Problema de Roteamento de Veículos Capacitados

Filipe da Silva Rocha[1], **Henrique Pereira Cristófaro**[1], **Vitor Theodoro Rocha Domingues**[1]

[1]Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG),
Belo Horizonte, MG, Brasil

filipesilva.tmsi@gmail.com, henrique.henri25@gmail.com, vitor-theodoro@homail.com

Abstract. *This article introduces a solution to the capacitated vehicle routing problem, employing the GORUBI mixed-integer linear programming solver in conjunction with the metaheuristic known as the ant colony algorithm. The primary goal of the implementation was to compare and evaluate the obtained results.*

Resumo. *Neste artigo, é apresentada uma solução para o problema de roteamento de veículos com capacidade, utilizando o solver GORUBI de programação linear inteira mista, juntamente com a meta-heurística conhecida como algoritmo de colônia de formigas. A implementação teve como principal objetivo a comparação e avaliação dos resultados obtidos.*

1. Introdução

A logística tem ganhado destaque tanto no meio acadêmico quanto empresarial, evidenciando sua importância crescente ao longo do tempo. Sua evolução reflete não apenas a complexidade em constante ascensão dos ambientes operacionais, mas também a compreensão mais refinada de sua significativa relevância e do impacto financeiro positivo derivado da otimização dos processos logísticos.

A gestão eficiente desses processos transcende a mera coordenação operacional, configurando-se como um diferencial competitivo essencial para o sucesso organizacional, especialmente na promoção da eficácia e resiliência das cadeias de suprimentos modernas.

Após examinar a evolução e relevância da logística, é imperativo abordar desafios específicos inerentes à gestão eficiente de processos logísticos. O Problema de Roteamento de Veículos (PRV), engloba o planejamento otimizado de entregas por meio de rotas ideais. Nesse cenário, veículos partem de um ou mais depósitos em direção a um número predeterminado de clientes, sujeitos a restrições específicas.

A versão capacitada deste problema, conhecida como Problema de Roteamento de Veículos Capacitados (PRVC), acrescenta complexidade ao considerar restrições nas capacidades dos veículos. A otimização de rotas é ampliada pela necessidade crucial de gerenciar eficientemente a capacidade de carga dos veículos durante todo o processo logístico. Essa análise aprofundada proporcionará insights essenciais para o desenvolvimento de estratégias eficazes de roteamento em contextos logísticos desafiadores.

Além de sua relevância teórica, o PRVC desempenha um papel vital em operações práticas, sendo particularmente aplicável em setores como transporte e logística. Empresas podem utilizar soluções derivadas do PRVC para otimizar operações diárias, reduzindo custos e aprimorando a eficiência no uso de frota. No setor de distribuição, o PRVC

aprimora a gestão de rotas de entrega, garantindo que veículos atendam às demandas dos clientes de maneira eficaz e econômica.

2. Trabalhos relacionados

O PRV foi formulado pela primeira vez por [Dantzig and Ramser 1959] , quando estudaram a aplicação real na distribuição de gasolina para estações de venda de combustíveis. O PRVC surgiu pela necessidade de considerar a capacidade dos veículos ao planejar as rotas. Desde então tem surgido diversas abordagens para a resolução desse problema, pode-se classificá-las em duas classes distintas: métodos exatos e métodos heurísticos.

Nesse contexto, os métodos exatos para resolver o Problema de Roteamento de Veículos Capacitados (PRVC) são geralmente baseados em programação linear inteira e são capazes de encontrar a solução ótima para o problema. No entanto, devido à complexidade do PRVC, esses métodos são computacionalmente intensivos e só podem ser usados para resolver instâncias de problemas de tamanho moderado. Um dos marcos na abordagem exata para o PRVC foi apresentado por [Christofides et al. 1981], que trabalhou com limitantes lagrangeanos para a geração de subrotas. Já, [Feillet et al. 2004] utilizou programação dinâmica para estender caminhos por meio da adição de vértices não visitados ao final desses caminhos.

Os métodos heurísticos são preferíveis em instâncias de grande porte, pois conseguem encontrar soluções de boa qualidade utilizando menos recursos computacionais do que algoritmos exatos. [Bullnheimer et al. 1999] apresentou uma solução baseada em colônias de formigas, na qual cada formiga constrói uma solução potencial ao criar uma sequência de clientes a serem visitados pelos veículos. Por outro lado, [Tao and Wang 2015] optaram por utilizar a Busca Tabu para abordar o problema de roteamento. Este método emprega uma busca local para explorar o espaço de solução além dos ótimos locais, demonstrando uma abordagem eficaz na resolução do problema.

Além disso, os algoritmos correspondentes e as suas implementações computacionais assumem um papel essencial na busca de soluções viáveis de alta qualidade para as instâncias do mundo real, e com tempo de computação aceitáveis. Comparado aos procedimentos não baseados em otimização combinatória, o ganho de economia e redução de custos é significativo quando na utilização de frotas de veículos.

Nos últimos anos, foram desenvolvidas muitas ferramentas de software integrando serviços de transmissão eletrônica de dados entre veículos e administradores de frotas. O objetivo dessas ferramentas é permitir uma reação mais rápida dos planejadores à dinâmica do sistema de transporte e às possíveis perturbações causadas pela avaria de veículos ou pelas condições ruins de tráfego nas estradas.

3. Definição formal

O PRVC modelado neste trabalho seguiu uma adaptação do modelo de índice triplo proposto por [Toth and Vigo 2002]

3.1. Modelo do problema

$$Minimizar \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} \sum_{k=1}^{K} x_{ijk} \tag{1}$$

Sujeito a:

$$\sum_{k=1}^{K} y_{ik} = 1 \qquad (i = 1, \ldots, n) \tag{2}$$

$$\sum_{k=1}^{K} y_{0k} = K \tag{3}$$

$$\sum_{j=0}^{n} x_{ijk} = \sum_{j=0}^{n} x_{jik} = y_{ik} \qquad i = 0, \ldots, n, \quad (k = 1, \ldots, K) \tag{4}$$

$$\sum_{i=1}^{n} d_i y_{ik} \leq Q \qquad (k = 1, \ldots, K) \tag{5}$$

$$\sum_{i=1}^{n} p_i y_{ik} \leq M_k \qquad (k = 1, \ldots, K) \tag{6}$$

$$y_{11} = y_{21} = 1 \tag{7}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijk} \leq |S| - 1 \qquad (\forall\, S \subseteq V \backslash \{0\}, |S| \geq 2, k = 1, \ldots, K) \tag{8}$$

$$x_{ijk} \in \{0, 1\} \qquad (i, j = 0, \ldots, n; i \neq j), (k = 1, \ldots, K) \tag{9}$$

$$y_{ik} \in \{0, 1\} \qquad (i = 0, \ldots, n), (k = 1, \ldots, K) \tag{10}$$

A função objetivo (1) busca a minimização da distância total percorrida, a restrição (2) desta formulação impõe que cada cidade seja visitada por apenas um veículo, a restrição (3) determina que os veículos devem deixar o depósito e a restrição (4) estabelece que o mesmo veículo deve entrar e sair de um determinado cidade. A restrição de capacidade (5) determina que a carga de qualquer veículo não pode ultrapassar a sua capacidade. A restrição (6) impede que o número máximo de entregas permitidas para caminhão seja excedido, e (7) obriga o caminhão a atender obrigatoriamente aos clientes i = 1 e i = 2. A restrição (8) elimina os subcircuitos ilegais (sub-rotas desconectadas do depósito), As últimas restrições, (9) e (10), indicam que as variáveis x_{ij} e y_{ik} são binárias.

3.2. Gurobi

O Gurobi é uma biblioteca de otimização matemática amplamente reconhecida por sua eficiência na resolução de problemas complexos. Com uma interface amigável, o Gurobi oferece suporte para várias linguagens de programação, incluindo Python e C++, tornando-o acessível a uma ampla comunidade de usuários. Sua capacidade de lidar com diferentes tipos de modelos, como programação linear, inteira e quadrática, e sua notável velocidade de resolução o tornam uma ferramenta valiosa em ambientes acadêmicos e industriais.

Ao aproveitar as capacidades diversificadas do Gurobi, buscamos otimizar a alocação de veículos e rotas de coleta de maneira eficaz, levando em consideração restrições específicas do PRVC. A abordagem em Python oferece uma interface acessível e familiar, permitindo uma implementação mais ágil e uma integração perfeita com outras ferramentas de desenvolvimento. Em resumo, nossa escolha do Gurobi no contexto

do Python visa maximizar a eficiência, proporcionando soluções otimizadas para os desafios logísticos e operacionais do PRVC.

3.3. Algoritmo Heurístico

O algoritmo de otimização da colônia de formigas ou, do acrônimo inglês, ACO – *ant colony optimization* é comumente utilizado na solução de problemas de procura de melhor caminho em grafos. Esse algoritmo é uma heurística baseada em probabilidade criada por [Dorigo and Gambardella 1997], com o objetivo de melhorar o até então conhecido sistemas de formigas.

No mundo natural, o caminho das formigas funciona da seguinte maneira: no início, elas saem do formigueiro em busca de alimentos. Como em sua maioria são cegas e não sabem onde encontrarão comida, elas andam dispersas, cada uma em uma direção. Durante esse trajeto, elas vão deixando no caminho um rastro de feromônio. Ao encontrar comida, as formigas retornam ao formigueiro pelo mesmo rastro deixado para trás.

Quando outras formigas passam pelo mesmo rastro de feromônio, a concentração deste aumenta, dando um feedback positivo para as demais formigas. Porém, essa é uma solução dinâmica. Como o rastro de feromônio é volátil, ele vai se dissipando, o que permite que uma solução inicialmente encontrada não seja obrigatoriamente uma solução ótima, visto que outra formiga pode encontrar outro caminho, mais curto, e o seu rastro será então mais forte.

Podemos então entender que esse algoritmo é indicado para grafos que tendem a mudar dinamicamente, como o caminho que o Waze pode indicar para você fazer no trajeto da sua casa. Mesmo escolhendo sempre o caminho mais rápido, ele provavelmente não envia você sempre pelo mesmo caminho, pois podem existir entraves nesse trajeto (feedbacks negativos), que tornam o grafo (trajeto para sua casa) algo dinâmico.

O ACO opera na prática de maneira a explorar eficientemente o espaço de soluções em problemas de otimização em grafos. Cada formiga, representando uma solução parcial, faz escolhas probabilísticas baseadas em feromônios deixados nas arestas do grafo. Esses feromônios indicam a qualidade das soluções anteriores.

Durante a construção da solução, as formigas depositam feromônios em suas trilhas, intensificando caminhos mais promissores. A evaporação natural dos feromônios simula a dinâmica do ambiente, permitindo que o algoritmo se adapte a mudanças nas condições. Esse ciclo de construção, avaliação e atualização contínua de feromônios possibilita que o ACO converge gradualmente para soluções de alta qualidade.

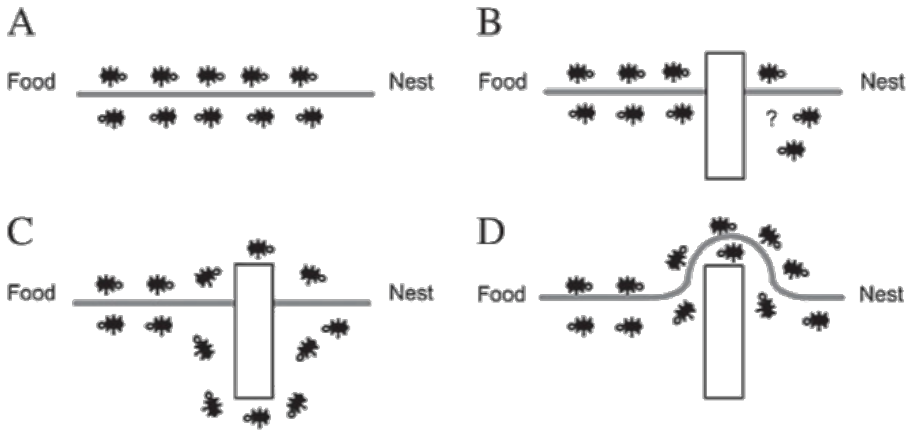


Figura 1. Caminhos das formigas

A aplicação prática do ACO pode ser exemplificada em roteamento de veículos, onde as formigas buscam caminhos eficientes para otimizar o transporte de mercadorias. Esse processo iterativo e adaptativo do ACO reflete sua capacidade de encontrar soluções robustas e eficazes em cenários dinâmicos do mundo real.

3.4. Datasets

Os datasets desempenham um papel crucial ao abordar o PRVC, fornecendo informações essenciais sobre locais, demandas e restrições logísticas. Esses conjuntos de dados são projetados para representar cenários do mundo real, oferecendo exemplos ricos e desafiadores para otimizar as operações de coleta e entrega. Utilizamos uma *library* com vários datasets produzidos por [de Bittencourt et al. 2012a].

4. Resultados

Nesta etapa, foi empreendido uma análise abrangente e minuciosa, explorando os desdobramentos resultantes da aplicação dos algoritmos Simplex e ACO (Ant Colony Optimization) no contexto do Problema de Roteamento de Veículos Capacitados (CVRP). O propósito primordial é avaliar o desempenho dessas abordagens em uma variedade de cenários, manipulando variáveis cruciais, tais como o tempo máximo de execução.

Ao submeter os algoritmos a diferentes conjuntos de dados, foram analisados os efeitos das alterações nos parâmetros fundamentais. A exploração não se limita apenas à influência do tempo máximo de execução; mas os desafios decorrentes do aumento no número de cidades e caminhões nos problemas tratados.

A análise sistemática dos resultados obtidos desempenhou um papel crucial na identificação de nuances e padrões emergentes, contribuindo significativamente para a compreensão do comportamento desses algoritmos. O foco foi direcionado para compreender como cada algoritmo respondeu a desafios específicos, permitindo-nos comparar os resultados alcançados em cada cenário com os valores ótimos conhecidos.

Ao executarmos os algoritmos em diversos conjuntos de dados, obtivemos os seguintes resultados, juntamente com suas informações associadas:

Tabela 1. Parâmetros do Algoritmo ACO

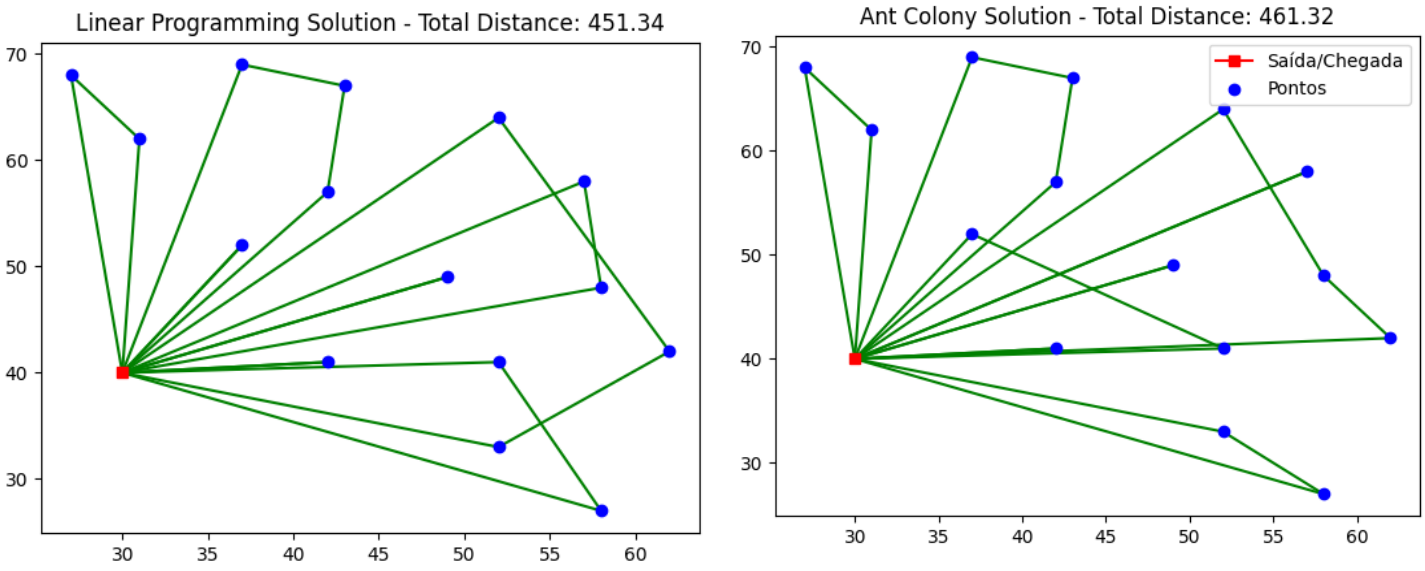
Parâmetro	Valor
Taxa de Evaporação	0.8
Alfa	2.0
Beta	5.0
Sigm	3.0

Tabela 2. Parâmetros do Gurobi

Parâmetro	Valor
Gap	0

Tabela 3. Datasets utilizados

Dataset	Solução ótima
E-n16-k8	450
E-n19-k2	212
E-n22-k4	375
E-n33-k4	835

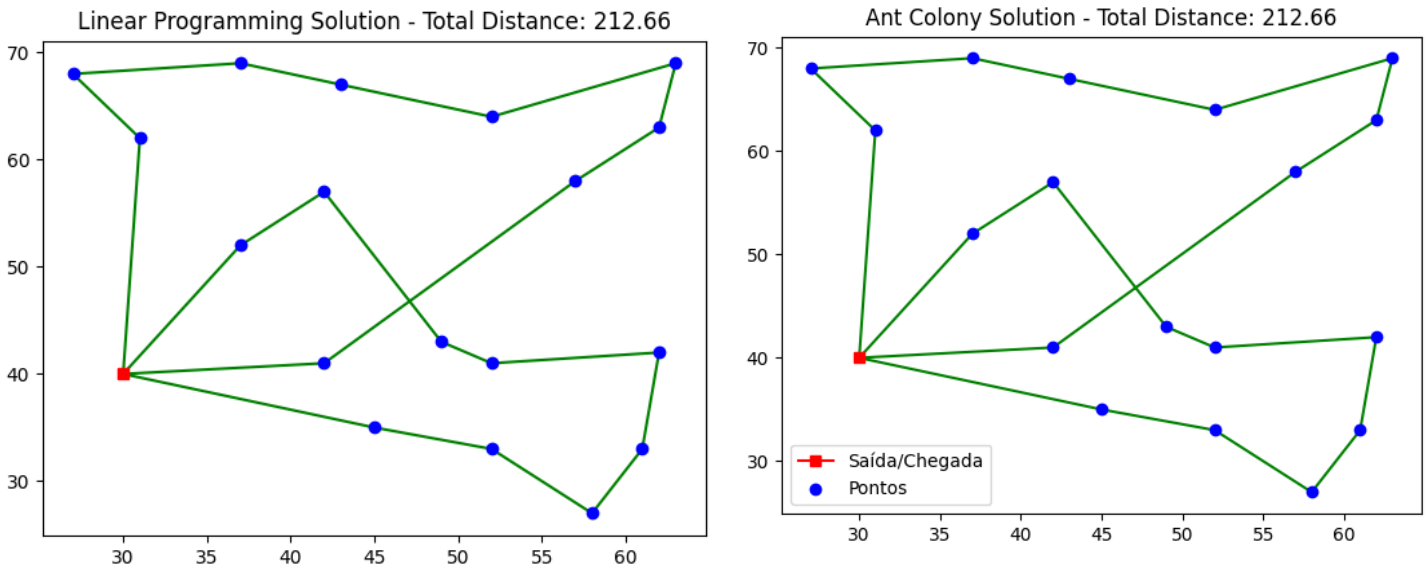


(a) Gurobi

(b) ACO

Figura 2. E-n16-k8

Para o caso acima, foi utilizada a instância menos complexa, que consiste em 16 cidades e 8 caminhões. O solver Gurobi levou 7,43 segundos para convergir para a solução ótima de 451,32. Por outro lado, o ACO encontrou uma solução de 461,32 ao ser executado durante o mesmo período de 7,43 segundos.



(a) Gurobi (b) ACO

Figura 3. E-n19-k2

Para o caso acima, foi utilizada uma instância composta por 19 cidades e 2 caminhões. O solver Gurobi levou 44,95 minutos para convergir para a solução ótima de 212,66. Por sua vez, o ACO também encontrou a solução ótima de 212,66 ao ser executado pelo mesmo período de 44,95 minutos.

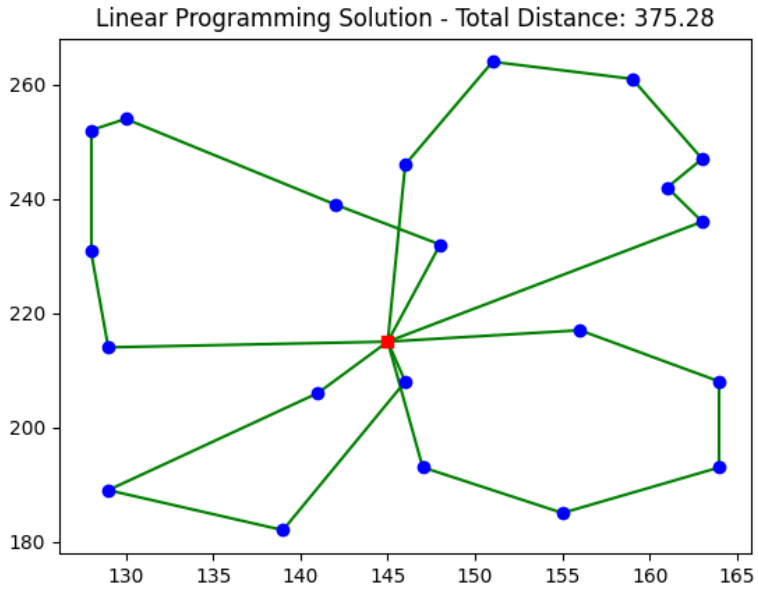


Figura 4. Tempo necessário para encontrar a solução ótima de cada dataset

Para o caso acima, foi utilizada uma instância composta por 22 cidades e 4 caminhões. O solver Gurobi levou 8,46 horas para convergir para a solução ótima de 375,28.

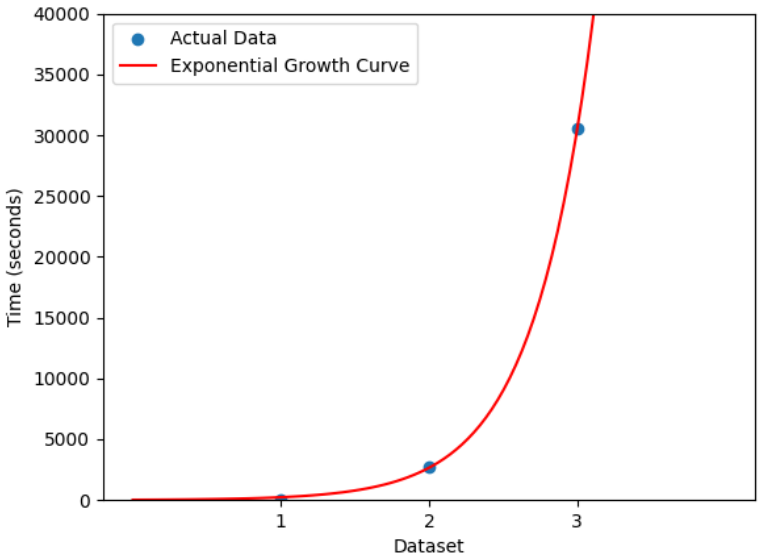
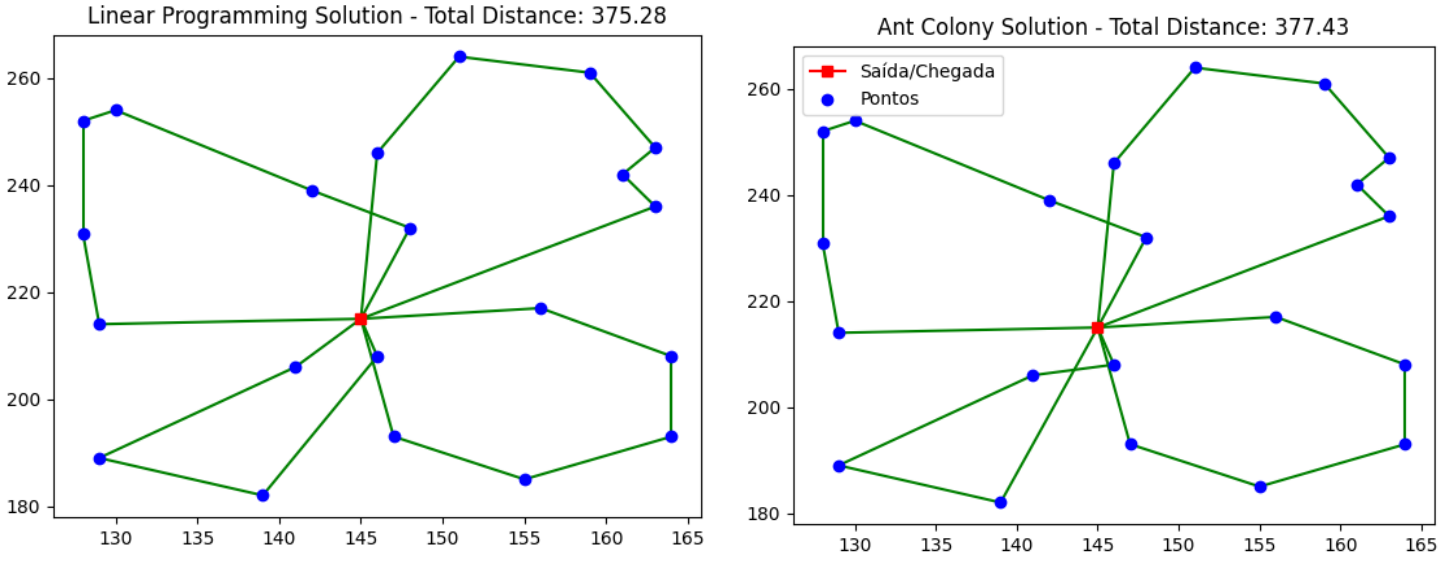


Figura 5. Tempo necessário para encontrar a solução ótima de cada dataset

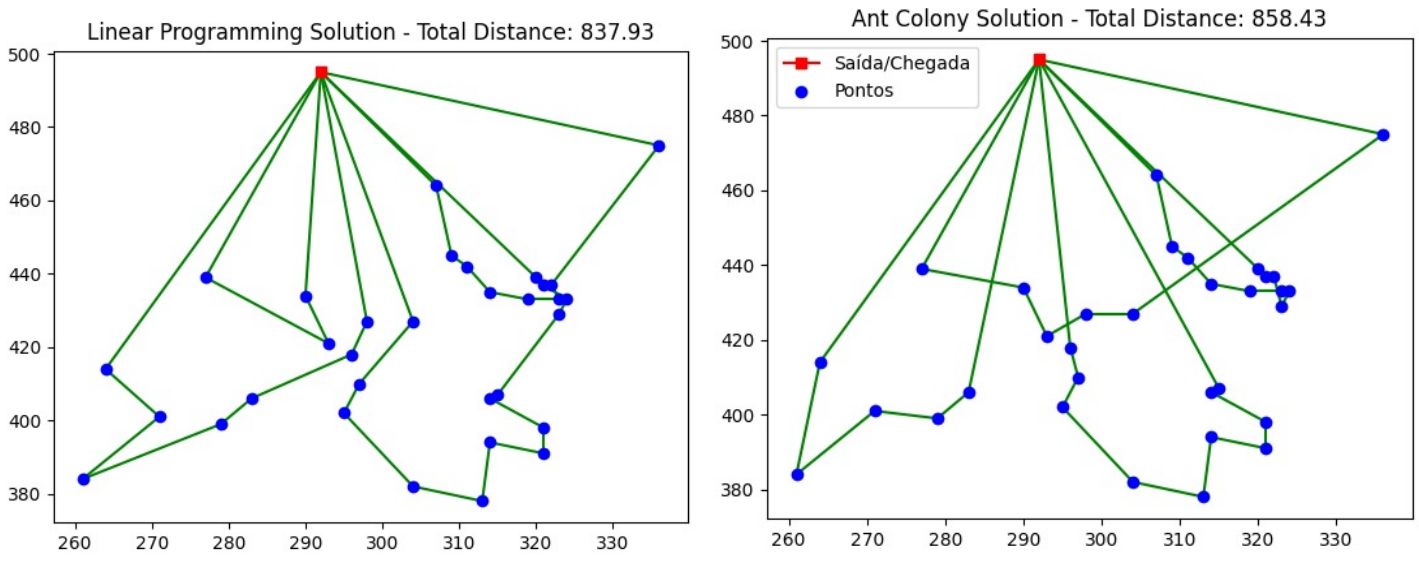
O gráfico acima foi gerado com base no tempo necessário para encontrar a solução ótima das três instâncias mais recentes apresentadas. Ao analisá-lo, observa-se um comportamento exponencial no tempo de resolução do solver Gurobi.



(a) Gurobi **(b) ACO**

Figura 6. E-n22-k4

Para o caso acima, foi utilizada uma instância composta por 22 cidades e 4 caminhões. Em ambos os casos, o algoritmo foi executado por 2 horas. O Gurobi foi interrompido antes de encontrar a solução ótima e convergiu para uma solução de 375,28. Por sua vez, o ACO encontrou uma solução de 377,43 durante o tempo estabelecido.



(a) Gurobi **(b) ACO**

Figura 7. E-n33-k4

Para o caso acima, foi utilizada uma instância composta por 33 cidades e 4 caminhões. Em ambos os casos, o algoritmo foi executado por 2 horas. O Gurobi foi interrompido antes de encontrar a solução ótima e convergiu para uma solução de 837,93. Por sua vez, o ACO encontrou uma solução de 853,43 durante o tempo estabelecido.

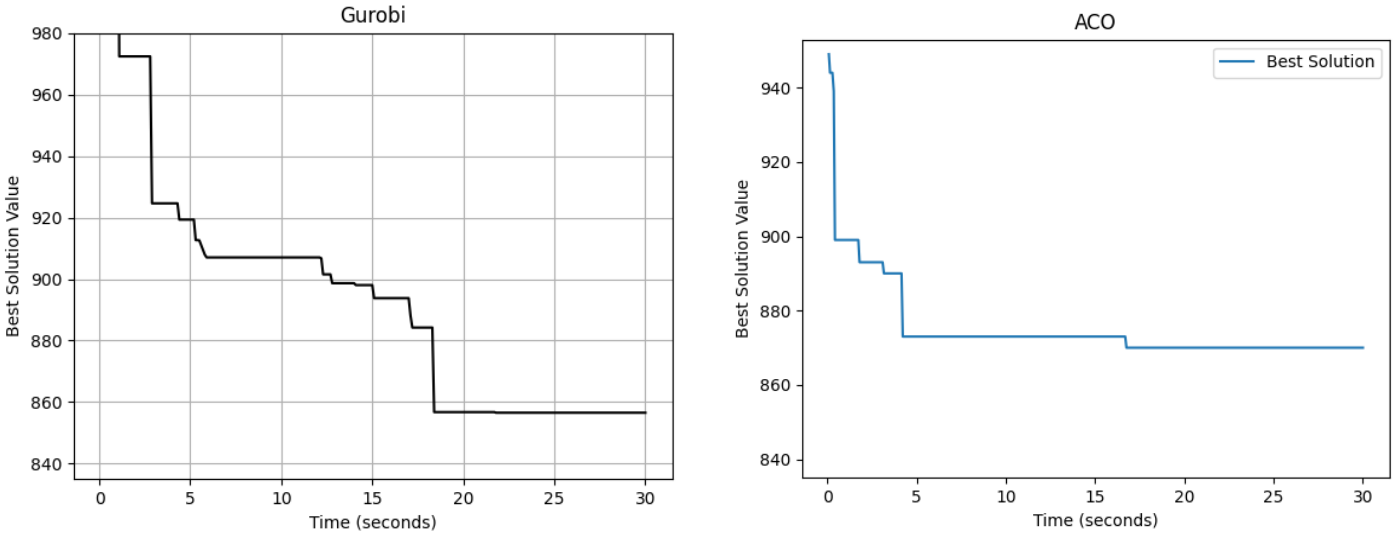


Figura 8. Melhor solução encontrada a cada segundo

Para gerar o gráfico acima, utilizou-se a instância anterior composta por 33 cidades e 4 caminhões. Analisou-se a melhor solução encontrada por ambos os algoritmos a cada segundo de execução. Ao observar o gráfico, percebe-se que a otimização por colônia de formigas apresenta resultados superiores nos primeiros segundos, mas posteriormente tende a ficar aquém em comparação ao solver Gurobi.

5. Considerações finais

Neste trabalho, apresentamos soluções para o Problema de Roteamento de Veículos Capacitados (PRVC), utilizando o solver Gurobi de programação linear inteira mista, juntamente com a meta-heurística conhecida como algoritmo de colônia de formigas.

O objetivo foi comparar e avaliar os resultados obtidos por essas duas abordagens em diferentes conjuntos de dados, representando cenários do mundo real. Os resultados mostraram que o algoritmo de colônia de formigas foi capaz de encontrar soluções de boa qualidade em um tempo computacional razoável, enquanto o solver Gurobi obteve soluções ótimas ou próximas do ótimo em alguns casos, mas demorou muito mais tempo para resolver as instâncias mais complexas. Infelizmente, não foi possível determinar soluções maiores, pois soluções com muitas variáveis e/ou restrições excederam a capacidade da versão gratuita do solver.

Então chegamos a conclusão que o algoritmo de colônia de formigas é uma ferramenta eficaz e robusta para lidar com o PRVC, especialmente em situações dinâmicas e complexas, onde o solver Gurobi pode não ser viável ou eficiente, desde que a solução ótima seja aproximada, caso contrário a solução por programação linear se sai melhor.

Além disso, este trabalho possibilitou a aplicação prática dos conceitos adquiridos nas disciplinas de Inteligência Artificial e Otimização I. Foi possível também investigar o uso de softwares na solução de problemas de otimização, ao mesmo tempo em que se pesquisava e desenvolvia algoritmos de Inteligência Artificial para o mesmo fim.

Referências

Bullnheimer, B., Hartl, R. F., and Strauss, C. (1999). An improved ant system algorithm for thevehicle routing problem. *Annals of operations research*, 89(0):89–319.

Christofides, N., Mingozzi, A., and Toth, P. (1981). State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11(2):145–164.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1):80–91.

de Bittencourt, G. C., Rodrigues, S., Netto, P. O. B., and Jurkiewicz, S. (2012a). Capacitated vehicle routing problem library. In *http://vrp.atd-lab.inf.puc-rio.br/index.php/en/*.

de Bittencourt, G. C., Rodrigues, S., Netto, P. O. B., and Jurkiewicz, S. (2012b). Problema de roteamento de veículos capacitados (prvc): solução manual x busca dispersa. In *Congresso Latino-Iberoamericano de Investigatión Operativa–CLAIO. Simpósio Brasileiro de pesquisa Operacional–SBPO: Rio de Janeiro*.

Dorigo, M. and Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66.

Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229.

Tao, Y. and Wang, F. (2015). An effective tabu search approach with improved loading algorithms for the 3l-cvrp. *Computers Operations Research*, 55:127–140.

Toth, P. and Vigo, D. (2002). The vehicle routing problem. siam monographs on discrete mathematics and applications, vol. 9. siam.