Sustainable Waste Collection Optimization

Tommaso Tarchi

October 28, 2024

University of Trieste

Original paper

Tirkolaee, E.B., Goli, A., Gütmen, S. et al. A novel model for sustainable waste collection arc routing problem: Pareto-based algorithms. Ann Oper Res 324, 189–214 (2023). https://doi.org/10.1007/s10479-021-04486-2

Problem description

General description

Sustainable **periodic capacitated arc routing problem (PCARP)** for municipal solid waste management.

General description

Sustainable **periodic capacitated arc routing problem (PCARP)** for municipal solid waste management.

Objectives:

1. Minimize total cost.
2. Minimize total pollution.
3. Maximize total job opportunities.
4. Minimize workload deviation (i.e. distribute work as evenly as possible).

Assumptions

1. Multiple planning periods.
2. Fleet of homogeneous vehicles, with usage cost, finite capacity and maximum service time.
3. Separate locations for *depot* and *disposal site*.
4. Multiple trips of same vehicle in one period allowed.

Assumptions

1. Multiple planning periods.
2. Fleet of homogeneous vehicles, with usage cost, finite capacity and maximum service time.
3. Separate locations for *depot* and *disposal site*.
4. Multiple trips of same vehicle in one period allowed.

Note 1: in each period, each vehicle starts its first trip from depot and ends it at disposal site. Following trips of the same vehicle will start and end at disposal site.

Note 2: single trips length is limited by both maximum service time and capacity; total length travelled in one period by one vehicle is limited by maximum service time.

Modeling of network

The network is represented as an undirected graph $G = (V, E)$, with $V = \{1...n\}$ set of nodes and E set of edges.

Node 1 is the depot, node n is the disposal site.

Not all edges have to be served: E_R^t is the set of *required edges* at planning period t ($E_R^t \subseteq E$, $\forall t \in T$).

Modeling of network

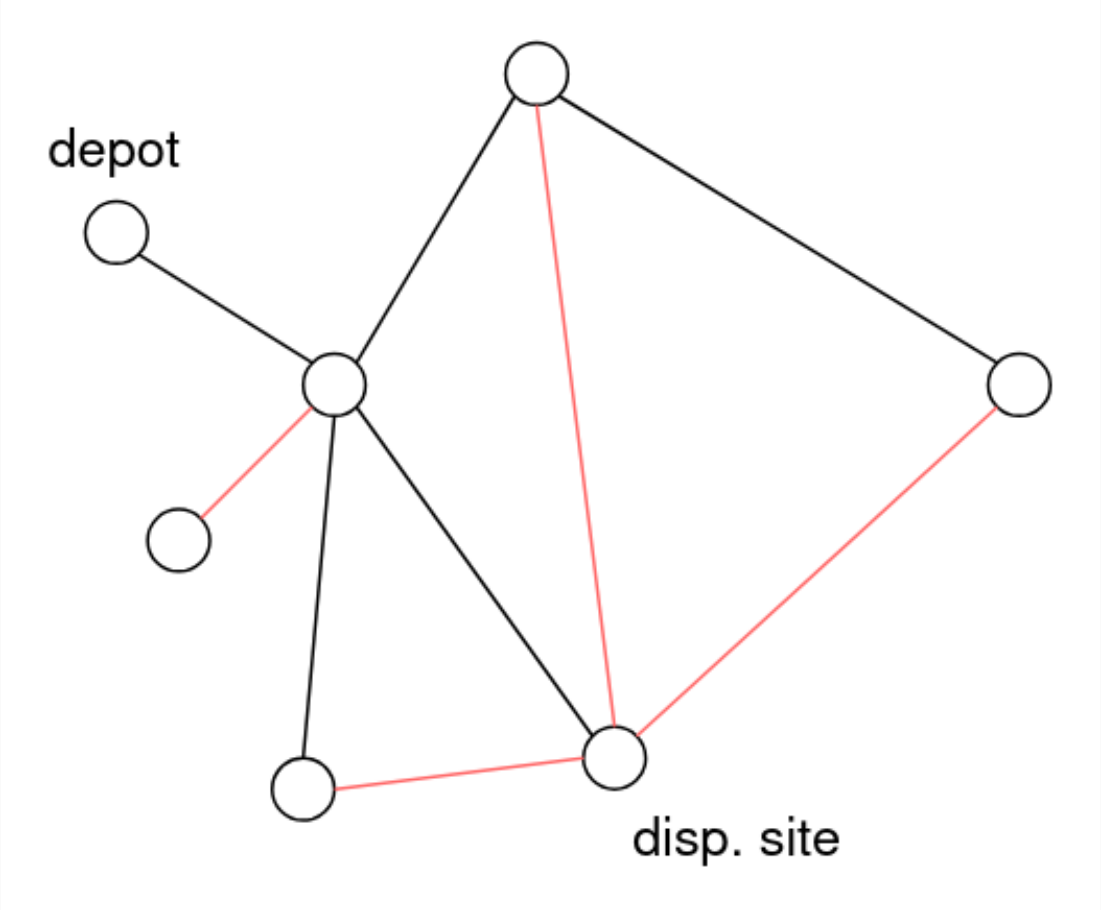


Figure 1: Sample network (red edges are the required ones).

Algorithms

1. **Epsilon-constraint method**: one of the objectives is chosen as main objective and the other ones are transformed into constraints. The resulting single-objective model can be solved exactly.

2. **MOSA-MOIWOA**: genetic algorithm, composed of:
   - 2.1 heuristic to produce initial solutions (similar to a greedy algorithm to find shortest total path);
   - 2.2 multi-objective simulated annealing (MOSA) to refine initial solutions;
   - 2.3 multi-objective invasive weed optimization algorithm (MOIWOA) to find approximated optimal solutions.

MILP formulation

Parameters and variables

Parameters

c_{ij}	Distance of edge (i, j)
W	Available capacity for each vehicle
d_{ijt}	Demand of edge (i, j) in tth period
T_{max}	Maximum available time for vehicles
M	A large number,
t_{ij}	Traversing time of edge (i, j)
cv_k	Usage cost of kth vehicle
θ	Conversion factor of distance to cost
G_{ij}	Amount of pollution emission released by traversing edge (i, j)
σ	Number of required workforce (drivers and crew) for each vehicle

Variables

x^p_{ijkt}	Number of traversing the edge $(i, j) \in E$ by kth vehicle in pth trip and tth period
y^p_{ijkt}	1 if edge $(i, j) \in E_R$ is served by kth vehicle in pth trip and tth period, otherwise 0
u_{kt}	1 if kth vehicle is employed in tth period, otherwise 0
LT^p_k	Total loading time of kth vehicle in pth trip and tth period
UT^p_k	Total unloading time of kth vehicle in pth trip and tth period
WT_{kt}	Total service time of kth vehicle in tth period;

$$WT_{kt} = \sum_{p \in P} LT^p_{kt} + \sum_{p \in P} UT^p_{kt} + \sum_{p \in P} \sum_{(i,j) \in E} t_{ij} x^p_{ijkt} \quad \forall k \in K, \forall t \in T,$$

Objective functions

$$\text{minimize } Z_1 = \theta \left(\sum_{(i,j)\in E} \sum_{p\in P} \sum_{t\in T} \sum_{k\in K} c_{ij} x_{ijkt}^{p} \right) + \sum_{t\in T} \sum_{k\in K} cv_k u_{kt}$$

$$\text{minimize } Z_2 = \sum_{(i,j)\in E} \sum_{p\in P} \sum_{t\in T} \sum_{k\in K} G_{ij} x_{ijkt}^{p}$$

$$\text{maximize } Z_3 = \sum_{t\in T} \sum_{k\in K} \sigma u_{kt}$$

$$\text{minimize } Z_4 = \sum_{t\in T} \sum_{k\in K} \frac{T_{max} - WT_{kt}}{T_{max}}$$

1. Minimize total cost.
2. Minimize total pollution.
3. Maximize employed workforce.
4. Minimize workload deviation.

Constraints - 1

$$\sum_{i \in V[E]} x_{ijkt}^{p} = \sum_{j \in V[E]} x_{jikt}^{p} \quad \forall i \in V[E]; (i,j) \in E, \forall k \in K, \forall p \in P, \forall t \in T, \tag{5}$$

$$\sum_{p \in P} \sum_{k \in K} (y_{ijkt}^{p} + y_{jikt}^{p}) = 1 \quad \forall (i,j) or (j,i) \in E_R, \forall t \in T, \tag{6}$$

$$\sum_{(i,j) \in E_R} d_{ijt} y_{ijkt}^{p} \leq W \quad \forall k \in K, \forall p \in P, \forall t \in T, \tag{7}$$

$$y_{ijkt}^{p} \leq x_{ijkt}^{p} \forall (i,j) \in E, \quad \forall k \in K, \forall p \in P, \forall t \in T, \tag{8}$$

$$\sum_{p \in P} \sum_{(i,j) \in E} x_{ijk}^{p} \leq M u_{kt} \quad \forall k \in K, \forall t \in T, \tag{9}$$

$$LT_{kt}^{p} = ul \sum_{(i,j) \in E_R} d_{ij} y_{ijkt}^{p} \quad \forall k \in K, \forall p \in P, \forall t \in T, \tag{10}$$

$$UT_{kt}^{p} = uu \sum_{(i,j) \in E_R} d_{ij} y_{ijk}^{p} \quad \forall k \in K, \forall p \in P, \forall t \in T, \tag{11}$$

$$\sum_{p \in P} LT_{kt}^{p} + \sum_{p \in P} UT_{kt}^{p} + \sum_{p \in P} \sum_{(i,j) \in E} t_{ij} x_{ijkt}^{p} \leq T_{max} \quad \forall k \in K, \forall t \in T, \tag{12}$$

$$\sum_{(j,h) \in S} x_{jhkt}^{p} \leq M \sum_{i \notin V[S], j \in V[S] \setminus \{1,n\}} x_{ijkt}^{p} \quad \forall S \subseteq E, \forall k \in K, \forall p \in P, \forall t \in T, \tag{13}$$

Constraints - 2

$$\sum_{j \in V[E]} x^1_{1jkt} \geq \sum_{j \in V[E]} x^2_{njkt} \quad \forall k \in K, \forall t \in T, \tag{14}$$

$$\sum_{j \in V[E]} x^p_{njkt} \geq \sum_{j \in V[E]} x^{p+1}_{njkt} \quad \forall p \in \{2, 3, \ldots, |P| - 1\}, \forall k \in K, \forall t \in T, \tag{15}$$

$$\sum_{\substack{(1,j) \in E \\ j \in V[E] \setminus \{1,n\}}} x^p_{1jkt} = u_{kt} \quad \forall k \in K, \forall p = 1, \forall t \in T, \tag{16}$$

$$\sum_{\substack{(j,n) \in E \\ j \in V[E] \setminus \{1,n\}}} x^p_{jnkt} = u_{kt} \quad \forall k \in K, \forall p = 1, \forall t \in T, \tag{17}$$

$$\sum_{\substack{(n,j) \in E \\ j \in V[E] \setminus \{1,n\}}} x^p_{njkt} \leq u_{kt} \quad \forall k \in K, \forall p \in P \setminus \{1\}, \forall t \in T, \tag{18}$$

$$\sum_{\substack{(j,n) \in E \\ j \in V[E] \setminus \{1,n\}}} x^p_{jnkt} \leq u_{kt} \quad \forall k \in K, \forall p \in P \setminus \{1\}, \forall t \in T, \tag{19}$$

$$x^p_{ijkt} \in Z^+, y^p_{ijkt} \in \{0, 1\}, u_{kt} \in \{0, 1\} \quad \forall (i, j) \in E, \forall k \in K, \forall p \in P, \forall t \in T, \tag{20}$$

$$LT^p_{kt}, UT^p_{kt} \geq 0 \quad \forall (i, j) \in E, \forall k \in K, \forall p \in P, \forall t \in T. \tag{21}$$

Corrections - constraint 5 (flow balance)

Adjusted indexing and flow balance in first trip (in the first trip there must be flow unbalance in starting and ending nodes):

$$\sum_{j \in V[E]:(i,j) \in E} x^p_{ijkt} + \delta_{1p,1i} = \sum_{j \in V[E]:(j,i) \in E} x^p_{jikt} + \delta_{1p,ni},$$

$$\forall i \in \hat{V}_p, \forall k \in K, \forall p \in P, \forall t \in T,$$

where $\hat{V}_p = V[E] \setminus \{1, n\}$ if $p = 1$ and $\hat{V}_p = V[E]$ otherwise.

Corrections - constraint 13 (subtours elimination)

Use only connected subsets S (also improves efficiency) and add D to make sure that traverses from origin of trip are counted as "incoming" traverses:

$$\sum_{(i,j)\in S} x^p_{ijkt} \leq M * \left(\sum_{i\notin V[S], j\in V[E]\setminus\{1,n\}} x^p_{ijkt} + D \right),$$

$$\forall S \subseteq E : Sconnected, \forall k \in K, \forall p \in P, \forall t \in T,$$

where D is equal to 1 if either $p = 1$ and an edge starting at the depot is in the subset S or $p > 1$ and an edge starting at the disposal site is in S, and it's 0 otherwise.

Epsilon-constraint method

Epsilon-constraint

ϵ-**constraint** for a certain objective f_i:

- $f_i \leq \epsilon_i$, if f_i should be minimized
- $f_i \geq \epsilon_i$, if f_i should be maximized

Algorithm

1. Choose one of the objective functions as the main objective (in our case Z_1, i.e. the total cost).
2. For each objective, solve the related single-objective problem.
3. For each obtained solution, compute the value of all objective functions. In this way, a list of values is obtained for each objective function.
4. For each list of values, compute the values of ϵ dividing the interval between best and second best homogeneously.
5. Solve single-objective problem with main objective as objective function and ϵ-constraints on other objectives, for each combination of values of ϵ's (in our case, for each combination (ϵ_2, ϵ_3, ϵ_4)).
6. Select Pareto solutions.

Implementation details

- As solver, we used *Gurobi* for python.
- To select Pareto solutions at the end, we sorted all the obtained solutions using *DEAP*'s implementation of non-dominated sorting, and selected only those belonging to the first Pareto front.
- For implementation of constraint 13 (subtours elimination), we used *networkx* library.
- Combinations (ϵ_2, ϵ_3, ϵ_4) resulting in no feasible solution, were ignored.

MOSA-MOIWOA

Solution representation

Similarly to the original paper, we represented a possible solution to the problem as **a list of arrays, one for each period**.

Each array is made of two parts of length equal to the number of required edges:

1. the **first part** represents the order in which the required edges are visited by the fleet of vehicles;
2. the **second part** represents the vehicles that visited each required edge.

Solution representation

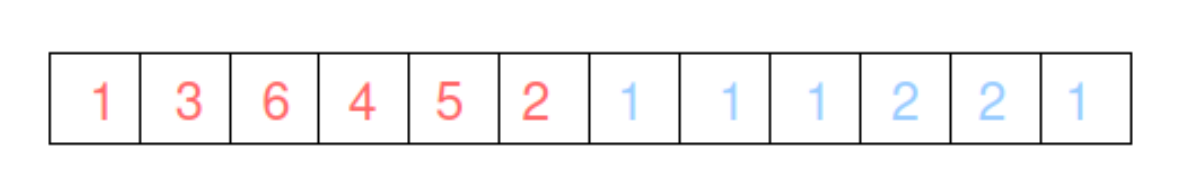


Figure 2: Example of representation of a period solution with 6 required edges and 2 vehicles; red indices refer to required edges and blue ones to vehicles employed.

Solution representation

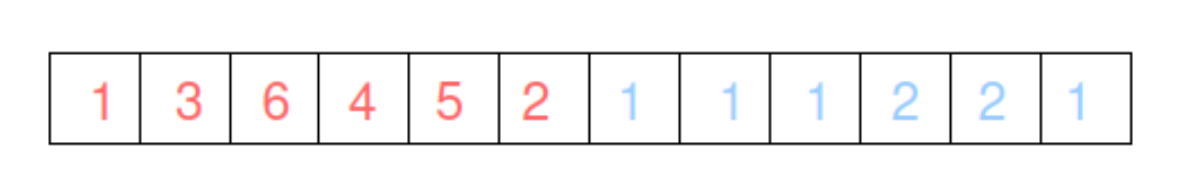


Figure 2: Example of representation of a period solution with 6 required edges and 2 vehicles; red indices refer to required edges and blue ones to vehicles employed.

Note: this way of representing solutions is not ambiguous, assumed that:

1. to move from a required edge to the following one, a vehicle always chooses the shortest path;
2. each vehicle serves the most edges it can according to its capacity.

Heuristic for initial solutions

A **greedy algorithm** in which, at each iteration, a random vehicles is chosen and trips are built to serve the maximum number of required edges possible, given the vehicle maximum capacity constraint.

Multiple consecutive trips of the same vehicle are allowed, given that the maximum service time constraint is respected.

Heuristic for initial solutions

A **greedy algorithm** in which, at each iteration, a random vehicles is chosen and trips are built to serve the maximum number of required edges possible, given the vehicle maximum capacity constraint.

Multiple consecutive trips of the same vehicle are allowed, given that the maximum service time constraint is respected.

To compute the shortest paths between edges we used *networkx* library.

MOSA - original algorithm

Given an initial solution s_0 and an initial *temperature* T_0:

```
s=s0
T=T0
Repeat
    Generate a neighbor s'=N(s)
    If C(s') dominates C(s)
                move to s'
    else if C(s) dominates C(s')
                move to s' with transition probability
            Pt(C(s), C(s'), T)
    else if C(s) and C(s') do not dominate each other
                move to s'
    end if
    T=annealing(T)
End repeat (until the termination are satisfied)
```

MOSA - implementation details

- To produce a neighbor we either shuffle the first part or change one vehicle in the second.
- To compute the shortest paths between edges we used *networkx* library.
- As stopping criteria, we used two:
  1. maximum number of iterations;
  2. maximum number of non-improving iterations (i.e. stop the algorithm if the solutions has not been modified for a given number of iterations).
- As temperature cooling strategy, we used the geometric one:

$$T_k = \alpha T_{k-1},$$

  starting from T_0, with α parameter chosen by the user.

MOSA - implementation details

- For acceptance of modified solutions we used the following probability formulation:

$$P_{accept} = \exp\left(-\frac{\|\hat{\Delta}f\|}{KT}\right),$$

  where $\hat{\Delta}f$ is the average difference between old and new solutions' objectives and K is a parameter chosen by the user.
- Each time we generate a neighbor solution, we check for its feasibility. If it is not, we generate a new one, and so on.

MOIWOA - original algorithm

Given an initial population of solutions (*seeds*), repeat the following until the given maximum number of iterations is reached:

1. **Reproduction**: each seed of the current population generates S children seeds by mutation (see next slide), where S is computed as:

$$S = S_{min} + (S_{max} - S_{min}) \frac{f - f_{worst}}{f_{best} - f_{worst}}.$$

   The generated seeds are added to the current population
2. **Competition**: the current population is sorted according to the following non-dominate sorting technique: a. sort by non-dominance (i.e. sort Pareto fronts), b. within each Pareto front, sort by crowding distance.
   Then, only the first N_{max} seeds are kept, where N_{max} is a parameter.

MOIWOA - mutations

Reproduction is carried out by applying **one** of the following mutations (inspired by, but not exactly the same as in the original paper), chosen randomly:

1. Swap the order of two edges in a trip.
2. Shuffle the order of edges in a trip.
3. Reverse the order of edges in a trip.
4. Combine two trips: select two random trips and divide them in two parts, then combine the first part of the first trip with the second of the second, and vice-versa.

The generated seed is checked for feasibility, and discarded if it fails the check.

MOIWOA - implementation details

- We compute the fitness of the current i-th solution in a "pool" of current solutions as:

$$fitness\left(sol_i\right) = \frac{1}{4}\sum_{j=1}^{4}\frac{f_j\left(sol_i\right)}{\hat{f}_j},$$

  where $f_j\left(sol_i\right)$ is the value of the j-th objective for the i-th current solution and $\hat{f}_j$ is the average of the j-th objective value over all current solutions.
  Rescaling is done in order to have a contribute of the same order of magnitude from each objective.
- *DEAP* library is used for non-dominate sorting and crowding distance computation.

MOSA-MOIWOA - complete algorithm

1. Generate initial solutions using the first heuristic.
2. Refine initial solutions applying MOSA to each one of them individually.
3. Use the refined solutions as input seed population for MOIWOA.
4. Output Pareto solutions.

Evaluation

Dataset

To compare efficiency and effectiveness of the two algorithms we used synthetic datasets of variable size.:

problem ID	N_{nodes}	N_{edges}	$N_{required\_edges}$	$N_{periods}$	$N_{vehicles}$
0	4	6	3	1	2
1	5	8	4	1	2
2	6	9	5	1	2
3	7	10	6	1	3
4	8	11	7	2	3
5	9	12	8	2	3
6	10	13	9	2	4
7	11	14	10	2	4

Dataset - good parameters

- Estimate of P as $0.3 * N_{required\_edges}$.
- Definition of T_{max} as:

$$T_{max} = \frac{1.2 * P * N_{edges} * t_{max}}{N_{vehicles}}$$

- uu and ul defined to satisfy:

$$uu*N_{required\_edges}*d_{max}+ul*N_{required\_edges}*d_{max}+P*N_{edges}*t_{max} < N_{vehicles}*T$$

- W defined to satisfy:

$$N_{required\_edges} * d_{max} < N_{vehicles} * W.$$

- θ defined to satisfy:

$$\hat{cv} \sim \theta * N_{edges} * \hat{c}.$$

Evaluation metrics

- Execution time.
- Number of Pareto solutions (NOS).
- Mean of ideal distance (MID):

$$MID = \frac{1}{NOS} \sum_{s=1}^{NOS} \sqrt{\sum_{m=1}^{4} f_{s,m}^2}.$$

- Solution distancing (D):

$$D = \sqrt{\sum_{m=1}^{4} \left(max_{s \in \{1,n\}} f_{s,m} - min_{s \in \{1,n\}} f_{s,m} \right)^2}.$$

Where $f_{s,m}$ represents the m-th objective value for the s-th solution.

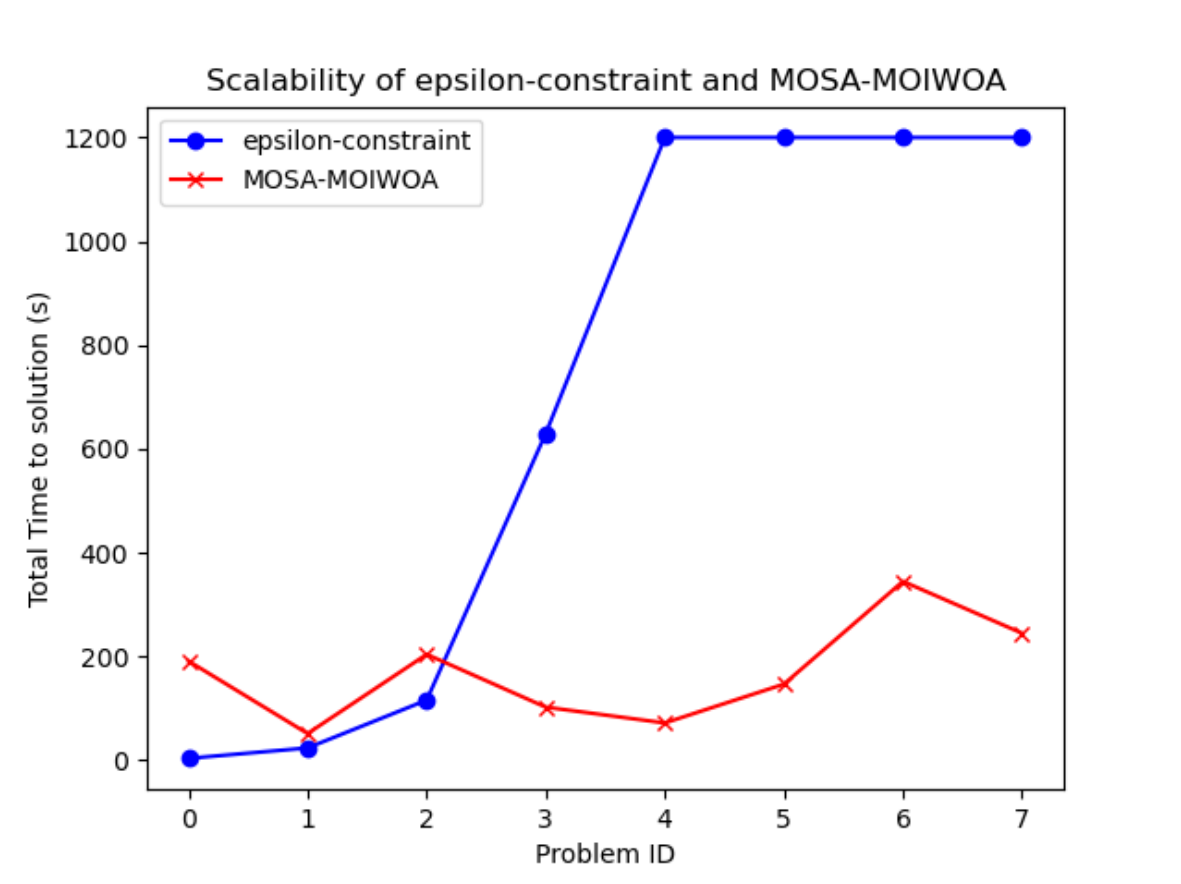
Results

Solvers' parameters

algorithm	parameter	value
ϵ-constraint	n_ϵ	10
heuristic	N_0	10
MOSA	T_0	800
MOSA	α	0.9
MOSA	K	70
MOSA	max_iter	200
MOSA	max_non_improv_iter	10
MOIWOA	S_{min}	9
MOIWOA	S_{max}	200
MOIWOA	N_{max}	100
MOIWOA	max_iter	300

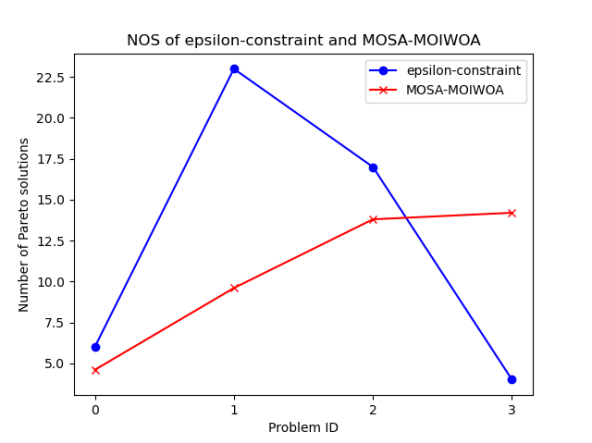
(Values of parameters for MOSA and MOIWOA were taken from the original paper).

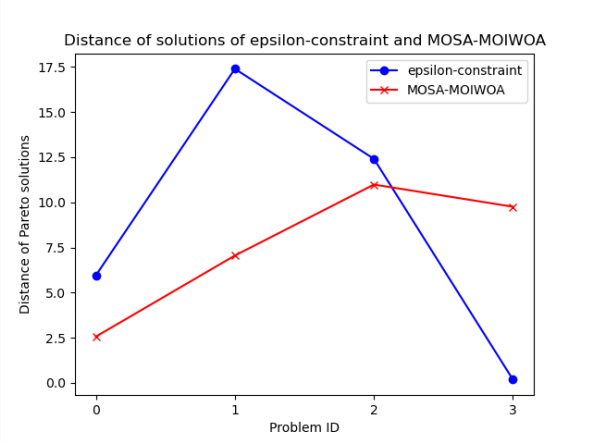
Time scalability

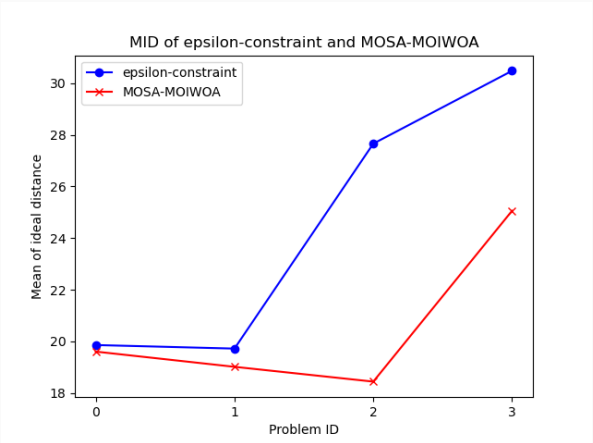


Note: MOSA-MOIWOA results were averaged over 8 runs.

Quality of solutions

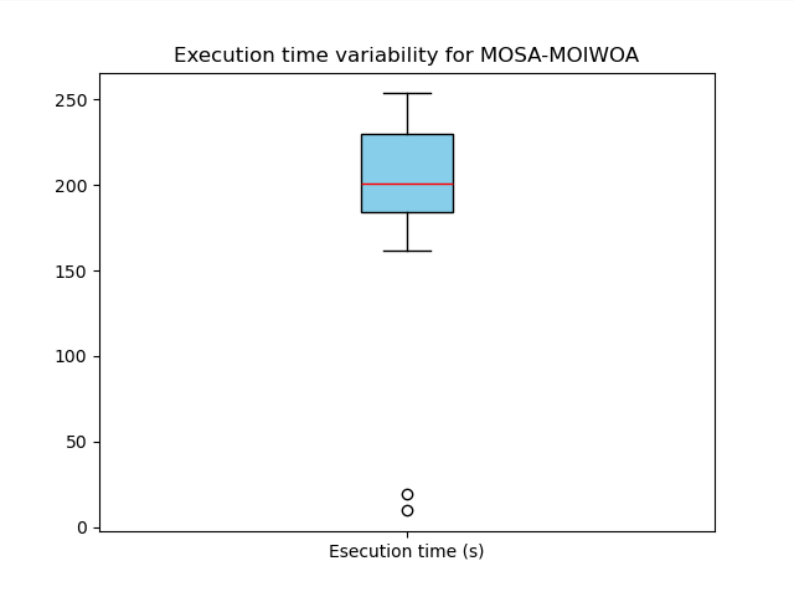


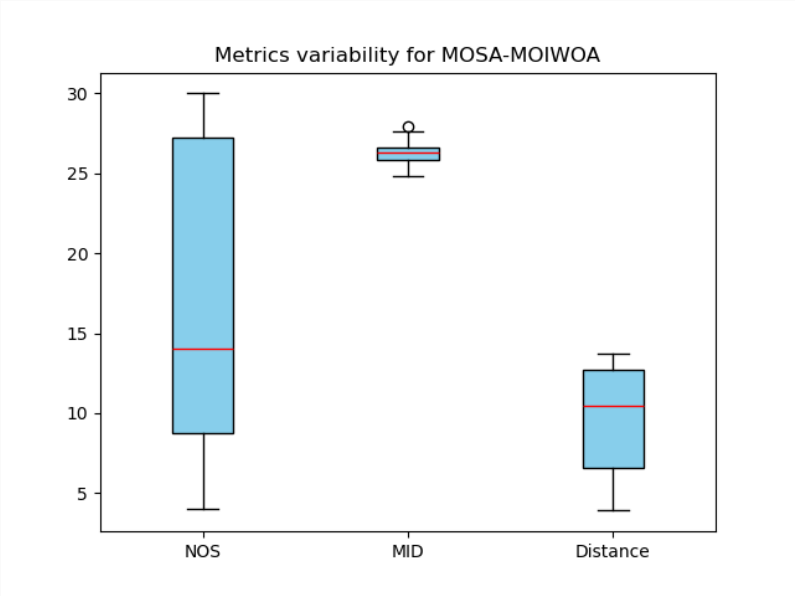




Note: MOSA-MOIWOA results were averaged over 8 runs.

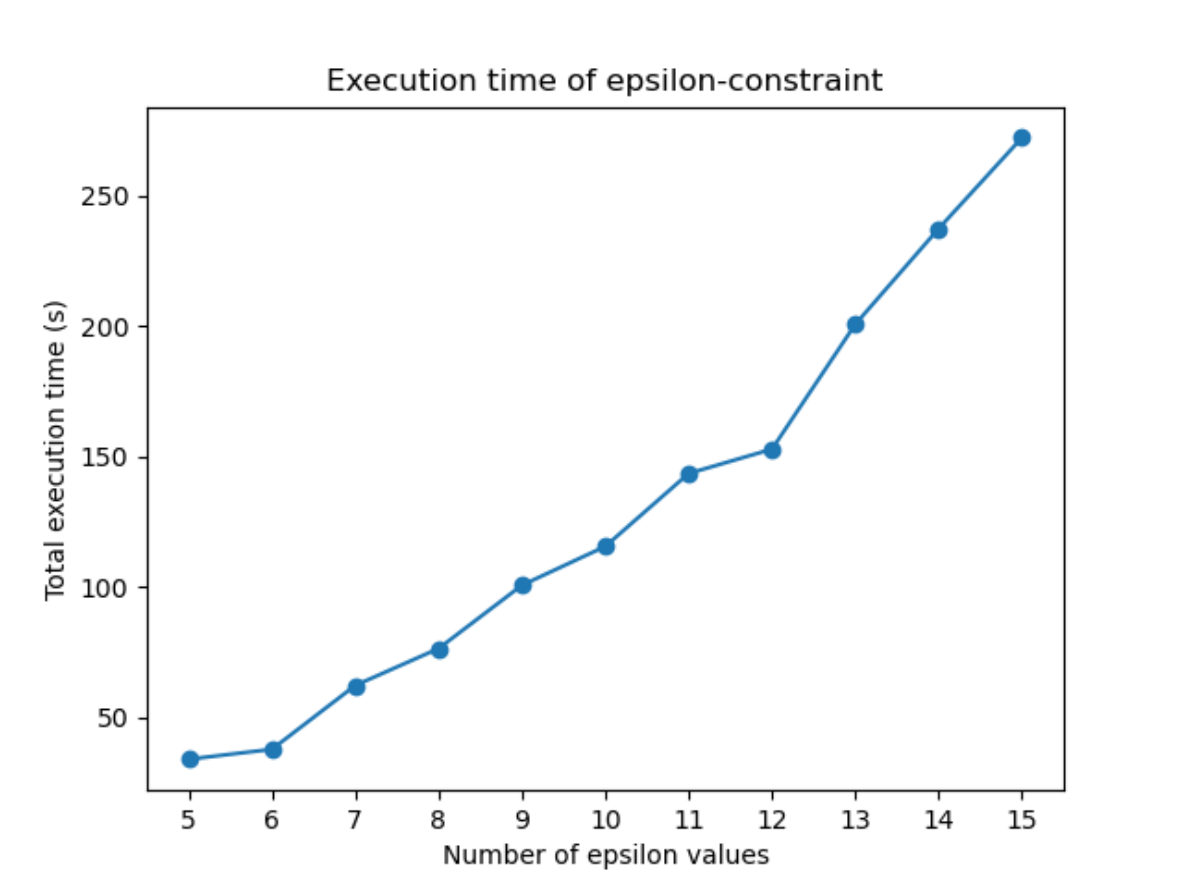
Variability of MOSA-MOIWOA





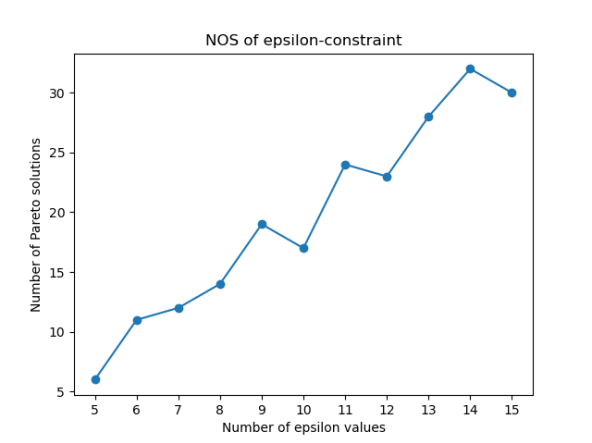
Note: results were obtained on problem 2, and averaged over 20 runs.

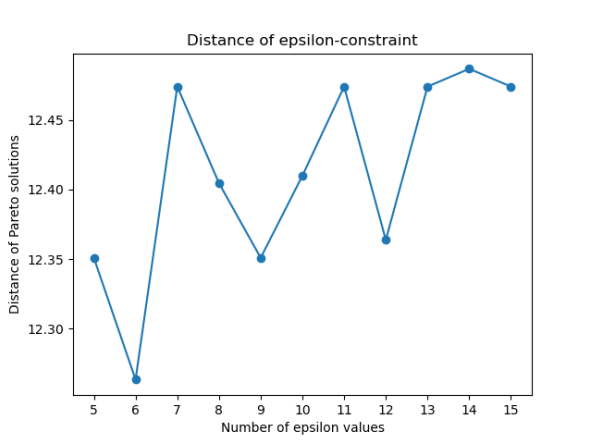
Epsilon scalability - time

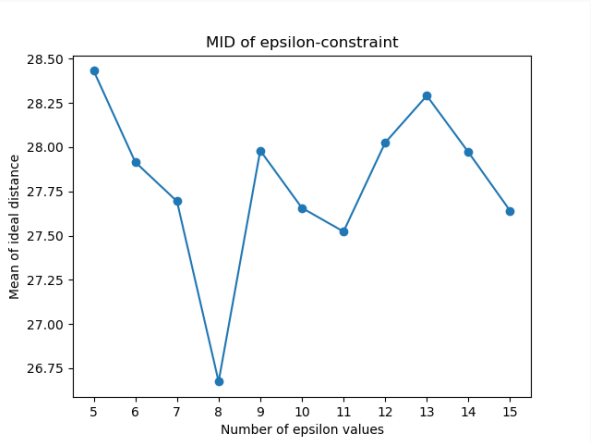


Note: results were obtained on problem 2. As a comparison, MOSA-MOIWOA needs an average of 80 seconds to solve the same problem.

Epsilon scalability - quality of solutions







Note: results were obtained on problem 2. As a comparison, MOSA-MOIWOA achieves on average: NOS=16, distance=11, MID=28.

Conclusions

- MOSA-MOIWOA produces in general better results, however, the outcome is strongly initialization-dependent.
- MOSA-MOIWOA is harder to implement and tune.
- On larger problems, MOSA-MOIWOA is by far more efficient, even if its execution time is strongly initialization-dependent.
- Increasing the number of ϵ values in ϵ-constraint method is not always enough to improve the quality of solutions.

References

- Tirkolaee, E.B., Goli, A., Gütmen, S. et al. A novel model for sustainable waste collection arc routing problem: Pareto-based algorithms. Ann Oper Res 324, 189–214 (2023). https://doi.org/10.1007/s10479-021-04486-2
- Amine, Khalil, Multiobjective Simulated Annealing: Principles and Algorithm Variants, Advances in Operations Research, 2019, 8134674, 13 pages, 2019. https://doi.org/10.1155/2019/8134674

Thank you!