Graph-Based Genetic Optimization for Vehicle Routing via Giant Tour Decomposition

1. Overview and Motivation

Classical Genetic Algorithms (GAs) for Vehicle Routing Problems (VRPs) typically rely on population-based search over route-level encodings, combined with crossover operators based on fixed cut points (e.g., one-point, two-point, or order-based crossover). While effective in many contexts, such operators suffer from two fundamental limitations. First, they treat the inheritance process syntactically rather than structurally, copying contiguous segments without explicitly evaluating their contribution to global solution quality. Second, feasibility and route structure are often enforced indirectly, requiring costly repair operators or post-processing steps.

To address these limitations, we propose a **graph-based genetic optimization framework** that operates on **giant tours** and uses a **graph-theoretic splitting mechanism** to generate feasible routes. The approach combines a **Genetic Algorithm for global exploration** with a **graph-based optimization layer** that simultaneously performs route splitting, local search, and crossover decisions. The key novelty lies in replacing classical crossover operators with a **shortest-path-driven inheritance mechanism**, where a graph explicitly decides which solution components are inherited from each parent.

The method is implemented in **Java 23**, exploits **multi-threaded graph construction and evaluation**, and integrates **intra-route and inter-route local search** directly inside the graph, rather than as an external improvement phase.

---

2. Giant Tour Representation and Global Search

The core genetic search operates on a **giant tour representation**, where a solution is encoded as a permutation of all customers, without explicit route delimiters. This encoding provides a compact and flexible representation that avoids feasibility constraints during the genetic search phase. Genetic operators act on permutations, enabling broad exploration of the solution space without prematurely enforcing capacity or route constraints.

The Genetic Algorithm evolves a population of giant tours using selection, mutation, and a **non-classical crossover operator** described later. Fitness evaluation is not performed directly on the giant tour; instead, each individual is decoded into a feasible VRP solution via a **graph-based splitting procedure**. This separation of concerns allows the GA to focus on global sequencing decisions, while the graph handles feasibility and cost optimization.

---

3. Graph-Based Splitting into Feasible Routes

For each giant tour, a **directed acyclic graph (DAG)** is constructed to model all feasible ways of splitting the tour into routes. Nodes represent positions in the giant tour, and an arc from node *i* to node *j* represents a feasible route covering customers from position *i+1* to *j*, respecting vehicle capacity and other constraints.

Each arc is weighted by the exact cost of the corresponding route, including depot connections. The problem of splitting a giant tour into optimal routes is then reduced to a **shortest path problem** on this graph, solvable efficiently using dynamic programming or label-setting algorithms.

This graph is not static. It is constructed using **multi-threading**, where arc evaluations (cost and feasibility checks) are distributed across threads. This design significantly reduces decoding time, which is critical given that decoding is performed for every individual in the population at every generation.

---

4. Embedded Local Search Inside the Graph

A key innovation of the proposed approach is that the graph is not merely a decoding structure but an **active optimization component**. During graph construction and evaluation, we perform:

- **Intra-route local search**, such as 2-opt or segment reversals within candidate routes represented by arcs.
- **Inter-route local search**, where alternative split points and customer reallocations are explored implicitly through competing arcs.

By embedding these local improvements directly into the graph, the shortest path computation naturally selects improved routes without requiring a separate local search phase after decoding. This tightly integrated design blurs the classical separation between construction, improvement, and evaluation, leading to more coherent optimization behavior.

---

5. Graph-Based Genetic Crossover

The most distinctive contribution of this work is a **graph-based genetic crossover operator**, fundamentally different from classical cut-point-based crossovers.

Given two parent giant tours, instead of exchanging contiguous segments at predefined cut points, we construct a **combined graph** that contains candidate arcs derived from both parents. Each arc is tagged with its parent of origin and represents a feasible route fragment or subsequence inherited from that parent.

The graph then evaluates all possible combinations of inherited components using a **shortest path algorithm**. The resulting offspring is not formed by blindly copying segments but by **selecting the best-performing parts of each parent**, as determined by global cost minimization. In other words, inheritance is decided by optimization, not by syntactic position.

This mechanism has several important properties:

- Good structural components (high-quality route fragments) are preserved even if they are non-contiguous in the original giant tour.
- Poor components are naturally discarded if they do not appear on the shortest path.
- Feasibility is guaranteed by construction, as only feasible arcs are included.

This crossover can be interpreted as a **recombination through optimization**, where the offspring is the best solution that can be assembled from the building blocks provided by both parents.

---

6. Parallelism and Scalability

The approach is designed with scalability in mind. Graph construction, arc evaluation, and local search operations are parallelized using Java 23's concurrency mechanisms. This is particularly important for large-scale instances, where the number of possible arcs grows quadratically with the tour length.

Parallelism is exploited at multiple levels:

- Arc feasibility and cost computation
- Local search evaluation
- Independent decoding of individuals in the population

This design allows the algorithm to scale efficiently on modern multi-core architectures without altering the underlying optimization logic.

---

7. Positioning with Respect to Existing Methods

Compared to classical Genetic Algorithms and Memetic Algorithms:

- Crossover is **optimization-driven**, not cut-point-driven.
- Local search is **embedded inside the decoding graph**, not applied afterward.
- Feasibility is enforced naturally through shortest path selection.

Compared to pure graph-based or dynamic programming approaches:

- Global exploration is maintained through population-based genetic search.
- Diversity is preserved via giant tour permutations and mutation.

The method can thus be seen as a **hybrid between evolutionary computation and graph-theoretic optimization**, where the graph plays a central decision-making role rather than serving as a passive decoder.

---

8. Conclusion

This approach introduces a new way of combining Genetic Algorithms and graph-based optimization for VRPs. By elevating the graph from a decoding tool to a core genetic operator, and by letting shortest path optimization decide inheritance, the method overcomes key limitations of classical crossover operators. The integration of multi-threading, embedded local search, and optimization-driven recombination results in a flexible, scalable, and conceptually unified framework for solving large-scale routing problems.

Othmane EL YAAKOUBI - MagicVRP TEAM