UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE
DEPARTAMENTO DE COMPUTAÇÃO

THIAGO RIBEIRO VALENTIM MARTINS

MÉTODO DOS PONTOS ROLANTES: UMA NOVA META-HEURÍSTICA BASEADA EM FÍSICA

ALEGRE - ES
2021

THIAGO RIBEIRO VALENTIM MARTINS

MÉTODO DOS PONTOS ROLANTES: UMA NOVA META-HEURÍSTICA BASEADA EM FÍSICA

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Geraldo Regis Mauri.

ALEGRE - ES

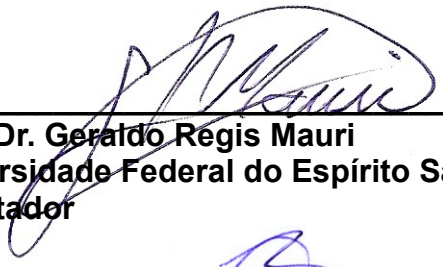
2021

THIAGO RIBEIRO VALENTIM MARTINS

MÉTODO DOS PONTOS ROLANTES: UMA NOVA META-HEURÍSTICA BASEADA EM FÍSICA

Trabalho de conclusão de curso apresentado ao Departamento de Computação do Centro de Ciências Exatas, Naturais e da Saúde da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 01 de setembro de 2021.



Prof. Dr. Geraldo Regis Mauri
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Dayan de Castro Bissoli
Universidade Federal do Espírito Santo

Prof. Dr. Edmar Hell Kampke
Universidade Federal do Espírito Santo

ALEGRE – ES

SETEMBRO 2021

AGRADECIMENTOS

Agradeço primeiramente meus pais, os quais me possibilitaram ter toda a educação, preparo e incentivos necessários para estar aqui.

À Natália, minha noiva, por sempre me dar forças para perseverar mesmo nos momentos mais difíceis. Sem seu apoio, provavelmente eu não concluiria este trabalho.

Ao meu orientador, Geraldo, que prontamente me auxiliou, corrigiu e guiou sempre que necessário, obrigado por ser zeloso nos detalhes.

Ao meu irmão, Caio, que sempre me inspirou a ser o melhor profissional possível.

A todos amigos e colegas, que estiveram comigo ao longo desse percurso e sabem como não foi fácil.

Finalmente, a todos professores que fizeram parte de minha graduação, que estejam certos que levarei comigo os valores, incentivos e conhecimento que proporcionaram.

RESUMO

Diversos problemas das áreas de engenharia, *design*, automatização e sistemas de informação podem ser entendidos como problemas de otimização. Os problemas de otimização são baseados no princípio de maximização ou minimização de uma ou mais funções-objetivo. Alguns algoritmos de solução exata são conhecidos para problemas específicos. No entanto, grande parte dos problemas de otimização são NP-Difíceis, ou seja, não se conhece algoritmos de tempo polinomial para encontrar o ótimo global. Sob essa perspectiva, as meta-heurísticas são alternativas de alto nível para encontrar soluções de qualquer tipo de problema de otimização, sem o acoplamento às particularidades dos mesmos, fornecendo bons resultados em tempos aceitáveis. Uma grande quantidade de meta-heurísticas é inspirada na natureza, como os Algoritmos Genéticos e Colônia de Formigas. Entretanto, ao longo dos últimos anos, diversas meta-heurísticas foram desenvolvidas com inspiração na física, desde o clássico *Simulated Annealing* até o mais recente e elegante *Big Bang - Big Crunch Optimization*. O presente trabalho desenvolve uma nova meta-heurística baseada em conceitos da cinética clássica para resolver o Problema da Latência Mínima, que é comprovadamente da classe NP-Difícil e de ampla aplicação, como no roteamento de veículos, distribuição de ajuda humanitária em catástrofes, distribuição de bens de consumo, escalonamento de processos, entre outros. O algoritmo desenvolvido, denominado *Método dos Pontos Rolantes*, simula que cada solução do problema é um ponto com massa, energia cinética e posição, e conforme soluções vizinhas são geradas, os pontos se movem ao longo do espaço de solução, acumulando energia cinética. O objetivo final é avaliar a meta-heurística proposta, em termos de performance, com os trabalhos anteriores sobre a mesma classe de problemas.

Palavras-chave: Meta-heurística, Física, Problemas de Latência Mínima, Otimização Combinatória.

ABSTRACT

Several problems in the areas of engineering, design, automation and information systems can be understood as optimization problems. Optimization problems are based on the principle of maximizing or minimizing one or more objective functions. There are some exact solution algorithms for specific problems. However, most optimization problems are NP-Hard, that is, there are no polynomial time algorithms to find the global optimum. From this perspective, meta-heuristics are high-level alternatives to find solutions to any type of optimization problem, without coupling to their particularities, providing good results in acceptable time. Many meta-heuristics are inspired by nature, such as Genetic Algorithms and Ant Colony. However, over the past few years, several meta-heuristics have been developed with inspiration from physics, from the classic Simulated Annealing to the most recent and elegant Big Bang - Big Crunch Optimization. The present work develops a novel meta-heuristic based on classical kinetics concepts to solve the Minimum Latency Problem, which is demonstrably of the NP-Hard class and of wide application, such as in vehicle routing, disaster relief distribution, distribution of consumer goods, process scheduling, and others. The developed algorithm, named Rolling Points Algorithm, simulates that each solution to the problem is a point with mass, kinetic energy and position, and as neighboring solutions are generated, the points move along the solution space, accumulating kinetic energy. The final objective is to evaluate the proposed meta-heuristic, in terms of performance, with previous work on the same class of problems.

Keywords: Metaheuristics, Physics, Minimum Latency Problems, Combinatorial Optimization.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Diversos problemas de áreas técnicas como processamento de imagens, roteamento de veículos, *design* de sistemas mecânicos e pesquisa operacional podem ser entendidos como problemas de otimização. Os problemas de otimização são baseados no princípio de maximização ou minimização de uma função-objetivo $f(x): x = (x_1, \dots, x_n)^t \in \tau \subset \mathbb{R}^n$ em que τ é o espaço de soluções possíveis e x_i as variáveis de decisão, geralmente sujeitos a um conjunto de restrições que reflete as restrições do mundo real (DRÉO *et al*. 2006).

Melhorn e Sanders (2007) apresentam diversos algoritmos “eficientes” para resolução de problemas específicos como *Shortest Path Problem* (SPP) e *Spanning Tree Problem* (STP), nos quais por “eficiente” se entende como algoritmos que finalizam sua execução em tempo polinomial, ou seja, se existe um polinômio p tal que o tempo de execução do algoritmo para uma entrada n seja $O(p(n))$. No entanto, grande parte dos problemas de otimização são NP-Difícies, ou seja, não se conhece algoritmos de tempo polinomial para sua resolução (ARAUJO *et al*., 2018). Um desse problemas é o Ciclo Hamiltoniano, um problema do tipo NP-Completo cujo objetivo é, dado um grafo $G = (V, E)$, determinar se é possível encontrar um ciclo que visita cada vértice $v \in V$ exatamente uma vez (EJOV *et al.*, 2006). Algoritmos convencionais baseados em gradiente não encontram bons resultados para esse problema, frequentemente caindo em ótimos locais. Nesse contexto, as meta-heurísticas superam os limites de otimização local e encontram bons resultados em tempo aceitável para uma ampla gama de problemas de otimização (GHOLIZADE *et al.*, 2020).

Diferentemente das heurísticas para solução de problemas específicos, as meta-heurísticas propõe metodologias de alto nível para a resolução de qualquer tipo de problema de otimização, sem precisar se adaptar às particularidades dos mesmos (BOUSSAID *et al*., 2013). Por conta disso, há aproximadamente 40 anos, as primeiras meta-heurísticas foram desenvolvidas e a eficiência desses métodos tem sido demonstrada repetidas vezes, desde o pioneiro e mais simples método *Simulated Annealing* (SA) proposto por Kirkpatrick *et al*. (1983), que é uma meta-heurística de solução única baseada no resfriamento de metais, até formulações mais complexas como o *Multi-Verse Optimizer* (MVO), que é baseado em conceitos da cosmologia

como buracos negros, buracos brancos e buracos de minhoca (MIRJALILI; MIRJALILI; HATAMLOU, 2015)

Salcedo-Sanz (2016) apresenta em seu extenso trabalho que apesar de a maioria das inspirações para meta-heurísticas bem-sucedidas serem a biologia, como os Algoritmos Genéticos e Colônia de Formigas, recentemente algoritmos de otimização baseados em física não-linear se mostram capazes de explorar o espaço de solução dos problemas de maneira completamente diferente, permitindo-os apresentar desempenho consideravelmente melhor que os algoritmos já existentes.

Com base nesse contexto, o presente trabalho tem como objetivo apresentar, testar e avaliar uma nova meta-heurística baseada em Física Newtoniana, aplicada no Problema de Latência Mínima (PLM) em grafos – um problema da classe NP-difícil (SAHNI e GONZALEZ, 1976) – e comparar sua eficiência com outros métodos de otimização.

1.2 OBJETIVOS

Os objetivos gerais e específicos do presente trabalho são descritos abaixo.

1.2.1 Objetivos Gerais

Este trabalho tem como objetivo geral apresentar uma nova meta-heurística baseada em conceitos da física, que não seja dependente de muitos parâmetros definidos pelo usuário e que apresente performance equivalente ou superior aos métodos do estado-da-arte, aplicando-a em Problemas da Latência Mínima que possui um conjunto de instâncias como *benchmark*.

1.2.2 Objetivos Específicos

a) Desenvolver uma nova meta-heurística baseada na cinética física de Newton com as devidas adaptações;

b) Realizar testes computacionais com o algoritmo desenvolvido para avaliar seu desempenho;

c) Comparar o desempenho da meta-heurística com outros algoritmos da literatura;

d) Avaliar as possíveis melhorias da solução em trabalhos futuros.

1.3 ESTRUTURA DO TRABALHO

O Capítulo 2 aborda a revisão da literatura, que está dividia em duas partes: a primeira analisa as meta-heurísticas baseadas em conceitos físicos que foram desenvolvidas ao longo dos últimos 20 anos; enquanto que a segunda parte trata da revisão de trabalhos propostos para resolução de PLMs e outros problemas similares. O Capítulo 3 discorre sobre a metodologia proposta para desenvolvimento da meta-heurística baseada em física e, a seguir, o Capítulo 4 aborda os testes computacionais e os resultados obtidos. O Capítulo 5 apresenta as conclusões, seguido pelas referências bibliográficas utilizadas.

2. REVISÃO DA LITERATURA

Para garantir o embasamento teórico e revisão bibliográfica consistentes, os artigos citados no presente trabalho foram selecionados com alguns critérios considerados relevantes para o assunto, utilizando duas principais ferramentas de pesquisa de trabalhos acadêmicos: *Google Scholar* e *Science Direct*.

O primeiro critério de avaliação foi a relação entre o artigo e as palavras-chaves pesquisadas, enquanto que o segundo critério baseou-se na quantidade de citações. Por último, foram analisadas as correlações de resultados entre as duas plataformas, dando prioridade aos trabalhos que apareceram em ambas. Cada conjunto de palavras-chave foi pesquisado de maneira idêntica nas duas plataformas, em inglês e português, utilizando-se como filtro o ano de publicação para garantir a confiabilidade e a atualização dos resultados.

A primeira palavra-chave pesquisada foi *Physics-based metaheuristics*, com filtro do ano de publicação entre 2015 e 2019. No *Google Scholar* foram obtidos mais de 10.000 trabalhos como resultado, dentre os quais seis estavam intimamente relacionados com o tema proposto pela palavra-chave. Esses artigos foram encontrados nas duas primeiras páginas de pesquisa, sendo que quatro destacaram-se pela completude e número de citações, como apresentado na Tabela 1.

Tabela 1 – Resultados da busca na plataforma *Google Scholar* para a palavra-chave *physics-based metaheuristic*.

Título	**Número de Citações**
Multi-Verse Optimizer: a nature-inspired algorithm for global optimization	258
Modern meta-heuristics based on nonlinear physics processes: A review of models and design procedures	39
Physics Based Metaheuristic Algorithms for Global Optimization	19
Physics - based search and optimization Inspirations from nature	9

No *Science Direct*, a mesma palavra-chave foi pesquisada, em inglês e português, utilizando como filtro o ano de publicação entre 2015 e 2020, obtendo como resultado 687 trabalhos. Destes, apenas dois destacaram-se pela relação com a pesquisa proposta, e são apresentados na Tabela 2. Nenhum desses trabalho apareceu em ambas as plataformas.

Tabela 2 – Resultados da busca na plataforma *Science Direct* para a palavra-chave *physics-based metaheuristic.*

Título	**Número de Citações**
Multi-scale quantum harmonic oscillator algorithm for global numerical optimization	3
A new Newton metaheuristic algorithm for discrete performance-based design optimization of steel moment frames	1

A segunda palavra-chave pesquisada foi *minimum latency problem* (problema da latência mínima), com e sem o acréscimo de *metaheuristics* no final, utilizando também o filtro do ano de publicação entre 2015 e 2020. No *Google Scholar* foram obtidos 2.300 resultados totais, dos quais cinco apresentaram maior relevância pelos critérios já mencionados anteriormente. Os títulos dos artigos são apresentados na Tabela 3.

Tabela 3 – Resultados da busca na plataforma *Google Scholar* para a palavra-chave *minimum latency problem.*

Título	**Número de Citações**
A simple and effective metaheuristic for the Minimum Latency Problem	52
Exact algorithms for the minimum latency problem	29
The Latency Location-Routing Problem	8
A DVND Local Search Implemented on a Dataflow Architecture for the Minimum Latency Problem	3
A mixed integer formulation and an efficient metaheuristic procedure for the k-Travelling Repairmen Problem – 1	1

Na plataforma *Science Direct*, por sua vez, foram obtidos 409 resultados, utilizando como filtro o ano de publicação entre 2015 e 2020. Deste montante, três artigos foram selecionados e dois destacaram-se por terem sido encontrados também no *Google Scholar*. Os títulos dos artigos são apresentados na Tabela 4.

Tabela 4 – Resultados da busca na plataforma *Science Direct* para a palavra-chave *minimum latency problem.*

Título	**Número de Citações**
A simple and effective metaheuristic for the Minimum Latency Problem	52
A multi-start procedure for the Minimum Latency Problem	11
The Latency Location-Routing Problem	8

Além dos trabalhos mencionados, outros artigos foram consultados ao longo da pesquisa para garantir o correto e consistente embasamento dos conceitos, estudos e fatos apresentados.

2.1 TRABALHOS CORRELATOS DE META-HEURÍSTICAS BASEADAS EM FÍSICA

Sacco e Oliveira (2005) afirmam que muitos algoritmos de otimização, como o *Simulated Annealing* (*SA*) e Algoritmo Genético (GA), frequentemente caem em ótimos locais e dependem de parâmetros definidas pelo usuário. Para resolver tais problemas, os autores modelaram uma meta-heurística que se assemelha ao SA, com menos variáveis ajustadas pelo usuário. O algoritmo denominado *Particle Collision Algorithm* (*PCA*) foi inspirado nas reações de fusão nuclear, principalmente nas etapas de espalhamento (um nêutron incidente é espalhado pela colisão com um núcleo) e absorção (o nêutron incidente é absorvido pelo núcleo alvo). A solução dos autores foi utilizada no problema de engenharia chamado *nuclear core design optimization*, que consiste em otimizar diversos parâmetros das células de um reator para minimizar os custos e maximizar a segurança do processo.

Luz *et al*. (2008) sugeriram melhorias ao PCA, implementando-o para multi-partículas (M-PCA) com o intuito de otimizar a rotina de busca de soluções e evitar cair em ótimos locais, metodologia que proporcionou melhores resultados em todos os testes realizados. Abuhamdah e Ayo (2009) também utilizaram o PCA, aplicando-o em problemas de Calendário de Cursos, obtendo melhores resultados que o SA em algumas instâncias.

No mesmo ano de publicação do PCA, o *Big Bang – Big Crunch Search (BB-BC)* foi apresentado por Erol e Eksin (2005). O BB-BC foi inspirado pela teoria de que o universo passa por ciclos de expansão (*Big-Bang*) e, posteriormente, encolhimento (*Big Crunch*). Analogamente, o algoritmo possui duas fases: a de expansão, na qual várias soluções aleatórias são geradas, e a de encolhimento, em que todas as soluções são reduzidas à uma "singularidade" – uma solução ótima local. Anos mais tarde, Alatas (2011) melhorou a convergência do algoritmo introduzindo um mapa caótico durante a fase do *Big Crunch*.

O *Big Bang – Big Crunch Algorithm* foi aplicado em um método híbrido por Kaveh e Talatahari (2009) em um problema de otimização de design de treliças, comparando os resultados com outros métodos como *Simulated Annealing, Colônia de Formigas* e *Harmony Search Algorithm.* Genc e Hocaoglu (2008) aplicaram o algoritmo em problemas de *Bearing-Only Target Motion Analysis (BO-TMA)* e concluíram que o BB-BC convergiu mais rapidamente que outros métodos computacionais evolutivos.

Formato (2007) apresentou em seu artigo uma nova meta-heurística para otimização multi-dimensional, baseada na cinética gravitacional dos corpos, nomeada *Central Force Optimization Algorithm* (*CFO*). O CFO busca o espaço de soluções por meio de "sondas" que são atraídas por outros corpos de massa maior por influência da gravidade, que possui uma formulação matemática bem definida. Por esse motivo, ele é considerado uma meta-heurística determinística. Esse método, apesar de não possuir prova de convergência, demonstrou ser de grande utilidade em trabalhos posteriores (ASI e DIB, 2010). Para melhorar a capacidade de convergência do CFO, Qian *et al*. (2015) introduziram pesos e constante gravitacional adaptativos, baseado na teoria de sistemas dinâmicos de variação de tempo discreta, demonstrando obter resultados ainda melhores que as reformulações anteriores do CFO.

Rabanal *et al*. (2008) construíram uma meta-heurística de comportamento semelhante ao Colônia de Formigas. O algoritmo baseia-se na formação de rios pelas gotas de água que erodem o solo e depositam sedimentos, alterando os níveis do solo e consequentemente o caminho que as gotas de água passarão. O processo termina quando todas as gotas percorrem o mesmo caminho, o que indica a melhor solução da execução atual. A meta-heurística proposta (*River Formation Dynamics Algorithm – RFDA)* foi testada pelos autores em problemas do Caixeiro Viajante, ou *Traveling Salesman Problem* (*TSP*), comparando os resultados com uma implementação do algoritmo de colônia de formigas. Em ambos grafos dinâmicos e estáticos, o ACO performou melhor que o RFDA no quesito tempo. No entanto, o RFDA encontrou soluções melhores a longo prazo. Dois anos após esse trabalho, Rabanal *et al*. (2010) também implementaram o RFDA para o problema *Minimum Steiner Tree Problem* (*STP*), onde em 84.2% dos experimentos a diferença entre a solução ótima e a fornecida pela meta-heurística foi menor que 5%, representando assim como a melhor taxa de performance até aquele momento do artigo para um algoritmo de tempo polinomial.

O *Eletromagnetism-like Algorithm* (*EMA*) foi proposto por Birbil e Fang (2003) como uma meta-heurística inspirada nas leis do eletromagnetismo, que utiliza da atração e repulsão para mover as soluções para o ponto ótimo. Wang *et al*. (2008) utilizaram o EMA no problema de treinamento de redes neurais de classificação. Como discutido em seu trabalho, técnicas de busca local geralmente caem em ótimos locais para esse tipo de problema, o que foi a base da motivação da utilização do algoritmo. Sua performance foi comparada com o *Back Propagation* e o Algoritmo Genético, demonstrando maior acurácia e menor tempo de computação que ambos. Tan *et al*. (2016) implementaram técnicas de Espalhamento, Sondagem e Comparação no algoritmo tradicional (*Split, Compare and Probe - SCP)* que substitui o método de busca local anterior, obtendo resultados melhores em todas as comparações realizadas.

Inspirados pelo trabalho de Spears *et al*. (2005) que apresentaram o *Artificial Physics Framework***,** Xie *et al*. (2009a) propuseram a meta-heurística *Artificial Physics Optimization Algorithm* (*APO*), que utiliza da simulação do framework proposto pelos

autores anteriores para modelar a solução de um problema de otimização como uma partícula *p* que possui massa, velocidade e aceleração causada pela força de atração gravitacional. Nesse algoritmo, a massa das partículas corresponde ao resultado da solução para o problema, ou seja, quanto maior a massa melhor é a solução representada por aquela partícula. As partículas se movem em direção às áreas do espaço de solução que possuem maior massa, e para longe das áreas de menor massa, que por consequência leva a áreas próximas da solução global. Posteriormente, Xie *et al*. (2009b) desenvolveram um modelo vetorizado do APO que facilitou a análise e construção das soluções.

Mais recentemente, Javidy *et al*. (2015) publicaram o *Ions Motion Algorithm* (IMO), cujo objetivo era criar uma meta-heurística com o menor número de hiper-parâmetros possíveis e baixa complexidade. O algoritmo foi baseado na dinâmica física de íons com carga positiva (Cátions) e carga negativa (Ânions), sendo cada partícula carregada uma solução do problema. Nesse *framework,* as forças eletrostáticas de atração e repulsão movem as partículas, o que representa a busca no espaço de soluções. O objetivo ao final de cada iteração do algoritmo é encontrar o melhor cátion e o melhor ânion, calcular sua força de atração e atualizar a posição de todas as demais cargas. Para implementar a diversificação e convergência, os autores modelaram uma fase líquida, onde os íons se movem mais livremente e podem explorar o espaço de soluções, e uma fase sólida, onde as soluções se consolidam e convergem a um novo valor ótimo. Uma importante aplicação do IMO pode ser observada pelo trabalho de Issa e Elaziz (2020), que o aplicaram como uma ferramenta para auxiliar no processo de análise biológica do COVID-19, obtendo resultados melhores que as metodologias anteriores.

Buch e Trivedi (2020) apresentaram o *Non-dominated Sorting Ions Motion Algorithm* (NSIMO), uma modificação do IMO com duas principais contribuições: Durante a fase de cristalização dos íons, uma força externa de impacto é aplicada aos cristais para desfazê-los em novas soluções, um mecanismo matemático para evitar que as mesmas fiquem presas a ótimos locais. Além disso, eles aplicaram o conceito de classificação “não-dominada”, que organiza as soluções em grupos de acordo com a qualidade relativa do valor de sua função objetivo. O desempenho da meta-heurística modificada foi medido comparando-a com a execução de outros métodos recentes

como o *Non-dominated sorting moth flame optimization* (NSMFO) apresentado por Savsani e Tawhid (2017) e o *Multi-objective sine-cosine algorithm* (MO-SCA) concebido pelo trabalho de Tawhid e Savsani (2019), além de outros problemas de *benchmark*. Ao todo foram executados oito problemas irrestritos, seis restritos e seis problemas de design de engenharia, obtendo resultados melhores ou equivalentes aos algoritmos comparados.

Mesmo com diversos métodos de otimização já existentes, ainda há espaço para o desenvolvimento de novas meta-heurísticas que podem demonstrar maior eficiência para a resolução de problemas específicos. Como apresentado por Wolpert e Macready (1997) em seu “Teorema do Almoço Grátis” (*No free lunch theorem),* não existe um algoritmo que resolva de maneira ótima todos os problemas de otimização e, portanto, sempre haverá espaço para pesquisa e desenvolvimento de novos algoritmos.

2.2 TRABALHOS CORRELATOS AO PROBLEMA DA LATÊNCIA MÍNIMA

O Problema da Latência Mínima - PLM (*Minimum Latency Problem*), também conhecido como *Delivery Man Problem* (TSITSIKLIS, 1992), é uma variante do problema do Caixeiro Viajante (*Traveling Salesman Problem* - TSP) cujo objetivo é encontrar um ciclo ou circuito hamiltoniano em um grafo que minimize a latência entre os vértices. A latência de um determinado vértice é a soma dos custos das arestas de um caminho entre o vértice atual e o vértice inicial do ciclo – denominado v_0. E, a latência total do grafo é definida como o somatório das latências de cada vértice do ciclo, i.e. $lt = \sum_{i=0}^{n} l(v_i)$, sendo lt a latência total do ciclo. A Figura 1 representa a latência de um determinado vértice v_2, enquanto que a Figura 2 representa a latência total de um ciclo hamiltoniano.

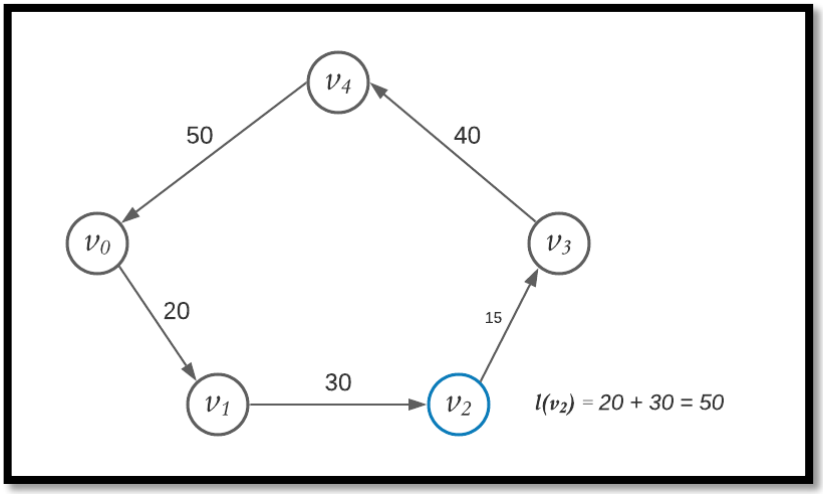


Figura 1 – Latência de um vértice v_2 em um dado ciclo hamiltoniano.

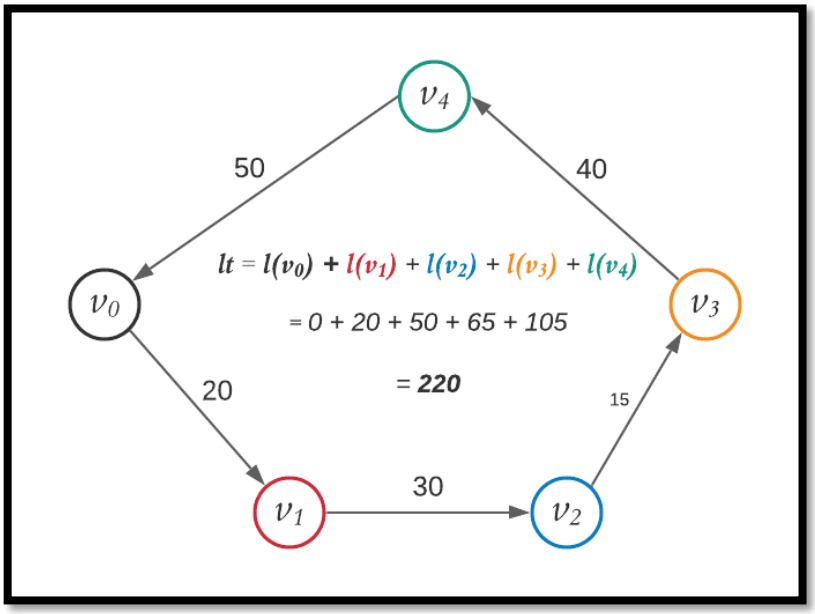


Figura 2 – Latência total de um ciclo hamiltoniano.

É importante destacar que a latência do vértice inicial v_0 é 0, ou seja, a última aresta do ciclo é ignorada para o cálculo da latência total do grafo. O PLM, assim como o TSP, é um problema NP-difícil (SAHNI e GONZALEZ, 1976).

Um dos primeiros trabalhos a tratarem o PLM foi o de Lucena (1990), que propôs um algoritmo exato (*branch-and-bound)* baseado em uma formulação não-linear obtida por meio de relaxação Lagrangiana, que basicamente dividia os limites inferiores em diversos componentes e otimizava cada um destes.

Wu *et al*. (2004) desenvolveram um algoritmo exato para superar as dificuldades de técnicas anteriores, como o *Branch-and-Bound (B&B)* e a programação dinâmica, que apesar de serem capazes de resolver esses problemas em tempos computacionais menores que a força bruta O(*n*!), ainda assim demandavam tempos não-polinomiais para a maioria dos problemas. Os autores introduziram uma técnica de "podagem" das soluções geradas pelo algoritmo de programação dinâmica, salvando o limite superior para a otimalidade e o limite inferior de cada solução gerada. Os resultados do algoritmo foram testados em dados reais e dados gerados aleatoriamente, obtendo soluções melhores e em menor tempo que o B&B e a programação dinâmica puros.

Silva *et al*. (2012), sob a perspectiva de que os algoritmos exatos até o momento não eram capazes de resolver consistentemente PLMs com mais de 100 vértices, desenvolveram duas contribuições para o problema: uma meta-heurística híbrida, baseada no *Greedy Randomized Adaptive Search Procedure* (GRASP), *Iterated Local Search* (ILS) e *Random Variable Neighborhood Descent* (RVND), e um tipo de movimento entre as soluções com complexidade O(1). A solução proposta pelos autores utiliza de apenas 3 hiper-parâmetros definidos pelo usuário. Testes foram realizados em instâncias com mais de 1000 vértices, obtendo boas soluções com pouca demanda computacional. Soluções ótimas para problemas conhecidos de até 107 vértices também foram obtidas em poucos segundos.

Angel-Bello *et al*. (2013), um ano após o trabalho de Silva *et al*. (2012), desenvolveram uma heurística de inicialização múltipla para resolver PLMs propostos anteriormente na literatura. Sua solução utiliza o GRASP para a geração de múltiplas soluções iniciais que são armazenas em um *pool* de soluções e, em seguida, aplica busca local em cada uma delas para melhorar a qualidade das mesmas. Além disso, foi aplicado um algoritmo de pós-processamento em cada par de soluções finais chamado de *Path Relinking*, que gera novas soluções a partir da combinação dos caminhos dos pares selecionados. A meta-heurística construída pelos autores foi aplicada em duas classes de instâncias: uma gerada aleatoriamente pelos autores e outra retirada do trabalho de Salehipour *et al*. (2008) e Negueveu *et al.* (2010), utilizando instâncias de 10 a 100 vértices. Por fim, os autores compararam os resultados com o algoritmo GRASP+VND (SALEHIPOUR *et al*., 2008) e o Algoritmo Memético (NGUEVEU *et al*.,

2010), obtendo melhores resultados em tempo de processamento e qualidade das soluções.

Mafort e Ochi (2016) exploraram o problema da latência mínima por meio de aplicação do Algoritmo de Colônia de Formigas (ACO). Nesse tipo de algoritmo, a fase construtiva das soluções é realizada por uma modelagem matemática e computacional do comportamento de formigas, que percorrem os vértices de um grafo de maneira aleatória, depositando feromônios no caminho em quantidade proporcional à qualidade da solução. Com o passar das iterações, as formigas tendem a escolher os caminhos que possuem maior quantidade de feromônio e, consequentemente, a melhor solução. No entanto, foi introduzido um fator aleatório na escolha do vértice inicial pelas formigas para evitar o elitismo precoce. O método de busca local utilizado leva em consideração as melhores soluções da iteração anterior, o que teve impacto na qualidade final da solução. Para melhorar a complexidade do algoritmo durante a fase de avaliação das soluções, foi utilizada uma técnica de otimização baseada no trabalho de Kindervater e Savelsbergh (1997) e estendido por Vidal *et al*. (2011) e Vidal (2013), que consiste em usar de programação dinâmica, uma estrutura de dados em matriz para representar as soluções e um operador de concatenação para evitar o cálculo redundante de custos de nós que se repetem.

Diversos trabalhos na literatura basearam-se no *k*-PLM, que é uma variante do PLM, com a diferença de que existem *k* “entregadores” ou “veículos” que devem percorrer cada um dos *n* vértices de um grafo uma única vez, ou seja, cada veículo deve formar *p* caminhos disjuntos de forma a minimizar o tempo total de espera de cada nó. Esse tipo de problema, como explicitado por Méndez-Díaz *et al*. (2008), possui alta aplicabilidade em cenários reais onde é necessário distribuir algum tipo de serviço ou produto para vários clientes, com o foco na satisfação dos usuários desses bens. A Figura 3 apresenta uma visualização desse tipo de problema com três tipos de veículos percorrendo rotas distintas.

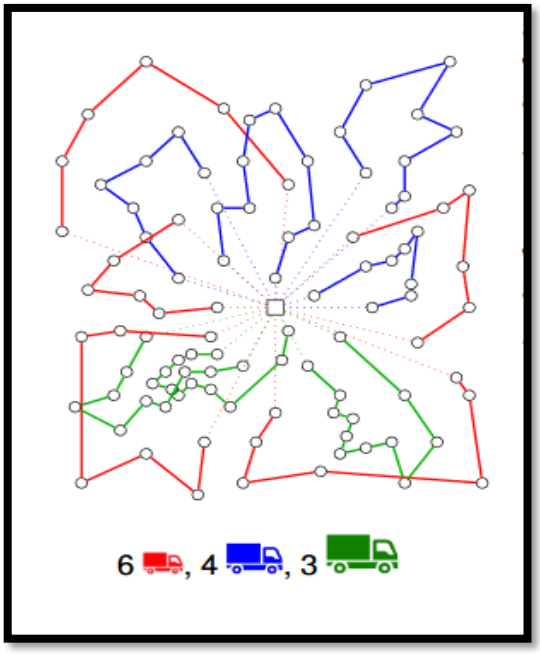


Figura 3 – Problema da classe *k*-PLM com 3 tipos de veículos.
Fonte: Sadykov *et al*. (2018).

Ban e Nguyen (2016) discorrem sobre o *k*-PLM e apresentam uma meta-heurística baseada no GRASP para construir as soluções iniciais e o *Variable Neighborhood Descent* (VND) para refinar as soluções geradas na fase anterior. O refinamento das soluções consiste principalmente em duas etapas: a primeira é a aplicação de 6 tipos de movimentos para encontrar soluções vizinhas (4 de aplicação em uma única rota e 2 de aplicação inter-rotas), e a segunda é o procedimento de agitação (*shaking procedure)*, onde dada uma rota R_i altera-se aleatoriamente a posição de cada nó dentro da mesma. O algoritmo encerra o seu processamento quando não há melhoria da função objetivo após *NL* iterações, parâmetro escolhido pelo usuário. A meta-heurística foi aplicada a *benchmarks* retirados do *Networking and Emerging Optimization* (NEO)[1], filtrando os problemas que possuíam entre 25 e 1200 nós. Os resultados demonstraram uma melhora entre 10,5% a 15,7% em instâncias pequenas, e significativas melhoras de 23,66% a 60% em instâncias grandes.

Nucamendi-Guillén *et al*. (2016) também apresentaram um trabalho sobre o *k*-PLM, reforçando a importância e aplicação desse tipo de problema em áreas como roteamento de veículos, distribuição de ajuda humanitária em áreas de desastre, entrega e coleta de produtos perecíveis e escalonamento de processos em máquinas.

[1] https://neo.lcc.uma.es/vrp/vrp-instances/

O artigo propõe uma formulação inteira mista com uma meta-heurística simples que se baseia no *Iterated Greedy Algorithm*, um caso particular de *Multi-Start Methods* (MS) em que cada nova solução é gerada a partir da "destruição" de uma solução anterior e conseguinte "reconstrução" de uma nova rota. A fase construtiva da meta-heurística é uma adaptação do algoritmo de construção paralela de rotas em conjunto com o *Generalized Regret Measure*, propostos por Potvin e Rousseau (1993). A fase de melhoria consiste em cinco estratégias distintas de busca local divididas entre movimentos inter-rotas e movimentos intra-rotas. A solução proposta conseguiu resolver instâncias até 3 vezes maiores que as resolvidas pela formulação KEE (KARA *et al*, 2008; EZZINE e ELLOUMI, 2012) e, nas instâncias de mesmo tamanho, obteve o resultado ótimo até 1000 vezes mais rápido.

No âmbito dos métodos exatos, Bulhões et. al (2018) utilizaram o método *Branch-and-Price-and-Cut* para resolver problemas de PLM. Até aquele momento, métodos exatos para esse tipo de problema não eram capazes de resolver instâncias com 150 nós, enquanto que instâncias de TSP com milhares de nós são resolvidas frequentemente por meta-heurísticas (ABELEDO *et al*., 2013). Por esse motivo, os autores propuseram a utilização de estratégias eficientes como *reduced cost fixing*, *dual stabilization, route enumeration* e *strong branching*. Além disso, foi utilizada uma relaxação de caminhos no grafo chamada de *ng-paths*, uma metodologia introduzida por Baldacci *et al*. (2011) que "memoriza" as vizinhanças de um determinado nó, cuja performance foi melhorada por Roberti e Mingozzi (2014). Os autores basearam-se no método usado por Roberti e Mingozzi (2014) com adição de três melhorias: *Multiple Partial Label Dominance*, *Arc-based ng-path Relaxation* e *Fully Dynamic ng-path Relaxation*. Os resultados alcançados pela execução do método excederam os algoritmos do estado da arte, resolvendo todas as 9 instâncias não resolvidas por Roberti e Mingozzi (2014) e outras 4 instâncias nunca consideradas anteriormente para métodos exatos (u159, si175, br180 e rat195).

Dada a considerável relevância de problemas de PLM e suas variações, como *k*-PLM*,* além da intrínseca complexidade da resolução desse tipo de problema, na qual uma pequena variação local da posição dos nós pode afetar a solução de maneira global, o presente trabalho intenta utilizar a nova meta-heurística proposta na resolução de

instâncias desse problema, comparando sua performance tanto com métodos exatos quanto heurísticos.

3. METODOLOGIA

A metodologia adotada neste trabalho foi de caráter exploratório e incremental, desenvolvendo-se uma meta-heurística baseada em conceitos de física newtoniana denominada Método dos Pontos Rolantes, ou *Rolling Points Algorithm* (RPA), em inglês. Além disso, outras técnicas de otimização foram empregadas para potencializar os resultados. O algoritmo desenvolvido pode ser divido em quatro principais fases, que serão detalhadas adiante:

1. Fase Exploratória
2. Busca Local Simples
3. Fase de Convergência
4. Busca Local Profunda

3.1 FASE EXPLORATÓRIA

A fase exploratória do algoritmo é a primeira etapa do mesmo, e caracteriza-se pela geração inicial de α soluções que serão utilizadas pelas etapas posteriores. A heurística construtiva utilizada nessa etapa foi aleatória, começando pelo vértice inicial V_0 e definindo como próximo vértice da solução um aleatório dentre os vértices que ainda não foram escolhidos. O pseudo-código a seguir exemplifica esse processo:

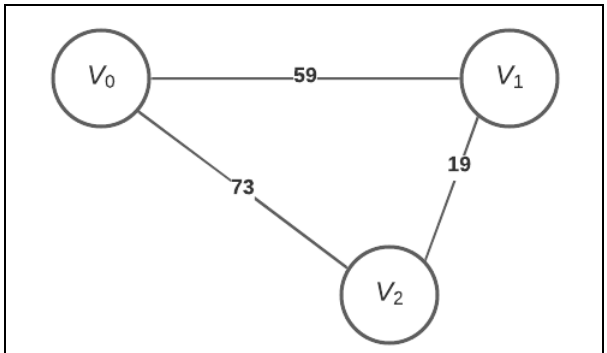
```
1  Input α, V, M
2  Output α soluções construídas com a heurística aleatória
3
4  solucoes = Ø
5  Para i = 0 Até α - 1 Faça
6      novaSolucao = constroiSolucaoAleatoria(M, V)
7      solucoes = { solucoes, novaSolucao }
8
9  Retorna solucoes
```

Figura 4 – Pseudocódigo da fase exploratória.
Fonte: O autor.

- V: Número de vértices do problema.
- M: Matriz de distâncias entre cada vértice.

A matriz de distâncias é obtida com base nas distâncias entre cada vértice do problema, sendo considerado que a distância de um vértice *i* para um vértice *j* é

representada pela posição M[*ij]* dessa matriz e, além disso, que essa distância é igual na direção oposta M[*ji*]. A figura abaixo ilustra a relação entre a matriz de distâncias e o grafo do problema:



0	59	73
59	**0**	19
73	19	**0**

Figura 5 – Exemplo de matriz de solução de um grafo.
Fonte: O autor.

Observa-se que a diagonal principal é constituída de zeros pois a distância de um vértice para ele mesmo é nula. Além disso, no contexto da heurística deste trabalho, cada solução do problema é considerada um corpo físico que possui uma massa. O valor da massa de cada solução pode ser inicializado de duas maneiras distintas, ambas baseadas em um delimitador *m*, definido pelo usuário:

1. Massa **aleatória**: cada solução é gerada com uma massa que varia entre 1 e *m*.
2. Massa **estática**: o valor da massa de toda solução gerada é *m*.

3.2 BUSCA LOCAL SIMPLES

A busca local simples é realizada pelo algoritmo com o intuito de melhorar a qualidade das soluções geradas na fase exploratória. Essa etapa consiste em gerar vizinhos aleatórios para cada solução do problema até que a condição de parada seja atingida. As vizinhanças utilizadas foram:

- **SWAP** - Troca a posição de um par aleatório de nós do vetor de solução

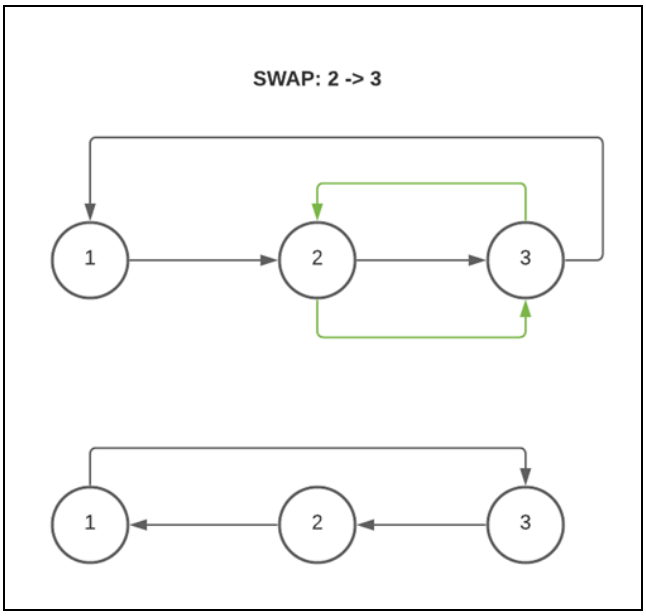


Figura 6 – Exemplo de vizinhança SWAP.
Fonte: O autor.

- **Reinsertion** - Examina uma possível reinserção de um nó em outra posição do ciclo. Quando o nó é reinserido, todos os demais nós são realocados uma posição na solução.

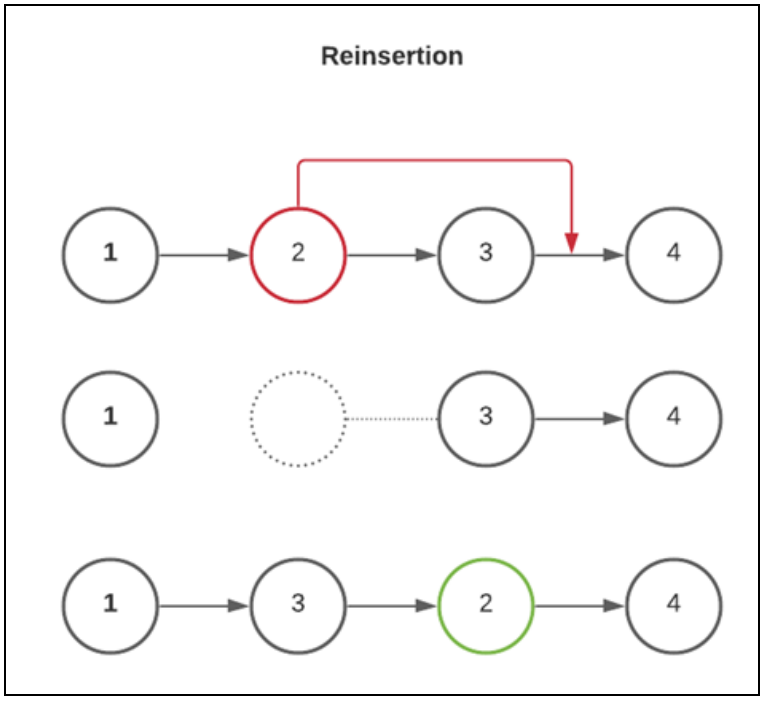


Figura 7 – Exemplo de vizinhança Reinsertion.
Fonte: O autor.

- **2-Optimal** - Remove duas arestas não-adjacentes do grafo, depois reconstrói o ciclo hamiltoniano novamente, religando os caminhos que sobraram de maneira que a solução continue viável.

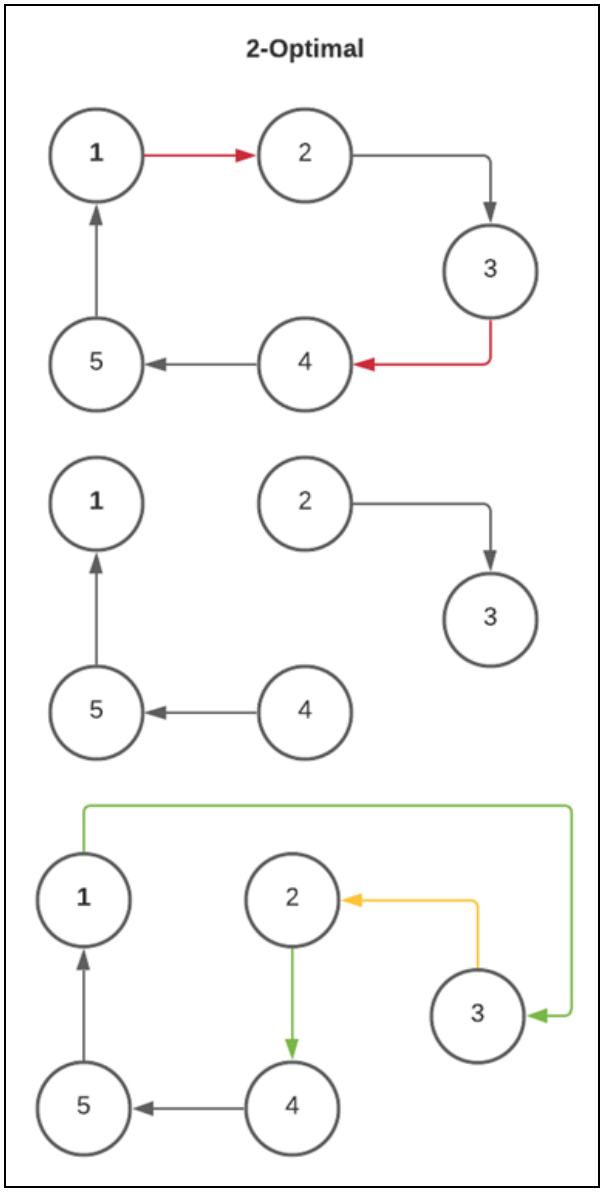


Figura 8 – Exemplo de vizinhança 2-Optimal.
Fonte: O autor.

- **Or-Opt 2** – Move dois vértices adjacentes para uma nova posição no vetor de solução. Quando a troca é realizada, os demais vértices são reajustados para que o ciclo continue válido.

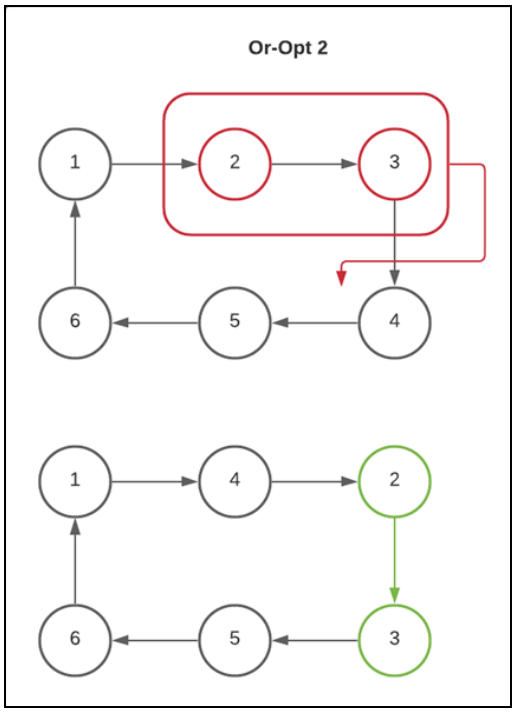


Figura 9 – Exemplo de vizinhança Or-Opt 2.
Fonte: O autor.

- **Or-Opt 3** - Uma variação da vizinhança Or-Opt 2, no entanto, no lugar de um par de vértices adjacentes serem movidos, um trio de vértices é selecionado.

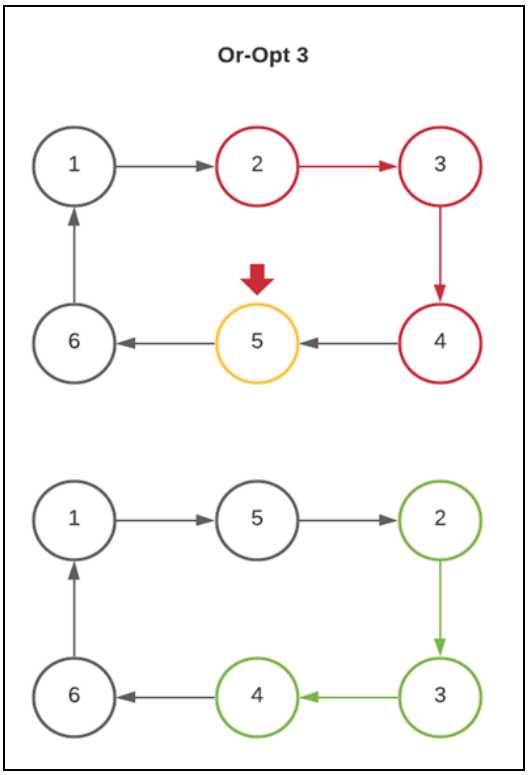


Figura 10 – Exemplo de vizinhança Or-Opt 3.
Fonte: O autor.

A condição de parada para a geração de vizinhanças é o ponto principal deste trabalho, que no lugar de definir um tempo máximo de execução ou número máximo de iterações, se baseia na energia cinética acumulada por cada solução. Os detalhes dessa modelagem são descritos na próxima seção.

3.2.1 Algoritmo para condição de parada

A analogia fundamental dessa fase é de que cada solução de um problema de otimização seja considerada um ponto em um determinado gráfico, onde o eixo *x* representa o espaço de soluções e o eixo *y* representa o valor da função objetivo. À priori, a meta-heurística aplica-se intuitivamente em problemas de minimização. No entanto, os conceitos podem ser facilmente convertidos em problemas de maximização, invertendo o sinal do cálculo de diferença entre duas “alturas” da solução. A Figura 11 representa uma solução em meio à um espaço de soluções desconhecido.

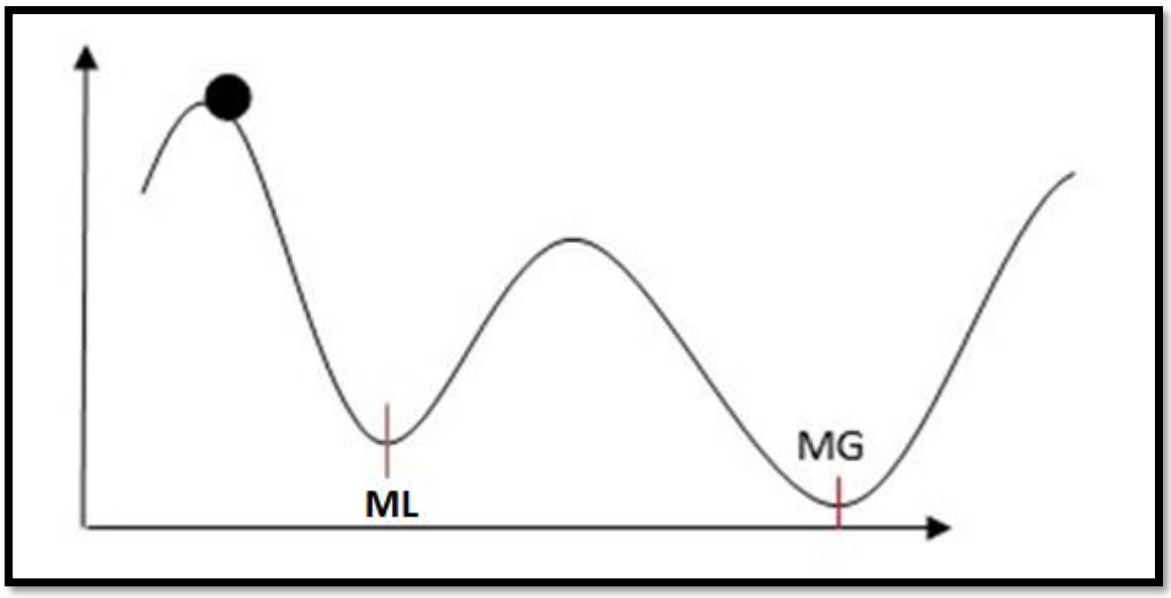


Figura 11 – Gráfico representando uma solução aleatória de um problema de otimização, o mínimo local e o mínimo global.
Fonte: o autor

Se uma solução aleatória é gerada no ponto determinado, seguindo uma heurística da primeira melhora, o ponto desceria até o mínimo local (ML), e não prosseguiria para encontrar o mínimo global (MG). No entanto, ao tratar o ponto inicial que foi gerado como um “corpo” físico sob atuação da gravidade, o mesmo desceria pelo gráfico até o mínimo local, e, por causa da energia cinética acumulada, seria capaz

de subir o morro adjacente para finalmente cair no mínimo global. A condição de parada, portanto, é quando a energia cinética do corpo chega a zero.

Para a modelagem matemática desse cenário foi considerado como prioritário a análise de proporcionalidade entre as grandezas, em contraste com a estrita conformidade com as fórmulas físicas, com o intuito de facilitar o cálculo da energia e não onerar o tempo de resolução do algoritmo. Seguindo essa abordagem, sabe-se que a energia cinética de um corpo pode ser calculada por:

$$E_c = \frac{mv^2}{2}$$

Como mencionado anteriormente, cada solução do problema já possui uma massa definida, restando ainda calcular a velocidade. Para todas as soluções, foi considerado um parâmetro ***V₀***, informada pelo usuário, que representa a velocidade inicial. Sendo assim, a energia cinética inicial pode ser calculada por:

$$E_{c_{Inicial}} = m * V_0$$

Para as demais iterações do algoritmo, é necessário calcular a velocidade média do corpo em movimento, que pode ser obtida por meio da fórmula:

$$V = \frac{\Delta S}{\Delta T}$$

Sendo que:

- ΔS: É a variação da posição de um corpo, medida em dois pontos diferentes do tempo.
- ΔT: É a variação do tempo entre as duas medições.

A variação da posição (ΔS) foi considerada como a variação percentual do valor da função-objetivo entre duas soluções vizinhas. Ou seja, dado que uma solução original S_1, com o valor da função-objetivo F_1 tenha gerado um vizinho S_2 com o valor da função-objetivo F_2, tem-se o seguinte cálculo para ΔS nessa iteração:

$$\Delta S_{(S_1,S_2)} = \frac{(F_1 - F_2)}{F_1}$$

Já a variação de tempo (ΔT) é considerada como 1 para cada vizinho gerado. Além disso, um fator de compensação **β**, definido pelo usuário, foi inserido no cálculo final da velocidade para potencializar o valor de $\Delta \boldsymbol{S}$. A equação final para a velocidade fica, portanto:

$$V = \frac{\Delta S}{\Delta T} * \beta$$

$$V = \frac{(F_1 - F_2)}{F_1} * \beta$$

Um possível problema relacionado a essa modelagem é o caso do "ponto" cair em um platô, ou seja, as soluções vizinhas geradas pelo algoritmo sempre retornam o mesmo valor de FO. Nesse caso, o valor de ΔS sempre seria zero e a energia cinética nunca decairia. A solução implementada foi utilizar um parâmetro **μ** para configurar um coeficiente de atrito do espaço que ele percorre, o que faz com que o corpo não se mova indefinidamente.

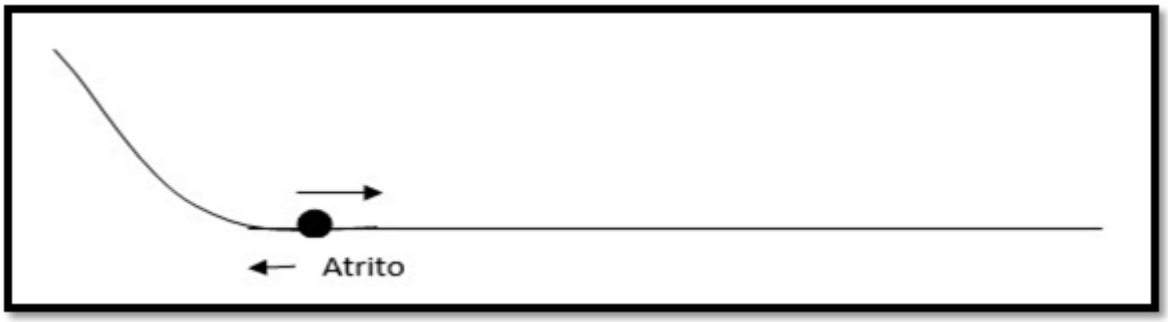


Figura 12 – Solução movendo-se em direção ao platô e consequente ação do atrito no corpo.
Fonte: o autor

Considerando essa nova variável, o cálculo final para o valor da velocidade de uma solução fica:

$$V = \frac{(F_1 - F_2)}{F_1} * \beta - \mu$$

E, por fim, a energia cinética de uma solução é calculada por:

$$E_c = \frac{m * \left(\frac{(F_1 - F_2)}{F_1} * \beta - \mu\right)^2}{2}$$

Como o cálculo da energia cinética é feito diversas vezes pelo algoritmo, optou-se por omitir o fator quadrático da velocidade e também a divisão por dois, chegando finalmente ao cálculo da energia cinética de uma solução:

$$E_c = m * \left(\frac{(F_1 - F_2)}{F_1} * \beta - \mu\right)$$

O cálculo acima é efetuado toda vez que uma solução vizinha é gerada para uma solução original, e o resultado desse cálculo é somado à energia cinética atual da solução. Dessa forma, caso uma solução pior seja gerada, ocorre um decaimento da energia cinética, e, caso uma solução melhor seja gerada, a energia cinética é incrementada. Sempre que uma solução melhor é encontrada, a mesma substitui a solução original. O pseudo-código abaixo ilustra esse comportamento:

```
1  Input S, V_0, α, M, V
2  Output Conjunto de soluções S melhoradas
3
4  Para i = 0 Até α - 1 Faça
5      solucao = S[i]
6      energiaCineticaInicial = V_0 * solucao.massa
7      energiaCinetica = energiaCineticaInicial
8
9      Enquanto energiaCinetica > 0 Faça
10         ultimaFo = solucao.funcaoObjetivo
11         vizinho = gerarVizinhoAleatorio(solucao, M, V)
12         novaFo = vizinho.funcaoObjetivo
13         Se ( novaFo < ultimaFo ):
14             S[i] = vizinho
15
16         energiaCinetica = energiaCinetica + calcularEnergia(solucao, ultimaFo)
17
18 Retorna S
```

Figura 13 – Pseudocódigo da fase de busca local simples.
Fonte: O autor.

Onde:

- S: É o conjunto de soluções geradas na fase exploratória
- V_0: É o valor da velocidade inicial definida pelo usuário
- α: É o número de soluções geradas na fase exploratória

- M: É a Matriz de distâncias entre cada vértice.
- V: É o Número de vértices do problema.

O procedimento apresentado acima é efetuado para cada solução do conjunto *S* para possibilitar uma maior diversidade na fase de exploração. A ideia principal é de que a partir de cada solução se possa explorar uma parte diferente do espaço de soluções, aumentando as chances de encontrar o mínimo global. Na Figura 14 é ilustrado o funcionamento dessa analogia.

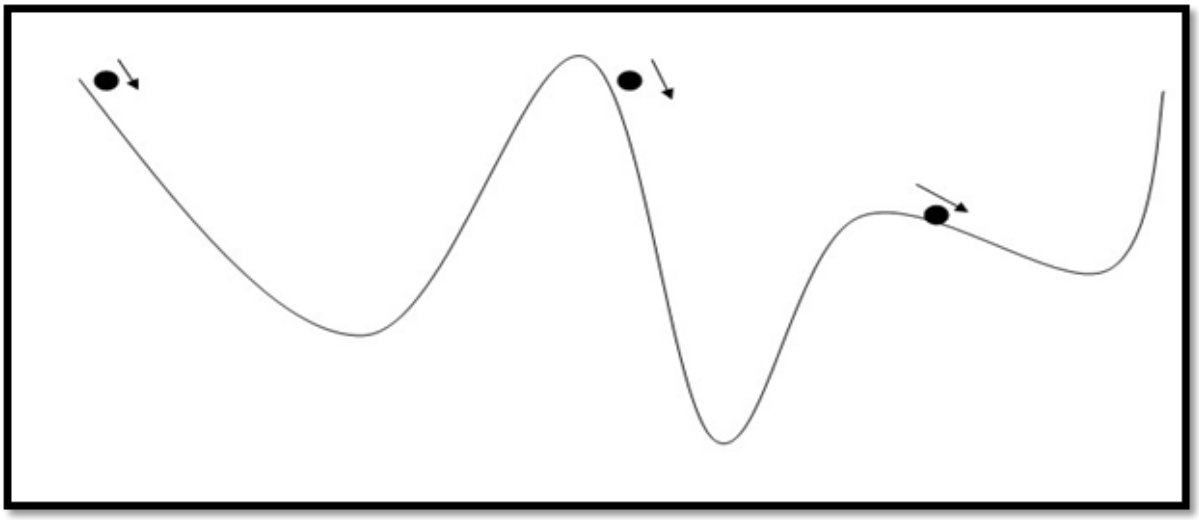


Figura 14 – Vários pontos que percorrem o espaço de soluções em busca do mínimo global.
Fonte: O autor.

3.3 FASE DE CONVERGÊNCIA

Após a exploração do espaço de solução pelo algoritmo descrito no tópico anterior, a heurística implementada busca encontrar a melhor solução. Para isso, foi utilizado o algoritmo de ordenação *Quick Sort*, considerado um algoritmo complementar ao *Merge Sort*, que utiliza a estratégia “dividir e conquistar” implementada de maneira rápida (complexidade do caso médio de $O(\log n)$) e com pouco custo de memória (MELHORN e SANDERS, 2007). Dessa maneira, a solução com o melhor valor da FO é obtida e repassada para a etapa final da heurística.

3.4 BUSCA LOCAL PROFUNDA

A última etapa do algoritmo consiste em realizar uma busca local mais robusta com a solução resultante da etapa anterior, utilizando o método *Variable Neighborhood Descent (VND)*. O VND é uma meta-heurística que consiste em explorar diversas

vizinhanças de maneira dinâmica, podendo ser definida uma sequência $N_1, N_2, \ldots, N_{max}$ a ser visitada ou decidindo-se aleatoriamente, avaliando a melhor melhora em cada uma. Quando uma vizinhança é totalmente explorada sem nenhuma melhora na solução, a próxima da lista é selecionada e esse processo é repetido até que condição de parada seja atingida (TALBI, 2009).

No presente trabalho, decidiu-se adicionar um fator aleatório ao VND (tornando-o *Random Variable Neighborhood Descent - RVND*), o que significa que quando uma vizinhança é totalmente explorada, a próxima selecionada seja uma aleatória dentre as vizinhanças remanescentes. Sempre que uma solução melhor é encontrada ao finalizar a exploração de uma vizinhança, o algoritmo RVND é reiniciado. As vizinhanças utilizadas foram as mesmas definidas na Seção 3.1, a saber:

- Swap
- Reinsert
- Or-Opt2
- Or-Opt3
- 2-Opt

```
1  Input solucao, M
2  Output Melhor Solução do algoritmo
3
4  vizinhancas = gerarVizinhancaAleatoria()
5  Para Cada vizinho Em vizinhancas Faça
6      novaSolucao = explorarVizinho(solucao, vizinho)
7      Se (novaSolucao.funcaoObjetivo < solucao.funcaoObjetivo)
8          solucao = novaSolucao
9          Volte Para (linha 4)
10
11 Retorna solucao
```

Figura 15 – Pseudocódigo da fase de busca local profunda.
Fonte: O autor.

O método **gerarVizinhancaAleatoria()** cria um conjunto contendo todas as estratégias de vizinhanças citadas acima, ordenadas de maneira aleatória. Dessa forma, a exploração dos vizinhos é sempre “reiniciada” quando uma solução melhor é encontrada. O algoritmo é encerrado quando todas as vizinhanças do conjunto “vizinhancas” são exploradas e não há nenhuma melhora. O pseudo-código abaixo demonstra uma visão geral do algoritmo implementado, com todas as fases presentes.

```
1  Input V_0, α, β, μ, m, M, V
2  Output Melhor solução
3
4  S = faseExploratoria(M, V, α)
5  buscaLocalSimples(S, V_0, α, M, V)
6
7  melhorSolucaoAtual = quickSort(S)
8
9  melhorSolucaoDefinitiva = RVND(melhorSolucaoAtual, M, V)
10
11 Retorna melhorSolucaoDefinitiva
```

Figura 16 – Pseudocódigo do Algoritmo dos Pontos Rolantes.
Fonte: O autor.

4. EXPERIMENTOS COMPUTACIONAIS

A meta-heurística desenvolvida nesse trabalho foi testada em instâncias de *Traveling Salesman Problem* apresentadas por Mafort e Ochi (2016), que por sua vez foram retiradas do TSPLib (REINELT, 1991). A escolha dessas instâncias deu-se pela estrutura em grafo se adequar bem ao problema da latência mínima e por serem uma coleção de instâncias comumente utilizadas para comparação na literatura.

As instâncias foram escolhidas com base no número de nós, de maneira crescente, até o limite de 439 nós. Para 7 das 9 instâncias, o algoritmo foi executado 100 vezes, e nas duas maiores (lin318 e pr439) foi executado 10 vezes.

O Método dos Pontos Rolantes foi implementado na linguagem C/C++, utilizando o ambiente MinGW versão 6.4, executando em uma máquina com o sistema operacional Windows 10 *Professional Edition* versão 20H2, CMAKE versão 3.19.2 e GCC versão 8.1.0. O hardware consiste em memória RAM DDR3 de 16GB (1600MHz) e processador Intel(R) Core(TM) i5-4690 de 3.50GHz.

Os cinco parâmetros do algoritmo, V_0 para velocidade inicial dos pontos, α para a população de soluções, β para o fator de compensação, μ para o coeficiente de atrito e *m* para massa das soluções, foram obtidos por processo empírico de teste e repetição do algoritmo em cada instância. Os valores iniciais foram decididos arbitrariamente e, a partir da primeira execução, cada parâmetro foi incremetado até que uma piora no valor médio da função objetivo ou do tempo de execução ocorresse. Os valores finais que melhor demonstraram um equilíbrio entre tempo de execução e qualidade das soluções foram:

- V_0 = 100
- α = 50
- β = 10
- μ = 1.5
- *m* = 10

Obs: O valor de *m* foi considerado constante para **todas** as soluções do problema.

Para comparar de maneira mais justa o tempo de execução do algoritmo proposto neste trabalho com os demais, utilizou-se a técnica apresentada em Souto *et al* (2021),

que considera um fator de proporção entre as CPUs utilizadas. Para tal, foi considerada a métrica *Single Thread Rating (STR)* de cada processador, que representa milhões de operações efetuadas por segundo, dividindo o valor de cada trabalho anterior pelo valor da métrica para a CPU utilizada neste. A Tabela 6 ilustra essas medidas, onde **STR** representa a métrica mencionada acima e **Fator** representa a proporção.

Tabela 5 – Resultados computacionais do Método dos Pontos Rolantes para as instâncias supracitadas.

	Salehipour *et al.* (2011)[a]	Silva *et al.* (2012)[b]	Mafort e Ochi (2016)[c]	Rolling Points Algorithm[d]
STR	385	1402	1973	2206
Fator	0,17	0,89	0,63	1

[a] Pentium 4 com 2.4GHz, 512 MB de RAM
[b] i7 de 2.93 GHz, 8GB de RAM – GNU/Linux Ubuntu 10.04
[c] i7 de 2.4 GHz, 8GB de RAM – Windows 10 (C#)
[d] i5 4690 de 3.5GHz, 16GB de RAM - Windows 10 Professional Edition 20H2

4.1 RESULTADOS OBTIDOS

A Tabela 7 exibe os resultados do algoritmo desenvolvido neste trabalho, separada em 4 colunas. A coluna **Instância** representa um subconjunto das instâncias do TSPLib (REINELT, 1991) utilizadas para os testes computacionais, ordenadas crescentemente pelo número de nós. A coluna **Melhor FO** representa os melhores valores de função-objetivo alcançados pelo algoritmo para cada instância. A coluna **Média FO** representa a média aritmética das soluções geradas e a coluna **Desvio (%)** apresenta o desvio percentual das médias em relação às melhores soluções, sendo que: Desvio = ((Média FO - Melhor FO) / Melhor FO) x 100. Por fim, a coluna **T. (s)** representa a média aritmética do tempo de execução do algoritmo, medido em segundos, já calculado considerando o fator citado acima.

Os resultados da Tabela 7 são referentes ao trabalho de Salehipour *et al*. (2011) em comparação com os valores obtidos neste trabalho. A coluna **DIF** representa a diferença percentual entre a melhor solução obtida pelo Algoritmo dos Pontos Rolantes e a melhor solução do trabalho anterior, sendo calculada pela fórmula:

$$DIF = \frac{(S_2 - S_1)}{S_1}$$

Sendo:

- S_1: Valor da função objetivo da melhor solução do trabalho anterior;
- S_2: Valor da função objetivo da melhor solução do trabalho atual.

Tabela 6 – Resultados computacionais do Método dos Pontos Rolantes para as instâncias supracitadas.

Instâncias	Melhor FO	Média FO	Desvio (%)	T. (s)
st70	19.710	22.682,10	15,08	0,36
rat99	57.414	59.814,61	4,18	1,43
kroD100	952.728	1.021.170,15	7,18	1,64
lin105	587.357	634.464,82	8,02	2,05
pr107	1.982.040	2.053.670,64	3,61	2,80
rat195	227.167	230.968,23	1,67	22,15
pr226	7.124.078	7.439.480,12	4,43	80,28
lin318	5.943.382	6.152.400,00	3,52	233,20
pr439	19.246.546	19.618.200,20	1,93	721,67
Média	4.015.602,44	4.136.983,43	5,51	118,40

Os valores em negrito de FO, média e DIF representam os melhores valores obtidos, comparando o presente trabalho com o de referência.

Tabela 7 – Comparação dos resultados deste trabalho com o de Salehipour *et al*. (2011).

Inst.	Salehipour *et al.* (2011)		ROLLING POINTS ALGORITHM			DIF
	Melhor FO	T. (s)	Melhor FO	Média FO	T. (s)	
st70	**19.553**	0,37	19.710	22.682,10	0,36	0,80%
rat99	**56.994**	2,44	57.414	59.814,61	1,43	0,74%
kroD100	976.830	0,82	**952.728**	1.021.170,15	1,64	**-2,47%**
lin105	**585.823**	1,96	587.357	634.464,82	2,05	0,26%
pr107	1.983.475	6,88	**1.982.040**	2.053.670,64	2,80	**-0,07%**
rat195	**213.371**	18,22	227.167	230.968,23	22,15	6,47%
pr226	7.226.554	39,93	**7.124.078**	7.439.480,12	80,28	**-1,42%**
lin318	**5.876.537**	69,99	5.943.382	6.152.400,00	233,20	1,14%
pr439	**18.567.170**	88,80	19.246.546	19.618.200,20	721,67	3,66%
Média	**3.945.145,22**	25,49	4.015.602,44	4.136.983,43	118,40	1,01%

Observa-se que o Algoritmo dos Pontos Rolantes obteve melhores resultados de FO em três das nove instâncias. Além disso, a diferença média entre os valores de FO é de aproximadamente 1% acima, o que demonstra pouca alteração dos ótimos encontrados.

A Tabela 8 exibe a comparação dos resultados com o trabalho de Silva *et al.* (2012), que utilizaram o mesmo conjunto de instâncias em seus testes computacionais.

Tabela 8 – Comparação dos resultados deste trabalho com o de Silva *et al.* (2012).

Inst.	Silva *et al*. (2012)		ROLLING POINTS ALGORITHM			DIF
	Melhor FO	**T. (s)**	**Melhor FO**	**Média FO**	**T. (s)**	
st70	**19.215**	0,96	19.710	22.682,10	0,36	2,58%
rat99	**54.984**	6,16	57.414	59.814,61	1,43	4,42%
kroD100	**949.594**	4,39	952.728	1.021.170,15	1,64	0,33%
lin105	**585.823**	3,93	587.357	634.464,82	2,05	0,26%
pr107	**1.980.767**	5,17	1.982.040	2.053.670,64	2,80	0,06%
rat195	**210.191**	48,02	227.167	230.968,23	22,15	8,08%
pr226	**7.100.308**	37,53	7.124.078	7.439.480,12	80,28	0,33%
lin318	**5.560.679**	140,19	5.943.382	6.152.400,00	233,20	6,88%
pr439	**17.688.561**	351,92	19.246.546	19.618.200,20	721,67	8,81%
Média	**3.794.458**	66,48	4.015.602,44	4.136.983,43	118,40	3,53%

Nesse contexto, os resultados de FO obtidos por Silva *et al*. (2012) superam o Algoritmo dos Pontos Rolantes em todas as instâncias, com as diferenças mais proeminentes nas instâncias pr439 (DIF de 8.81%), rat195 (8.08%), lin318 (6.88%), rat49 (4.42%) e st70 (2.58%). Não obstante, as diferenças entre os ótimos nos demais casos foram relativamente pequenas, abaixo de 1%, sendo que o tempo médio de 6 das 9 instâncias foi inferior para o algoritmo proposto.

A Tabela 9 compara as métricas com o trabalho mais recente de Mafort e Ochi (2016), que utilizaram de programação dinâmica e um algoritmo baseado em colônia de formigas.

Tabela 9 – Comparação dos resultados com o trabalho de Mafort e Ochi. (2016).

Inst.	Mafort e Ochi (2016)			ROLLING POINTS ALGORITHM			DIF
	Melhor FO	**Média FO**	**T. (s)**	**Melhor FO**	**Média FO**	**T. (s)**	
st70	20030,68	20411,97	0,57	**19710**	22682,10	0,36	**-1,60%**
rat99	**56989,57**	57809,65	1,88	57414	59814,61	1,43	0,74%
kroD100	**951731,45**	984153,58	1,57	952728	1021170,15	1,64	0,10%
lin105	**587015,15**	594785,60	1,82	587357	634464,82	2,05	0,06%
pr107	1984540,47	2004629,31	1,74	**1982040**	2053670,64	2,80	**-0,13%**
rat195	**221239,43**	226040,78	10,73	227167	230968,23	22,15	2,68%
pr226	**7117374,39**	7185368,02	12,41	7124078	7439480,12	80,28	0,09%
lin318	**5670374,32**	5857527,31	45,05	5943382	6152400,00	233,20	4,81%
pr439	**18128690,01**	18627114,28	145,95	19246546	19618200,20	721,67	6,17%
Média	**3859776,15**	3950871,17	22,03	4015602,44	4136983,43	118,40	1,44%

Nota-se que o Algoritmo dos Pontos Rolantes obteve melhores valores de FO para 2 das 9 instâncias (st70 e pr107), além de ter executado em tempo médio inferior na instância st70. Com exceção das três instâncias maiores (pr439, lin318 e rat195), os

resultados deste trabalho aproximaram-se dos demais valores obtidos, com a diferença percentual abaixo de 1%.

5. CONCLUSÕES

Este trabalho abordou a otimização de problemas da latência mínima, cuja aplicação pode ser observada em diversas áreas, como transporte, redes de computadores, distribuição de carga, roteamento de veículos, etc. Diversas propostas de solução para esse tipo de problema foram revisadas, tanto em métodos exatos quanto meta-heurísticas baseadas em comportamentos naturais.

O Algoritmo do Pontos Rolantes foi apresentado como uma alternativa aos métodos já existentes, caracterizando-se como um método populacional baseado em física clássica. Além do método proposto, o algoritmo de ordenação *Quick Sort* foi utilizado para selecionar a melhor solução da segunda etapa, e a busca em vizinhança variável aleatória (RVND) foi utilizada como última fase do algoritmo para melhorar a solução ótima local.

A meta-heurística proposta foi avaliada com testes computacionais executados em instâncias do TSPLib, um repositório público de instâncias para o *Traveling Salesman Problem*, que se assemelha ao problema da latência mínima. Nove instâncias foram selecionadas deste acervo, variando de 70 até 439 nós. Além disso, os resultados foram comparados com trabalhos anteriores de Salehipour *et al*. (2011), Silva *et al*. (2012) e Mafort e Ochi (2016).

Os resultados dos testes computacionais mostraram que o algoritmo proposto converge para as instâncias citadas, chegando a obter resultados melhores em três delas em comparação com o trabalho de Salehipour *et al*. (2011), chegando em melhora de até 2,47% na FO da instância com 100 nós (kroD100). Em comparação com o trabalho de Mafort e Ochi (2016), o presente trabalho obteve melhora em duas das nove instâncias, chegando a um valor de 1,60% para a instância st70.

Em contraste, todos os valores de FO obtidos pelo trabalho de Silva *et al*. (2012) foram superiores para as instâncias avaliadas, apesar de serem executados em tempo médio superior em 66,66% das instâncias. Além disso, nos problemas com mais de 226 nós, os trabalhos anteriores obtiveram resultados melhores de FO e tempo de execução.

Como possíveis melhorias do algoritmo apresentado, destaca-se a aplicação de programação dinâmica para diminuir o tempo de exploração das vizinhanças na etapa

do RVND, como avaliado por Mafort e Ochi (2016), e a paralelização do processo de busca local simples, onde o caminho percorrido por cada solução seja calculado de maneira assíncrona. Outra alternativa é a aplicação de perturbações nas soluções, como o *Double Bridge* (BOUSSAÏD *et al*, 2013), que pode melhorar o desempenho do RNVD.

REFERÊNCIAS

ABELEDO, H.; FUKASAWA, R.; PESSOA, A.; UCHOA, E. The time dependent traveling salesman problem: polyhedra and algorithm. **Mathematical Programming Computation**, v. 5, n. 1, p. 27-55, 2013.

ABUHAMDAH, A.; AYOB, M. Experimental result of particle collision algorithm for solving course timetabling problems. **International Journal of Computer Science and Network Security**, v 9, p. 134-142, 2009.

ALATAS, B. Uniform big bang–chaotic big crunch optimization**. Communications in Nonlinear Science and Numerical Simulation**, v. 16, p. 3696-3703, 2011.

ANGEL-BELLO, F.; ALVAREZ, A.; GARCÍA, I. A multi-start procedure for the minimum latency problem, **IFAC Proceedings Volumes**, v. 46, n. 9, p. 436-441, 2013.

ARAUJO, R. P.; COELHO, I. M.; MARZULO, L. A. A DVND local search implemented on a dataflow architecture for the minimum latency problem. **IEEE International Parallel and Distributed Processing Symposium Workshops**, Vancouver, BC, p. 1250-1259, 2018.

ASI, M. J; DIB, N. I. Design of multilayer microwave broadband absorbers using central force optimization. **Progress in Electromagnetics Research B**, v. 26, p. 101-113, 2010.

BALDACCI, R.; MINGOZZI, A.; ROBERTI, R. New route relaxation and pricing strategies for the vehicle routing problem. **Operations Research**, v. 59, n. 5, p. 1269-1283, 2011.

BAN, H. B; NGUYEN, D. N. An effective GRASP+VND metaheuristic for the k-minimum latency problem. **IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future**, Hanoi, p. 31-36, 2016.

BIRBIL, Ş.İ.; FANG, S. An electromagnetism-like mechanism for global optimization, **Journal of Global Optimization**, v. 25, p. 263-282, 2003.

BOUSSAÏD, I.; LEPAGNOT, J; SIARRY, P**.** A survey on optimization metaheuristics. **Information Sciences**, v. 237, p. 82-117, 2013.

BUCH, H.; TRIVEDI, I. A new non-dominated sorting ions motion algorithm: Development and applications. **Decision Science Letters**, v. 9, p. 59-76, 2020.

BULHÕES, T.; SADYKOV, R.; UCHOA, E. A branch-and-price algorithm for the Minimum Latency Problem, **Computers & Operations Research**, v. 93, p. 66-78, 2018.

DRÉO, J.; PÉTROWSKI, A.; SIARRY, P.; TAILLARD, E. **Metaheuristics for Hard Optimization: Methods and case studies**. 1 ed. Paris: *Springer-Verlag Berlin Heidelberg,* 2006.

EJOV, V.; FILAR, J. A.; LUCAS, S. K.; NELSON, J. L. Solving the hamiltonian cycle problem using symbolic determinants. **Taiwanese Journal of Mathematics,** v. 10, n. 2, p. 327-338, 2006.

EROL, O. K.; EKSIN, I. A new optimization method: Big Bang–Big Crunch. **Advances in Engineering Software**, v. 37, n. 5, p. 106-111, 2006.

EZZINE, I.; ELLOUMI, S. Polynomial formulation and heuristic based approach for the k-travelling repairman problem**. International Journal of Mathematics in Operational Research**, v. 4, n. 5, p. 503-514, 2012.

FORMATO, R. A. Central force optimization: a new metaheuristic with applications in applied electromagnetics. **Progress in Electromagnetics Research**, v. 77, p. 425-491, 2007.

GENC, H. M.; HOCAOGLU, A. K. Bearing-Only target tracking based on Big Bang – Big Crunch algorithm. **The Third International Multi-Conference on Computing in the Global Information Technology**, Athens, p. 229-233, 2008.

GHOLIZADEH, S.; DANESH, M.; GHEYRATMAND, C. A new Newton metaheuristic algorithm for discrete performance-based design optimization of steel moment frames. **Computers & Structures**, v. 234, 2020.

ISSA, M; ELAZIZ, M. A. Analyzing COVID-19 virus based on enhanced fragmented biological Local Aligner using improved Ions Motion Optimization algorithm. **Applied Soft Computing**, v. 96, 2020.

JAVIDY, B.; HATAMLOU, A.; MIRJALILI, S. Ions motion algorithm for solving optimization problems. **Applied Soft Computing**, v. 32, p. 72-79, 2015.

KARA, I; KARA, B.Y.; YETIŞ, M.K. Cumulative vehicle routing problems. **Vehicle Routing Problem**, p. 85-98, 2008.

KAVEH, A.; TALATAHARI, S. Size optimization of space trusses using Big Bang–Big Crunch algorithm. **Computers & Structures**, v. 87, p. 1129-1140, 2009.

KINDERVATER, G.; SAVELSBERGH, M. Vehicle routing: handling edge exchanges. **Local Search in Combinatorial Optimization**, p. 337-360, 1997.

KIRKPATRICK, S.; GELATT, C; VECCHI, M. Optimization by simulated annealing. **Science**, v. 220, p. 671-680, 1983.

LUCENA, A. Time-dependent traveling salesman problem - The deliveryman case. **Networks**, v. 20, p. 753-763, 1990.

LUZ, E. F. P.; BECCENERI, J. C.; CAMPOS VELHO, H. F. A new multi-particle collision algorithm for optimization in a high performance environment. **Journal of Computational Interdisciplinary Sciences**, v. 1, p. 3-10, 2008.

MAFORT, R. L.; OCHI, L. S. An efficient ant colony algorithm for the Minimum Latency Problem. **Anais do XLVIII SBPO - Simpósio Brasileiro de Pesquisa Operacional**, p. 1752-1763, Vitória, 2016.

MELHORN, K.; SANDERS, P. Algorithms and Data Structures: 1. ed. Springer-Verlag Berlin Heidelberg, 2007.

MÉNDEZ-DÍAZ, I.; ZABALA, P.; LUCENA, A. A new formulation for the traveling deliveryman problem. **Discrete Applied Math**, v. 156, p. 3223-3237, 2008.

MIRJALILI, S.; MIRJALILI, S.; HATAMLOU, A. Multi-Verse Optimizer: a nature-inspired algorithm for global optimization. **Neural Computing and Applications**, v. 27, p. 495-513, 2015.

NGUEVEU, S.U.; PRINS, C.; WOLFLER-CALVO, R. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. **Computers & Operations Research**, v.37, p. 1877-1885, 2010.

NUCAMENDI-GUILLÉN, S.; MARTÍNEZ-SALAZAR, I; ANGEL-BELLO, F.; MORENO-VEGA, M. M. A mixed integer formulation and an efficient metaheuristic procedure for the k-Travelling Repairmen Problem. **Journal of the Operational Research Society**, v. 67, p. 1121-1134, 2016.

POTVIN, J.Y.; ROUSSEAU, J.M. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. **European Journal of Operational Research**, v. 66, n. 5, p. 331-340, 1993.

QIAN, W.; WANG, B.; FENG, Z. Adaptive Central Force Optimization Algorithm Based on the Stability Analysis. **Mathematical Problems in Engineering**, p. 1-10, 2015.

RABANAL, P.; RODRIGUEZ, I.; RUBIO, F. Solving Dynamic TSP by using River Formation Dynamics. **Fourth International Conference on Natural Computation**, Jinan, p. 246-250, 2008.

RABANAL, P.; RODRIGUEZ, I.; RUBIO, F. Applying River Formation Dynamics to the Steiner Tree Problem. **IEEE International Conference on Cognitive Informatics (ICCI)**, Beijing, p. 704-711, 2010.

REINELT, G. TSPLIB - A Traveling Salesman Problem Library. **ORSA Journal on Computing**, v. 3, n. 4, p. 376-384, 1991.

ROBERTI, R.; MINGOZZI, A. Dynamic ng-path relaxation for the delivery man problem. **Transportation Science**, v. 48, n. 3, p. 413-424, 2014.

SACCO, W. F.; DE OLIVERIA, C. R. E. A New Stochastic Optimization Algorithm based on a Particle Collision Metaheuristic. **6th World Congresses of Structural and Multidisciplinary Optimization**. Rio de Janeiro - Brasil, 2005.

SADYKOV, R.; TAHIRI, I.; VANDERBECK, F.; DUCLOS, R.; PESSOA, A.; UCHOA, E. Branch-cut-and-price solver for vehicle routing problems. **ISMP 2018 - 23th International Symposium on Mathematical Programming**, Bordeaux – França, 2018.

SAHNI, S.; GONZALEZ, T. P-complete approximation problems. **Journal of the ACM**, v. 23, p. 555-565, 1976.

SALCEDO-SANZ, S. Modern meta-heuristics based on nonlinear physics processes: A review of models and design procedures. **Physics Reports,** v. 655, p. 1-70, 2016.

SALEHIPOUR, A.; SORENSEN, K.; GOOS, P.; BRAYSY, O. An efficient GRASP + VND metaheuristic for the traveling repairman problem. Working paper, University of Antwerp, Faculty of Applied Economics, 2008.

SALEHIPOUR, A.; SORENSEN, K.; GOOS, P.; BRAYSY, O. An efficient GRASP + VND metaheuristic for the traveling repairman problem. **4OR - A Quarterly Journal of Operations Research**, v. 9, p. 189-209, 2011.

SAVSANI, V.; TAWHID, M. A. Non-dominated sorting moth flame optimization (NS-MFO) for multi-objective problems. **Engineering Applications of Artificial Intelligence**, v. 63, p. 20-32, 2017.

SILVA, M. M.; SUBRAMANIAN, A.; VIDAL, T.; OCHI, L. S. A simple and effective metaheuristic for the Minimum Latency Problem. **European Journal of Operational Research**, v. 221, n. 3, p. 513-520, 2012.

SOUTO, G.; MORAIS, I.; MAURI, G. R.; RIBEIRO, G. M.; GONZÁLEZ, P. H.; A hybrid matheuristic for the Two-Stage Capacitated Facility Location problem. **Expert Systems with Applications**, v. 185, 115501, 2021.

SPEARS, W. M.; SPEARS, D. F.; HEIL, R.; KERR, W.; HETTIARACHCHI, S. An Overview of Physicomimetics. **Lecture Notes in Computer Science**, v. 3342, 2005.

TAN, J. D.; DAHARI, M.; KOH, S. P.; KOAY, Y. Y.; ABED, I. A. An improved electromagnetism-like algorithm for numerical optimization. **Theoretical Computer Science**, v. 641, p. 75-84, 2016.

TALBI, E. G. Metaheuristics: From Design to Implementation. 1ª ed. New Jersey: Wiley-Blackwell, 2009.

TAWHID, M. A; SAVSANI, V. Multi-objective sine-cosine algorithm (MO-SCA) for multi-objective engineering design problems. **Neural Computing and Applications**, v. 31, p. 915-929, 2019.

TSITSIKLIS, J. N. Special cases of traveling salesman and repairman problems with time windows. **Networks**, v. 22, p. 263-282, 1992.

VIDAL, T. **Approches générales de résolution pour les problèmes multi-attributs de tournéesde véhicules et confection d'horaires**. *Ph.D. Thesis* (Informatique), CIRRELT - Université de Montréal, 2013.

VIDAL, T.; CRAINIC, T.; GENDREAU, M.; PRINS, C**. A unifying view on timing problems and algorithms**. Ed. CIRRELT, Canadá, 48p, 2011.

WANG X. J.; GAO L.; ZHANG C.Y. Electromagnetism-Like Mechanism Based Algorithm for Neural Network Training. **Lecture Notes in Computer Science**, v. 5227, 2008.

WOLPERT, D. H.; MACREADY, W. G. No Free Lunch Theorems for Optimization. **IEEE Transactions on Evolutionary Computation 1**, p. 67, 1997.

WU, B. Y.; Huang, Z.N.; Zhan, F.J. Exact algorithms for the minimum latency problem. **Information Processing Letters**, v. 92, n. 6, p. 303-309, 2004.

XIE, L.; ZENG J.; CUI, Z. General framework of Artificial Physics Optimization Algorithm. **World Congress on Nature & Biologically Inspired Computing**, Coimbatore, p. 1321-1326, 2009.

XIE, L.; ZENG, J.; CUI, Z. The Vector Model of Artificial Physics Optimization Algorithm for Global Optimization Problems. **Intelligent Data Engineering and Automated Learning**, v. 5788, p. 610-617, 2009.