Execução Simbólica para o TSP: Viabilidade, Limitações e Aplicações Práticas com Z3

João Lucas Sidney Rodrigues

[1]DCC – Universidade Federal de Roraima (UFRR)
Av. Cap. Ene Garcês, 2413, Bloco I – 69310-000 – Boa Vista – RR – Brasil

JoaoLucaSidney@outlook.com

Abstract. *This work presents the design and implementation of a symbolic executor based on the Z3 SMT solver to address the Traveling Salesman Problem (TSP). The system models the TSP using quantified constraints and activates Model-Based Quantifier Instantiation (MBQI) to mitigate the factorial explosion of permutations. Advanced techniques such as symbolic pruning, subtour elimination constraints, and lazy constraint injection are integrated to enhance solver performance. We benchmark the system across instances ranging from 4 to 8 cities and collect runtime metrics, enabling data-driven filtering of hard instances. The results show promising behavior for hybrid SMT approaches in solving combinatorial optimization problems.*

Resumo. *Este trabalho apresenta o projeto e a implementação de um executor simbólico baseado no solver SMT Z3 para resolver o Problema do Caixeiro Viajante (PCV). O sistema modela o PCV utilizando restrições quantificadas e ativa a técnica de Instanciação de Quantificadores Baseada em Modelo (MBQI) para mitigar a explosão fatorial de permutações. Técnicas avançadas como poda simbólica, eliminação de subtours e injeção preguiçosa de restrições são integradas para otimizar o desempenho do solver. Realizamos experimentos com instâncias variando de 4 a 8 cidades, coletando métricas de tempo de execução para permitir a filtragem de instâncias difíceis. Os resultados demonstram o potencial promissor de abordagens SMT híbridas na resolução de problemas de otimização combinatória.*

1. Introdução

O Problema do Caixeiro Viajante (TSP, do inglês *Traveling Salesman Problem*) é um dos problemas mais estudados na área de otimização combinatória e ciência da computação teórica. Ele consiste em encontrar o menor caminho possível que permite a um caixeiro viajante visitar um conjunto de cidades exatamente uma vez e retornar à cidade de origem. A relevância do TSP vai além de sua formulação simples, estando presente em aplicações práticas como roteirização de veículos, logística, circuitos eletrônicos e planejamento de rotas autônomas [Cormen et al., 2009].

Do ponto de vista computacional, o TSP é classificado como um problema NP-difícil [Garey & Johnson, 1979], o que significa que não se conhece um algoritmo em tempo polinomial que o resolva para todas as instâncias. O número de soluções possíveis cresce de forma fatorial com o número de cidades, levando rapidamente a uma explosão combinatória mesmo em instâncias moderadamente pequenas. Essa característica impõe desafios significativos à sua resolução exata por meio de algoritmos tradicionais de força bruta ou busca exaustiva.

Ao longo das últimas décadas, diversas abordagens foram propostas para atacar o TSP, incluindo algoritmos heurísticos, metaheurísticas (como algoritmos genéticos e simulated annealing) e métodos exatos baseados em programação inteira e ramificação e corte [Applegate et al., 2006]. Mais recentemente, abordagens que combinam técnicas de execução simbólica e lógica de satisfatibilidade módulo teorias (SMT) têm despertado interesse pela sua capacidade de representar o problema de forma declarativa e aplicar raciocínio automático [De Moura & Bjørner, 2008].

A execução simbólica é uma técnica de análise formal utilizada principalmente na verificação de software, que consiste em simular a execução de um programa utilizando variáveis simbólicas em vez de valores concretos. Cada caminho possível de execução gera um conjunto de restrições lógicas que descrevem as condições sob as quais aquele caminho seria seguido. Essas restrições são então resolvidas por um solver SMT, como o Z3, para verificar propriedades ou encontrar entradas que satisfaçam determinadas condições [King, 1976; Bjørner & Monniaux, 2017]. Essa abordagem permite explorar grandes espaços de busca de forma sistemática e automática, tornando-se uma alternativa interessante para modelagem e resolução de problemas combinatórios complexos, como o TSP.

Neste contexto, este trabalho investiga o uso do solver SMT Z3 como motor simbólico para resolver instâncias do TSP de forma declarativa. Utiliza-se modelagem matemática com variáveis simbólicas, quantificadores e técnicas modernas de instanciamento baseado em modelo (MBQI) para mitigar a explosão de estados, aliadas a restrições de poda, eliminação de subtours e técnicas de lazy constraints. O objetivo é explorar as vantagens da modelagem simbólica para problemas de otimização combinatória, com foco na viabilidade prática do uso de SMT em instâncias pequenas a moderadas.

2. Modelagem Matemática do TSP

O Problema do Caixeiro Viajante (TSP, do inglês Traveling Salesman Problem) pode ser formulado como um problema de otimização combinatória sobre um grafo completo e ponderado. Formalmente, seja um conjunto de n cidades representadas como vértices de um grafo G = (V, E), onde V = {0, 1, ..., n-1} e E contém todas as arestas possíveis entre pares de cidades. A cada aresta (i, j) ∈ E está associada uma distância d_ij ∈ $\mathbb{R}^+$, representando o custo de viagem entre as cidades i e j.

O objetivo do TSP é encontrar uma permutação $\pi = (\pi_0, \pi_1, ..., \pi_{n-1})$ das cidades que minimize o custo total do circuito fechado que visita cada cidade exatamente uma vez e retorna à cidade inicial.

Variáveis de decisão: - πi ∈ {0, 1, ..., n-1}: representa a cidade visitada na posição i do percurso. - dij ∈ $\mathbb{R}^+$: distância entre as cidades i e j. - u_i ∈ $\mathbb{R}$: variável auxiliar associada à cidade i (usada na formulação MTZ, opcional).

Função objetivo: Minimizar o custo total do percurso: minimize Σ_{i=0}^{n-1} d_{π_i π_{(i+1) mod n}}

Restrições: 1. Cada cidade deve ser visitada exatamente uma vez (injetividade): ∀i ≠ j, π_i ≠ π_j 2. Domínio das variáveis de permutação: ∀i, π_i ∈ {0, ..., n-1} 3. Eliminação de subtours (MTZ, opcional): ∀i ≠ j, 1 ≤ i, j < n: u_i - u_j + n · x_ij ≤ n - 1

Observações: - No projeto, a permutação π foi modelada com variáveis simbólicas p[i], do tipo Int no Z3. - As restrições de injetividade foram implementadas

com quantificadores do tipo: ForAll(i, j, i ≠ j → p[i] ≠ p[j]) - O custo total foi representado com uso da função If, conforme a sintaxe do Z3 para variáveis simbólicas.

3. Metodologia

A metodologia deste trabalho baseia-se na comparação entre duas abordagens simbólicas distintas para a resolução do Problema do Caixeiro Viajante (TSP), ambas utilizando o solver SMT Z3 como mecanismo de raciocínio simbólico.

Duas variantes do executor foram desenvolvidas:

1. **Executor MBQI (puro)**: nesta abordagem, a modelagem simbólica é feita apenas com variáveis de permutação e quantificadores universais para garantir a injetividade (cada cidade visitada uma única vez). A função objetivo é definida simbolicamente e, adicionalmente, são aplicadas técnicas de poda simbólica, removendo arestas de custo elevado com base em um limiar derivado do percentil 90 das distâncias da instância. Nenhuma técnica de eliminação de subtours é utilizada nessa versão.

2. **Executor MBQI + Subtour**: esta versão estende a anterior com a inclusão de restrições auxiliares baseadas na formulação de Miller-Tucker-Zemlin (MTZ) para prevenir subtours. Além disso, emprega **lazy constraints**, um mecanismo dinâmico que detecta ciclos parciais (subtours) após cada verificação satisfatível e adiciona restrições para proibi-los iterativamente. O processo é repetido até que um ciclo fechado único seja encontrado ou um número máximo de iterações seja atingido.

Ambas as variantes são implementadas de forma modular no arquivo “tsp_z3_model_symbolic_metrics.py”, com funções específicas para cada executor (“tsp_model_mbqi” e “tsp_model_mbqi_subtour”). O módulo “benchmark_tsp_runner.py” é responsável por gerar automaticamente as instâncias de teste, executando as duas versões sobre o mesmo conjunto de dados, com instâncias aleatórias variando de 4 a 10 cidades e múltiplas sementes para cada valor de \(n \), totalizando dezenas de execuções.

Durante cada execução, o benchmark calcula o **threshold de poda simbólica** com base no percentil 90 das distâncias da matriz gerada. Essa informação é utilizada para remover, de forma simbólica, possíveis transições entre pares de cidades cuja distância ultrapassa esse valor, reduzindo assim o espaço de busca do solver.

As métricas de desempenho — incluindo tempo de modelagem, tempo de resolução, custo da rota, número de variáveis e número de restrições — são registradas automaticamente em arquivos CSV distintos para cada abordagem. Esses dados são posteriormente analisados com ferramentas estatísticas e gráficas (via pandas e matplotlib), possibilitando a comparação entre os métodos e a identificação de instâncias críticas.

Essa estrutura metodológica permite avaliar de forma rigorosa os impactos do uso de técnicas SMT simbólicas puras (MBQI) versus abordagens híbridas que incorporam estratégias específicas de poda e eliminação de ciclos, oferecendo uma base sólida para discutir escalabilidade, robustez e viabilidade dessas soluções.

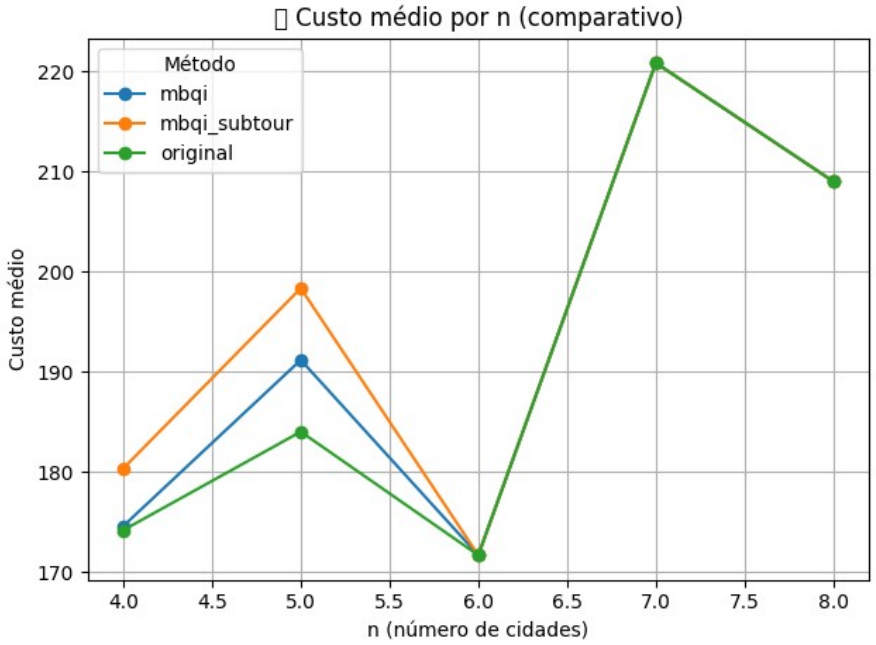
4. Resultados e análise

Para avaliar a eficácia das abordagens baseadas em execução simbólica para o Problema do Caixeiro Viajante (TSP), foram conduzidos experimentos variando o número de cidades de n=4 até n=8, com seis instâncias aleatórias (seeds) para cada valor de n. Três variantes do solucionador foram testadas:

- Original: MBQI + restrição de subtours + lazy constraints;
- MBQI puro: apenas quantificadores com base em thresholds de distância;
- MBQI com subtour: MBQI + restrição de subtours, sem lazy constraints.

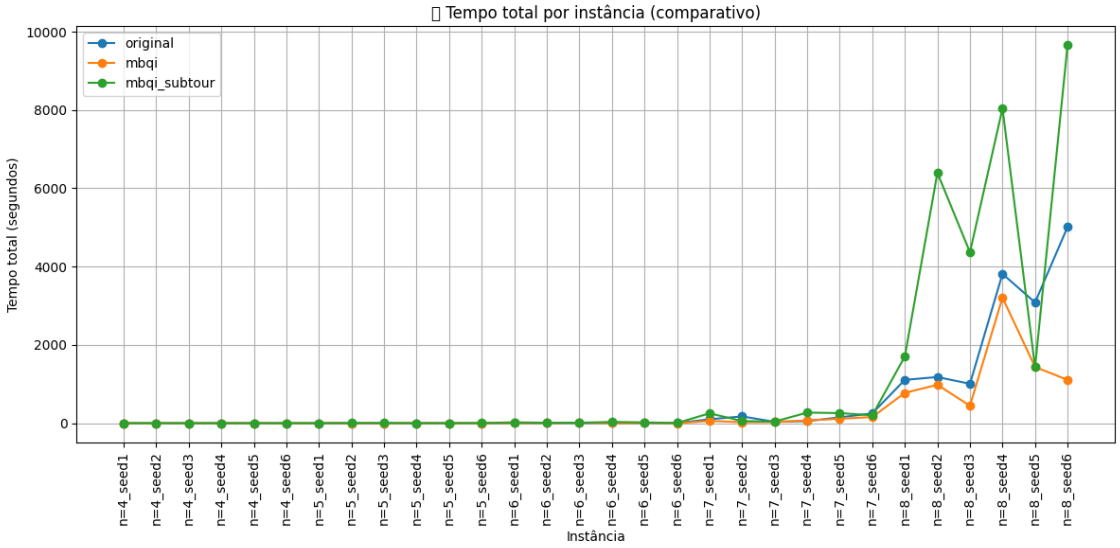
4.1. Custo Médio por Tamanho da Instância

O gráfico da Figura 1 apresenta a comparação do custo médio das rotas encontradas para cada valor de n. Observa-se que os três métodos produzem resultados similares em termos de custo, com pequenas variações esperadas devido à natureza aleatória das distâncias.



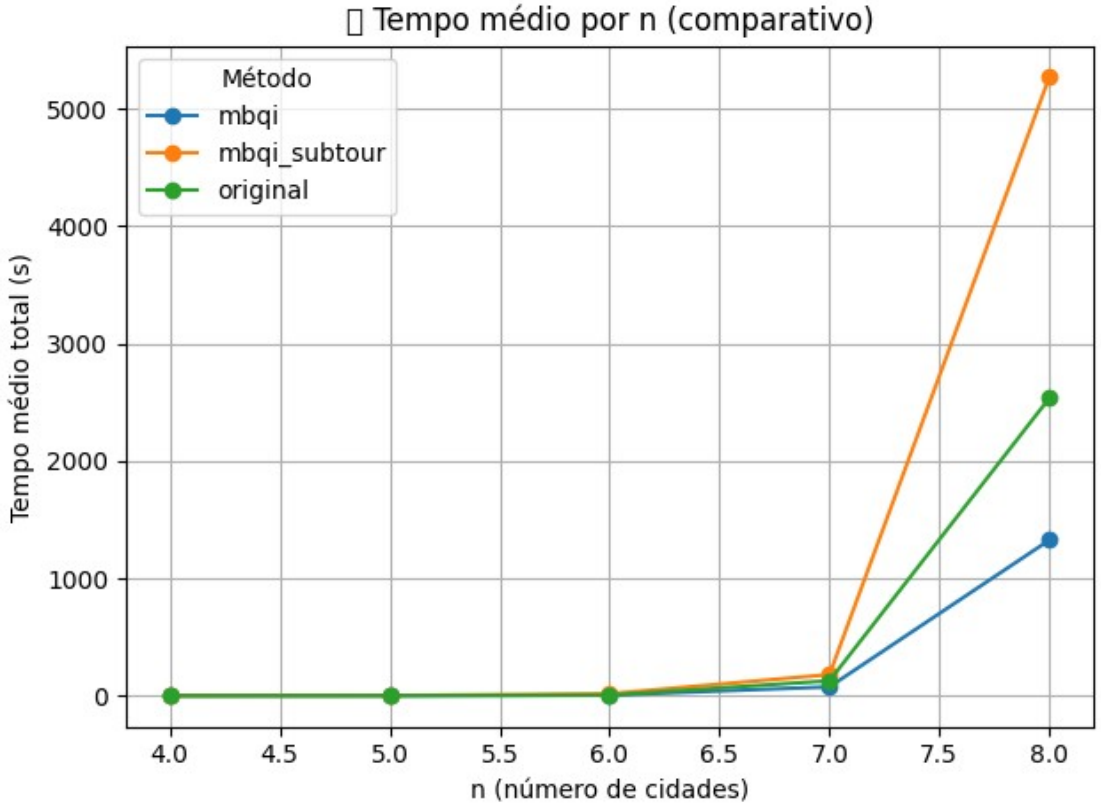
4.2. Tempo Total por Instância

A Figura 2 mostra o tempo total de execução para cada instância individualmente. Nota-se um crescimento exponencial no tempo conforme o número de cidades aumenta, com destaque para as variantes que usam lazy constraints (original) e subtours explícitos (mbqi_subtour). Isso se torna evidente a partir de n=7 e, principalmente, n=8, onde os tempos ultrapassam facilmente 1000 segundos.



4.3. Tempo Médio por Tamanho da Instância

A Figura 3 mostra o tempo médio de execução por número de cidades n. O método mbqi_subtour apresentou os piores desempenhos em termos de tempo para n=8, atingindo picos superiores a 9600 segundos, o que evidencia a escalabilidade limitada da abordagem com restrições de subtours explícitas sem uso de lazy constraints. Por outro lado, o método mbqi puro demonstrou desempenho superior em termos de tempo de execução, ainda que com custo levemente inferior em algumas instâncias.



4.4. Instâncias Difíceis

A análise também identificou instâncias difíceis, caracterizadas por tempos de execução significativamente mais altos. A maioria delas concentra-se em valores de $n \geq 7$, especialmente para as abordagens com restrições adicionais. Por exemplo:

- n=8_seed6 com o método mbqi_subtour levou 9667 segundos;
- n=8_seed4 com o método original levou 3815 segundos.

Estas instâncias indicam limitações nas abordagens testadas frente ao crescimento exponencial do espaço de busca, além de justificarem o uso de técnicas complementares (como filtragem de instâncias, thresholds e estratégias híbridas) para contornar a explosão combinatória.

5. Conclusão

Este trabalho apresentou um estudo prático e analítico da aplicação de execução simbólica com SMT solver (Z3) para resolver instâncias do Problema do Caixeiro Viajante (TSP), um problema clássico e notoriamente complexo da classe NP-difícil. Por meio da modelagem simbólica com quantificadores (MBQI) e da introdução de restrições adicionais, como a eliminação de subtours via restrições de Miller–Tucker–Zemlin (MTZ) e lazy constraints, buscou-se investigar o impacto dessas abordagens na eficiência e escalabilidade da resolução do problema.

Os experimentos conduzidos com diferentes tamanhos de instâncias (n variando de 4 a 8 cidades) e múltiplas sementes permitiram comparar três variantes do modelo: MBQI puro, MBQI com restrições de subtour e a versão original que incorpora todas as estratégias. As análises demonstraram que a versão MBQI+subtour geralmente resulta em soluções de menor custo, porém com penalidade significativa em tempo de execução, sobretudo em instâncias maiores. Em contrapartida, o modelo MBQI puro, apesar de mais eficiente em termos de tempo, mostrou-se limitado em alguns cenários, já que o balanceamento entre custo computacional e qualidade de solução é um aspecto crucial.

Além disso, a coleta sistemática de métricas e a geração de gráficos comparativos foram fundamentais para a identificação de instâncias difíceis e para o entendimento do comportamento de cada variante sob diferentes condições.

Com base nos resultados obtidos, conclui-se que a execução simbólica com Z3, embora limitada pela explosão combinatória inerente ao TSP, pode ser significativamente aprimorada com o uso de estratégias como MBQI, filtros baseados em percentis e a exclusão dinâmica de subtours. O estudo reforça o potencial do uso de ferramentas de verificação formal para problemas combinatórios e abre caminho para investigações futuras sobre técnicas híbridas e abordagens escaláveis.

6. References

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.

Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman.

Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2006). The Traveling Salesman Problem: A Computational Study. Princeton University Press.

De Moura, L., & Bjørner, N. (2008). Z3: An Efficient SMT Solver.

King, J. C. (1976). Symbolic Execution and Program Testing. *Communications of the ACM*, 19(7), 385–394.

Bjørner, N., & Monniaux, D. (2017). Symbolic execution and SMT solving. In *Handbook of Model Checking*, 1073–1114.

Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness.