UNIVERSITÀ
DEGLI STUDI DI TRIESTE
Dipartimento di Ingegneria e Architettura

Solving the maximum edge disjoint path problem using a modified Lagrangian particle swarm optimization hybrid

by Jake Weiner, Andreas T. Ernst, Xiaodong Li, Yuan Sun, Kalyanmoy Deb

Michele Scomina

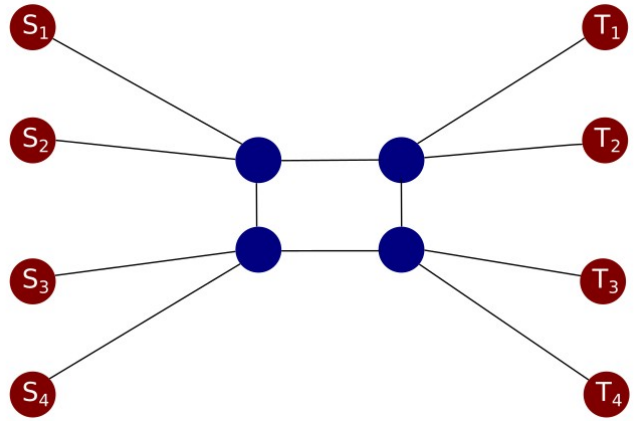
Maximum Edge Disjoint Path (MEDP)

Maximum Edge Disjoint Path:

- Set of source-terminal node pairs has to be connected on a graph.

- Edges can only be used by at most one source-destination pair.

Objective: Connect the most amount of node pairs.



Applications: Optical and energy-efficient networks
(Raghavan & Upfal – 1994)
(Gerez – 1998)

Integer Problem formulation

$$\max \sum_{k \in K} \sum_{j:(s_k, j) \in E} x_{s_k jk}$$

Vars:
x = edge use for each commodity pair

Parameters:
k = source-terminal pair number
i,j = node number
s_k = source node of pair k

Assumptions:
- All edges have a cost of 0, therefore all paths are equally good as long as they satisfy the constraints

Node flow: $$\sum_{j:(i,j) \in E} x_{ijk} - \sum_{j:(j,i) \in E} x_{jik} = 0 \qquad \forall\ k \in K, i \in V \setminus \{s_k, t_k\}$$

Source flow: $$\sum_{j:(s_k, j) \in E} x_{s_k jk} \leq 1 \qquad \forall\ k \in K$$

Capacity constraint: $$\sum_{k \in K} (x_{ijk} + x_{jik}) \leq 1 \qquad \forall\ (i, j) \in E : i < j$$

$$x_{ijk} = \{0, 1\} \qquad (i, j) \in E, k \in K$$

Multi-start Simple Greedy Algorithm (MSGA)

MSGA:
- One of the first approaches *(Kleinberg – 1996, Blesa & Blum – 2007)*
- Tries to connect the commodity pairs using Dijkstra in random orders, then removes the edges used from the graph if a path was found.

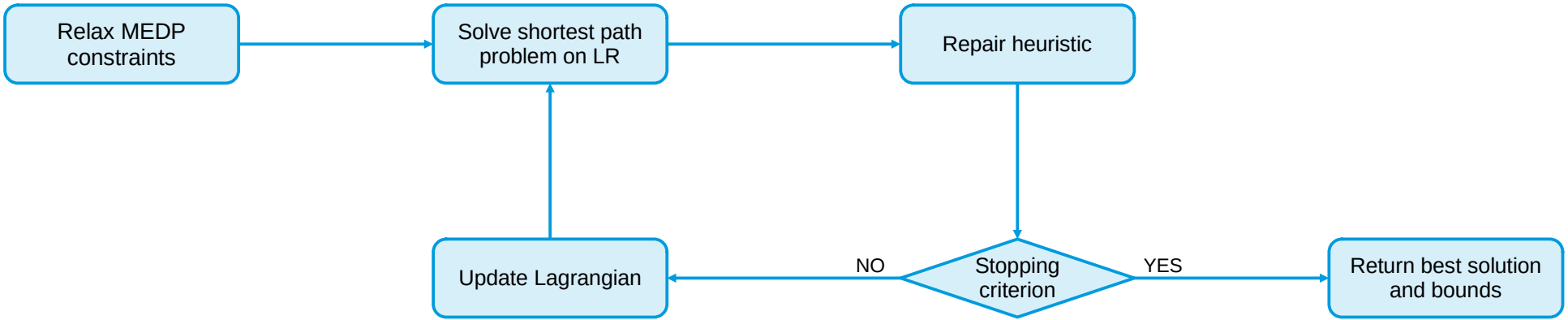
Advantages:
- Very easy to implement.
- Very fast.
- Works very well in small and well-connected graphs.

Disadvantages:
- Poor solutions on large or poorly connected graphs.
- Does not provide an upper bound solution.
- Does not explore the entire solution space.

```
S_best = 0
while n < max_iter:
        reset Graph
        shuffle pairs
        S_current = 0
        for pair in pairs:
                path = dijkstra(Graph, pair)
                if path exists:
                        S_current += 1
                        remove(Graph, path)
        if S_current > S_best:
                S_best = S_current
```

LaPSO

LaPSO *(2021)*:
- Short for Lagrangian Relaxation and Particle Swarm Optimisation.
- Consists in relaxing the capacity constraints and reformulating the problem as a minimization problem.
- The MEDP turns in a series of shortest path problems with variable weights.
- If solutions are infeasible, apply a repair heuristic.
- Given the solutions and infractions, update the Lagrangian through PSO framework.
- If solution hasn't improved, max time or iterations have been reached, return the best solution found.



Lagrangian Relaxation

Relaxing the capacity constraint:

$$\min_{\lambda \geq 0} LR(\lambda) = \max_{x \in \mathcal{F}} \sum_{k \in K} \sum_{(s_k, j) \in E} x_{s_k jk} + \sum_{(i,j) \in \bar{E}} \lambda_{ij} \left(1 - \sum_{k \in K} (x_{ijk} + x_{jik}) \right)$$

$$= \sum_{(i,j) \in \bar{E}} \lambda_{ij} + \max_{x \in \mathcal{F}} \left\{ \sum_{(s_k, j) \in E} x_{s_k jk} (1 - \lambda_{s_k j}) - \sum_{k \in K} \sum_{(i,j) \in \bar{E}, i \neq s_k} \lambda_{ij} x_{ijk} \right\}$$

$$= \sum_{(i,j) \in \bar{E}} \lambda_{ij} - \sum_{k \in K} \left\{ \min_{x_k \in \mathcal{F}_k} \sum_{(s_k, j) \in E} (\lambda_{s_k j} - 1) x_{s_k jk} + \sum_{(i,j) \in \bar{E}, i \neq s_k} \lambda_{ij} x_{ijk} \right\}$$

$$\min_{\lambda \geq 0} LR(\lambda) = \sum_{(i,j) \in E} \lambda_{ij} + |K| - \sum_{k \in K} \min \left\{ 1, \min_{s_k - t_k \text{path} P_k} \sum_{(i,j) \in P_k} \lambda_{ij} \right)$$

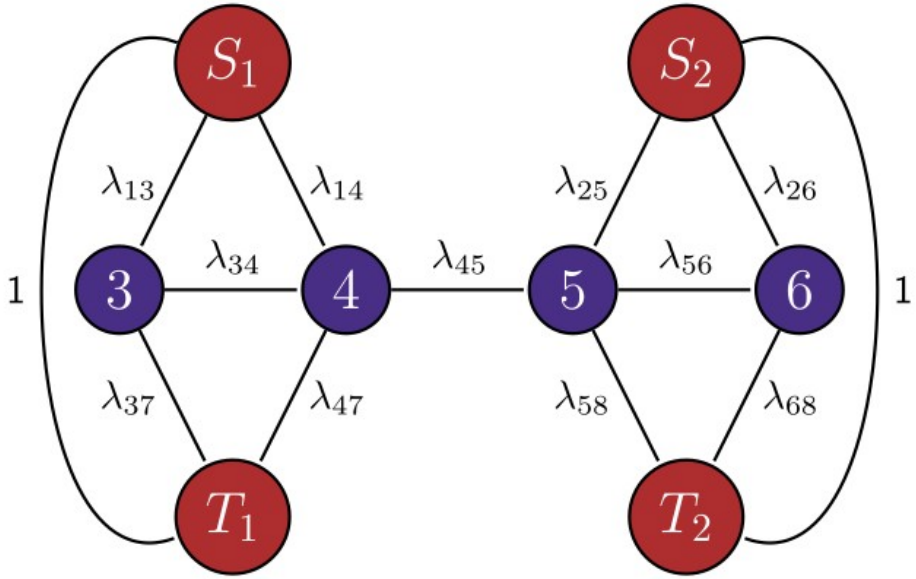
Parameters:

$\bar{E} = \{(i, j) \in E : i < j\}$

$\mathcal{F}$ set of feasible solutions to all constraints except capacity

Graph representation

Graph representation of the relaxed Lagrangian minimization problem:

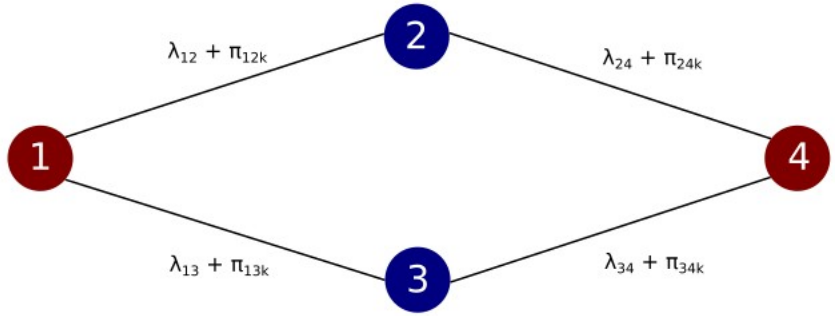


Perturbations

New problem:
If the edge weights are all the same for all commodities, then they will most likely all be connected through the same lowest cost edges, leading to poor diversity and high violation count.

Solution: Perturbations

By adding small perturbations to the graph's edges, commodity pairs will diversify their paths, leading to overall better results. These perturbations are learned through the PSO framework to penalize edges already used by other commodity pairs.



Repair heuristics

Since the MEDP problem has been relaxed, generated solutions may (and very often are) infeasible.

Solution: Repair heuristic

A repair heuristic is applied to the relaxed solution in order to make it feasible.
It takes the infeasible paths and tries to reroute them onto unused edges using Dijkstra.

There are three possible repair heuristics:
- **Rand**: Randomly picks an infeasible path and tries to reroute it.
- **LVA** (Largest Violation Arbitrary): Picks the path with the largest amount of violations and tries to reroute it.
- **LVP** (Largest Violation Perturbation): Picks the path with the largest amount of violations and tries to reroute it using the perturbation values as edge weights.

Particle Swarm Optimization

$$\boldsymbol{v}_i \leftarrow \omega \boldsymbol{v}_i + r^L \alpha_i \frac{UB - LB_i}{||\boldsymbol{g}_i||^2} \boldsymbol{g}_i + r^G \phi (\boldsymbol{\lambda}_i^G - \boldsymbol{\lambda}_i)$$

$$\boldsymbol{w}_i \leftarrow \omega \boldsymbol{w}_i + r^G \phi (\boldsymbol{\pi}_i^G - \boldsymbol{\pi}_i) + \delta r^G \phi (1 - 2\boldsymbol{x}_i^G)$$

$$\boldsymbol{\lambda}_i \leftarrow \boldsymbol{\lambda}_i + \boldsymbol{v}_i$$

$$\boldsymbol{\pi}_i \leftarrow \theta \boldsymbol{\pi}_i + \boldsymbol{w}_i$$

Parameters:

v_i, w_i = Particle speeds
ω = Velocity factor
r^L, r^G = random uniform variables – [0,1]
g_i = subgradient vector
UB, LB_i = Upper and lower bounds

α_i = Subgradient factor
ϕ = Global factor
δ = Perturbation factor
θ = Perturbation decay

Pseudocode

Algorithm 1 MEDP LaPSO Algorithm.

Require: Number of particles n
Require: CPU Runtime Limit T
Require: Velocity parameters $\omega, \phi, \delta > 0$
Require: α^0, β the initial value and update factor of α
Define global best $\lambda^G = 0$, $p^G = 0$
Set initial perturbation and Lagrangian Multipliers for $p_i(\lambda_i, \pi_i) \in n : \lambda_i \in [0, 0.1]$, $\pi_i = 0$
Set initial velocities $v_i = 0$, $w_i = 0$
Set maxIter and LBC $maxIter = 30000$, $LBC = 10$
for $iter = 0 \ldots maxIter$ **and** while $LB \leq UB - \epsilon$ **do** and while $CPU() \leq T$
  for $p_i(\lambda_i, \pi_i) \in n$ **do**
    if $iter \mod LBC = 0$ **then** // Lower Bound check
      Solve subproblem $L(\lambda_i, 0)$ // using Dijkstra
    else
      Solve subproblem $L(\lambda_i, \pi_i)$ // using Dijkstra
    end if
    Let x' be the optimal solution to theminimisation subproblem
    $g \leftarrow b - Ax'$ // update subgradient g
    if $L(\lambda_i, \pi_i) > LB$ **then**
      $LB \leftarrow L(\lambda_i, \pi_i)$
    else if LB_i not improved for 10 iterations of particle p_i **then**
      $\alpha_i \leftarrow \beta \alpha_i$ // reduce step size
    end if
    if x' is not feasible **then**
      $x' \leftarrow$ RepairHeuristic(x') // repair solution
    end if
    if x' is feasible and $(\bar{c}x' < UB)$ **then**
      $UB \leftarrow \bar{c}x'$ // update best Upper Bound
      $p^G \leftarrow p_i$ // update best particle
    end if
    Randomly generate numbers $r^G, r^L \in [0, 1]$
    Update particle velocities v_i, w_i using Eq. (13) and (14)
    Update the particle positions using Eq. (15) and (16)
  end for
end for

Experimental setup

Parameter choices (PSO):

Parameter	Tuning Domain	Value Chosen
Velocity Factor - ω	{0.01,0.05,0.1,0.5}	0.1
Global Factor - ϕ	{0.01,0.05,0.1,0.2,0.3}	0.05
Subgradient Factor - α_i	{1.5,2,3,3.5,5}	2
Perturbation Factor - δ	{0.01,0.05,0.1,0.5}	0.5
Swarm Size - p	{1,8,16,32,64,96}	8

Maximum iterations: 1000
Maximum time: 20 minutes

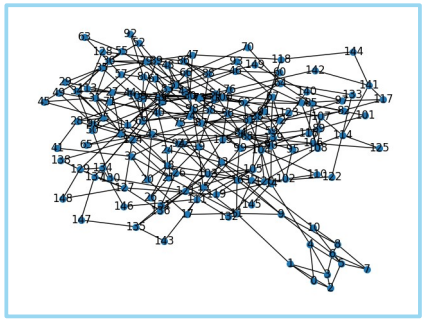
3 runs per graph combination
66% nodes used for pairs

Computational:

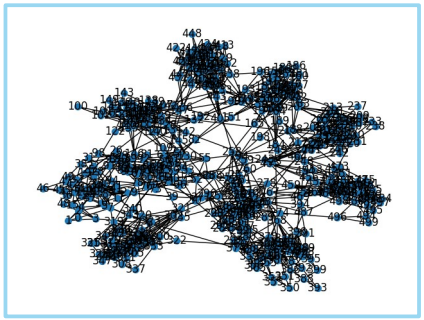
- Python codebase, with RustworkX for fast graph operations
- Ryzen 7 7800X3D 8-cores – 32 GBs of RAM
- MIP solver: Gurobi

Graph choices:

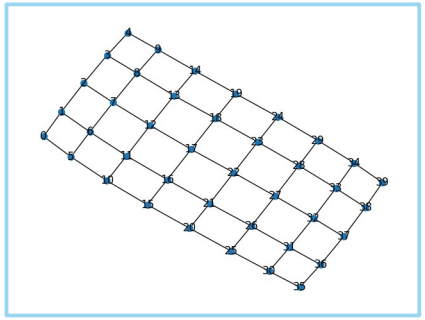
Ordinary$_N$:



Collated$_{C,N}$:



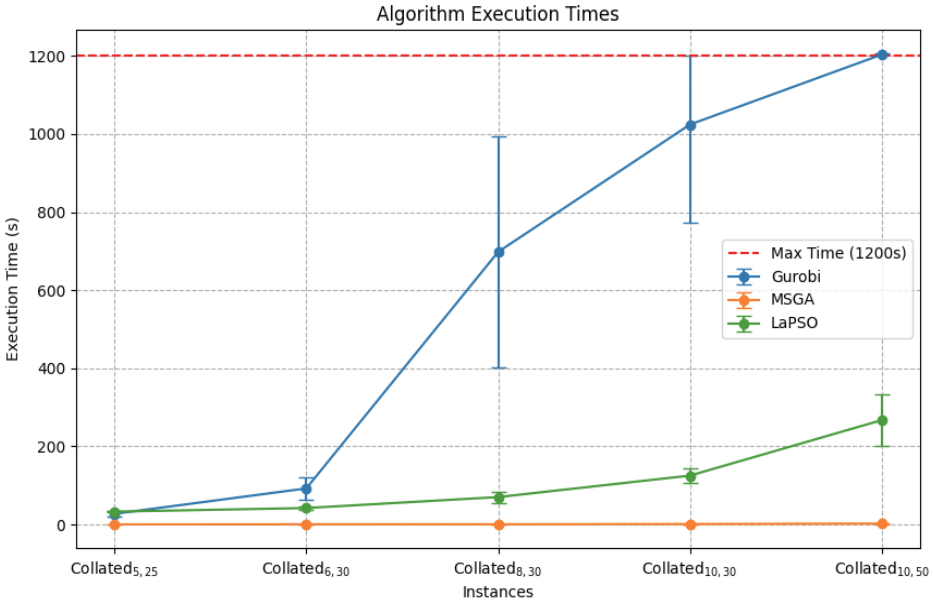
Grid$_{W,h}$:

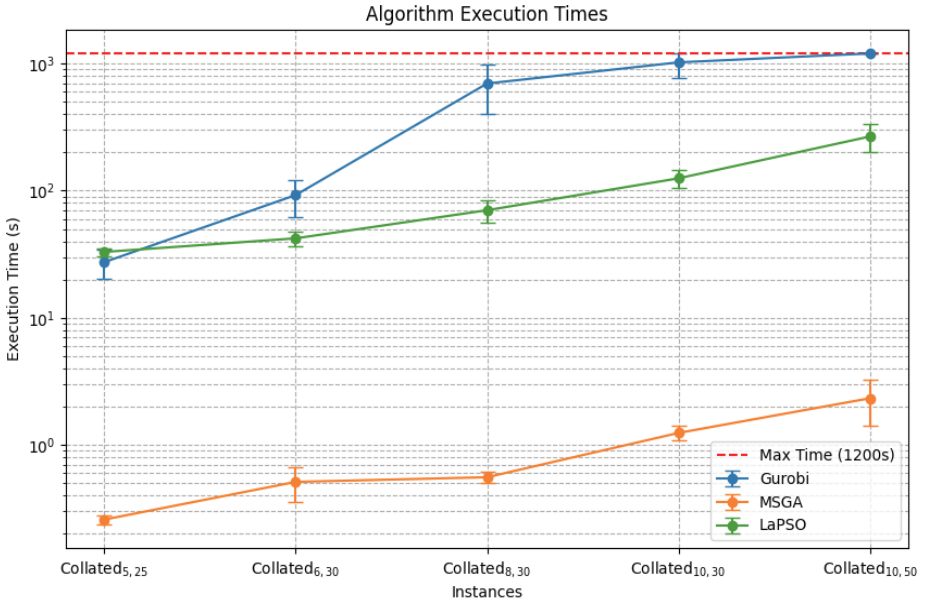


Results

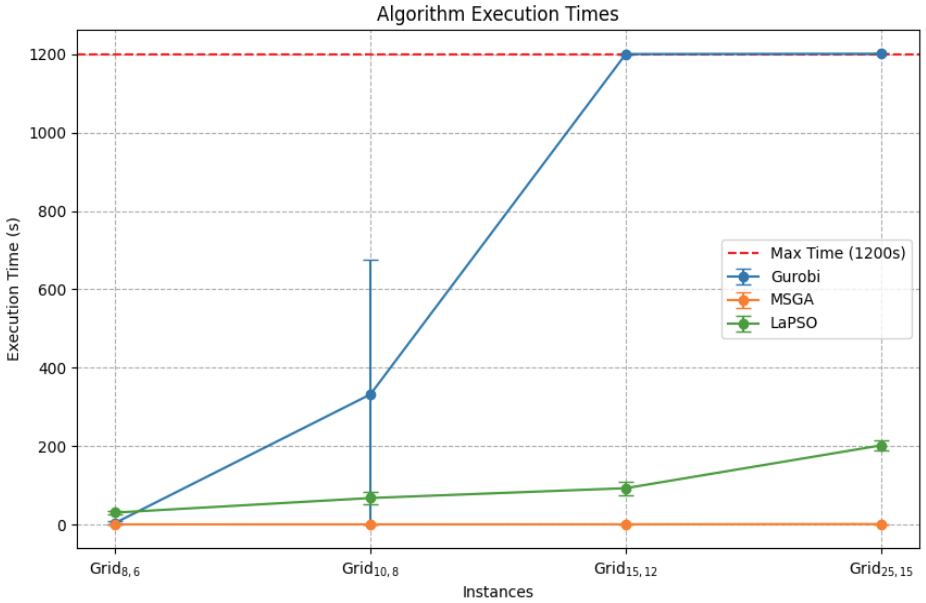
CASE:	Gurobi - Avg	MSGA - Avg	LaPSO - Avg	Gurobi - Gap	MSGA - Gap	LaPSO - Gap	Gurobi - Time	MSGA - Time	LaPSO - Time
Collated$_{5,25}$	35.6666	35.6666	35.6666	0.0000	0.2866	0.1160	27.3711	0.2577	32.9486
Collated$_{6,30}$	49.6666	49.3333	49.6666	0.0000	0.2850	0.1240	92.3655	0.5105	42.1783
Collated$_{8,30}$	55.0000	55.0000	55.0000	0.0000	0.2361	0.1628	697.9784	0.5555	70.2271
Collated$_{10,30}$	75.3333	72.6666	75.3333	0.0492	0.1925	0.0848	1024.0420	1.2402	125.1619
Collated$_{10,50}$	124.3333	117.6666	124.3333	0.1907	0.4260	**0.1077**	1203.2094	2.3249	267.2233
Grid$_{8,6}$	11.3333	11.3333	11.3333	0.0000	0.2444	0.1288	3.7978	0.0748	30.0885
Grid$_{10,8}$	17.6666	16.0000	17.6666	0.0000	0.3846	0.1180	331.8981	0.1690	67.3477
Grid$_{15,12}$	32.3333	26.0000	28.6666	0.1101	0.5593	0.2252	1200.2497	0.3349	92.4551
Grid$_{25,15}$	54.0000	39.3333	47.0000	0.3799	0.6802	**0.2703**	1201.1051	0.8911	201.6951
Regular$_{50}$	16.0000	16.0000	15.6666	0.0000	0.0000	0.0208	0.3434	0.0870	4.0694
Regular$_{100}$	31.3333	28.3333	28.0000	0.0000	0.1414	0.1515	255.6080	0.2212	31.5226
Regular$_{150}$	43.0000	38.0000	36.6666	0.0720	0.2245	0.2517	1200.2062	0.3314	23.6007
Regular$_{250}$	65.6666	57.6666	55.0000	0.1244	0.2967	0.3293	1200.5760	0.9000	70.6878

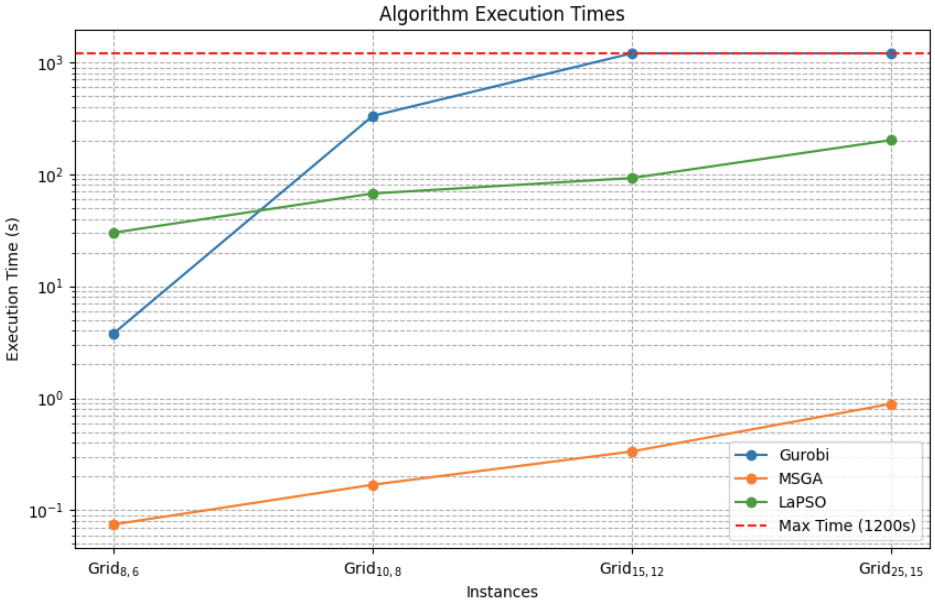
Graph plots (Collated)





Graph plots (Grid)





Conclusions

Thank you for your attention.