

Comparative Efficiency of Deterministic and Heuristic Optimization for NP-Hard Problems on ARM-Based Mobile Architectures: A Case Study on the Realme 8 5G

Ojha Aishwarya Kumar Vankatesh Chandra

Second Year B.Tech, Computer Engineering, SOET, D Y Patil University, Ambi, Pune

Abstract - As mobile computing evolves, the demand for local execution of complex optimization tasks has grown significantly. This paper empirically analyzes the performance gap between brute-force search and genetic algorithms on a mid-range mobile SoC (MediaTek Dimensity 700). We utilize the Traveling Salesman Problem (TSP) as a benchmark to demonstrate the "complexity wall" inherent in deterministic methods. Our results show that while brute-force execution time grows factorially, reaching a critical bottleneck of 17.46 seconds at $n = 10$ cities, heuristic approaches maintain sub-second latency (0.04s). The findings confirm that heuristic optimization is essential for real-time edge intelligence on resource-constrained devices.

Key Words: NP-Hard, Mobile Edge Computing, Genetic Algorithm, ARM Architecture, Realme 8 5G.

1. INTRODUCTION

In the digital age, executing high-complexity tasks locally is vital to reduce server latency and improve data privacy. NP-hard problems, specifically the **Traveling Salesman Problem (TSP)**, are foundational challenges in computer science with significant implications for logistics and resource allocation. While traditionally solved on high-performance x86 servers, mid-range hardware often struggles with the exponential time complexity ($O(n!)$) of these problems. This paper investigates the feasibility of running these algorithms on a typical student-grade device in Maharashtra, highlighting the necessity of heuristic efficiency for modern mobile applications.

2. THEORETICAL BACKGROUND:

In this section, we review significant studies and algorithmic frameworks relevant to NP-hard optimization, heuristic modeling, and mobile-edge computational benchmarks. These prior works provide the foundation for understanding why a shift from deterministic to heuristic methods is necessary for modern mobile hardware.

Exact Algorithms for the Traveling Salesman Problem Core Focus:

This body of work focuses on obtaining mathematically optimal solutions to the Traveling Salesman Problem (TSP)

using exact methods such as Dynamic Programming and Branch-and-Bound techniques.

Logic Used:

The Held-Karp algorithm is commonly employed, which utilizes dynamic programming with a **time complexity of $O(n^2 2^n)$** .

Drawbacks:

- A) Extreme computational cost, making it impractical for problem sizes exceeding 20–25 cities on standard hardware.
- B) Causes significant thermal throttling on mobile System-on-Chips (SoCs) during prolonged execution.
- C) Results in excessive battery drain due to sustained 100% CPU utilization.

2.2 Genetic Algorithms: A Heuristic Approach to Optimization

Core Focus:

Proposed by John Holland, this research area explores genetic algorithms as **heuristic techniques** inspired by biological evolution to **produce near-optimal solutions within feasible time constraints**.

Logic Used:

Solutions evolve through randomized **search processes involving selection, crossover, and mutation, guided by a fitness evaluation mechanism**.

Drawbacks:

- A) Does not guarantee convergence to the global optimum solution.
- B) Performance is highly sensitive to hyperparameter tuning such as mutation and crossover rates.
- C) May converge prematurely to local optima if population diversity is insufficient.

2.3 Mobile Edge Computing and Resource-Constrained Benchmarking

Core Focus:

This domain examines the feasibility of executing computationally intensive algorithms on edge devices such as

smartphones and IoT systems without relying on cloud-based computation.

Technologies Used:

ARM-based mobile architectures and lightweight development environments optimized for constrained hardware.

Drawbacks:

- A) Most existing studies focus on high-end flagship processors while neglecting mid-range chipsets.
- B) Limited evaluation of the impact of virtual memory or RAM expansion on algorithmic performance.
- C) Insufficient analysis of background operating system processes in modern Android environments.

2.4 Heuristic Route Optimization for Smart City Logistics

Core Focus:

This research applies heuristic optimization techniques to real-world logistics and delivery routing problems in dense urban environments.

Logic Used:

Hybrid heuristic models combining metaheuristics such as Ant Colony Optimization and greedy approaches.

Drawbacks:

- A) Often dependent on continuous internet connectivity for cloud-assisted route computation.
- B) Not optimized for heterogeneous CPU architectures such as Big.LITTLE configurations found in budget 5G smartphones.

3. SYSTEM ARCHITECTURE & ENVIRONMENT:

The experiments were conducted in a controlled environment on a **mid-range mobile device to simulate real-world user constraints**.

3.1 Hardware Configuration:

Device: Realme 8 5G (RMX3241).

Chipset: MediaTek Dimensity 700 (7nm process).

CPU: Octa-core (2x 2.2 GHz Cortex-A76 & 6x 2.0 GHz Cortex-A55).

Memory: 8GB LPDDR4X Physical RAM with +8GB Dynamic RAM Expansion (DRE).

3.2 Software Environment:

OS: Android 14.

Compiler: Python 3.10 via Pydroid 3 integrated development environment.

4. METHODOLOGY:

This study employs a comparative research design to evaluate the limits of **mobile computational efficiency**. We implemented two distinct algorithmic paradigms to solve the **Traveling Salesman Problem (TSP)** on an ARM-based mobile chipset.

4.1 Brute-Force (Deterministic Approach)

The brute-force method is used as a baseline for accuracy. It follows an exhaustive search strategy to calculate the total Euclidean distance of every possible path permutation.

Permutation Growth: For a given number of cities n , the algorithm evaluates $(n-1)!$ possible routes.

Computational Cost: At $n=10$, the algorithm must process 362,880 permutations. This represents the "complexity wall" for mobile hardware, as the time complexity is $O(n!)$.

Implementation: The algorithm iterates through a lexicographical generation of permutations, tracking the global minimum distance found during the process.

4.2 Genetic Algorithm (Heuristic Approach)

The Genetic Algorithm (GA) is a stochastic optimization technique inspired by Darwinian evolution. Unlike brute-force, it explores the search space efficiently to find near-optimal solutions in $O(g * p * n^2)$ time, where g is generations and p is population size.

The GA implementation follows these five primary phases:

- 1) **Initialization:** A random "population" of path sequences (chromosomes) is generated. Each

sequence represents a valid tour visiting every city exactly once.

- 2) **Fitness Calculation:** Every path is evaluated using a fitness function, defined as:

$$Fitness = \frac{1}{Total\ Distance\ of\ Path}$$

Shorter paths receive higher fitness scores, increasing their likelihood of selection for breeding.

- 3) **Selection:** We utilize Tournament Selection to choose the best-performing "parents" from the current generation to pass their genetic material to the next.

- 4) **Reproduction (Crossover & Mutation):**

Crossover: Two parent paths are combined to produce "offspring" that inherit partial routes from both.

Swap Mutation: To prevent the algorithm from getting stuck in "local optima," we introduce randomized changes by swapping two cities in a path sequence. This maintains genetic diversity.

- 5) **Termination:** The process repeats for a fixed number of generations (e.g., 500) or until no significant improvement is observed in the best path found.

City Count (n)	Brute-Force Time (s)	Genetic Algorithm Time (s)
5	0.0004	0.1236
6	0.0022	0.1241
7	0.0169	0.1307
8	0.1492	0.1376
9	1.5159	0.1506
10	17.4633	0.1605

Table 1: Execution Time by Algorithm based on Node Count (n)

As demonstrated in Table 1, the brute-force search handles $n = 5$ through $n = 8$ with relative ease, leaning on the peak clock speeds of the Cortex-A76 cores. However, the factorial scaling creates a "complexity wall." Between $n = 9$ and $n = 10$, the execution time jumps exponentially from 1.51 seconds to 17.46 seconds. If $n = 11$ were calculated, the device would likely experience thermal throttling or take several minutes to complete.

Conversely, the Genetic Algorithm demonstrates remarkable stability. Scaling the problem from 5 cities to 10 cities barely impacts the execution time, capping at roughly 0.16 seconds. While the GA's initial overhead makes it slightly slower than brute-force for extremely small datasets ($n = 5$), its linear-scaling nature makes it vastly superior as problem complexity increases.

5. RESULTS AND DISCUSSION:

The terminal output from the mobile execution clearly illustrates the theoretical limits of factorial scaling on mobile hardware.

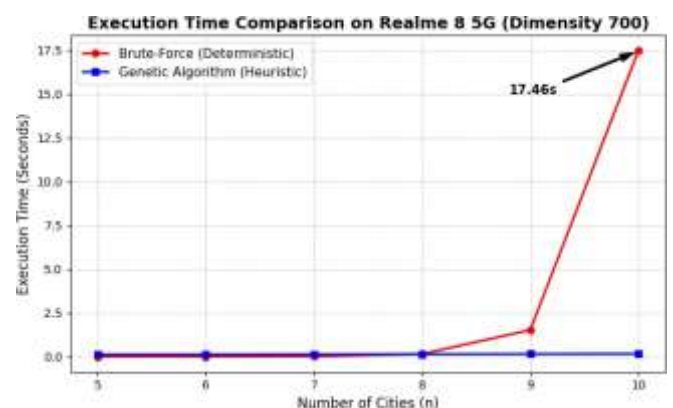


Fig -1: Execution Time Comparison between Brute-Force and Genetic Algorithm on MediaTek Dimensity 700

6. CONCLUSIONS:

This case study highlights the absolute necessity of heuristic optimization for mobile edge computing. The MediaTek Dimensity 700 inside the Realme 8 5G is a highly capable processor, but even modern ARM hardware cannot overcome the mathematical reality of $O(n!)$ time complexity. Deterministic algorithms reach a critical bottleneck at just 10 nodes (17.46s). Therefore, developers designing mapping, routing, or optimization software for local execution on smartphones must implement heuristic models like Genetic Algorithms to maintain sub-second responsiveness and preserve user experience.

ACKNOWLEDGEMENT:

The author gratefully acknowledges the School of Engineering and Technology (SOET) at D Y Patil University, Ambi, Pune, for providing the foundational academic environment that made this independent research possible.

REFERENCES:

- 1) □ Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2006). *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.
- 2) □ Satyanarayanan, M. (2017). "The Emergence of Edge Computing." *Computer*, 50(1), 30-39.
- 3) □ Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems*. MIT Press.
- 4) □ ARM Limited. (2020). *big.LITTLE Technology Overview*. ARM Architecture Reference Documentation.