Resolução do Problema do Caixeiro Viajante Multi-objetivo aplicando *Simulated Annealing*

Ariel François Maia Domingues
Engenharia de Sistemas
Universidade Federal de Minas Gerais
Belo Horizonte, MG 31270-901
Email: ariel.f.m.domingues@gmail.com

Hernane Braga Pereira
Engenharia de Sistemas
Universidade Federal de Minas Gerais
Belo Horizonte, MG 31270-901
Email: hernanepereira@hotmail.com

Nikolas Dias Magalhães Fantoni
Engenharia de Sistemas
Universidade Federal de Minas Gerais
Belo Horizonte, MG 31270-901
Email: nikolasfantoni@gmail.com

***Resumo*—Este trabalho[1] tem por finalidade apresentar uma solução para o Problema do Caixeiro Viajante a partir da metaheurística Simulated Annealing Multi-Objetivo, visando otimizar a distância percorrida e o tempo do trajeto do Caixeiro, utilizando seis estruturas de vizinhanças diferentes, cada uma com um nível de perturbação distinto das outras. A linguagem utilizada para a implementação dos algoritmos foi o R e os resultados mostraram uma boa otimização em relação à resposta inicial, além de uma otimização excelente caso escolhida uma solução completamente aleatória.**

***Palavras-Chave*–Otimização Multi-objetivo, Problema do Caixeiro Viajante, Simulated Annealing, Estrutura de Vizinhança**

I. Introdução

O problema do caixeiro viajante (PCV) é um problema clássico na literatura, onde um entregador ou vendedor deve percorrer o caminho de menor custo a partir de uma cidade inicial, visitando uma única vez todas as outras *n* cidades, e então retornar para o ponto de partida. A solução do problema é combinatória e ele é classificado como NP-completo, pois não há como resolver esse problema utilizando um algoritmo de ordem de complexidade polinomial. Assim, são aplicadas meta-heurísticas, de forma a tentar otimizar a solução do problema para um valor considerado ótimo, dadas determinadas métricas inicialmente definidas. [1]

Para este trabalho será usada uma instância de 250 cidades, todas interligadas entre si (*i.e.* pode-se ir, a partir de qualquer cidade A, para qualquer cidade B, com $A \neq B$), de valores assimétricos, ou seja, apesar de a distância da cidade A até B ser idêntica a de B até A, o tempo entre os deslocamentos de ida e de volta podem ser diferentes. Nesta etapa do trabalho, foram fornecidos dois arquivos **.csv* contendo os valores das distâncias e tempos entre todas as cidades do problema. Cabe ressaltar que a distância e o tempo entre uma cidade e ela mesma é $d = t = 0$.

O problema será tratado como multi-objetivo e, portanto, a função objetivo será uma minimização simultânea da distância total e do tempo total gastos pelo Caixeiro, através da técnica de meta-heurística *Simulated Annealing* (SA) e utilizando um total de seis estruturas de vizinhança alternadas, a fim de obter um resultado mais próximo de um possível ótimo global e evitar resultados ótimos locais.

Além disso, serão utilizadas duas abordagens escalares: a Soma Ponderada (P_w) e ϵ-restrito (P_ϵ) para a otimização multi-objetivo.

[1]Trabalho realizado para a disciplina ELE088 - Teoria da Decisão, ministrada pelo professor Lucas de Souza Batista do Departamento de Engenharia Elétrica da Universidade Federal de Minas Gerais para o segundo semestre do ano de 2019.

II. Metodologia

Todos os algoritmos aqui apresentados foram implementados na linguagem *R*, um ambiente de desenvolvimento para aplicações estatísticas e gráficas[2]. Para entender a representação do problema e a solução proposta, são necessários o entendimento de quatro tópicos, explicados posteriormente nesta sessão: a descrição matemática do problema do PCV, descrita em *Modelagem do Problema do Caixeiro Viajante*; a descrição do algoritmo de otimização utilizado, descrito em *Simulated Annealing (SA)*, a descrição do algoritmo de solução inicial proposto, descrito em *Solução Inicial* e por fim a descrição das estruturas de vizinhanças propostas para a resolução do problema, encontradas em *Estruturas de Vizinhanças*.

A. Modelagem do PCV

O problema do caixeiro viajante, também conhecido como *Traveling-Salesman Problem* (TSP), é um dos problemas que mais recebe atenção para sua resolução, visto que há grande aplicabilidade na indústria, devido ao seu grande potencial de retorno econômico. O PCV foi inicialmente abordado por Dantzig's [1] como um problema de otimização linear e solucionado utilizando o método simplex para uma instância de 49 cidades. Entretanto, por se tratar de um problema combinatório, esta abordagem não é viável ao ser utilizada em larga escala, portanto atualmente são mais comuns abordagens do problema utilizando de meta-heurísticas, que apesar de não garantirem otimalidade, entregam bons resultados em um tempo computacional viável. O problema é representado como um grafo, onde cada vértice representa uma cidade e cada aresta é equivalente a uma viagem da cidade *i* até *j*. Aqui, o grafo será representado como uma estrutura de matriz de adjacências, com o tamanho $[n \times n]$. Esta estrutura foi escolhida, pois todas as cidades estão interligadas entre si. A figura 1 mostra um exemplo de um grafo.

[2]Mais sobre o R em: https://www.r-project.org/about.html

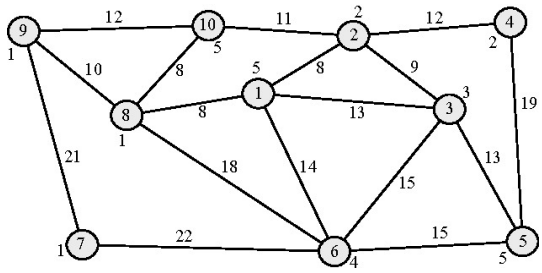

Fig. 1. Exemplo de um grafo.

Uma das maneiras de modelar o problema [2] encontra-se a seguir:

$$min \sum_{j=1}^{m} \sum_{i=1}^{m} c_{ij} x_{ij} \tag{1}$$

$$Sujeito\ a : \sum_{j=1}^{m} x_{ij} = 1 \qquad \forall i = 1, ..., m \tag{2}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j = 1, ..., m \tag{3}$$

$$\sum_{i \in K} \sum_{j \in K} x_{ij} \leq |K| - 1 \qquad \forall i, j \tag{4}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \tag{5}$$

onde K , é um subconjunto não vazio das cidades $1, .., m$ Para o caso onde será avaliado o tempo entre cidades, adiciona-se a restrição:

$$d_{ij} \neq d_{ji} \qquad \forall i, j \tag{6}$$

Pois apesar das distâncias entre cidades serem iguais, o tempo entre elas é diferente, variando de acordo com o ponto de partida.

B. Simulated Annealing (SA)

Para a resolução do PCV será o utilizado o método do *Simulated Annealing* (SA) multi-objetivo [3] [4] baseado em vizinhanças. O algoritmo SA pertence à classe dos algoritmos de busca local e é muito importante por possuir uma componente estocástica envolvida, o que facilita sua análise de convergência, além de ser aplicável a uma vasta quantidade de problemas. Sua maior vantagem é que a busca é feita procurando sempre aproximar do ótimo global, ao invés de aceitar soluções ótimas locais, o que é possível pois o algoritmo aceita soluções intermediárias cujo custo da função objetivo pode ser maior que o de uma solução intermediária anteriormente encontrada, para então refinar a pesquisa em busca de um ótimo. O algoritmo baseia-se na ideia do resfriamento lento de peças metálicas, usado na indústria metalúrgica com o objetivo de obter características como dureza e tenacidade do material diferentes das obtidas se fosse utilizado o resfriamento rápido.

Para entender a implementação geral do algoritmo, são necessárias algumas definições abaixo:

- Temperatura inicial T_{0d} e T_{0t}: define o grau de aceitação de soluções cujo resultado da função objetivo é pior que a solução anterior, para a distância e para o tempo, de forma separada.
- Iterações por temperatura m: define a quantidade de iterações que serão feitas a cada temperatura definida.
- Função de mudança de temperatura $f(T_k)$: define a temperatura T_{k+1} a partir de um T_k dado.
- Iterações totais n: define o total de mudanças de temperatura máximas possíveis.
- Função de Vizinhanças $\aleph(\bar{x}, lvl)$: gera uma solução $\bar{x}_2$, com nível de perturbação lvl, a partir de uma solução $\bar{x}$.
- Função de solução inicial $init()$: gera uma solução inicial para o problema.
- Nível da vizinhança lvl: define o nível de perturbação possível que cada vizinhança pode fazer, sendo o nível 1 considerado perturbações leves (que geram soluções parecidas com a da entrada) até o nível 6 (que gera perturbações consideravelmente distintas da solução de entrada).
- Diferença dos custos ΔE: define a diferença entre os custos da solução atual em relação ao tempo e à distância, simultaneamente, e da solução calculada a cada iteração.
- Função aleatória $rnd(0..1)$: gera um número aleatório entre 0 e 1 para comparar à função de probabilidade de aceitação para uma solução cujo custo é menor que o custo da solução atual.

O algoritmo do SA utilizado é uma variação do algoritmo clássico mostrado na literatura, mostrado a seguir. A variação consiste no uso de seis estruturas de vizinhança, classificadas pelo nível de perturbação *lvl*, que são escolhidas de acordo com a quantidade de soluções aceitas por iteração. Ou seja, se o algoritmo estiver com dificuldades de encontrar uma nova solução utilizando a estrutura de nível baixo, ele aumenta o nível para tentar encontrar uma nova solução. Assim, a melhor descrição do algoritmo implementado neste trabalho consiste em:

1) procura uma solução inicial e a considera a melhor;
2) encontra uma solução na vizinhança, com um nível de perturbação baixo;
3) calcula o ΔE;
4) se ΔE for menor ou igual a zero, a solução da vizinhança se torna a melhor até então e substitui a solução analisada. Se ΔE for maior que zero, um número aleatório é gerado, entre 0 e 1. Se for menor que a função de probabilidade $p = e^{\frac{-\Delta E_d}{T_{kd}}} \cdot e^{\frac{-\Delta E_t}{T_{kt}}}$, então a solução é aceita e substituída como a solução a ser analisada, caso contrário nada acontece;
5) repete os itens de 2 ao 4 para o total de *m* iterações;
6) atualiza a temperatura e o nível, este caso nenhuma solução tenha sido aproveitada utilizando o nível anterior;

7) repete os itens do 2 ao 6 até o total de *n* iterações ou até a temperatura ser suficientemente baixa;
8) retorna a melhor solução encontrada.

A função de variação de temperatura é dinâmica [5] [6], mostrada na equação 7, em que $\bar{E}$ corresponde a média de todas as diferenças de custos entre as soluções aceitas e as soluções atuais, $min(E)$ corresponde à menor diferença encontrada e D_0 é um valor fixado (entre 0,5 e 0,9). Assim, decide-se o menor valor entre a razão entre o custo mínimo e o custo médio ou entre o valor fixo D_0 e multiplica-se pela temperatura atual para obter a próxima temperatura. Esta operação é feita para cada temperatura, relacionada à distância e ao tempo, separadamente.

$$T_{k+1} \leftarrow min(\frac{min(E)}{\Delta\bar{E}}, D_0) \times T_k \tag{7}$$

A temperatura inicial foi calculada conforme a equação 8, em que o coeficiente τ_0 é uma taxa de aceitação inicial. Para este problema, foi considerado $\tau_0 = 0,5$. Assim, são feitas 100 pequenas perturbações na solução inicial, obtendo-se uma série de ΔE. Após isso, obtém-se a média $\Delta\bar{E}$. Foram calculadas duas temperaturas iniciais distintas, um T_{0d} para as distâncias e um T_{0t} para o tempo.

$$e^{\frac{-\Delta\bar{E}}{T_0}} = \tau_0 \tag{8}$$

Por fim, foi necessária uma normalização da função de custo, uma vez que as dimensões da distância destoam muito em relação ao tempo, o que pode prejudicar no balanceamento da otimização. A função normalizada é mostrada na equação 9, uma vez que o problema é de minimização.

$$f_i(x) = \frac{f_i(x) - minf_i(x)}{maxf_i(x) - minf_i(x)} \tag{9}$$

C. Abordagem Soma Ponderada P_w

Nesta abordagem, a função objetivo passa a ser

$$min \;\; w_t f_t(\bar{x}) + w_d f_d(\bar{x}), \tag{10}$$

em que $w_i \; \forall \; i \in \{d, t\}$ é o peso considerado para a otimização da distância e do tempo, respectivamente, e $f_i \; \forall \; i \in \{d, t\}$ são as funções dos custos da distância e do tempo, também respectivamente. Serão utilizados $w_d = w_t = 0,5$, uma vez que não há preferência *a priori* de otimização entre o tempo e a distância. Assim, ambas as restrições são otimizadas simultaneamente e o algoritmo normaliza a função de custo para definir se possui uma solução x^* melhor que a anterior.

D. Abordagem ϵ-Restrito P_ϵ

De acordo com Pantuza [7] o método ϵ-Restrito consiste na otimização do objetivo mais importante, sujeitando-se às restrições de outros objetivos. Considerando um problema de minimização, sua formulação seria:

$$minimizar \;\; f_1(x) \tag{11}$$

Algoritmo 1: SIMULATED ANNEALING

Entrada: $f(T_k)$, T_0d, T_0t, $init()$, $\aleph(\bar{x},\ lvl)$
Saída: Melhor solução x^*

início
  $\bar{x_1} \leftarrow init()$
  $\bar{x}^* \leftarrow \bar{x}_1$
  $T_{kd} \leftarrow T_{0d}$
  $T_{kt} \leftarrow T_{0t}$
  $lvl \leftarrow 1$
  para *cada* $i \in n$ **faça**
    $aceito \leftarrow 0$
    para *cada* $j \in m$ **faça**
      $x_2 \leftarrow \aleph(\bar{x}_1,\ lvl)$
      $\Delta E \leftarrow$ custo norm. $\bar{x}_2 -$ custo norm. $\bar{x}_1$
      se $\Delta E \leq 0$ **então**
        $aceito \leftarrow aceito + 1$
        $\bar{x}^* \leftarrow \bar{x}_2$
        $\bar{x}_1 \leftarrow \bar{x}_2$
      fim
      senão
        se $rnd(0..1) < e^{\frac{-\Delta E_d}{T_{kd}}} \cdot e^{\frac{-\Delta E_t}{T_{kt}}}$ **então**
          $\bar{x}_1 \leftarrow \bar{x}_2$
          $aceito \leftarrow aceito + 1$
        fim
      fim
    fim
    $T_k \leftarrow f(T_k)$
    se $aceito = 0$ **então**
      $lvl \leftarrow lvl + 1$
    fim
    senão
      $lvl \leftarrow 1$
    fim
  fim
fim
retorna $\bar{x}^*$

sujeito a:

$$f_i(x) \leq \epsilon_i \qquad \forall \; i = 2, 3, ..., q \tag{12}$$

Onde, ϵ_i é o limite superior do objetivo i e q o número de objetivos. Para construir o conjunto Pareto-ótimo, mesmo quando o espaço objetivo é não convexo, deve-se apenas variar o limite superior. Porém, se este limite não é adequado, o subconjunto de possíveis soluções obtido pode ser vazio, ou seja, não existe solução viável.

Neste trabalho, como as dimensões do tempo e da distância são de duas ordens de grandeza diferentes, o algoritmo foi implementado de forma automática em relação ao ϵ. Ou seja, se o ϵ é da ordem de grandeza do tempo, significa que está sendo pedido para restringir as horas gastas e otimizar a distância. Caso seja da ordem da distância, ele otimiza o tempo, restringindo a quilometragem.

E. Solução Inicial

A função de solução inicial usada no SA retornar uma matriz de tamanho $[3 \times 250]$ de cidades visitadas, onde a coluna *j* representa a cidade de origem e as linhas *i* representam a cidade de destino e o custo até a mesma. A função calcula a solução da seguinte forma: para cada cidade visitada, o custo entre cidades é ordenado de forma crescente e a próxima cidade é escolhida aleatoriamente entre as *r* cidades mais próximas, onde *r* é um parâmetro de entrada da função e quanto maior o valor de *r*, mais aleatória será a solução inicial gerada. Ao final da função, é calculada a distância para se retornar à cidade de origem. Como a função não é determinística, ela foi executada 5 vezes para verificar sua robustez com valores de $r = 3$ e $r = 10$. Os resultados são mostrados na tabela I.

	r=3		r=10	
TESTE	**DISTÂNCIA**	**TEMPO**	**DISTÂNCIA**	**TEMPO**
1	2432,8	44,1	4131,7	71,8
2	2608,7	44,6	4138,9	72,8
3	2366	42,5	4427,1	69,4
4	2583,1	43,7	4338,9	72,5
5	2579,1	43,5	4168,5	69,2
Média	2513,94	43,68	4241,02	71,14
Desvio Padrão	107,79	0,78	134,02	1,72

TABLE I
RESULTADOS DOS TESTES PARA O ALGORITMO DA SOLUÇÃO INICIAL.

F. Estrutura de Vizinhanças

As estruturas de vizinhança são peças fundamentais no funcionamento esperado do algoritmo SA. Elas são as responsáveis por modificar as soluções candidatas de modo a navegar pelo espaço de alternativas em busca dos ótimos locais, com o objetivo de mapear melhores soluções e de preferência o ótimo global. [8]

É preciso haver um equilíbrio nas estruturas de vizinhança. Elas devem modificar a solução candidata o suficiente para o prosseguimento do algoritmo, porém não tanto para que ocorram saltos grandes muito frequentemente, o que atrapalha o refinamento de ótimos locais.

O ideal, portanto, é ter várias estruturas de vizinhança com níveis de perturbação diferentes, para cada ocasião. Níveis de perturbação baixos são melhores para refinar soluções em uma mesma bacia que contém um ótimo local. Porém, quando o SA passa a não melhorar a solução é preciso buscar sair dessas bacias em busca de novas soluções, potencialmente melhores. Assim, níveis de perturbação crescentes são definidos, para que seja possível fazer esses saltos entre bacias. Depois o nível pode ser diminuído novamente para refinamento da nova bacia encontrada, e o processo se reinicia.

No algoritmo proposto, foram definidos seis estruturas de vizinhança. A ordem de perturbação é definida analisando dois critérios: o número de arestas modificadas (são elas que contém o custo de ir de uma cidade a outra) e a distância para seus vizinhos (quanto mais longe o vizinho, maior é a perturbação). As estruturas, de menor nível de perturbação para maior, são: Troca Vizinha Simples, Deslocamento Simples, Inversão, Troca Vizinha Dupla, Troca Intervalada e Deslocamento Duplo.

- *Troca Vizinha Simples*: Estrutura de nível de perturbação 1. Uma cidade é escolhida aleatoriamente e troca de lugar com seu vizinho da frente. Ou seja, se a ordem de um caminho for A > B > C > D > E, e B é a cidade escolhida, então B troca com C e a nova ordem passa a ser A > C > B > D > E.
  Na troca, são modificadas 3 arestas (3 retiradas e trocadas por outras 3) e o vizinho é o imediato da frente (baixa perturbação). Dessas 3 arestas, 1 só inverte seu sentido (se fosse de A para B, passa a ir de B para A), o que quer dizer que se a matriz de custos for simétrica (custo de A para B igual a custo de B para A), apenas 2 arestas afetam de fato o custo da nova solução.

- *Deslocamento Simples*: Estrutura de nível de perturbação 2. Uma cidade é escolhida aleatoriamente e sofre um deslocamento para frente de 3 a 7 cidades (probabilidade do tamanho do deslocamento distribuída uniformemente). Ou seja, se a ordem do caminho for A > B > C > D > E > F, e B é a cidade escolhida, então B é deslocada e a ordem passa a ser A > C > D > E > F > B, se o deslocamento for de 4 cidades por exemplo.

  O deslocamento modifica 3 arestas, assim como o anterior, porém não é afetado pela simetria dos custos. Além disso, os vizinhos são mais distantes (em média 5 cidades de distância), aumentando a perturbação.

- *Inversão*: Estrutura de nível de perturbação 3. Uma cidade é escolhida aleatoriamente e tem o trecho subsequente de 3 a 7 cidades invertido (probabilidade do tamanho do trecho distribuída uniformemente). Ou seja, se o caminho for A > B > C > D > E > F, e B é a cidade escolhida, então o caminho seguinte a B é invertido de forma a se tornar A > E > D > C > B > F, se o trecho for de 4 cidades por exemplo.
  No pior caso, são modificadas 8 arestas, porém 6 delas apenas com inversão de sentido, o que reduz a 2 arestas a perturbação para matrizes de custo simétricas. Para matrizes assimétricas, a perturbação passa a ser mais significante, porém, como a inversão ocorre com cidades vizinhas, a perturbação é baixa, mesmo em relação a outras estruturas que alteram menos arestas que esta.

- *Troca Vizinha Dupla*: Estrutura de nível de perturbação 4. É idêntica à estrutura de nível 1, com a diferença que neste caso ocorrem duas trocas aleatórias ao invés de uma só. Assim, 6 arestas são modificadas, sendo 2 invertidas apenas, mas ambas trocas ocorrem com vizinhos imediatos.

- *Troca Intervalada*: Estrutura de nível de perturbação 5.

Uma cidade é escolhida aleatoriamente e é trocada de lugar com outra cidade a sua frente, com um intervalo de 3 a 7 cidades entre elas (probabilidade do tamanho do intervalo distribuída uniformemente). Ou seja, se o caminho for A > B > C > D > E > F > G, e B é a cidade escolhida, então ocorre a troca e o novo caminho passa a ser A > G > C > D > E > F > B, se o intervalo for de 4 cidades por exemplo. Nesta troca, 4 arestas são modificadas e a distância entre os vizinhos é grande (em média 5 cidades), o que representa uma perturbação superior (em média) às anteriores.

- *Deslocamento Duplo*: Estrutura de nível de perturbação 6. É idêntica à estrutura de nível 2, com a diferença que neste caso ocorrem dois deslocamentos aleatórios ao invés de um só. Assim, 6 arestas são modificadas, e as trocas ocorrem com vizinhos distantes (de 5 cidades em média). Desta forma, é a estrutura que tende a perturbar mais as soluções candidatas.

III. Resultados

O algoritmo foi executado 5 vezes para cada método de abordagem. Para a abordagem de Soma Ponderada, as superfícies Pareto Ótimas encontradas são mostradas na figura 2, sendo cada conjunto de dados da mesma cor uma iteração diferente do método.

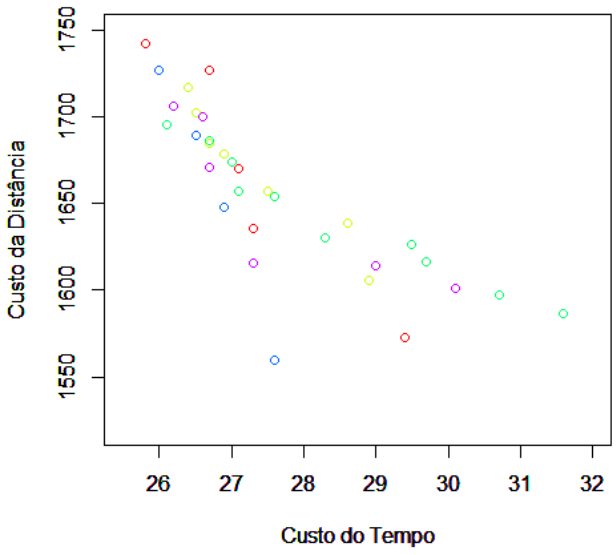

Fig. 2. Superfície Pareto Ótima para cada iteração do método P_w.

Ainda que separadas, cada superfície possa parecer uma curva diferente, ao juntar as cinco iterações vemos o princípio de uma exponencial decrescente se formando, o que indicaria que a superfície Pareto Ótima real para o problema pode ter este formato.

O algoritmo rodou por $56s$ em um PC da família $Intel\ core\ i5@3,5Ghz$, equipado com $16GB$ de RAM, sob o ambiente do Windows 10. Os parâmetros para a execução do algoritmo foram:

- $n = 1000$
- $D_0 = 0,8$
- $m = 20$
- w_t e $w_d \in \{0,03; 0,06; ...; 0,99\}$, sendo que $w_t = 1 - w_d$

O melhor resultado encontrado dentre todas as iterações foi o de 1559,9km e 27,6h. Considerando que a solução inicial retornava 1624,7km e 33h, o algoritmo conseguiu uma otimização de 4% em relação à distância e de 16,36% para a distância no melhor caso, em relação a uma solução inicial já pré-otimizada com um método guloso com uma variação pequena estocástica.

Para a abordagem de ϵ-Restrito, as superfícies Pareto Ótimas encontradas são mostradas na figura 3, sendo cada conjunto de dados da mesma cor uma iteração diferente do método.

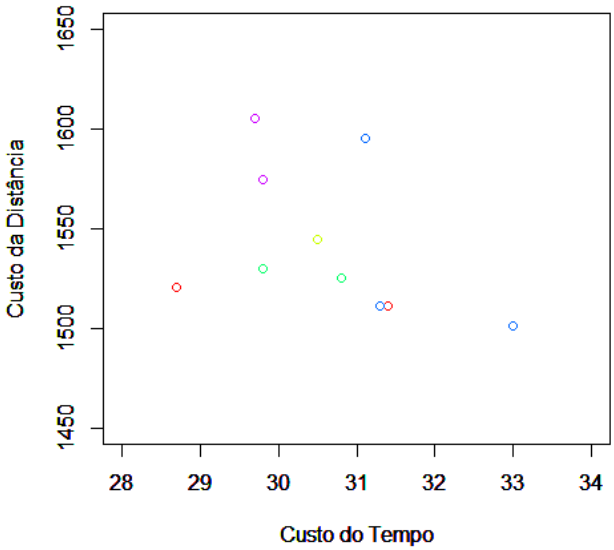

Fig. 3. Superfície Pareto Ótima para cada iteração do método P_ϵ.

O algoritmo foi executado por $8min26s$ para executar as 5 iterações, no mesmo ambiente que a abordagem anterior. Os parâmetros utilizados foram:

- $n = 1000$
- $D_0 = 0,8$
- $m = 20$
- $\epsilon \in \{27; 27,2; ...; 34\} \cup \{1580, 1582, ..., 1650\}$

O melhor resultado encontrado dentre todas as iterações foi o de 1520,6km e 28,4h. Considerando que a distância da solução inicial era de 1627,8km e o tempo era de 34,5h, o algoritmo conseguiu uma redução em forma de otimização aproximada de 6,6% em relação à distância e 17,7% em relação ao tempo.

IV. Conclusão

O Problema do Caixeiro Viajante é muito comum na literatura por ser um problema NP-completo, cuja solução em tempo polinomial implicaria na solução de todos os outros problemas NP-completos. Por não se saber se é possível essa solução exata, algoritmos de otimização são utilizados para aproximar soluções encontradas cada vez mais a um ótimo global.

O presente trabalho apresentou uma variante do PCV e otimizou uma solução inicial factível do problema utilizando o algoritmo *Simulated Annealing* Multi-objetivo. Em conjunto, foram implementados seis estruturas de vizinhanças com diferentes níveis de classificação cada uma. Os resultados mostraram uma robustez nos algoritmos implementados, uma vez que foram obtidas fronteiras Pareto Ótimas correspondentes visíveis. Os algoritmos mostraram um *tradeoff* entre uma fronteira Pareto-Ótima melhor representada na metodologia de Soma Ponderada e a solução global melhor no método de ϵ-Restrito. Todavia, ambos algoritmos podem ser melhor refinados ao longo dos seus estudos, uma vez que vários parâmetros podem ser testados novamente e novas estruturas de vizinhanças podem ser implementadas. Por fim, a solução multi-objetivo mostra-se mais realista para fins de uma posterior Tomada de Decisão, uma vez que considera todas as restrições existentes no problema.

Referências

[1] F. D. M. Calado and A. P. Ladeira, "Problema do caixeiro viajante: Um estudo comparativo de tÉcnicas de inteligÊncia artificial," *e-xacta*, vol. 4, no. 1, 2011.

[2] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *Journal of the ACM*, vol. 7, no. 4, pp. 326–329, 1960.

[3] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[4] K. Amine, "Multiobjective simulated annealing: Principles and algorithm variants," *Advances in Operations Research*, vol. 2019, pp. 1–13, 2019.

[5] W. Ben-Ameur, "Computing the initial temperature of simulated annealing," *Computational Optimization and Applications*, vol. 29, no. 3, pp. 369–385, 2004.

[6] D. Bertsimas and O. Nohadani, "Robust optimization with simulated annealing," *Journal of Global Optimization*, vol. 48, no. 2, pp. 323–334, 2009.

[7] G. Pantuza Júnior, "Uma abordagem multiobjetivo para o problema de sequenciamento e alocação de trabalhadores," *Gestão Produção*, vol. 23, no. 1, pp. 132–145, 2016.

[8] K. Helsgaun, "General k-opt submoves for the lin–kernighan tsp heuristic," *Mathematical Programming Computation*, vol. 1, no. 2-3, pp. 119–163, 2009.