Relatório sobre a implementação de algoritmos para o problema do caixeiro viajante e suas conclusões

Luis P. A. Afonso, Vitor A. C. Silva

Departamento de Informática – Universidade Estadual de Maringá (UEM)
Av. Colombo, 5790 – Jd. Universitário CEP 87020-900
Maringá – Pr - Brasil

{ra77429,ra104409}@uem.br

Abstract. *This report describes the implementation and testing of heuristic algorithms for the traveling salesman problem presented for the course "Algorithmic Modeling and Optimization" taught by Prof. Dr. Ademir Aparecido Constantino, as a requirement for completing the subject.*

Resumo. *Este relatório descreve a implementação e testes de algoritmos heurísticos para o problema do caixeiro viajante apresentados para a disciplina "Modelagem e otimização algorítmica" ministrada pelo Prof. Dr. Ademir Aparecido Constantino, como requisito para conclusão da matéria.*

1. Introdução

1.1. Visão geral

Este relatório tem por finalidade descrever e apresentar questões relacionadas ao problema do caixeiro viajante, sendo elas: abordar as implementações propostas de algoritmos heurísticos para obter uma solução geral viável para casos conhecidos da literatura; sendo eles implementados usando heurísticas de caráter construtivo e melhorativo (totalizando 5 algoritmos); e ainda uma comparação entre os algoritmos implementados.

1.2. Algoritmos implementados

Os algoritmos implementados são métodos heurísticos classificados em heurísticas construtivas e heurísticas melhorativas. Dentre as heurísticas construtivas foram implementados os algoritmos *Vizinho mais próximo*, *Inserção do mais distante* e *Inserção do mais barato*. Para as heurísticas melhorativas foram implementados os algoritmos *2-opt* e *3-opt*. Para fins de testes, para cada algoritmo foi criada uma versão que utiliza ou não a matriz de distâncias calculada.

1.2.1 Vizinho mais próximo

O algoritmo do Vizinho mais próximo foi um dos primeiros algoritmos construtivos utilizados para determinar uma solução para o problema do caixeiro viajante, de natureza gananciosa (ou gulosa), o algoritmo inicia o trajeto em um vértice arbitrário (neste trabalho o vértice de origem é 1) e até que todos os vértices estejam na rota, é adicionado à rota um vértice não visitado cuja aresta tenha menor custo a partir do ultimo vértice visitado.

1.2.2 Inserção do mais distante

Também um algoritmo construtivo, inicia-se o percurso com uma subrota $[v_1, v_2, v_3]$ e a cada iteração é inserido um vértice k não visitado e mais distante da subrota tal que a relação de custo (i, k) + (k, i + 1) – (i, i + 1) seja mínima.

1.2.3 Inserção do mais barato

O algoritmo construtivo *Inserção do mais barato* consiste em construir uma rota, a partir de uma subrota inicial [v_1, v_2, v_3], onde a cada iteração um vértice k não visitado é adicionado entre os vértices já visitados onde relação de custo (i, k) + (k, i + 1) – (i, i + 1) seja mínima.

1.2.4 2-opt

No algoritmo melhorativo *2-opt*, eliminam-se 2 arestas não adjacentes, reconecta-as usando duas outras arestas formando um ciclo e verifica se há melhorias. O processo é repetido para todos os pares de arestas e a melhor troca, onde há ciclo com menor custo, é realizada. Para fins de testes nossa implementação deste algoritmo é limitada a no máximo 20 melhorias.

A decisão sobre a definição do parâmetro máximo de melhorias no valor 20 foi definida em testes preliminares, onde constamos muitos pontos de estagnação na curva (i x s*) acarretando em muitos pontos de busca cega na execução do algoritmo, acarretando em maior tempo de execução sem melhoria alguma.

1.2.5 3-opt

Basicamente o mesmo procedimento que o algoritmo *2-opt*, mas são realizadas eliminações e trocas de 3 arestas não adjacentes resultando em 8 combinações possíveis, das quais 4 combinações são idênticas ao algoritmo *2-opt,* portanto, neste trabalho foram consideradas apenas 4 combinações distintas. Para fins de testes nossa implementação deste algoritmo é limitada a no máximo 1200 melhorias.

Bem como no algoritmo *2-opt*, a definição do parâmetro de máximo de melhorias foi definida em testes anteriores, a diferença entre os parâmetros do algoritmo *2-opt* e do algoritmo *3-opt* devem-se às suas entradas que são as saídas dos algoritmos construtivos *Vizinho mais próximo* e *Inserção do mais distante* respectivamente.

1.3. Casos de testes usados para validação

Os casos de testes requisitados para o relatório foram:

pr1002 (1002-city problem (Padberg/Rinaldi))

fnl4461 (Die 5 neuen Laender Deutschlands (Ex-DDR) (Bachem/Wottawa))

pla7397 (Programmed logic array (Johnson))

brd14051 (BR Deutschland in den Grenzen von 1989 (Bachem/Wottawa))

d15112 (Deutschland-Problem (A.Rohe))

d18512 (Bundesrepublik Deutschland (mit Ex-DDR) (Bachem/Wottawa))

pla33810 (Programmed logic array (Johnson))

pla85900 (Programmed logic array (Johnson))

Os resultados obtidos a partir de cada teste, assim como a execução ou não de cada um será discutido posteriormente.

1.4. Especificações das máquinas usadas para os testes

Por conta da limitação de tempo e para a execução de um teste não interferir com a execução de outro, e para com a completude do relatório, eles (os testes) foram executados em máquinas diferentes com especificações diferentes de hardware. Segue a nomenclatura de cada máquina:

Máquina 1 - Intel(R) Core(TM) i7-6500U CPU @2.50GHz, 4 Threads

Máquina 2 - Intel(R) Celeron(R) CPU N2808 @1.58GHz, 2 Threads

Máquina 3 - Intel(R) Core(TM) i7-3770 CPU @3.40GHz, 8 Threads

2. Descrição do problema

2.1. O problema do caixeiro viajante

O problema do caixeiro viajante consiste em visitar n cidades diferentes, iniciando e encerrando sua viagem na mesma cidade (neste trabalho o vértice de origem é 1). As cidades podem ser visitadas em qualquer ordem e de cada uma delas pode-se ir diretamente a qualquer outra.

O problema do caixeiro viajante consiste em descobrir a rota que torna mínima a viagem total. As cidades e distâncias entre elas são representadas por um grafo G = (V,A) onde cada vértice $v \in V$ é representado por coordenadas cartesianas (x,y) e cada aresta $c_{i,j} \in A$ é a distância euclidiana entre um vértice $v_i \in V$ e um vértice $v_j \in V$.

2.2. Implicações sobre obtenção de resultados viáveis

Formulação de Miller–Tucker–Zemlin (1960) do problema clássico do TSP:

$$\min Z = \sum_{i=1}^{n} \sum_{j=1}^{n} CijXij$$

$$\text{sujeito a} \quad \sum_{i=1}^{n} Xij = 1 \quad , \qquad j = 1, 2, \ldots, n$$

$$\sum_{j=1}^{n} Xij = 1 \quad , \qquad i = 1, 2, \ldots, n$$

$$u_i - u_j + nx_{ij} \leq n - 1, \qquad 2 \leq i \neq j \leq n,$$

$$x_{ij} \in \{0,1\} \; i,j = 1, 2, \ldots, n, \qquad i \neq j$$

$$u_i \in R^+ \; i = 1,2,\ldots,n.$$

O problema do caixeiro viajante é de difícil solução, pois nele é possível obter um grande número de soluções possíveis, principalmente quando é utilizado em grandes e diversas distâncias, considerado sua complexidade NP–Difícil (BELFIORE; FÁVERO, 2013) então, para dimensões elevadas, a resolução por métodos de programação matemática fica impeditivo em termos de tempo computacional. Portanto, a abordagem de solução heurística e aproximativa é mais útil para as aplicações que preferem o tempo de execução do algoritmo em relação à precisão do resultado.

3. Resultados obtidos por experimentações

3.1. Considerações sobre a experimentação

As máquinas que foram usadas para a testagem não eram dedicadas para isso. Contudo, durante as execuções, foi dada prioridade para o processo do teste, assim como o uso da máquina foi reduzido durante o período dos testes, de forma a buscar o desempenho ideal para tais execuções.

Ademais, mesmo com a natureza das heurísticas melhorativas propostas por esse trabalho (os algoritmos funcionam com a fase de melhoramento limitado a um máximo pré-definido) o tempo de execução de cada teste é relativamente alto. Portanto, casos com muitos vértices não foram testados, sendo eles: pla33810 e pla85900.

Para todos os algoritmos foram criadas uma versão que utiliza a matriz de distâncias previamente calculada e outra versão que não utiliza. Para fins de testes, por conta de tempo e recursos limitados, foram inclusos neste trabalho apenas os testes das versões sem a matriz de distâncias.

3.2. Gráficos para o par algoritmo construtivo vizinho mais próximo e melhorativo 2opt limitado (máximo 20 melhoramentos)

3.2.1 Caso PR1002.tsp

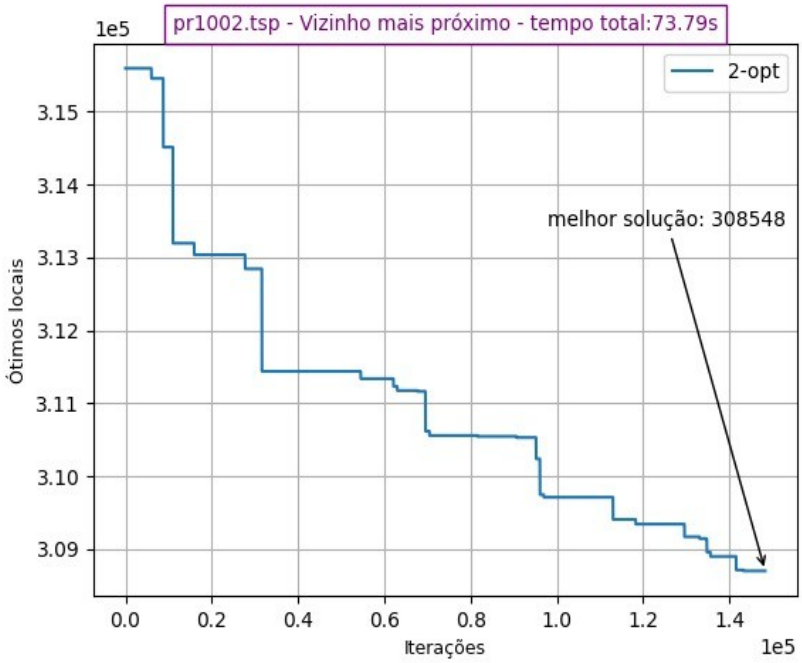

Figura 1. caso pr1002 – máquina 1

3.2.2 Caso FNL4461.tsp

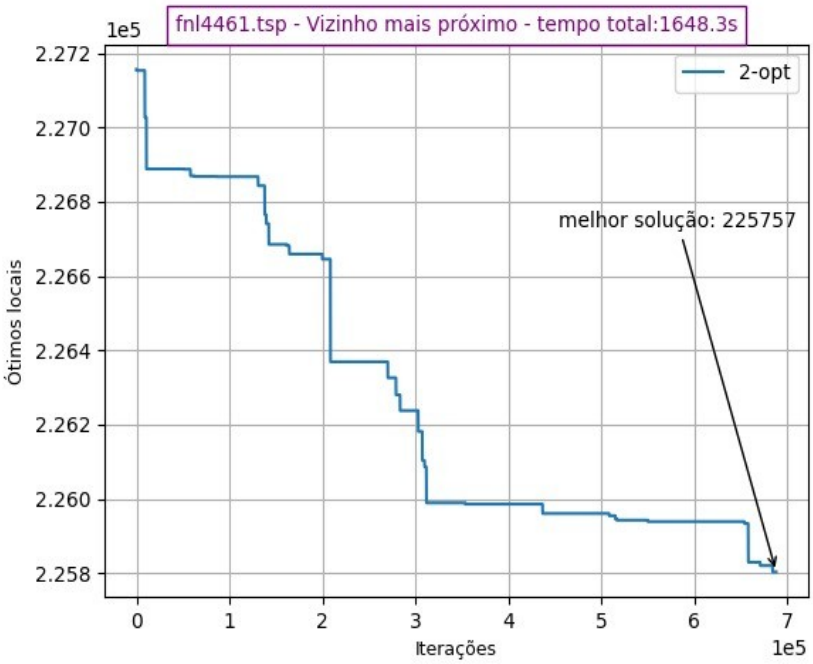

Figura 2. caso fnl4461 – máquina 1

3.2.3 Caso PLA7397.tsp

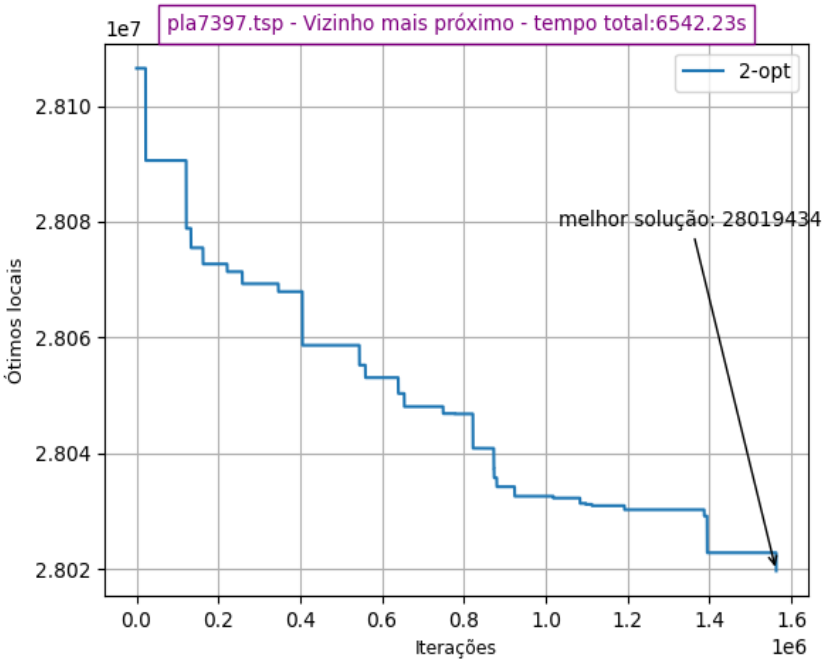

Figura 3 – caso pla7397 – máquina

3.2.4 Caso BRD14051.tsp

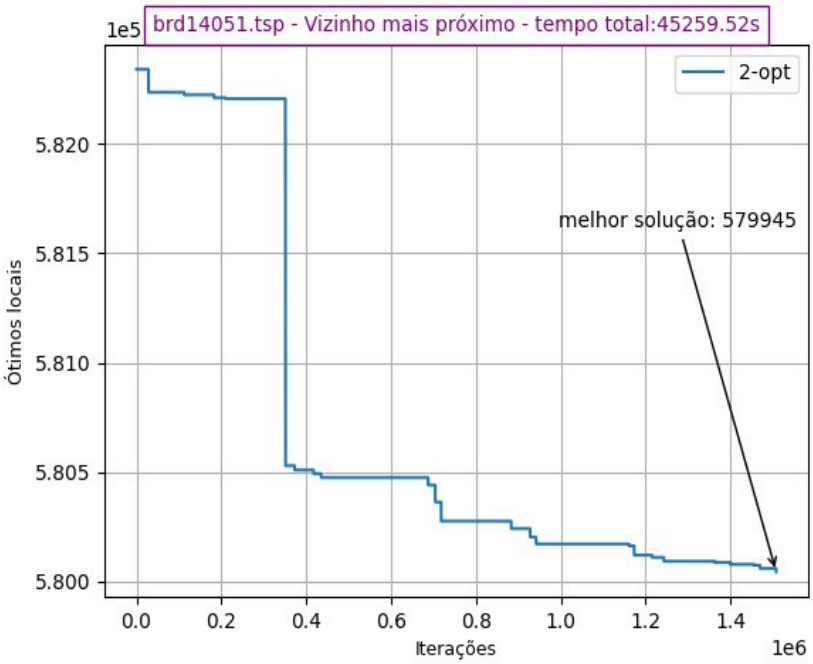

Figura 3 – caso brd14051 – máquina 2

3.2.5 Caso D15112.tsp

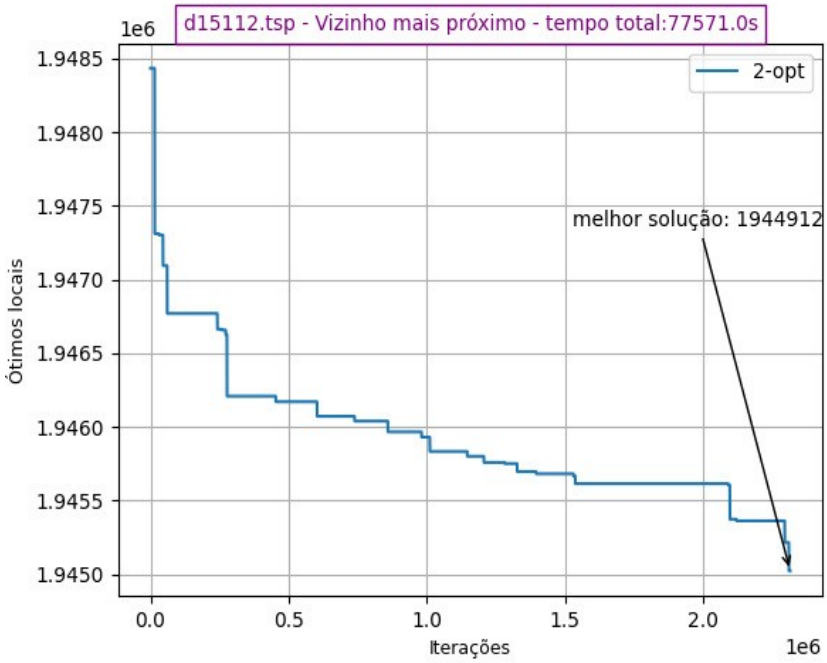

Figura 4 – caso d15112 – máquina

3.2.6 Caso D18512.tsp

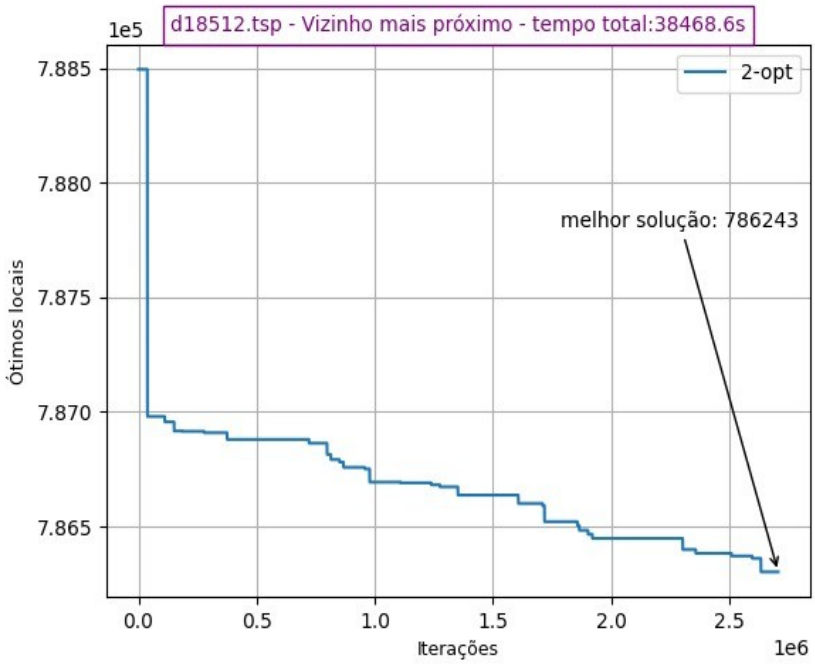

Figura 5 – caso pr18512 – máquina 2

3.3. Gráficos para o par algoritmo construtivo inserção do mais distante e melhorativo 3opt limitado (máximo 1200 melhoramentos)

3.3.1 Caso PR1002

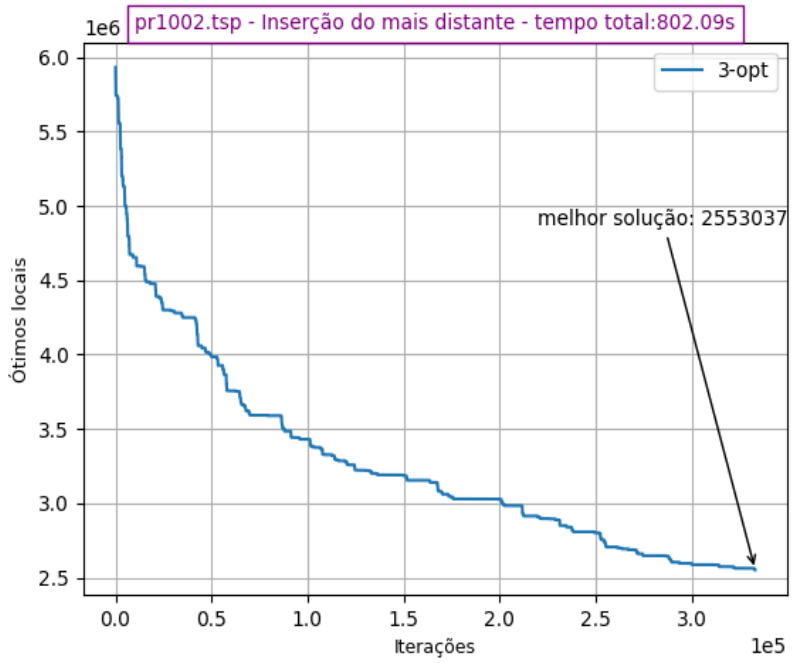

Figura 6 – caso pr1002 – máquina 3

3.3.2 Caso FNL4461

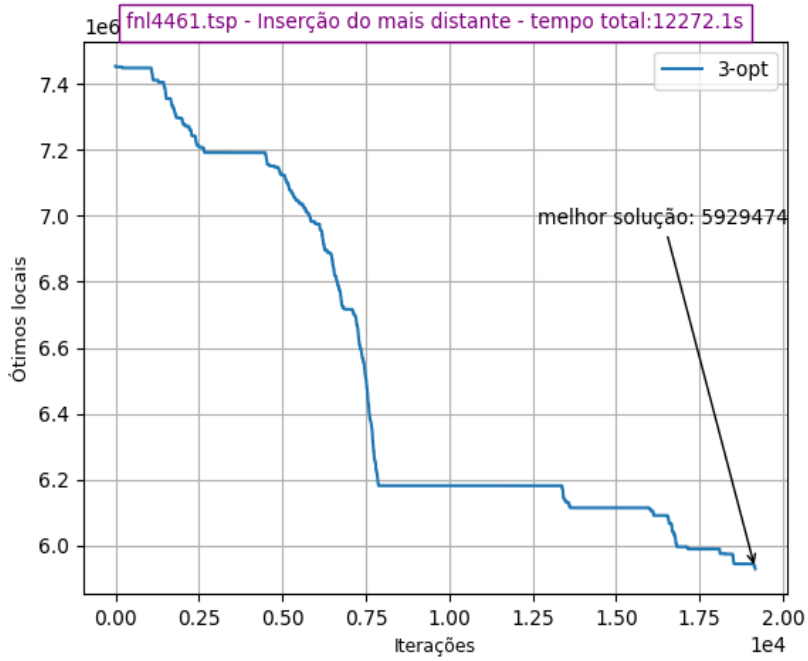

Figura 6 – caso fnl4461 – máquina 3

3.3.3 Caso PLA7307

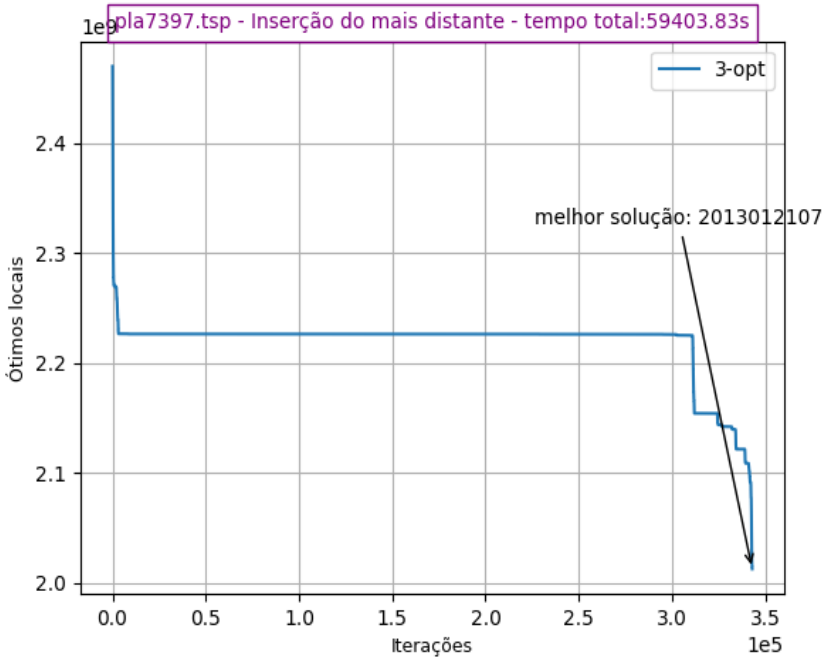

Figura 7 – caso pla7307 – máquina 3

3.3.4 Caso D15112

Caso estava sendo executado pela *máquina 1* mas foi abandonado por conta de após 627 minutos e 49 segundos ainda estar na fase construtiva. Não considerado um tempo viável para no final, alcançar uma solução indesejável.

3.3.5 Caso D18512

Caso estava sendo executado pela *máquina 1* mas foi abandonado por conta de após 1119 minutos e 23 segundos ainda estar na fase construtiva. Não considerado um tempo viável para no final, alcançar uma solução indesejável.

3.4. Tabela comparativa entre os casos

CASO	MS	VP_2OPT	ID_3OPT	GAP_VP_2OPT%	GAP_ID_3OPT%
pr1002	259045	308548	2553037	19.10	885.5
fnl4461	182566	225757	5929474	23.65	3147,8
pla7397	81438	28019434	2013012107	34305.84	2471733.9
brd14051	46985	579945	-	1134.31	-
d15112	1573084	1944912	-	23.63	-
d18512	645238	786243	-	21.85	-

4. Conclusões finais

Dadas os resultados obtidos pela fase de experimentação e o embasamento teórico, também discutido anteriormente, de que dado certos problemas a serem resolvidos computacionalmente, não é possível obter a melhor solução em tempo desejável/possível. Portanto, são propostas heurísticas, que aplicadas a esses problemas, produzem, em questão de qualidade, soluções mistas.

Ademais. Fomos levados a acreditar que existem heurísticas mais apropriadas que outras, visto que para os casos de teste o par *Vizinho mais próximo – 2opt limitado* se saíram consideravelmente melhor (mesmo que alguns casos não foram ideais, como o pla7397 e o brd14051) que o par *inserção do mais distante – 3opt limitado* tanto em tempo de execução quanto em custo final obtido. Ainda sobre o par *Vizinho mais próximo – 2opt limitado*, os casos mais apropriados mantiveram um percentual de GAP entre *19.10* e *23.65,* e os casos com resultados não adequados (possivelmente por conta da condição inicial do algoritmo, que levou a geração de um caminho pelo algoritmo construtivo em uma vizinhança com melhores locais ruins. Contudo essa afirmação é puramente especulativa.).

Por fim. Podemos concluir que a máquina usada para a execução da experimentação (no contexto deste trabalho, onde uma linguagem interpretada, Python, foi usada para a implementação dos algoritmos) foi um fator de peso no tempo de

execução, mas não no resultado final do custo obtido (por conta da característica determinista dos algoritmos). Como exemplo temos a execução do caso de teste D18512 executado pela máquina 1, que demorou aproximadamente 10 horas, e o caso D15112 executado pela máquina 2, que demorou aproximadamente 21 horas. Onde o segundo caso deveria ter rodado de forma mais rápida dado o tamanho de entrada com um número de 3 mil vértices a menos que o primeiro caso descrito.

References

Volmir E. Wilhelm. “The Travelling Salesman problem - TSP - Problema del viajante”, Departamento de engenharia de produção, UFPR. https://docs.ufpr.br/~volmir/PO_II_12_TSP.pdf.

Giuseppe Lancia, and Marcello Dalpasso. (2020). “Finding the Best 3-OPT Move in Subcubic Time”. MDPI open access journals. https://www.mdpi.com/1999-4893/13/11/306/htm.

Ademir Constantino. “Grafos Hamiltoniano e o Problema do Caixeiro Viajante – PCV”, Departamento de Informática, UEM. Disponibilizado aos alunos desta disciplina por meio da ferramenta Moodle.