Meta-Heurística NSGA-II Aplicada ao Problema de Caminhos Dissimilares

Tomás Loureiro Gomes[1], **Ana Paula Canal**[1]

[1]Ciência da Computação – Universidade Franciscana (UFN)
Caixa Postal 151 – 97.010-032 – Santa Maria – RS – Brasil

tomas.gomes@unifra.edu.br, apc@unifra.br

Abstract. *The Path Dissimilarity Problem is a generalization of the Shortest Path Problem and it is known as a bi-objective optimization problem. It must find paths in a graph with maximum dissimilarity between arcs and a minimum distance between nodes. The solution for problems with several objectives should consider Pareto dominance. Therefore, it was used the algorithm NSGA-II developed in Python to solve the Path Dissimilarity Problem. As a result, the algorithm is described and Pareto Frontier approximation is illustrated in charts with variations in the number of generations and population.*

Resumo. *O Problema de Caminhos Dissimilares é uma generalização do Problema do Caminho Mínimo e é conhecido por ser um problema de otimização bi-objetivo. Deve-se encontrar caminhos em um grafo com a máxima dissimilaridade entre arcos e a mínima distância entre os nodos. A solução para problemas com diversos objetivos deve considerar a dominância de Pareto. Portanto, foi utilizado o algoritmo NSGA-II desenvolvido em Python para resolver o Problema de Caminhos Dissimilares. Como resultado, o algoritmo é descrito e a aproximação da fronteira de Pareto é ilustrada em gráficos com variações na quantidade de gerações e população.*

1. Introdução

Nas últimas décadas, o interesse pela logística aumentou tanto para academia quanto para a indústria e um dos principais motivos é a evolução do modo como as pessoas consomem. O desenvolvimento da gestão de sistemas logísticos ocorre pelo avanço da comunicação e do transporte. O objetivo da logística é gerir o fluxo de produtos partindo de uma origem até seu destino, obedecendo alguns critérios [Labadie et al. 2016].

Segundo o mesmo autor, com o avanço da globalização, alguns problemas de logística começaram a ocorrer nas cidades, causado principalmente pelo trânsito congestionado. Sistemas modernos de entregas de produtos passaram a otimizar este processo. Deu-se início ao campo de estudo conhecido como Otimização Combinatória (OC), que tem como objetivo tentar encontrar soluções ótimas de problemas, dentro das soluções aceitáveis encontradas. Algoritmos são usados para realizar a busca no espaço de soluções, porém não se consegue encontrar e provar a otimalidade em tempo polinomial[1]. Problemas como esse são conhecidos como NP-difíceis (*NP-hard*).

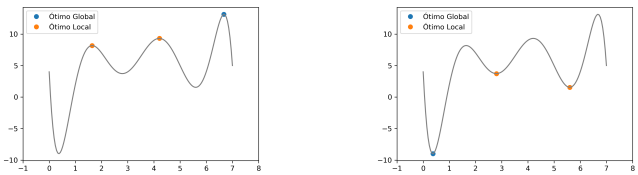
Um problema de otimização, mais precisamente um Problema do Caminho Mínimo (*Shortest Path Problem* ou SPP) pode ser exemplificado como um carro-forte, que

[1]Segundo [Cobham 1965], tempo polinomial é sinônimo de tempo hábil ou tratável.

sai de um ponto de origem e chega ao destino final para transportar dinheiro, buscando o menor caminho. O mesmo exemplo pode ser analisado como um Problema de Caminhos Dissimilares (*Path Dissimilarity Problem* ou PDP), no qual a empresa muda a estratégia, e em função de segurança ou por obstrução do caminho, busca-se alternativas. O cálculo realizado para encontrar novos caminhos dissimilares é mais um critério a ser otimizado. Estes dois problemas são apenas dois exemplos de uma grande lista de problemas de OC.

Em problemas de OC, formula-se um modelo matemático composto de pelo menos uma função objetivo[2], variáveis de decisão[3] e restrições. As soluções possíveis para o problema são determinadas através das variáveis de decisão, que podem receber qualquer tipo de valor e são atribuídas para a função objetivo a fim de encontrar a solução sem violar as restrições empregadas [Labadie et al. 2016].

Otimização é a tarefa de tentar encontrar soluções ótimas, normalmente, procurando-se o ótimo global, que é a melhor solução do espaço de soluções. Porém, este procedimento pode levar algum tempo, já que problemas de otimização possuem restrições, instabilidades e um elevado número de ótimos locais[4]. Em problemas de OC, utiliza-se métodos exatos ou aproximados. Métodos exatos são conhecidos por conseguirem localizar ótimos globais em problemas com baixo esforço computacional, já os métodos aproximados são heurísticas ou meta-heurísticas, usadas em problemas complexos, os quais computadores atuais são incapazes de resolvê-los em tempo tratável. Essas estratégias tentam encontrar ótimos locais e chegar o mais próximo de ótimos globais em tempo hábil. Heurísticas são conhecidas como algoritmos que fornecem soluções aceitáveis para um determinado problema, porém muitas vezes estes algoritmos ficam presos em ótimos locais. Meta-heurísticas são conhecidas como algoritmos que fornecem soluções aceitáveis muito próximas do ótimo, pois possuem estratégias para escapar de ótimos locais prematuros [Labadie et al. 2016]. A Figura 1(a) exemplifica os ótimos locais e globais no espaço de soluções de uma função objetivo, sendo esta, de maximização. Entretanto, a Figura 1(b) exemplifica o mesmo problema com uma visão diferente, a de minimização de uma função objetivo.



(a) Maximização da função (b) Minimização da função

Figura 1. Exemplo de uma função objetivo

Se modelados corretamente, os Algoritmos Genéticos (AG) podem resolver a maior parte dos problemas de OC [Kramer 2017]. Seguindo a afirmação de Kramer, foi utilizado neste estudo a meta-heurística conhecida como *Non-dominated Sorting Genetic Algorithm II* (NSGA-II), um Algoritmo Genético (AG) utilizado para problemas com

[2]O principal objetivo do problema, pode ser de maximização ou de minimização.
[3]Incógnitas a serem determinadas pela solução do modelo.
[4]Soluções que não são o ótimo global, porém, representam bons resultados.

mais de um objetivo. Apesar disso, poucos trabalhos foram publicados sobre a utilização de AG em PDP. É uma estratégia conhecida de otimização baseada nos princípios de seleção natural e no mecanismo de evolução da natureza, podendo trazer bons resultados para o experimento proposto neste trabalho.

1.1. Objetivo Geral

Aplicar a meta-heurística *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) no Problema de Caminhos Dissimilares (PDP).

1.2. Objetivos Específicos

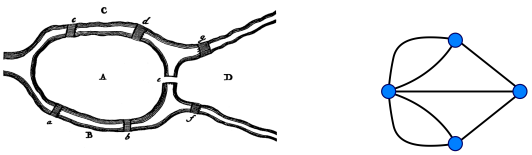
- Implementar o Algoritmo Genético (AG) multi-objetivo NSGA-II;
- Verificar o desempenho do algoritmo proposto no PDP;

1.3. Estrutura do Trabalho

O Trabalho está estruturado em: Referencial Teórico com os temas Problema do Caminho Mínimo, Problema de Caminhos Dissimilares e Algoritmos Genéticos; Trabalhos Relacionados com uma comparação dos trabalhos correlatos com o presente trabalho; Metodologia onde está descrito os passos seguidos e quais as ferramentas que foram utilizadas para alcançar os objetivos; Desenvolvimento com a descrição detalhada da implementação; Resultados onde é feita a avaliação dos gráficos obtidos e dos tempos de execução; Por último, Conclusões e Trabalhos Futuros.

2. Referencial Teórico

Leonhard Euler, em 1736, associou os termos vértices (porções de terra) e arestas (pontes) a um grafo no problema das Sete Pontes de Königsberg. A partir deste feito, definiu-se um grafo como sendo um conjunto $G = (V, A)$, onde V são os vértices (nós) e A as arestas (arcos) [Dhein et al. 2016]. A Figura 2(a) mostra o problema das Sete Pontes, representa-se as pontes (a, b, c, d, e, f, g) em letras minúsculas e as porções de terra (A, B, C, D) com letras maiúsculas. A Figura 2(b) mostra a representação do problema através de um grafo, onde os vértices são os pontos em azul (porções de terra) e as arestas os caminhos que as conectam (pontes).



(a) Desenho feito por Euler (b) Grafo

Figura 2. Representação do Problema das Sete Pontes de Königsberg

No caso do SPP, deve-se escolher um vértice de origem e um de destino para encontrar o menor caminho possível entre eles. Na Figura 3 é representado o mapa da Romênia como um grafo, as cidades como os vértices e os arcos como as estradas que ligam os pontos com custos de distância. O mapa é amplamente utilizado para representação de busca do caminho mínimo e, usualmente, o ponto inicial é Arad e o ponto final

Bucharest. Existem várias formas de chegar até Bucharest partindo de Arad. SPP busca encontrar o menor custo possível para sair do ponto inicial e chegar no ponto final. Neste mapa, partindo de Arad até chegar a Bucharest, o menor caminho possível é Arad - Sibiu - Rimnicu Vilcea - Pitesti - Bucharest com custo de 418. Existe um outro problema de otimização conhecido como PDP, que será abordado na próxima seção.

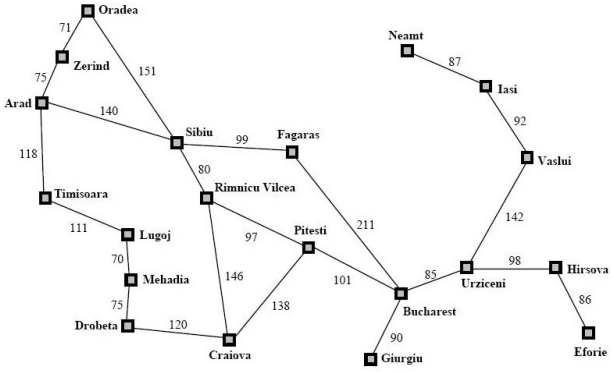


Figura 3. Representação do mapa da Romênia

2.1. Problema de Caminhos Dissimilares

Segundo [Corberán et al. 2012], Problema de Caminhos Dissimilares (PDP) é um problema de otimização bi-objetivo, onde se busca gerar um numero k de caminhos da origem até um destino com a menor distância possível e a máxima dissimilaridade. Encontrar diferentes caminhos em um grafo é um problema clássico de OC. Um dos mais conhecidos é o Problema dos k-Caminhos Mínimos (*k-Shortest Path Problem* ou k-SPP), onde se deve encontrar um conjunto de caminhos mais curtos entre dois vértices, ou seja, encontra-se o mais curto e outros $k-1$ em ordem crescente de distância. Porém, muitas vezes estes caminhos compartilham os mesmos vértices, e no caso do PDP o que se deseja é encontrar k-Caminhos Mínimos (*k-Shortest Paths* ou k-SPs), porém, com maior índice de dissimilaridade nos caminhos gerados.

O Problema de Caminhos Dissimilares (PDP) faz parte de problemas NP-difíceis, ou seja, o tempo de execução cresce exponencialmente de acordo com a quantidade de instâncias utilizadas. [Corberán et al. 2012] realizou um estado da arte sobre o PDP e neste, encontrou, na resolução deste problema, os métodos presentes na Tabela 1.

Tabela 1. Métodos que foram utilizados no PDP

Métodos	Artigos
Iterative Penalty Method (ITS)	Johnson, P., Joy, D., and Clarke, D. (1992). HIGHWAY 3.01, An Enhancement Routing Model: Program, Description, Methodology and Revised User's Manual, Arbeitsbericht, Oak Ridge National Laboratories, Washington, DC.
Gateway Shortest Path (GTS)	Lombard, K. and Church, R. (1993). The Gateway Shortest Path Problem: Generating Alternative Routes for a Corridor Location Problem. Geographical systems, 1(1):25-45.
Minimax (MM)	Kuby, M., Zhongyi, X., and Xiaodong, X. (1997). A Minimax Method for Finding the k Best "Differentiated"Paths. Geographical Analysis, 29(4):298-313.
IPM for P-dispersion Problem (IPMpD)	Akgün, V., Erkut, E., and Batta, R. (2000). On Finding Dissimilar Paths. European Journal of Operational Research, 121(2):232-246.
Multicriteria Shortest Path Algorithm (MSPA)	Dell'Olmo, P., Gentili, M., and Scozzari, A. (2005). On Finding Dissimilar Pareto-Optimal Paths. European Journal of Operational Research, 162(1):70-82.
Greedy Randomized Adaptive Search Procedure (GRASP)	Martí, R., Velarde, J. L. G., and Duarte, A. (2009). Heuristics for the Bi-objective Path Dissimilarity Problem. Computers & Operations Research, 36(11):2905-2912.
Multiobjective GRASP with Path Relinking (GRASP+PR)	Martí, R., Campos, V., Resende, M. G., and Duarte, A. (2015). Multiobjective GRASP with Path Relinking. European Journal of Operational Research, 240(1):54-71.

Segundo [Corberán et al. 2012], define-se, de modo geral, o PDP, como um grafo $G = (V, A)$ sendo V um conjunto de vértices e A um conjunto de arestas com um custo C_{ij} para $(i, j) \in A$. Este custo, normalmente, é calculado como a distância euclidiana dos vértices. Um par de vértices origem e destino origina todo o conjunto de caminhos possíveis no grafo, ou seja, $P(o, d)$. Uma possível solução para o PDP é um conjunto $S \subseteq P(o, d)$ onde $|S| = p$ e $p > 1$. A partir de uma solução $S = \{P_1, P_2, P_p\}$, pode-se definir seu valor $f_1(S)$ como a média dos custos dos caminhos em S:

$$f_1(S) = \frac{\sum_{t=1}^{p} c(P_t)}{p} \quad \text{onde} \quad c(P_t) = \sum_{(i,j) \in Pt} c_{ij} \tag{1}$$

Define-se também o valor da dissimilaridade $f_2(S)$ como a dissimilaridade média entre dois pares de caminhos distintos $\binom{p}{2}$ em S:

$$f_2(S) = \frac{\sum_{i=1}^{p-1} \sum_{j=i+1}^{p} dis(P_i, P_j)}{\binom{p}{2}} \tag{2}$$

Como citado em [Martí et al. 2009], a dissimilaridade $dis(P_i, P_j)$ entre dois caminhos P_i e P_j é calculado através da média das distâncias de cada vértice em P_i até P_j, somado as médias das distâncias de cada vértice em P_j até P_i. Portanto, o cálculo da dissimilaridade é definido como:

$$dis(P_i, P_j) = \frac{1}{2} \left(\frac{\sum_{v_i \in P_i} \delta(v_i, P_j)}{|P_i|} + \frac{\sum_{u_j \in P_j} \delta(u_j, P_i)}{|P_j|} \right) \tag{3}$$

onde:

$P_i = \{v_1, v_2, ..., v_n\}$ e $P_j = \{u_1, u_2, ..., u_m\}$ são os caminhos.

$\delta(u, v)$ calcula a distância euclidiana entre dois pontos em um plano cartesiano, como mostra a Equação 4. A partir desta distância, é definido uma distância entre um vértice $v \in V$ (conjunto de vértices) para um caminho $P_i = v_1, v_2, ..., v_n$ como $\delta(v, P_i) = \min_{v_j \in P_i} \delta(v, v_j)$.

$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2} \tag{4}$$

Nota-se que as funções objetivos são $f_1(S)$ e $f_2(S)$, as quais, deseja-se, respectivamente, minimizar o custo e maximizar a dissimilaridade. Uma solução S^* do PDP é eficiente se não houver nenhuma outra solução $S \subseteq P(o, d)$ como $f_1(S^*) > f_1(S)$ e $f_2(S^*) < f_2(S)$. Portanto, S^* é eficiente se não tiver solução em $P(o, d)$ melhor que S^* referente a ambos objetivos.

A maioria das técnicas de programação multi-objetiva tenta encontrar o conjunto de pontos eficientes de um determinado problema, pertencentes a fronteira de Pareto, que será caracterizada na Seção 2.2.1. Na resolução do problema, o objetivo principal é encontrar uma aproximação dessa fronteira, para isso são utilizadas heurísticas e meta-heurísticas e, nesta última classe, encontram-se os AG.

2.2. Algoritmos Genéticos

Abordagem de busca meta-heurística amplamente utilizada em problemas de otimização, o Algoritmo Genético (AG) se baseia no conceito de Evolução, o qual foi proposto em [Darwin 1859]. Nesta obra, se explica o desenvolvimento biológico das espécies com seleção para o acasalamento e a sobrevivência do mais apto. Acabou-se desenvolvendo uma representação conhecida como o Ácido Desoxirribonucleico (DNA). O DNA é o código genético de cada ser vivo, está presente no cromossomo e é a base para o processo evolutivo [Kramer 2017].

Um AG foi descrito pela primeira vez por John Holland na década de 1970 tendo como principal objetivo entender o fenômeno de adaptação, o mesmo que ocorre na natureza, e tentar representá-lo em sistemas computacionais. Em [Holland 1975] foi apresentado um AG como uma abstração da evolução biológica e se demonstrou o processo de evolução de uma população inicial (conjunto de cromossomos ou indivíduos - Figura 4), normalmente representadas por *string* de *bits*, as quais representam as possíveis soluções para o problema, até a geração de uma nova população, utilizando-se de operadores genéticos como seleção, recombinação (*crossover*) e mutação, a fim de tornar a nova população com maior aptidão [Mitchell 1995]. Segundo a mesma autora, cada cromossomo é formado por bits (genes ou alelos), que podem receber os valores 0 ou 1 como na Figura 4. Entretanto na área de OC, pode-se encontrar também cromossomos com valores inteiros, valores de pontos-flutuantes e caracteres.

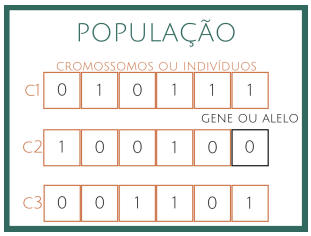


Figura 4. Representação de uma população inicial

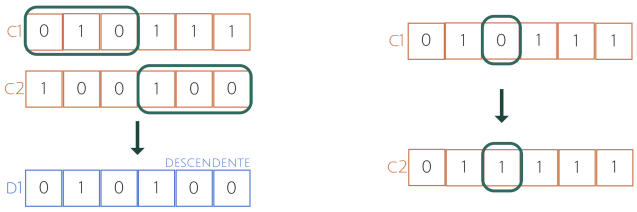
Na etapa de seleção são escolhidos aqueles cromossomos que serão utilizados na reprodução (pais). Uma das técnicas de seleção é a por *Mating Pool*, onde são combinadas a população pai e a descendente selecionando as melhores soluções de acordo com o cálculo de aptidão (*fitness*), que muitas vezes é a função objetivo do problema. Calcula-se o valor de aptidão para cada indivíduo e se separa os indivíduos de acordo com seu fitness. Esta técnica se assemelha a seleção por torneio, onde são selecionados N indivíduos para competição, o mais apto ganha e é elegido para se juntar ao grupo dos mais aptos. Este processo de seleção é repetido até todos os indivíduos mais aptos terem sido escolhidos. Na etapa de *crossover* são trocadas subpartes de dois cromossomos para gerar um descendente, imitando a recombinação biológica entre dois organismos vivos. A etapa de mutação ocorre quando se altera o valor do alelo randomicamente em uma determinada posição do cromossomo [Deb et al. 2002, Mitchell 1995].

O Algoritmo 1 mostra a estrutura de um AG e exemplifica as etapas que ele necessita para encontrar soluções adequadas. O primeiro passo é inicializar (representar) o conjunto de soluções (população) com valores aleatórios. Para representações do tipo *string* de *bits* é aconselhada uma combinação randômica de zeros e uns. Por exemplo, um

cromossomo randômico com os valores 1001001001 é uma string de bits típica de tamanho 10. O principal ciclo (*loop*) de um AG gera novas soluções descendentes com seleção, recombinação e mutação até que a nova população esteja completa [Kramer 2017]. A Figura 5 exemplifica as etapas de recombinação (*crossover*) e mutação.

Algoritmo 1: AG (ADAPTADO DE [KRAMER 2017, MITCHELL 1995])

```
1 início
2     inicialização randômica de uma população (N indivíduos)
3     repita
4         cálculo do fitness dos indivíduos
5         gera indivíduos descendentes usando seleção, crossover e mutação
6         combinação da população atual com a população descendente
7     até condição de parada alcançada;
8 fim
```



(a) *crossover* (b) mutação

Figura 5. Operadores Genéticos

2.2.1. *Non-dominated Sorting Genetic Algorithm II* (NSGA-II)

Para resolução do PDP, um problema bi-objetivo, procura-se minimizar o custo (distância do caminho) e maximizar a dissimilaridade dos caminhos gerados entre dois vértices. Portanto, o método utilizado para resolução do problema foi o algoritmo NSGA-II. Este algoritmo usa em sua implementação o critério de dominância a fim de estabelecer o conceito de qualidade no seu espaço de soluções [Marinho 2009].

Segundo [Marinho 2009], algoritmos bi-objetivos ou multi-objetivos levam em consideração a otimização de mais de um objetivo, os quais geralmente são conflitantes, pois normalmente não se pode maximizar ou minimizar todos eles juntos. Em outras palavras, quando se encontra uma solução, ela pode ser melhor em um objetivo do que em outro, tornando impossível afirmar qual solução é superior à outra, ou seja, nenhum dos objetivos domina o outro. As soluções não dominadas podem ser traçadas em um gráfico através dos objetivos, formando a fronteira de Pareto, que são conhecidas como as melhores soluções no espaço viável de soluções. A Figura 6 representa um problema com duas funções objetivo ($f1(x)$ e $f2(x)$) de minimização, onde a solução $A1$ é a melhor solução para minimizar a função $f1(x)$ e a $A2$ é a melhor solução para minimizar $f2(x)$.

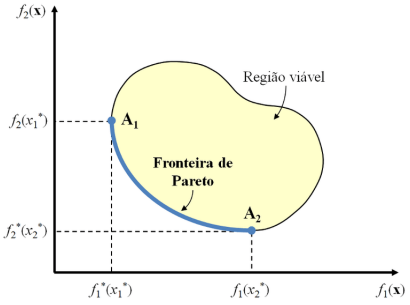


Figura 6. Representação da fronteira de Pareto de um problema bi-objetivo

Para tornar mais justa a comparação entre os objetivos, foi definido o conceito de dominância no algoritmo NSGA-II em [Deb et al. 2002]. O critério de dominância afirma que um indivíduo p não pode possuir nenhum objetivo com menos qualidade do que q, sendo p e q indivíduos pertencentes a uma mesma população P. Após o término desta classificação para descobrir quem domina e quem é dominado, todos os indivíduos são classificados em N fronteiras, sendo a primeira fronteira de todas as soluções não dominadas (aproximação da fronteira de Pareto) e as $N+1$ fronteiras com as demais soluções. Os indivíduos localizados na primeira fronteira são as melhores soluções daquela geração, enquanto que na última fronteira são localizadas as piores [Marinho 2009]. Através deste conceito de dominância, que é a classificação dos indivíduos em diferentes categorias de qualidade, agrega-se o conceito de elitismo, que distingue indivíduos mais aptos e menos aptos. Com isso, permite-se priorizar os indivíduos superiores, o que garante melhores soluções [Marinho 2009].

Segundo o mesmo autor, o algoritmo NSGA-II possui dois mecanismos importantes no processo de seleção, que o diferencia de um AG convencional: *Fast Non-Dominated Sorting* (FNDS) e *Crowding Distance* (CD). FNDS é a etapa descrita anteriormente, onde são classificadas as soluções em diferentes fronteiras de acordo com o critério de dominância. Na Figura 7 está a representação de duas funções objetivo de minimização z_1 e z_2, sendo (A, B, C, D, E, F, G, H, I, J, K, L, M, N) as soluções possíveis na primeira geração. A técnica FNDS classifica as fronteiras em F_1, F_2, F_3 e F_4, sendo a F_1 a fronteira de todas as soluções não dominadas e a F_4 as piores soluções da geração.

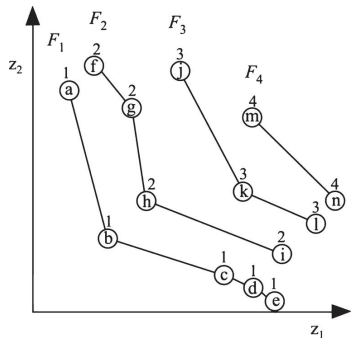


Figura 7. Representação da técnica *Fast Non-Dominated Sorting* (FNDS)

Crowding Distance (CD) é um operador de diversidade, que ordena cada indivíduo de acordo com sua distância em relação aos pontos vizinhos da mesma fronteira.

Quanto mais distante o ponto central está dos seus vizinhos, maior a probabilidade de serem escolhidos. Com a utilização desta técnica é favorecido soluções em áreas pouco povoadas, ou seja, soluções que não tem vizinhos próximos. Nas extremidades não há vizinho de um dos lados (infinito), então as soluções nas pontas das fronteiras tem vantagem. Na Figura 6 os indivíduos A1 e A2 têm maior probabilidade de serem escolhidos. É apresentado no Algoritmo 2 as etapas do NSGA-II.

Algoritmo 2: NSGA-II (adaptado de [Deb et al. 2002])

```
1  início
2      inicialização randômica de uma população (N indivíduos)
3      repita
4          cálculo do fitness dos indivíduos
5          classifica a população em N fronteiras (FNDS)
6          calcula a distância de todos indivíduos (CD)
7          combinação da população atual com a população descendente
8          seleção dos mais aptos
9      até condição de parada alcançada;
10 fim
```

3. Trabalhos Relacionados

Nesta seção foram selecionados diferentes trabalhos que utilizaram o algoritmo NSGA-II ou o PDP. Foi realizada uma correlação destes com o presente trabalho. O primeiro, "*A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II*"apresenta a técnica NSGA-II, servindo como base para a implementação deste trabalho. O segundo estudo intitulado "Uma Aplicação do Algoritmo Genético Multiobjetivo NSGA II Para Seleção de Imagens de Satélite de Trechos de Mata Atlântica"trata de um trabalho nacional onde foi utilizado a técnica NSGA-II em um problema multi-objetivo. Por último, "*Heuristics for the bi-objective path dissimilarity problem*", apresenta o modelo matemático utilizado neste trabalho, sendo um dos poucos artigos que trataram o PDP como um problema bi-objetivo, levando em conta questões discutidas nas Sessões 2.1 e 2.2.1.

3.1. *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II*

Anteriormente, o NSGA recebeu muitas críticas por não possuir capacidade de elitismo, ter uma complexidade computacional $O(MN^3)$ e necessitar da utilização de um parâmetro de compartilhamento [Deb et al. 2002]. Por isso, [Deb et al. 2002] apresenta o algoritmo NSGA-II com seus melhoramentos, uma complexidade de $O(MN^2)$ e suas técnicas FNDS e CD explicadas por etapas. O método de seleção utilizada é a de *Mating Pool*, onde são combinadas a população pai e a descendente selecionando as melhores soluções de acordo ao *fitness* e o espalhamento dos pontos. Esta seleção garante o elitismo, ou seja, os melhores resultados continuarão presentes nas próximas gerações.

Nas simulações realizadas em [Deb et al. 2002], em quase todos os problemas escolhidos, o NSGA-II conseguiu localizar soluções com maior espalhamento e uma melhor convergência à fronteira de Pareto, do que outros algoritmos evolucionários multiobjetivos. Todos os detalhes do NSGA-II apresentados em [Deb et al. 2002] foram utilizados no desenvolvimento do algoritmo proposto neste trabalho.

3.2. Uma Aplicação do Algoritmo Genético Multiobjetivo NSGA II Para Seleção de Imagens de Satélite de Trechos de Mata Atlântica

Este estudo descreve a necessidade da utilização de fotografias da Mata Atlântica brasileira para detectar trechos de florestas e tentar diminuir o impacto da degradação ambiental. Portanto, necessitou-se selecionar fragmentos nessas imagens de forma a atender critérios de similaridade [Marinho 2009]. Como diversos objetivos foram levados em conta, os autores optaram pelo uso do algoritmo NSGA-II.

Foi demonstrado que o algoritmo utilizado conseguiu, de forma eficiente, selecionar 20 trechos de Mata Atlântica minimizando para isso três objetivos: diferença de forma, área e conectividade. Quanto menor forem os objetivos, maior será a similaridade existente entre os fragmentos. De acordo com esse princípio o AG estabeleceu o padrão de busca, encontrar indivíduos que possuem fragmentos com maior grau de similaridade entre eles [Marinho 2009].

3.3. *Heuristics for the bi-objective path dissimilarity problem*

Primeiro artigo que introduz o PDP como um problema de OC bi-objetivo e afirma que uma solução consiste em um conjunto p de diferentes caminhos, gerando dois objetivos conflitantes: média de distância dos caminhos deve permanecer baixa e a dissimilaridade entre caminhos no conjunto deve permanecer alta [Martí et al. 2009].

Neste trabalho foi realizado uma revisão da literatura dos métodos descritos na Tabela 1, sendo feita ao final uma comparação com a implementação do método GRASP, desenvolvido pelos autores [Martí et al. 2009]. O modelo matemático presente neste artigo, foi utilizado neste estudo e foi descrito na Seção 2.1.

4. Metodologia

O presente trabalho teve ampla pesquisa bibliográfica, incluindo artigos, livros e vídeos. Iniciou-se com a definição do problema que se desejava trabalhar, focando em problemas de busca em grafos. Obteve-se conhecimento do PDP durante este estudo e por se tratar de um tema com poucos trabalhos publicados, tornou-se o foco principal deste trabalho. O principal artigo que serviu como base para este trabalho foi [Martí et al. 2009], que trata o problema como bi-objetivo.

Inseriu-se neste contexto um tema para tornar o estudo inédito: AG aplicado ao PDP. Como um AG convencional não possui capacidade para lidar com problemas multiobjetivos, obteve-se conhecimento de técnicas específicas para isso. Optou-se pela metaheurística NSGA-II apresentada por [Deb et al. 2002]. Esta técnica ficou conhecida por ser um AG específico para problemas multi-objetivos que possui estratégias para encontrar soluções eficientes. É uma técnica amplamente utilizada em diversos problemas de OC. A linguagem de programação Python foi escolhida por ser uma linguagem orientada à objetos que possui uma variedade de bibliotecas científicas, facilitando o processamento e a representação dos resultados encontrados através de gráficos.

5. Desenvolvimento

Nesta seção é descrita a preparação da estrutura de dados e o desenvolvimento do algoritmo NSGA-II aplicado ao PDP. As etapas que resultaram na obtenção dos resultados finais são descritas em forma de passos.

Passo 1 – Leitura da Instância: Foi escolhido uma instância chamada *ali535.tsp* retirada da biblioteca *tsplib95*[5], como mostra a Figura 8. Escolheu-se essa instância pois é uma das utilizadas no artigo de [Martí et al. 2009] e representa 535 aeroportos ao redor do mundo. Realizou-se a leitura das posições (x,y) do arquivo .tsp que representam coordenadas sendo calculada a distância euclidiana dos nodos. Foi considerado que todos os nós têm conexão. A maior distância encontrada foi de 338.58 unidades.

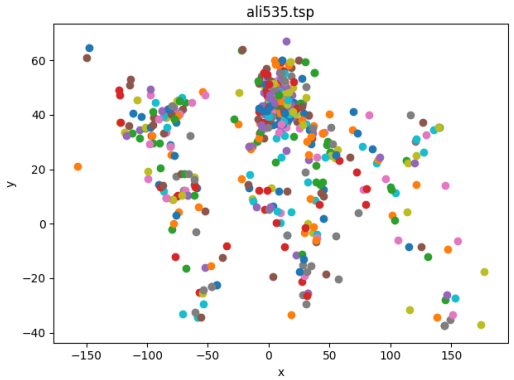


Figura 8. Pontos gerados com a leitura da instância ali535.tsp

Passo 2 – Definição da Estrutura de Dados: Para estrutura de dados foi escolhida a representação dos nodos e suas conexões através de uma matriz de adjacência. Uma vez que a distância de um nodo A até um nodo B é a mesma do que um nodo B até um nodo A, considerou-se apenas a parte triangular inferior desta matriz (representada na Figura 9), ou seja, apenas as posições onde a linha é maior que a coluna ($i > j$). O valor não nulo da distância euclidiana é representada na Figura 9 como a letra a e os valores dentro dos parênteses são respectivamente a posição da linha e coluna da matriz (i,j). Esta escolha reduziu a quantidade de armazenamento na memória de $N \times N$ (dimensões da matriz) para $(N \times N)/2 - N$. Evitando aproximadamente 143.000 cálculos de distância euclidiana com a utilização da instância *ali535.tsp*.

$$\begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ a_{(1,0)} & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{(534,0)} & a_{(534,1)} & \cdots & a_{(534,533)} & 0 \end{bmatrix}$$

Figura 9. Matriz densa triangular inferior

Passo 3 – Remoção das Arestas: Para tornar o problema mais interessante e mais semelhante possível do artigo [Martí et al. 2009] ainda foram feitas adaptações onde apenas os vértices com menos de 10% da maior distância calculada permanecem, zerando os outros valores. Assim é formada uma matriz esparsa, onde se tem apenas os valores

[5]Biblioteca de instâncias para o problema do Caixeiro Viajante e problemas relacionados. É disponibilizado arquivos de texto com coordenadas (x, y) referente a localização dos vértices [Reinelt 1995]. Encontra-se em https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/

diferentes de zero, tornando a estrutura de dados ainda mais eficiente. Para criar a matriz esparsa foi utilizado *SciPy*, que possui ferramentas computacionais científicas para a linguagem Python [Jones et al. 2001]. Os valores na matriz passam a ser representados apenas pelas coordenadas e os valores não nulos, como: $(1,0)a_{(1,0)}...(534,533)a_{(534,533)}$.

Passo 4 – Desenvolvimento do NSGA-II: O desenvolvimento do algoritmo NSGA-II, que foi implementado neste estudo é descrito no Algoritmo 2 e neste tópico será discutido por etapas:

Passo 4.1 – Inicialização randômica da população (N indivíduos): para a formação do espaço de soluções, primeiramente, foi realizado uma busca dos k-SPs, selecionando o Aeroporto Ajaccio-Napoleão Bonaparte na França como ponto de origem (18) e o Aeroporto Internacional de Acapulco no México como ponto de destino (7). Representados como os pontos mais distantes em [Martí et al. 2009]. Foi utilizado o método *all shortest path* presente no pacote NetworkX[6] para buscar os k-SPs. No final da busca, formou-se todo o espaço de soluções possível entre os pontos escolhidos. Para a formação da população inicial de tamanho N foi escolhido aleatoriamente, no espaço de soluções, caminhos que formaram cromossomos com 5 caminhos (*p = 5*, este *p* utilizado em [Martí et al. 2009]) como mostra na Figura 10. Escolheu-se neste estudo a representação do gene como um caminho, uma vez que o caminho pode ter diferentes tamanhos e o cálculo da dissimilaridade é obtido através de um conjunto *p* de caminhos.

$$\big[[18,2,1,5,6,7][18,2,9,1,3,4,7][18,1,5,4,2,7][18,9,5,6,8,7][18,5,3,1,7]\big]$$

Figura 10. Representação de um cromossomo com 5 caminhos

Passo 4.2 – Cálculo do fitness dos indivíduos: Este cálculo foi realizado para obtenção dos valores de custo e dissimilaridade. Tem a finalidade de ser utilizado nas próximas etapas e serve para selecionar os indivíduos mais aptos. Maiores valores de dissimilaridade e menores valores de custo são os objetivos a serem encontrados. Este cálculo é realizado através das Equações 1 e 2 descritas na Seção 2.1.

Passo 4.3 – Classifica a população em fronteiras (FNDS): Classificou-se a fronteira de Pareto e as fronteiras subsequentes. Nesta etapa são verificados os indivíduos que dominam e os que são dominados. Cada cromossomo possui um contador que é incrementado a cada indivíduo dominado por ele. Cromossomos não dominados, possuem o contador com os maiores valores e por isso fazem parte da fronteira de Pareto.

Passo 4.4 – Cálculo da distância de todos indivíduos (CD): Através da técnica CD foi calculado a distância entre os pontos gerados. Estas distâncias são ordenadas do maior valor para o menor. Maior distância significa maior espalhamento, ou seja, melhores soluções. Tanto a técnica anterior quanto esta são utilizadas na seleção dos mais aptos.

Passo 4.5 – Combinação da população atual com a população descendente: foi gerada a população descendente com o dobro do tamanho da população atual, onde a

[6]Pacote Python para criação, manipulação e estudo de estruturas, dinâmicas e funções de redes complexas, como Grafos, Dígrafos e etc. [Hagberg et al. 2008].

primeira metade tem os mesmos valores presentes na população atual e a segunda metade com indivíduos descendentes, modificados a partir de operadores genéticos como o *crossover* e a mutação, representados na Figura 11 e Figura 12, respectivamente.

Na Figura 11 é possível notar a seleção de subpartes de dois cromossomos para a formação de um indivíduo descendente.

$$\boxed{[18,2,6,8,1,7]}\ \boxed{[18,5,1,2,8,7]}\ [18,5,4,3,8,7]\ \ [18,4,2,9,1,7]\ \ [18,4,6,1,8,7]$$

$$[18,9,2,6,8,7]\ \ [18,1,4,5,9,7]\ \ \boxed{[18,3,2,1,9,7]}\ \boxed{[18,1,4,9,2,7]}\ \boxed{[18,4,2,8,9,7]}$$

$$\boxed{[18,2,6,8,1,7]}\ \boxed{[18,5,1,2,8,7]}\ \boxed{[18,3,2,1,9,7]}\ \boxed{[18,1,4,9,2,7]}\ \boxed{[18,4,2,8,9,7]}$$

Figura 11. Representação do operador genético *crossover*

Na Figura 12 ocorre a mutação do alelo de um indivíduo.

$$[18,5,9,3,1,7]\ \ [18,3,9,5,2,7]\ \ \boxed{[18,9,1,5,6,7]}\ \ [18,6,5,9,1,7]\ \ [18,9,6,1,2,7]$$

$$[18,5,9,3,1,7]\ \ [18,3,9,5,2,7]\ \ \boxed{[18,1,4,9,2,7]}\ \ [18,6,5,9,1,7]\ \ [18,9,6,1,2,7]$$

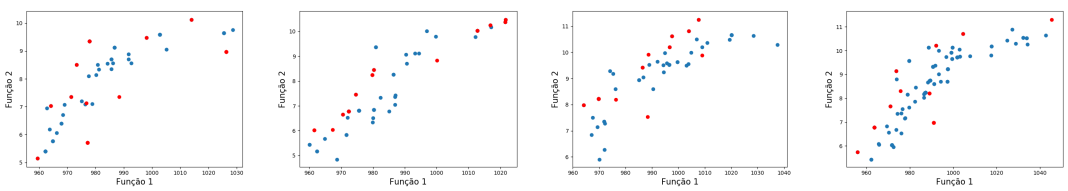
Figura 12. Representação do operador genético mutação

Passo 4.6 – Seleção dos mais aptos: A seleção ocorreu da mesma forma apresentada no artigo [Deb et al. 2002] onde é utilizado um *Mating Pool*. Essa técnica garante a perpetuação dos melhores indivíduos. Essa nova população gerada é utilizada na próxima geração.

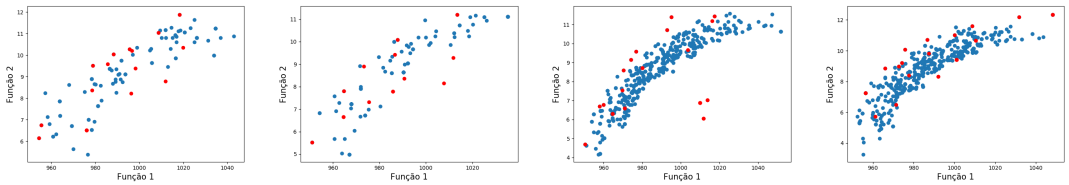
Passo 4.7 – Condição de Parada: As etapas do *Passo 4* descritas acima, se repetiram até que o valor máximo das gerações foi alcançado.

6. Resultados

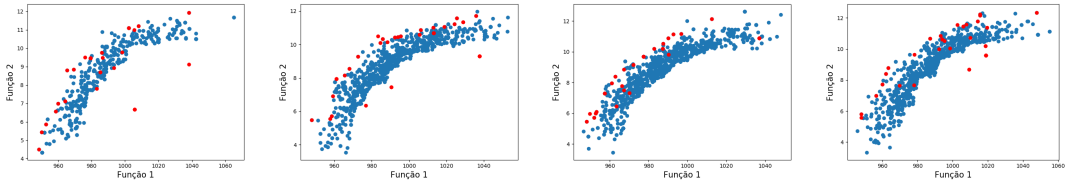
O tamanho da população (N cromossomos) e a quantidade de gerações neste trabalho foi 50, 100, 500 e 1000 e 20, 50 e 100, respectivamente. Na Figura 13 estão os gráficos gerados para cada uma das execuções realizadas, onde foi minimizada a Função 1 (custo) e maximizada a Função 2 (dissimilaridade). Os pontos vermelhos pertencem a fronteira de Pareto enquanto os pontos azuis representam as outras fronteiras. A legenda de cada gráfico é representada pelo tamanho da população vs. a quantidade máxima de gerações.



(a) 50 vs. 20 (b) 50 vs. 50 (c) 50 vs. 100 (d) 100 vs. 20



(e) 100 vs. 50 (f) 100 vs. 100 (g) 500 vs. 20 (h) 500 vs. 50



(i) 500 vs. 100 (j) 1000 vs. 20 (k) 1000 vs. 50 (l) 1000 vs. 100

Figura 13. Gráficos para cada execução (População vs. Gerações)

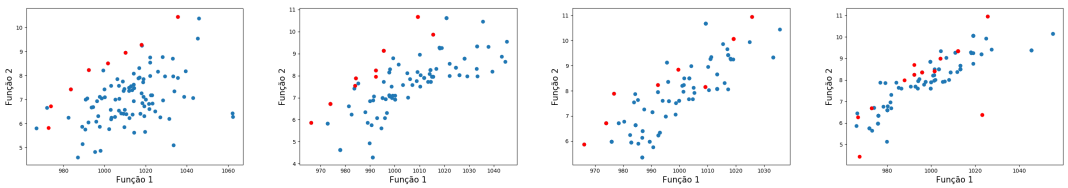
Enfim, observa-se na Tabela 2 os diferentes tempos de execução para cada um dos valores citados anteriormente, sendo a coluna da esquerda o número máximo de gerações e o cabeçalho o tamanho da população. Os testes foram realizados em um computador com processador i5-8250U e 8GB de memória RAM. Para armazenamento, foi utilizado um disco rígido de 5400 RPM.

Tabela 2. Tempos de execução do método proposto

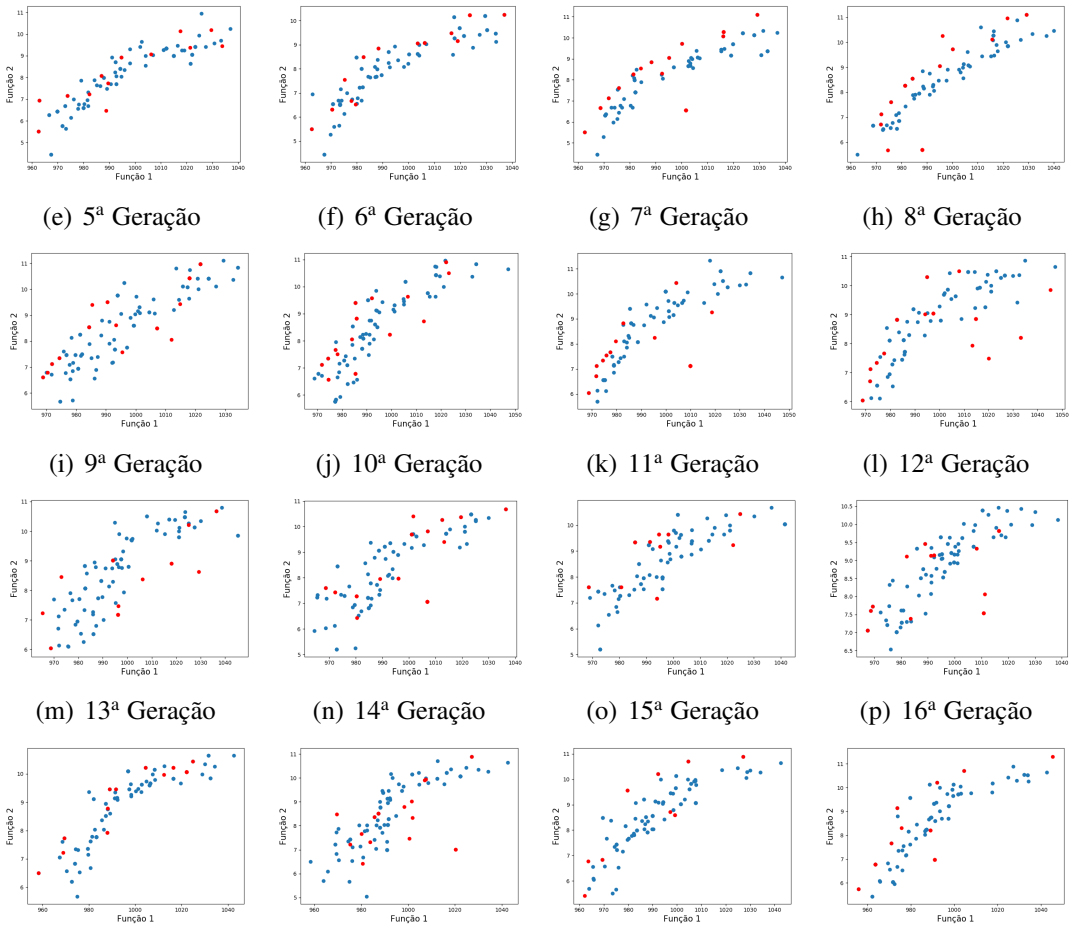
	50	100	500	1000
20	11.82s	26.15s	393.06s	1637.44s
50	29.26s	64.71s	963.20s	2933.17s
100	57.55s	129.70s	1953.27s	5758.06s

Para a montagem da Tabela 2 foi realizada uma média aritmética. Foram somados os valores de tempo encontrados para cada execução e dividido pelo valor total de execuções. Neste estudo foram realizadas 5 execuções para cada configuração, ou seja, $(t_1 + t_2 + t_3 + t_4 + t_5)/5$, sendo t_i o tempo de cada execução (i).

Para melhor compreensão dos gráficos gerados foi escolhida a Figura 13(d) para realizar uma análise dos resultados encontrados por geração. Uma vez que o gráfico possui 20 gerações, a Figura 14 apresenta os 20 gráficos gerados. Nota-se que o comportamento dos gráficos é de aleatoriedade em algumas gerações devido a probabilidade do *crossover* e da mutação ser calculado randomicamente. Porém é notável que os pontos estão se aproximando do eixo y e se afastando do eixo x, comportamento causado pela dissimilaridade que está sendo maximizada e o custo que está sendo minimizado.



(a) 1ª Geração (b) 2ª Geração (c) 3ª Geração (d) 4ª Geração



(e) 5ª Geração (f) 6ª Geração (g) 7ª Geração (h) 8ª Geração

(i) 9ª Geração (j) 10ª Geração (k) 11ª Geração (l) 12ª Geração

(m) 13ª Geração (n) 14ª Geração (o) 15ª Geração (p) 16ª Geração

(q) 17ª Geração (r) 18ª Geração (s) 19ª Geração (t) 20ª Geração

Figura 14. Gráficos para cada geração da configuração 100 vs. 20

7. Conclusões e Trabalhos Futuros

Foi abordado neste trabalho a possibilidade de resolução do Problema de Caminhos Dissimilares (PDP) com o algoritmo NSGA-II. Nota-se com o comportamento dos gráficos encontrados que foi possível maximizar a dissimilaridade entre os arcos e minimizar a distância entre os vértices, utilizando-se do método proposto. Dificuldades como a representação do cromossomo (caminhos podem ter diferentes tamanhos), a montagem das funções objetivo de acordo ao artigo de [Martí et al. 2009] e o tema tratado como bi-objetivo, foram alguns empecilhos superados.

Métricas de avaliação de Pareto, são ferramentas importantes para se certificar que o algoritmo está funcionando de forma matematicamente correta. Estas foram utilizadas no trabalho de [Marinho 2009] e são conhecidas como *HyperVolume*, *Maximum Spread* e *Spacing*, bem como, a paralelização em placas gráficas para obtenção de melhores resultados são algumas possibilidades de trabalhos futuros.

Uma vez que o PDP vem sendo estudado nos últimos anos e possui uma ampla gama de aplicações reais, concluiu-se que o presente trabalho pode vir a auxiliar no processo de logística de uma organização sempre que a mesma desejar calcular caminhos diferentes para entrega ou recolhimento de produtos mantendo a distância reduzida.

Referências

Cobham, A. (1965). The intrinsic computational difficulty of functions.

Corberán, Á., Peiró, J., Campos, V., Glover, F., and Martí, R. (2012). Optsicom project. Disponível em: <http://www.optsicom.es/pdp/>. Acesso em: 05 Abr. 2018.

Darwin, C. (1859). On the origin of species by means of natural selection. *Murray, London*.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

Dhein, G. et al. (2016). *Problemas de roteamento de veículos com dependência temporal e espacial entre rotas de equipes de campo*. PhD thesis, Universidade Federal de Santa Maria.

Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. pages 11–15, Pasadena, CA USA. 7th Python in Science Conference (SciPy2008).

Holland, J. (1975). Adaptation in natural and artificial systems: an introductory analysis with application to biology. *Control and artificial intelligence*.

Jones, E., Oliphant, T., Peterson, P., et al. (2001). Scipy: Open source scientific tools for python. Disponível em: <https://www.scipy.org/>. Acesso em: 20 Agosto. 2018.

Kramer, O. (2017). *Genetic algorithm essentials*, volume 679. Springer.

Labadie, N., Prins, C., and Prodhon, C. (2016). *Metaheuristics for Vehicle Routing Problems*. John Wiley & Sons.

Marinho, D. (2009). Uma aplicaçao do algoritmo genético multiobjetivo nsga ii para seleçao de imagens de satélite de trechos de mata atlântica. B.s. thesis, Universidade de Pernambuco.

Martí, R., Velarde, J. L. G., and Duarte, A. (2009). Heuristics for the bi-objective path dissimilarity problem. *Computers & Operations Research*, 36(11):2905–2912.

Mitchell, M. (1995). Genetic algorithms: An overview. *Complexity*, 1(1):31–39.

Reinelt, G. (1995). Tsplib95. *Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR), Heidelberg*, 338.