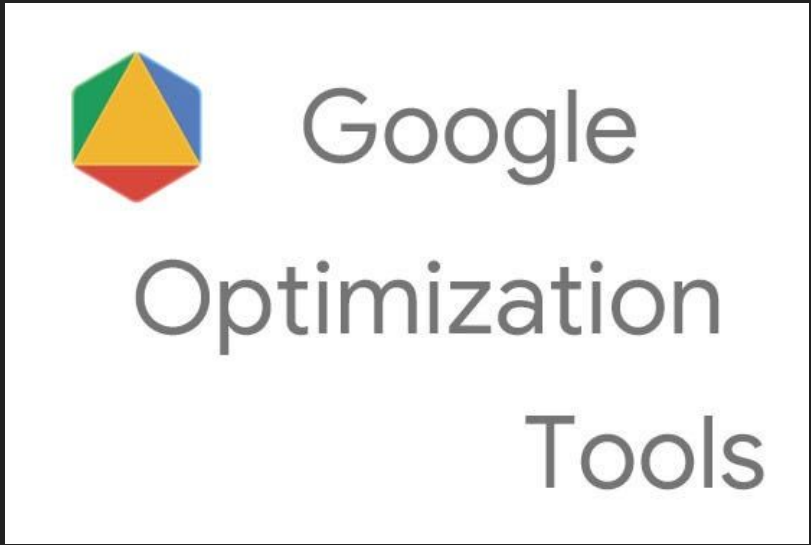
Vehicle Routing Problem

Julian and Alex

Baselines

Can’t solve even the small ones



Pretty good (but only works with integers!?!)

Instances	or-tools-60s
101_11_2.vrp	1959
101_8_1.vrp	800
121_7_1.vrp	1407
135_7_1.vrp	1243
151_15_1.vrp	3089
16_5_1.vrp	329
200_16_2.vrp	-1
21_4_1.vrp	351
241_22_1.vrp	629
262_25_1.vrp	5973
30_4_1.vrp	494
386_47_1.vrp	27105
41_14_1.vrp	845
45_4_1.vrp	712
51_5_1.vrp	528
76_8_2.vrp	719

Large Neighborhood Search

Construct, then Perturb

Construction:

- Random → Polar → Clarke Wright

Perturbation:

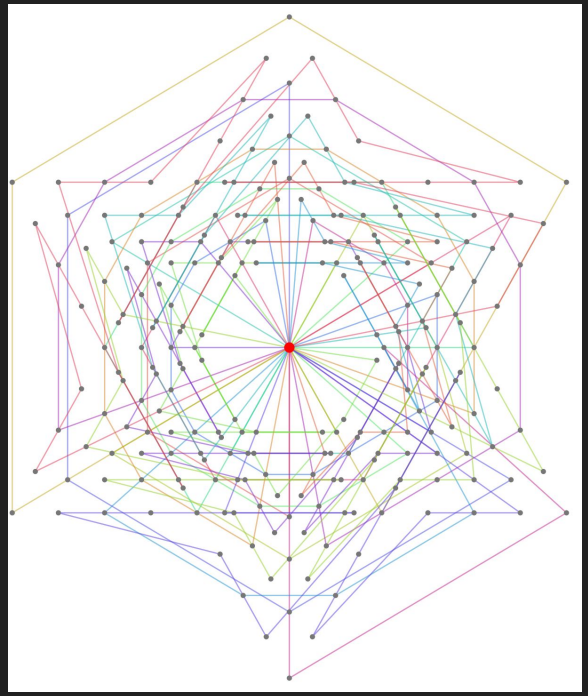
- Destroy/Repair Operators

Exploration:

- Simulated Annealing, Random Destruction

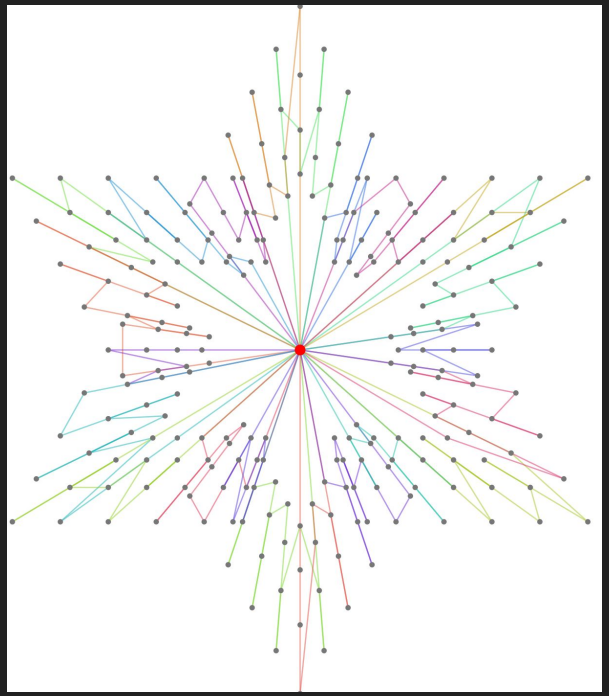
Constructors

Random



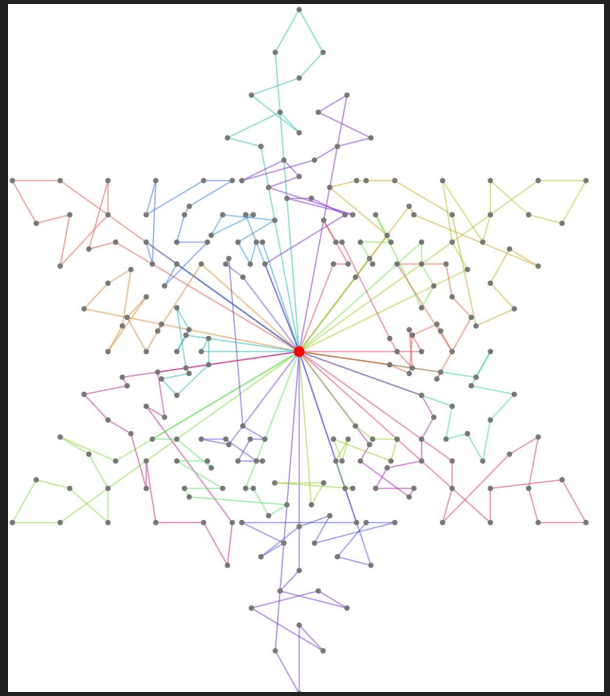
2047

Sweep (Polar Angle)



1253

Clarke-Wright (Fancy)



954

Destroy Operators

- Random 1-Removal
- Random K-Removal
- Random Shaw K-Remove

Repair Operators

- Random Insertion
- Random K-Insertion
- Greedy Insertion
- K-Regret Insertion

- Swaps

Adaptiveness: Intelligently select operators based on the moving history of their performance

Very **Greedy** Exploration

Feasible: Never allow infeasible solutions

Patience: If there are no local improvements in **10** iterations, jump to a new space

Simulated Annealing: Allow worse solutions **10%** of the time

Jumping: Jump back to the **best solution** we've found so far

Engineering Optimizations 🤓

Exploit Data Structures

- **remove redundant computations** in sanity checks (~60x faster)
- don't **filter all possible moves by tabu**, keep a list of those in and out of tabu (~38% faster)

Avoid Allocations

- **initialize vectors** to avoid resizing (7% faster)
- **reuse existing memory** allocations when copying solutions (16% faster)

Help the Compiler

- don't do **bounds checking** on indexing (~20% faster)
- **manually inline** hot distance calculation functions

Multithreading

Engineering Optimizations 🤓

pre

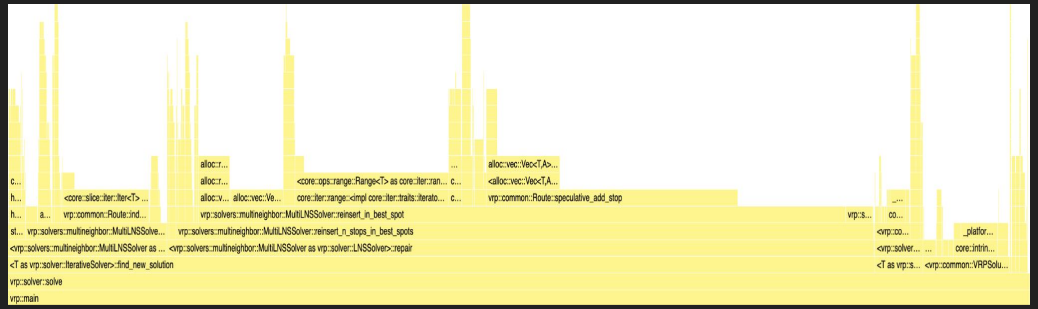
50,000 search iterations

in 60s

post

50,000 search iterations

in 385ms



on *input/386_47_1.vrp*

Engineering Optimizations 🤓

pre

50,000 search iterations

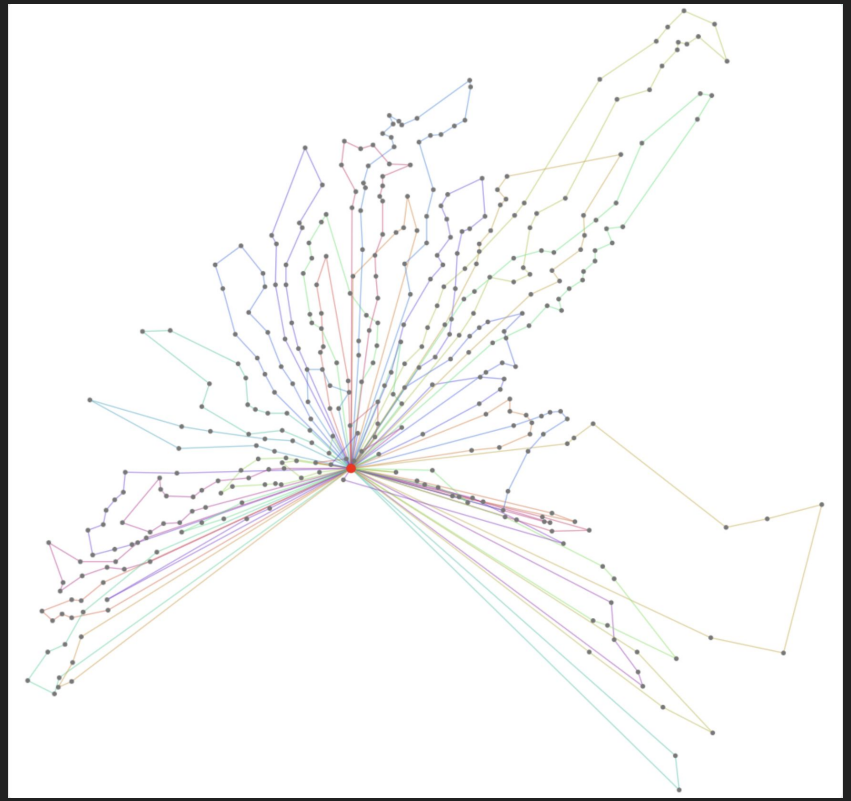
in 60s

post

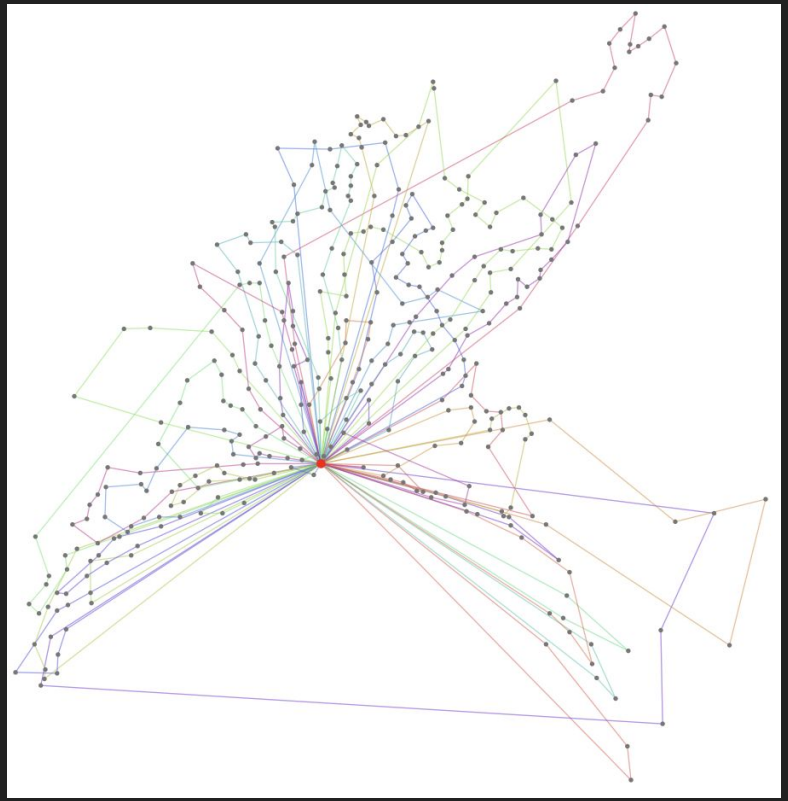
50,000 search iterations

in 385ms

```
Benchmark 1: no-shuffle
  Time (mean ± σ):     839.5 ms ±   7.4 ms    [User: 834.5 ms, System: 2.1 ms]
  Range (min … max):   817.1 ms … 847.6 ms    15 runs

Benchmark 2: clone-cap
  Time (mean ± σ):     766.3 ms ±   4.5 ms    [User: 761.0 ms, System: 1.5 ms]
  Range (min … max):   759.8 ms … 774.3 ms    15 runs

Benchmark 3: no-bounds-checks
  Time (mean ± σ):     688.3 ms ±   5.2 ms    [User: 683.0 ms, System: 1.7 ms]
  Range (min … max):   677.7 ms … 697.4 ms    15 runs

Benchmark 4: clone-from
  Time (mean ± σ):     589.3 ms ±   4.1 ms    [User: 584.9 ms, System: 1.4 ms]
  Range (min … max):   584.8 ms … 601.7 ms    15 runs

Benchmark 5: inline-dist
  Time (mean ± σ):     385.1 ms ±  14.9 ms    [User: 381.3 ms, System: 1.6 ms]
  Range (min … max):   371.2 ms … 419.9 ms    15 runs

Summary
  inline-dist ran
    1.53 ± 0.06 times faster than clone-from
    1.79 ± 0.07 times faster than no-bounds-checks
    1.99 ± 0.08 times faster than clone-cap
    2.18 ± 0.09 times faster than no-shuffle
```



Our solver (120s, 8 threads)



OR Tools (300s)

With the default settings as shown here: https://developers.google.com/optimization/routing/cvrp

Thank You!