Bi-Level Optimisation for Assignment and Routing Problems

Tom Kent

August 7, 2018

Report Overview

Report Overview

→ Strand 1 - Hierarchical Decomposition for Allocation and Routing Problems

- Strand 2 - Intelligent Agents & Hierarchical Authority
- Strand 3 - Hierarchical Reinforcement Learning
- Strand 4 - The Role of Expert Feedback and Imitation Learning
- Strand 5 - Hierarchical Emergence, Evolution & Adaptation

Assignment Problems

Assignment Problem

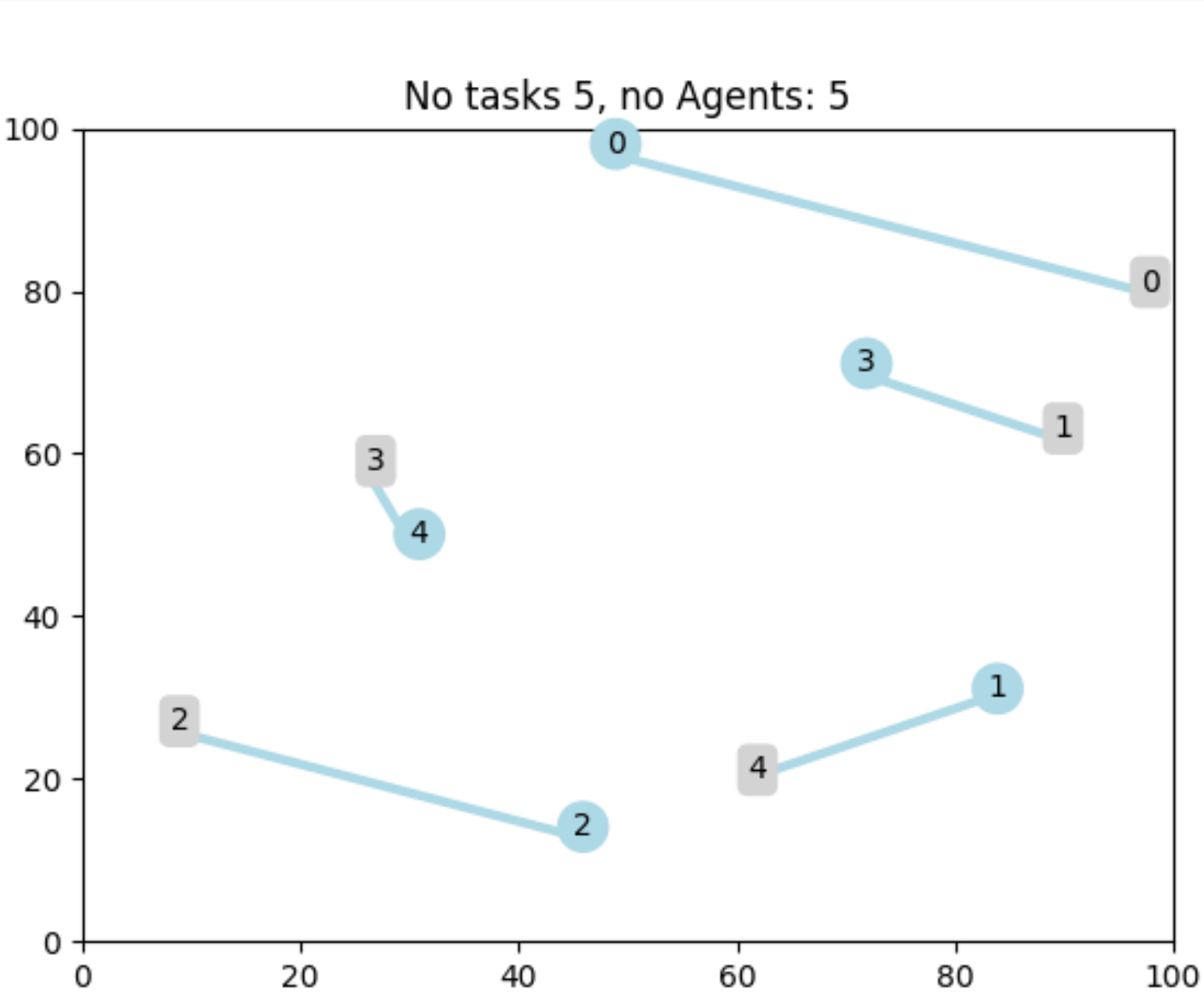
T - set of Tasks, A - set of Agents, V - values

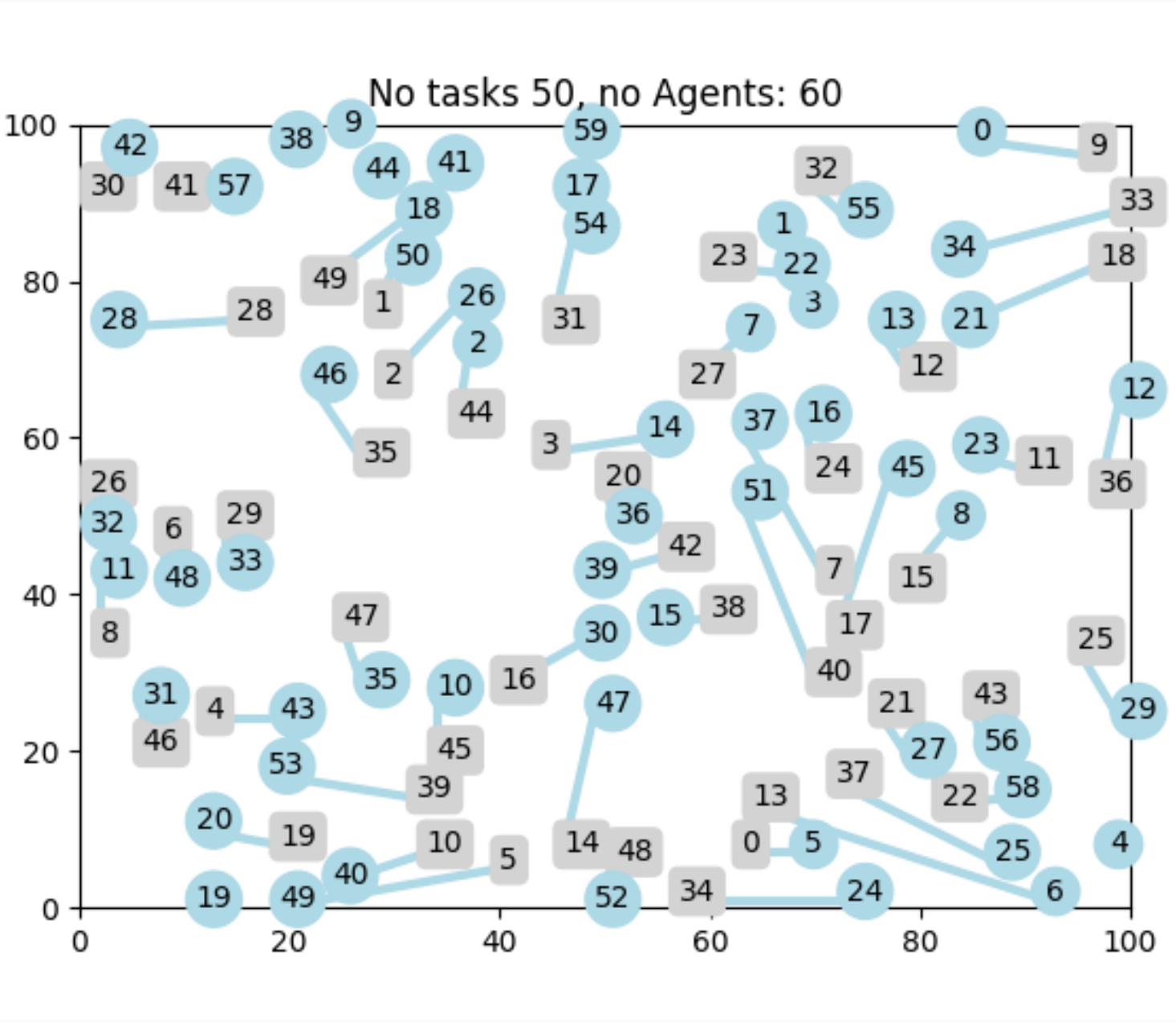
$$\textbf{max} \sum_{i \in A} \sum_{j \in T} V_{ij} x_{ij} \tag{1}$$

$$\sum_{j \in T} x_{ij} \leq 1 \qquad \text{for } i \in A \tag{2}$$

$$\sum_{i \in A} x_{ij} = 1 \qquad \text{for } j \in T \tag{3}$$

$$x \in \{0, 1\} \tag{4}$$


No tasks 5, no Agents: 5
0
0
3
1
3
4
1
2
4
2
0
20
40
60
80
100


No tasks 50, no Agents: 60
0
20
40
60
80
100

Bilevel Optimisation

Bilevel Optimisation

Many situtations involve the analysis of several objectives that reflect a hierarchy of decision-makers.

Multilevel optimization techniques **partition control over decision variables** amongst the levels. Decisions at each level may be **constrained by decisions at other levels**, and objectives for each level may account for decisions made at other levels.

In practice, multilevel problems are difficult to solve - most of the literature focused on bilevel programs [2].

Bilevel Optimisation

$$\max_{x \in X, y} F(x, y) \tag{5}$$
$$\text{s.t.} \, G(x, y) \leq 0 \tag{6}$$
$$y \in P \tag{7}$$

where

$$P(x) = \arg\min_{y \in Y} f(x, y) \tag{8}$$
$$g(x, y) \leq 0 \tag{9}$$

$P(x)$ defines a lower-level problem, which may have multiple solutions.

Here x is the primary upper-level decision, and y is the anticipated lower-level decision.

Pyomo Modelling

Pyomo is a Python-based open-source software package that supports a diverse set of optimization capabilities for formulating, solving, and analyzing optimization models.

Pyomo can be used to define general symbolic problems, create specific problem instances, and solve these instances using commercial and open-source solvers.

Pyomo Modelling Example

```
# Top level
m = ConcreteModel()
m.x = Var(bounds=(1,2))
m.y = Var(bounds=(1,2))
m.o = Objective(expr=m.x + m.y, sense=minimize)

# Sub Level
m.sub = SubModel(fixed=m.x)
m.sub.z = Var(bounds=(-1,1))
m.sub.o = Objective(expr=m.x*m.sub.z, sense=maximize)
m.sub.c = Constraint(expr=model.y + model.sub.z <= 2)

instance = m.create_instance()
results = opt.solve(instance)
```

Semi-Autonomous AP (SAAP) [1]

The Semi-Autonomous Assignment Problem (SAAP) adds *Selfish* behaviour in assignment scenarios.

Here some agents are *autonomous* (or Free) and some are *controlled*, and the central planner makes an assignment only for the controlled agents.

These *autonomous* agents have private incentives, unknown to the central planner, and look to optimise for their own objective.

Semi-Autonomous AP (SAAP) [1]

By assigning the controlled agents, the central planner can block some tasks and in this way essentially determine the outcome to a certain extent.

The central planner need not form a belief about utility functions of the autonomous agents, nor does it assume the autonomous agents to be expected utility maximisers.

SAAP Formulation i

T - Tasks, C - Controlled Agents, F - Free Agents, V - Values

Top level optimisation:

$$\textbf{min}_x \sum_{i \in C} \sum_{j \in T} V_{ij} x_{ij} + \sum_{i \in F} \sum_{j \in T} V_{ij} y_{ij} \tag{10}$$

$$\sum_{j \in T} x_{ij} \leq 1 \qquad \text{for } i \in C \tag{11}$$

$$\sum_{i \in C} x_{ij} \leq 1 \qquad \text{for } j \in T \tag{12}$$

$$x_{ij} \in \{0, 1\} \qquad \text{for all } (i, j) \in (C \times T) \tag{13}$$

SAAP Formulation ii

Lower level optimisation:

$$\textbf{y solves max} \sum_{i \in F} \sum_{j \in T} U_{ij} y_{ij} \tag{14}$$

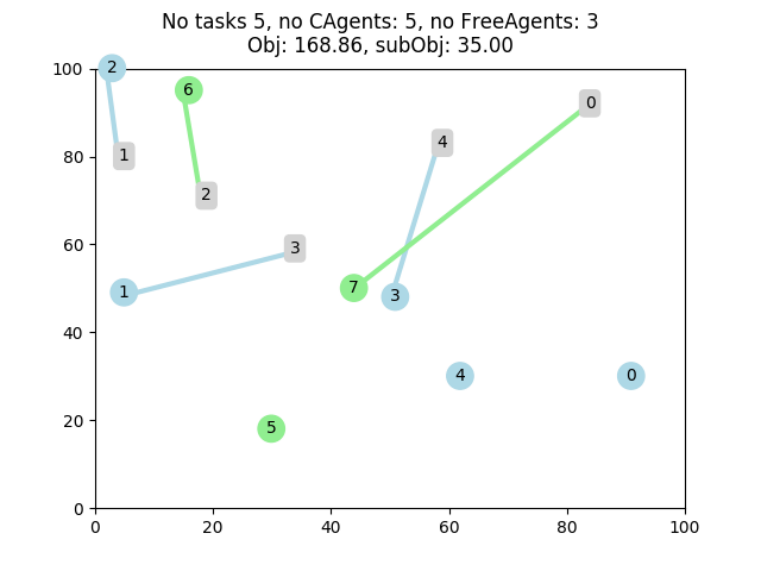
$$\sum_{j \in T} y_{ij} \leq 1 \qquad \text{for } i \in F \tag{15}$$

$$\sum_{i \in F} x_{ij} + \sum_{i \in C} x_{ij} \leq 1 \qquad \text{for } j \in T \tag{16}$$

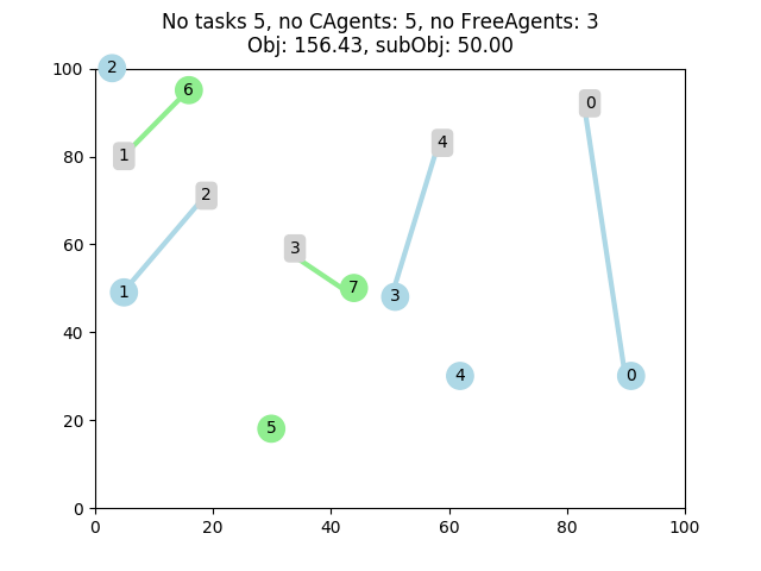
$$x_{ij} \in \{0, 1\} \qquad \text{for all } (i, j) \in (C \times T) \tag{17}$$

This is a Mixed Integer Linear Program. The paper focuses on converting it in a disjoint bilinear program, by replacing the lower LP with its dual so that they are both maximisation problems and can then be reduced to a single stage optimisation.

SAAP example 1

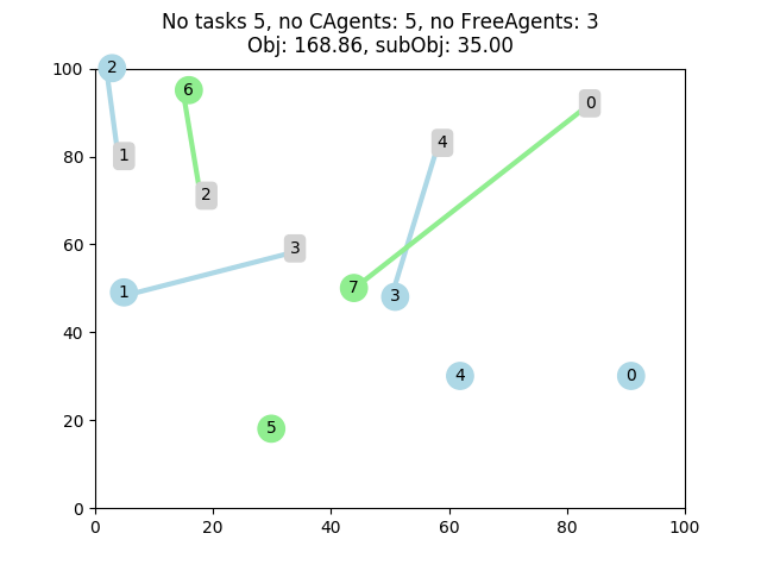


(a) Original SAAP

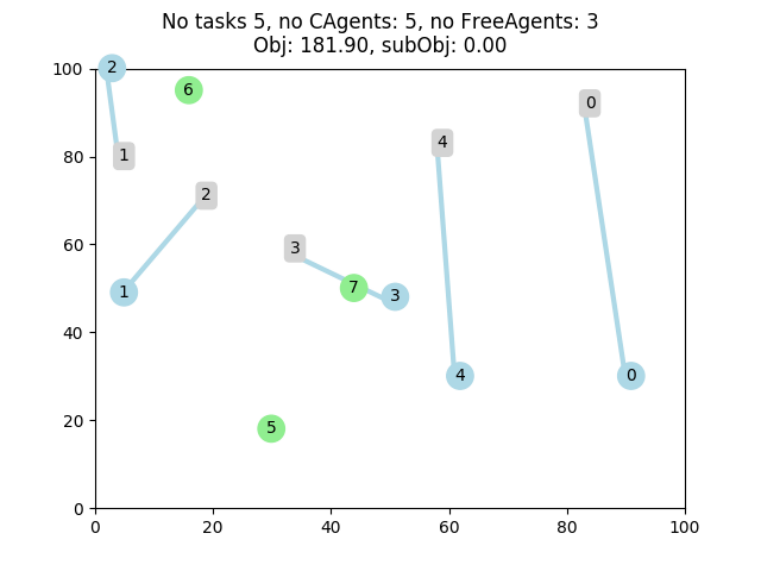


(b) Different Utility Values

SAAP example 2

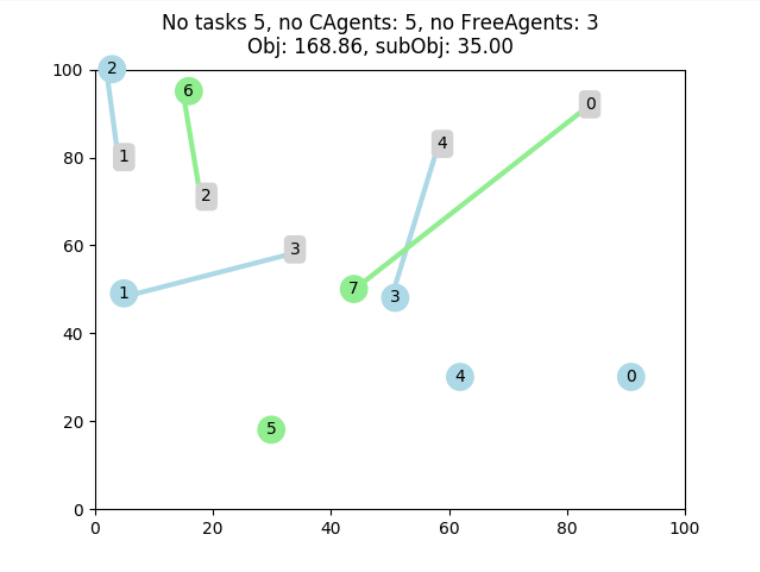


(a) Original SAAP

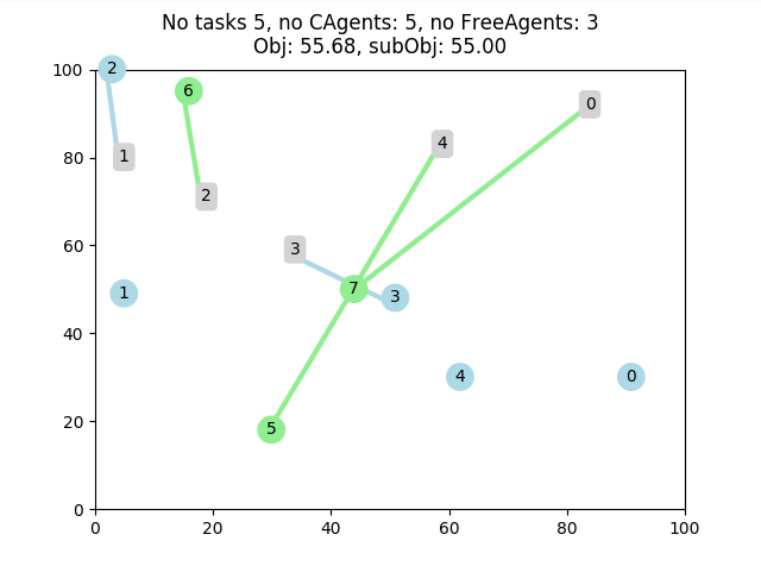


(b) x10 Free Agent Cost

SAAP example 3



(a) Original SAAP



(b) x0.1 Free Agent Cost

SAAP summary

- Model-based hierarchy - Can define constraints, variables and objectives in a way that outlines their interactions.
- Solution method requires essentially converting to a single layer optimisation (the SAAP paper converts the original Bi-level into a single level problem)
- Its not tractable - can handle at most about 10 tasks and 10 agents - Free agents are more 'costly'
- Global solution method: BLP $\rightarrow$ MPEC $\rightarrow$ GDP $\rightarrow$ MIP

Routing Problems

Idea

Assignment Problems work well for assigning tasks to agents based on a cost, but what if we want the cost to reflect the agent needed to actually travel.

Can a similar bi-level approach be taken?
- Top level: Assignment - optimised for some metric;
- Bottom level: Routing - optimised to minimise distance/fuel

... maybe

Multi-Agent Travelling Salesman (MATSP)

$$\min \sum_{i \in T} \sum_{j \in T} c_{ij} \sum_{a \in A} x_{ija} \tag{18}$$

$$\text{s.t.} \sum_{i \in T} \sum_{a \in A} x_{ija} = 1, \ \forall j \in T \tag{19}$$

$$\sum_{i \in T} x_{ipa} - \sum_{j \in T} x_{pja} = 0, \ \forall a \in A, p \in T \tag{20}$$

$$\sum_{j \in T} x_{1ja} = 1, \ \forall a \in A \tag{21}$$

$$u_i - u_j + n \sum_{a \in A} x_{ija} \leq n - 1, \ \forall i \neq j \in T \tag{22}$$

$$x_{ija} \in \{0, 1\} \ \forall i, j, a \tag{23}$$

Bi-Level MATSP i

Top level is the Assignment Problem:

$$\max \sum_{a \in A} \sum_{i \in T} v_{ij} x_{ai} \tag{24}$$

$$+ \sum_{a \in A} \sum_{j \in T} \sum_{i \in T} v_{ij} y_{aij} \tag{25}$$

$$\text{s.t.} \sum_{i \in T} x_{ai} = 1 \ \forall a \in A \tag{26}$$

$$\sum_{a \in A} x_{ai} = 1 \ \forall i \in T \tag{27}$$

Bi-Level MATSP ii

Bottom level is the Travelling Salesman:

$$\textbf{y solves min} \sum_{a\in A}\sum_{j\in T}\sum_{i\in T} D_{ij}y_{aij} \tag{28}$$

$$\text{s.t.} \sum_{i\in T}\sum_{a\in A} y_{aij} = 1,\ \forall j \in T \tag{29}$$

$$\sum_{i\in T} x_{aip} - \sum_{j\in T} x_{apj} = 0,\ \forall a \in A, p \in T \tag{30}$$

$$\sum_{j\in T} x_{a1j} = 1,\ \forall a \in A \tag{31}$$

$$u_i - u_j + n\sum_{a\in A} x_{aij} \leq n-1,\ \forall i \neq j \in T \tag{32}$$

$$x_{aij} \in \{0,1\}\ \forall a,i,j \tag{33}$$

$$\text{s.t. assignment } x \text{ matches route } y \tag{34}$$

Bi-Level MATSP iii

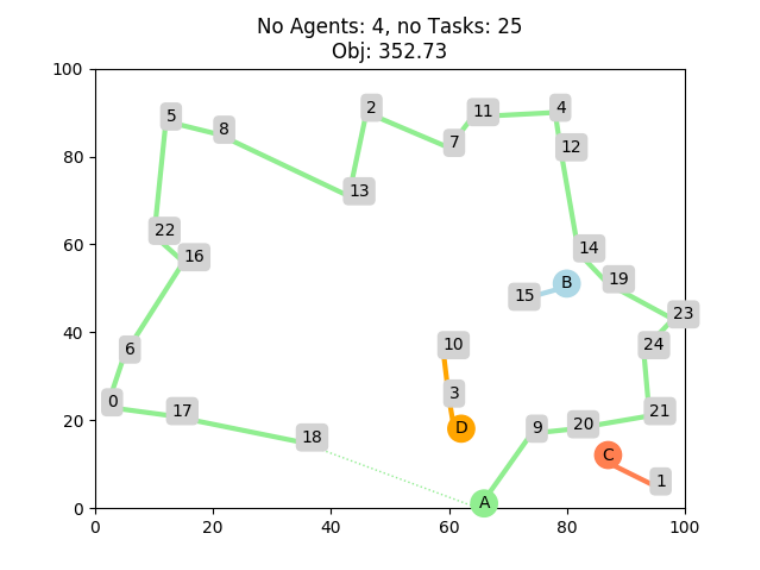
- Still trying to get this to work.
- Might be intractable, might just be poorly formulated.
- If a simpler problem like SAAP struggles at larger sizes a much more complicated AP + TSP won't be very scalable.

Next Direction

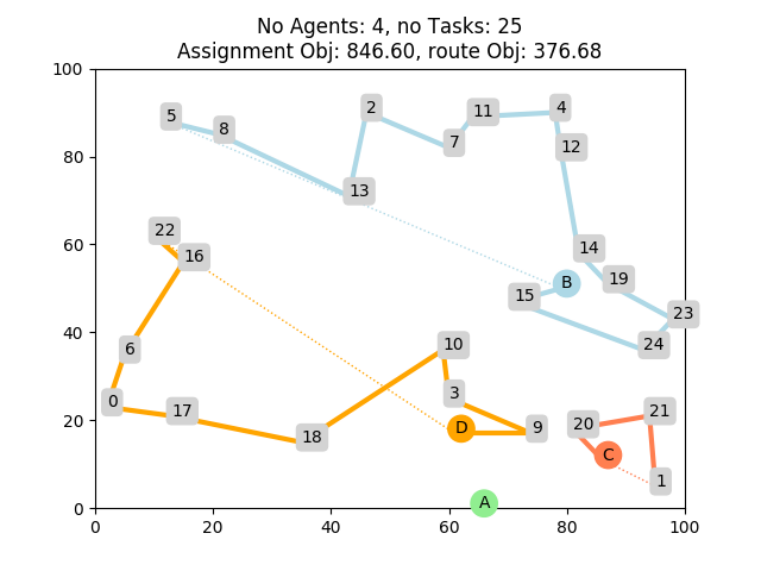
If there is no ‘silver bullet’ for modelling and solving these types of problems into a single optimisation, providing global optimality is that a deal breaker?

- Can the AP and the TSP remain seperated?
- Either the AP and TSP are optimised for different metrics e.g. risk, cost, distance.
- Or the AP and TSP are designed to better approximate the MATSP in a hierarchical way.

MATSP vs AP+TSP i

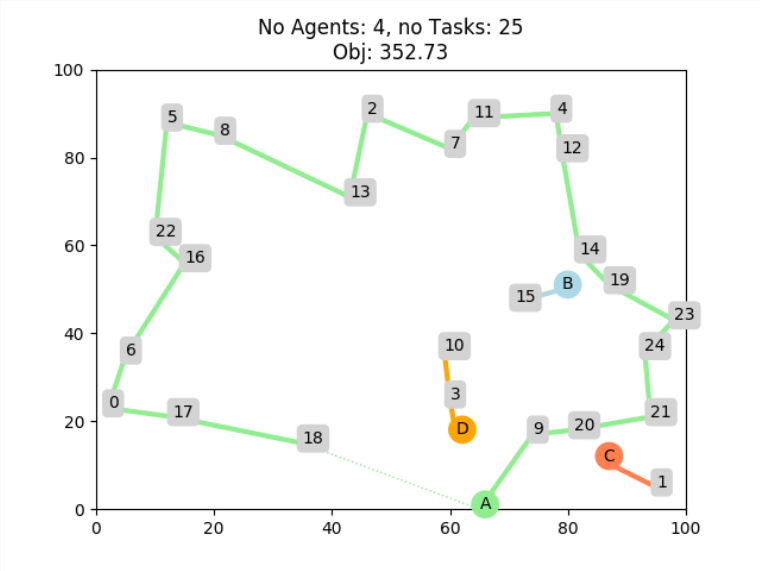


(a) MATSP

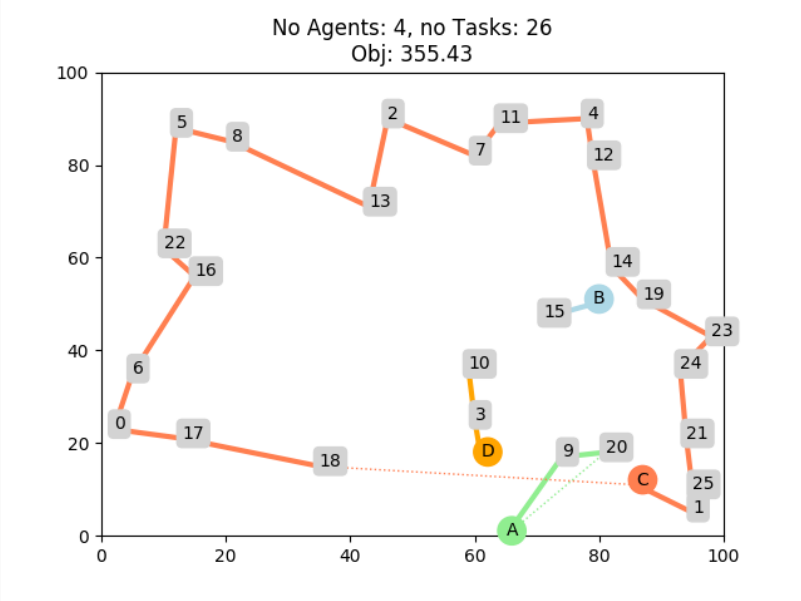


(b) AP + TSP

MATSP vs AP+TSP ii

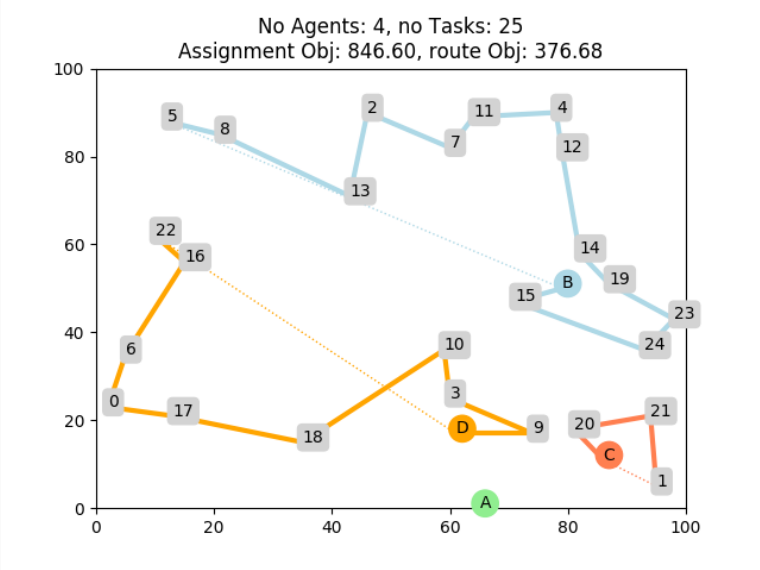


(c) MATSP

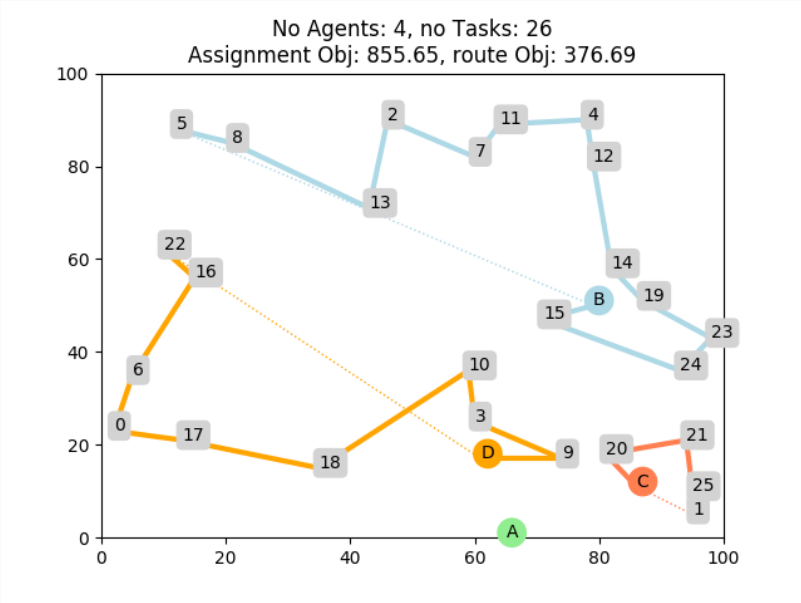


(d) MATSP Added Task

MATSP vs AP+TSP iii



(e) AP + TSP



(f) AP + TSP Added Task

Allocation and Routing Summary

- AP problems are straight forward to solve (for 100s of agents and tasks)
- Objective function needs to successfully quantify underlying 'costs'.
- Routing problems are slightly more complex: TSP $\rightarrow$ VRP etc
- Multi-Agent TSPs (and VRPs) have an implicit Assignment based on the routes
- Allocation, as a heuristic, has it's benefits
- MATSP can be split into a Two-Stage AP and TSP - Can the original allocation be improved after knowlege of the cost of routing? Perhaps there is need for feedback.

Multi Objective Evolutionary Algorithms with Decomposition

MOEA/D i

$$\text{minimize } F(X) = (f_1(x), \ldots, f_n(x))$$

- In most practical Multi-objective optimization problems, the objectives are mutually conflicting. [4]
- Aim to find Pareto-Optimal trade-off solutions and then an approximate set to the Pareto Front
- Classical MO approaches produce a single PO solution per run, MOEA can obtain several.

MOEA/D ii

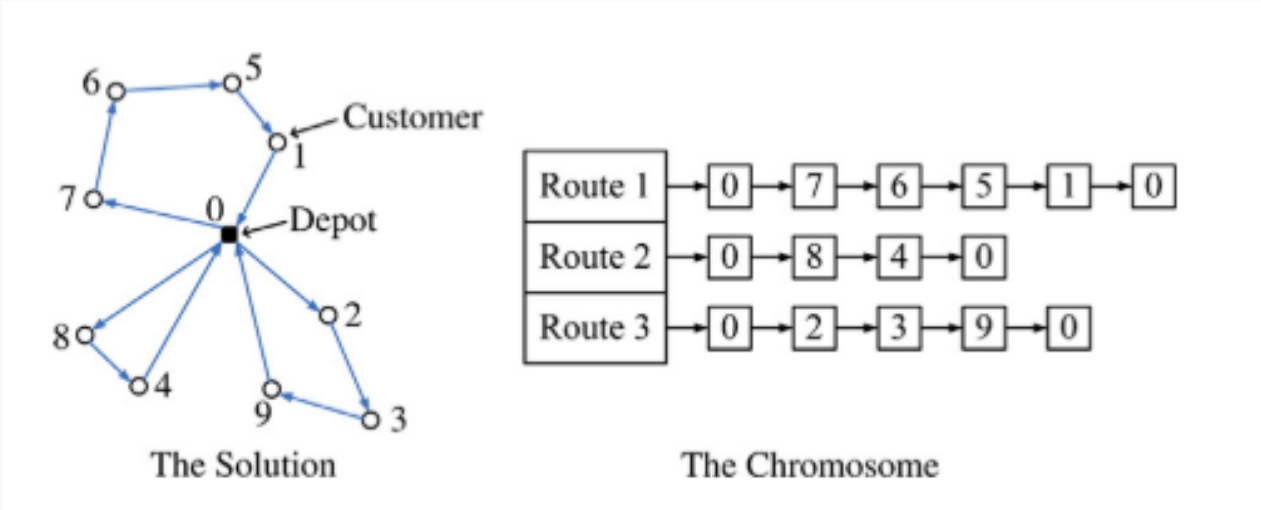
- Decomposition-Based MOEA: scalarizing functions convert the MOP into single-objective optimization subproblems solved using an EA and evolving a population of solutions.
- The objective function of each sub-problem is an aggregation of all f:

$$\Phi(x) = \sum_{i=1}^{m} \lambda_i f_i(x)$$

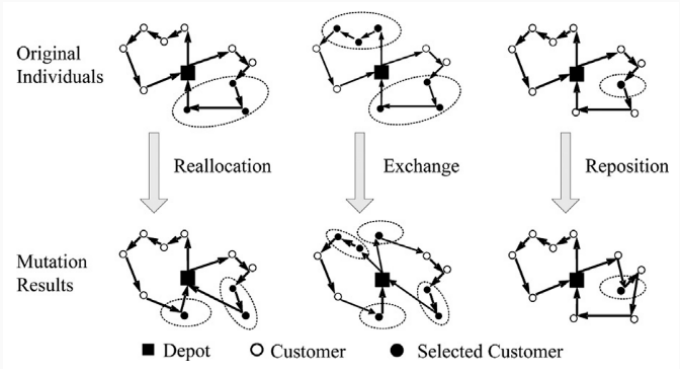
Qi et al. outline a Multi-objective VRP with time windows [3] using MOEA/D

EA ideas of how to handle crossover or two parents, and mutations among populations

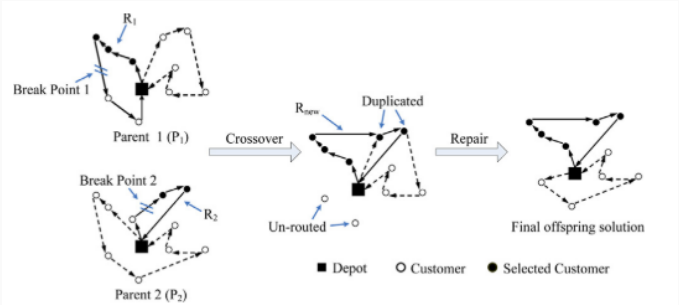
MOEA/D iii



(a) Chromosome Structure



(b) Mutation Method



(c) Crossover Method

Questions?

References i

F. M. Biermann, V. Naroditskiy, M. Polukarov, T. D. Nguyen, A. Rogers, and N. R. Jennings.
Task assignment with controlled and autonomous agents.

Mathematical Social Sciences, 71:116–121, 2014.

B. Colson, P. Marcotte, and G. Savard.
An overview of bilevel optimization.
Annals of Operations Research, 153(1):235–256, 2007.

References ii

Y. Qi, Z. Hou, H. Li, J. Huang, and X. Li.
A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows.
Computers and Operations Research, 62:61–77, 2015.

Q. Zhang and H. Li.
MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition.
IEEE Transactions on Evolutionary Computation, 11(6):712–731, 2007.