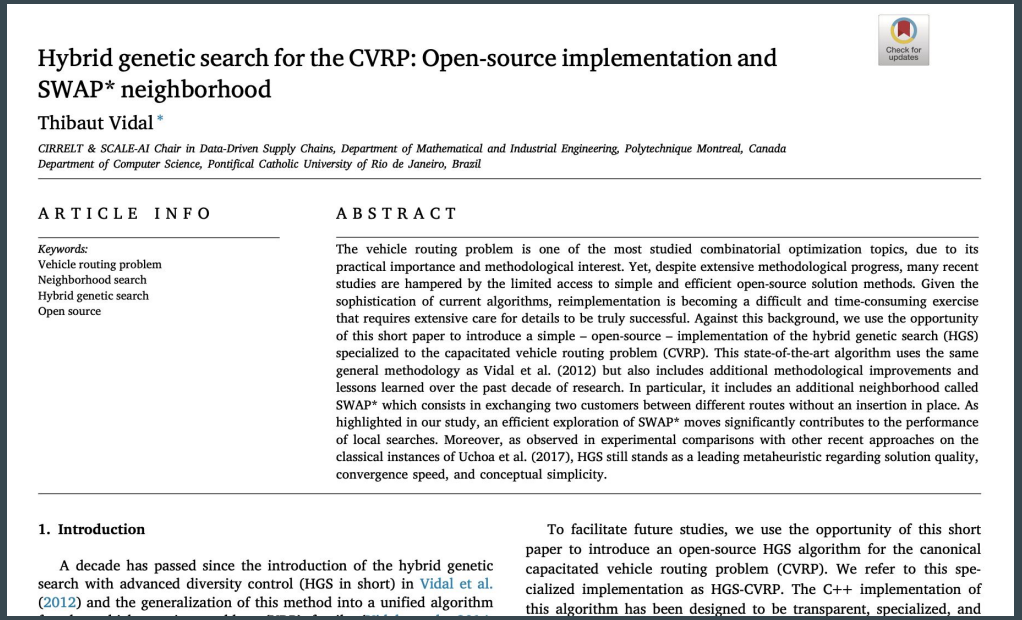
Genetic Algorithms

The algorithm that we developed is based on Vidal 2021, Computers & Operations Research



Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood

Thibaut Vidal *

CIRRELT & SCALE-AI Chair in Data-Driven Supply Chains, Department of Mathematical and Industrial Engineering, Polytechnique Montreal, Canada
Department of Computer Science, Pontifical Catholic University of Rio de Janeiro, Brazil

ARTICLE INFO

Keywords:
Vehicle routing problem
Neighborhood search
Hybrid genetic search
Open source

ABSTRACT

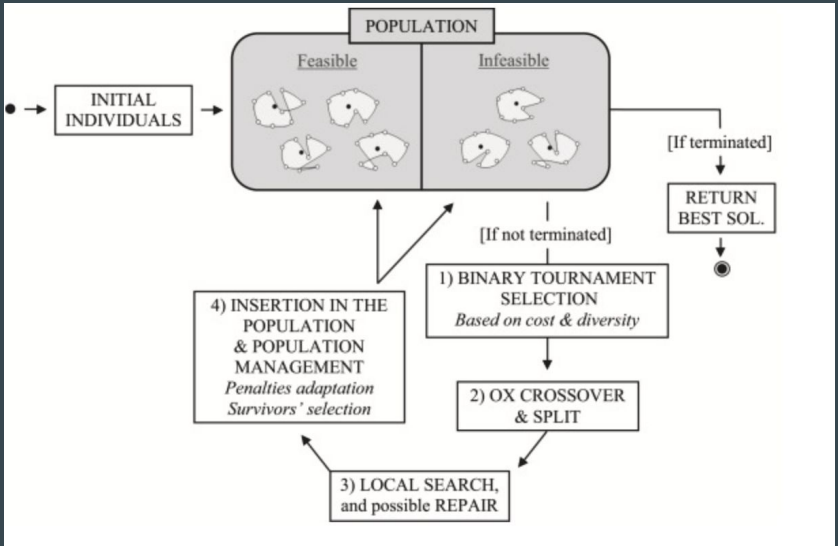
The vehicle routing problem is one of the most studied combinatorial optimization topics, due to its practical importance and methodological interest. Yet, despite extensive methodological progress, many recent studies are hampered by the limited access to simple and efficient open-source solution methods. Given the sophistication of current algorithms, reimplementation is becoming a difficult and time-consuming exercise that requires extensive care for details to be truly successful. Against this background, we use the opportunity of this short paper to introduce a simple – open-source – implementation of the hybrid genetic search (HGS) specialized to the capacitated vehicle routing problem (CVRP). This state-of-the-art algorithm uses the same general methodology as Vidal et al. (2012) but also includes additional methodological improvements and lessons learned over the past decade of research. In particular, it includes an additional neighborhood called SWAP* which consists in exchanging two customers between different routes without an insertion in place. As highlighted in our study, an efficient exploration of SWAP* moves significantly contributes to the performance of local searches. Moreover, as observed in experimental comparisons with other recent approaches on the classical instances of Uchoa et al. (2017), HGS still stands as a leading metaheuristic regarding solution quality, convergence speed, and conceptual simplicity.

1. Introduction

A decade has passed since the introduction of the hybrid genetic search with advanced diversity control (HGS in short) in Vidal et al. (2012) and the generalization of this method into a unified algorithm

To facilitate future studies, we use the opportunity of this short paper to introduce an open-source HGS algorithm for the canonical capacitated vehicle routing problem (CVRP). We refer to this specialized implementation as HGS-CVRP. The C++ implementation of this algorithm has been designed to be transparent, specialized, and

Implementation



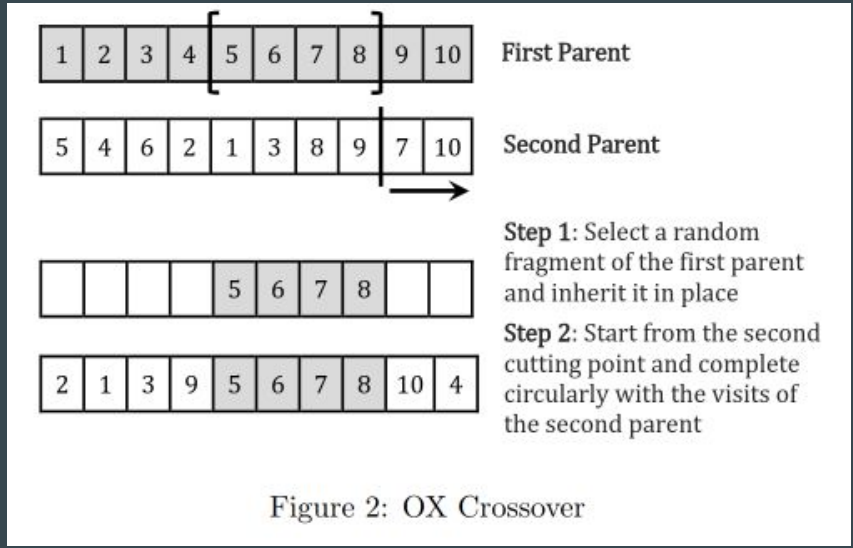


Figure 2: OX Crossover

Some Tuning

- The SWAP* technique mentioned in the paper did not actually perform well (maybe due to our smaller instances and Python overhead)

- Stop condition: 290 seconds passed OR 200 iterations without improvement

- Lots and lots of data structures

Results

- Reasonably competitive solutions for all instances

- Overhead of Python was a major bottleneck

- Move-testing (working with linked lists) in local search step is very slow

```
Collecting samples from 'python src/main.py input/386_47_1.vrp' (python v3.9.21)
Total Samples 28100
GIL: 100.00%, Active: 100.00%, Threads: 1

  %Own   %Total  OwnTime  TotalTime  Function (filename)
 14.00%  14.00%   34.16s    35.98s   move3 (hga_local.py)
  8.00%   8.00%   32.95s    34.49s   move5 (hga_local.py)
  5.00%   8.00%   29.37s    40.68s   move1 (hga_local.py)
 13.00%  13.00%   28.66s    31.33s   move2 (hga_local.py)
  6.00%  78.00%   21.59s    257.6s   run (hga_local.py)
  5.00%   5.00%   18.68s    23.89s   updateRouteData (hga_local.py)
  5.00%   5.00%   17.83s    18.49s   move6 (hga_local.py)
  2.00%   2.00%   16.58s    18.80s   move4 (hga_local.py)
  5.00%   7.00%   14.89s    19.46s   move8 (hga_local.py)
  5.00%   5.00%   13.30s    13.30s   setLocalVariables (hga_local.py)
  9.00%   9.00%   12.53s    15.87s   move9 (hga_local.py)
  7.00%   7.00%    8.27s     8.27s   propagate (hga_split.py)
  1.00%   1.00%    6.66s     6.66s   calc_penalty (hga_local.py)
  2.00%  21.00%    5.85s    21.43s   split_lf (hga_split.py)
  0.00%   0.00%    4.01s     5.90s   move7 (hga_local.py)
  8.00%   8.00%    3.30s     3.30s   dominates (hga_split.py)
  0.00%   0.00%    2.36s     3.04s   is_enclosed (hga_circle.py)
  0.00%   0.00%    1.61s     4.92s   extend (hga_circle.py)
  0.00%   0.00%    1.51s     1.51s   dominates_right (hga_split.py)
  1.00%   1.00%    1.08s     1.08s   brokenPairsDistance (hga_population.py)
  0.00%   0.00%   0.950s    0.950s   positive_mod (hga_circle.py)
  2.00%   2.00%   0.870s    0.870s   size (hga_split.py)
  0.00%   1.00%   0.840s     1.33s   loadIndividual (hga_local.py)
  1.00%   1.00%   0.790s    0.790s   get_front (hga_split.py)
  1.00%   1.00%   0.350s    0.350s   get_back (hga_split.py)
  0.00%   0.00%   0.250s    0.250s   evaluateCompleteCost (hga_structures.py)
```

Thank you py-spy

Reflection

- Time Spent: Too long (90+ student hours)

- Many sunrises witnessed

- Slept for 1 hour night before a final

- Code that got nuked harder than our subpopulations