA, João André, *Uma Api Rest Para Consumo de Dados Georreferenciados Utilizando ArangoDB*. *2018. 68f. Monografia* – Universidade do Estado do Rio Grande do Norte, Mossoró, 2018.