Lab AI: The Cambio Problem

2018-2019

Olivier Van den Eede & Dries Kennes

Content

- **Data Structures**
  - Request, Zone & Car objects
  - Solution object
  - Overlap Matrix
- Algorithm
  - Greedy Assign
  - Local Moves
  - Metaheuristic
- Results
  - Performance
  - Search space
  - Possible improvements

Request, Zone & Car objects

- Parsed from input once
- Immutable objects for Request & Zone
- Car is a string
- Request & Zone stored as
  - Immutable list
  - id→object maps

Solution class

- Car→Zone & Request→Car maps
- Only object copied for each local move
  - Input data is not copied
  - If local move was not successful no new copy

Overlap Matrix

- Pre-calculates overlapping requests
  - Calculated only once
- Stored in 2D boolean matrix
- Used in local moves
- Allows for faster feasibility check

Content

- Data Structures
  - Request, Zone & Car objects
  - Solution object
  - Overlap Matrix
- **Algorithm**
  - Greedy Assign
  - Local Moves
  - Metaheuristic
- Results
  - Performance
  - Search space
  - Possible improvements

Greedy Assign

- Loop over unassigned requests
  1) Find suitable car assigned to zone
  2) Find suitable car assigned to neighbor
  3) Assign a new car to zone
  4) Leave unassigned
- Used for initial solution
- Guaranties a feasible solution
- Called after each local move

Local Moves

- Moves
  1) Move request from ‘optimal’ zone to neighbor
  2) Move request from neighbor to ‘optimal’ zone
  3) Swap car assigned to request with another car
  4) Unassign 1 car from all its requests †
  5) Unassign 1 request †
- All guarantie a feasible solution

†: Twice as likely to be picked

Metaheuristic

- Simulated annealing
  1) Random initial solution
  2) Generate new solution with random local move
  3) Solution accepted
     - if cost is lower; or
     - with ever decreasing random probability
  4) Go back to 2 unless stop condition reached
- "Ideal" parameters found after a lot of testing†

†: See results on cambio.dries007.net/sa_parameter_tests

Content

- Data Structures
  - Request, Zone & Car objects
  - Solution object
  - Overlap Matrix
- Algorithm
  - Greedy Assign
  - Local Moves
  - Metaheuristic
- **Results**
  - Performance
  - Search space
  - Possible improvements

Results

- Results generated with 300 seconds on 2 threads.
- Median of a number of runs
- Impressive, or so we think 😀

Input File	Toledo	Our Score	Improvement
100_5_14_25	10015	8795	12.18%
100_5_19_25	6700	6045	9.78%
210_5_33_25	14325	11180	21.95%
210_5_44_25	5890	4045	31.32%
360_5_71_25	12955	7490	**42.18%**

Performance

- Performance is key!
  - Python generators
  - Limiting copying to absolute minimum
  - Profiling of the algorithm
  - No feasability check!
    - Not required
    - Overlap checking is still expensive, even with matrix.
- 12k; 10k; 5k lm/s for 100; 210; 360 requests
  - Complete simulated annealing takes ~10 seconds

Search space

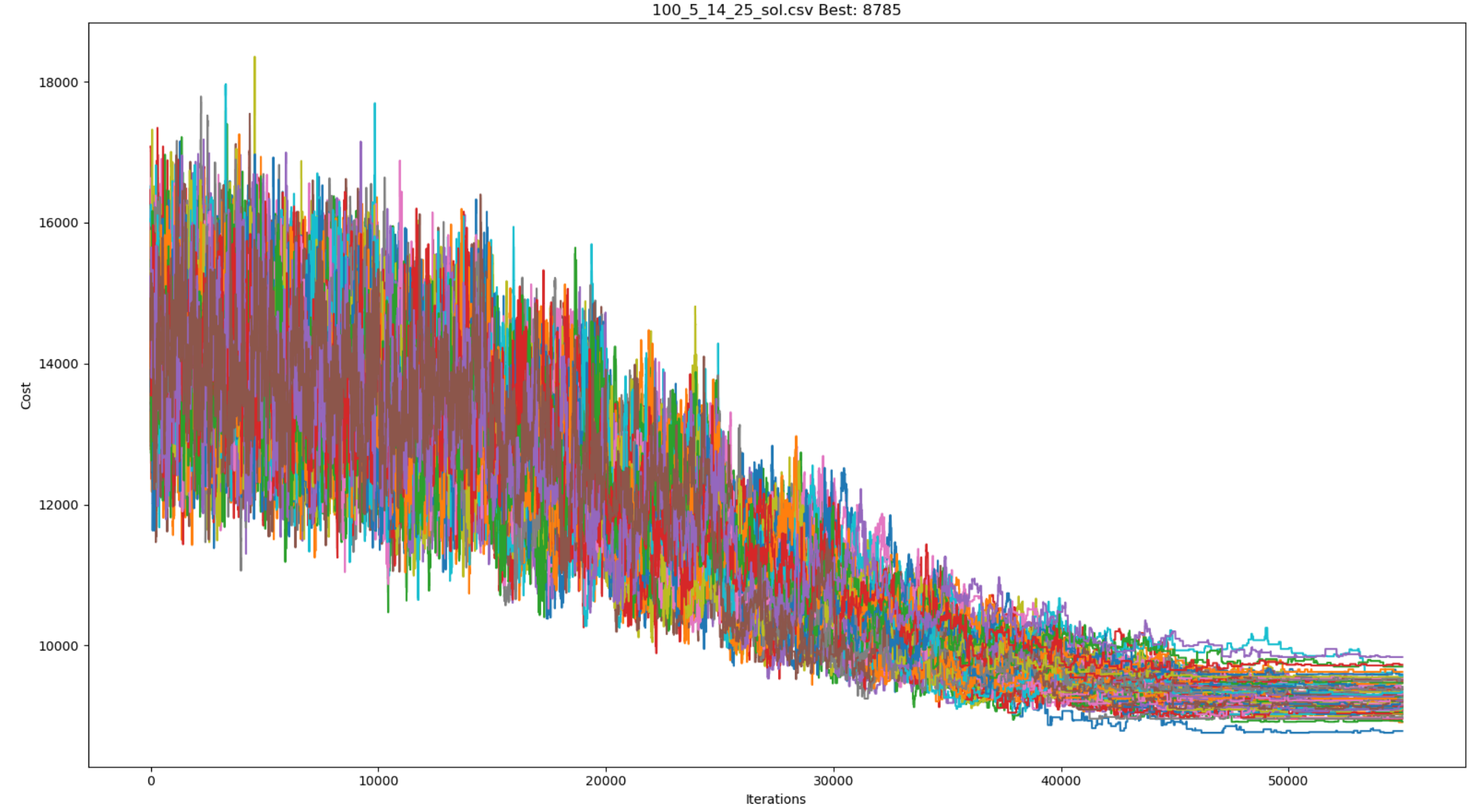
- Greety assign after every move
  - Limits search space
  - Example: No unassign two cars without attempting to assigning requests to them.
- Limited local moves
- No crossing "gaps of infeasibility"

Possible improvements

- Remove search space limitations
- Multipliers for local moves
  - Picked 2x on unassign moves with limited testing

Questions?

Example results for `100_5_14_25`



Example results for `360_5_71_25`