CS606 Project
Grid Operation-based Outage Maintenance Planning

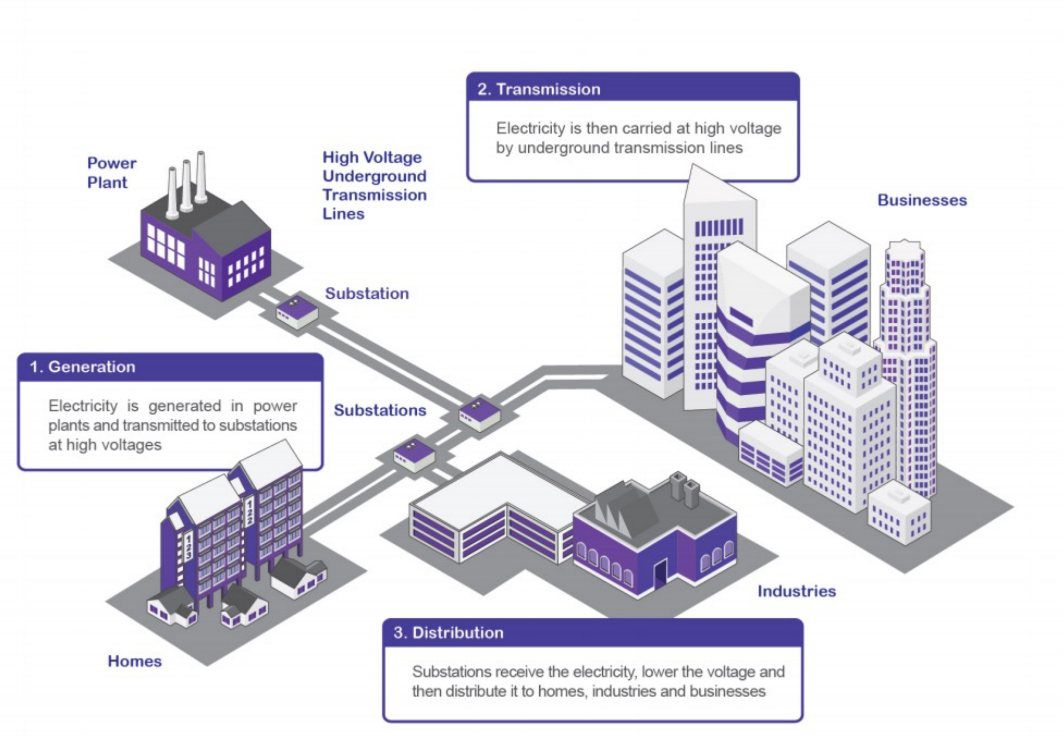
Group 1:

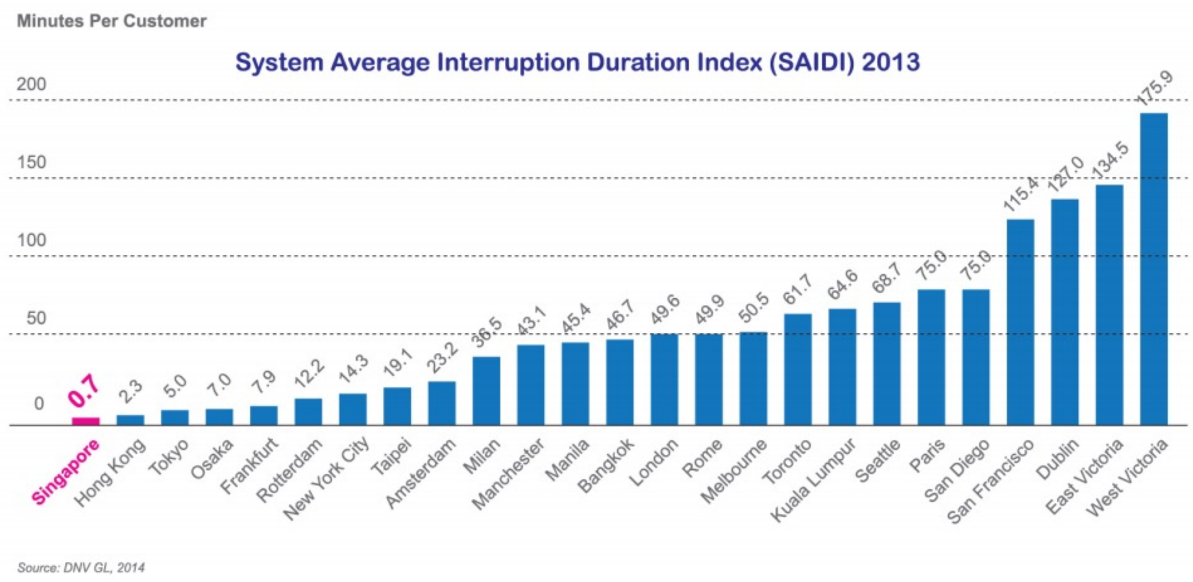
Wong Songhan, Xie Fei, Yan Jin

24 Apr 2021

Motivation & Objective

- Create robust, advanced and extensive electricity grids
- More complex grids due to greater diversity of energy sources
- Improve grid reliability via **efficient and comprehensive maintenance planning**





Source: EMA

Challenges in Maintenance Planning

- How to guarantee grid reliability with a mix of live-line and off-line works ?
- Ensuring network resilience to endure unexpected contingency
- Anticipating future maintenance operations due to aging network, new generation and consumption plans
- New integration of renewable energies

- Yet keeping cost affordable with <u>limited resources and time constraints</u>

Source: EMA

ROADEF Grid Maintenance Planning Challenge

- Biennial open competition (on-going since year 2020) by French Operational Research (OR) and Decision Support Society (ROADEF)
- Objective: Generate a maintenance interventions schedule that minimize risk for different simulated scenarios

Outage risk calculation for different scenarios
- Simulator

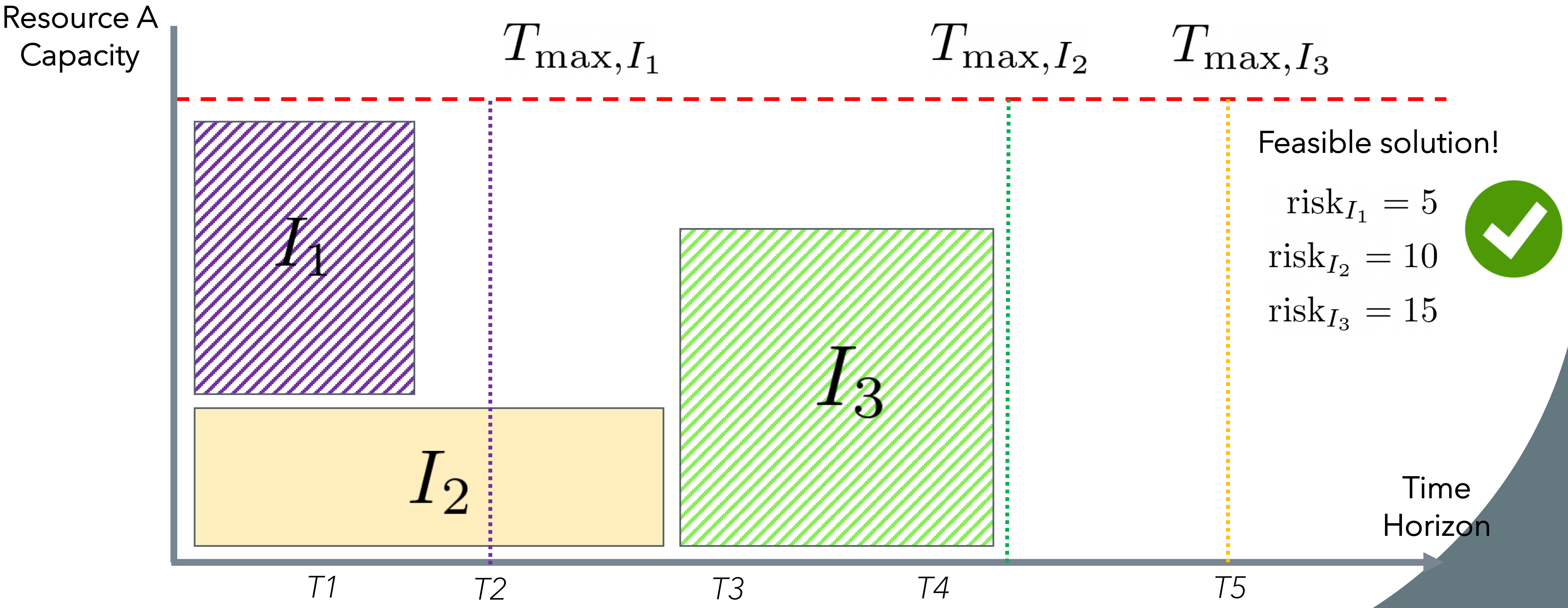
Maintenance planning
- Optimization

Validation and robustness
- Ground-truthing

Focus of the challenge!

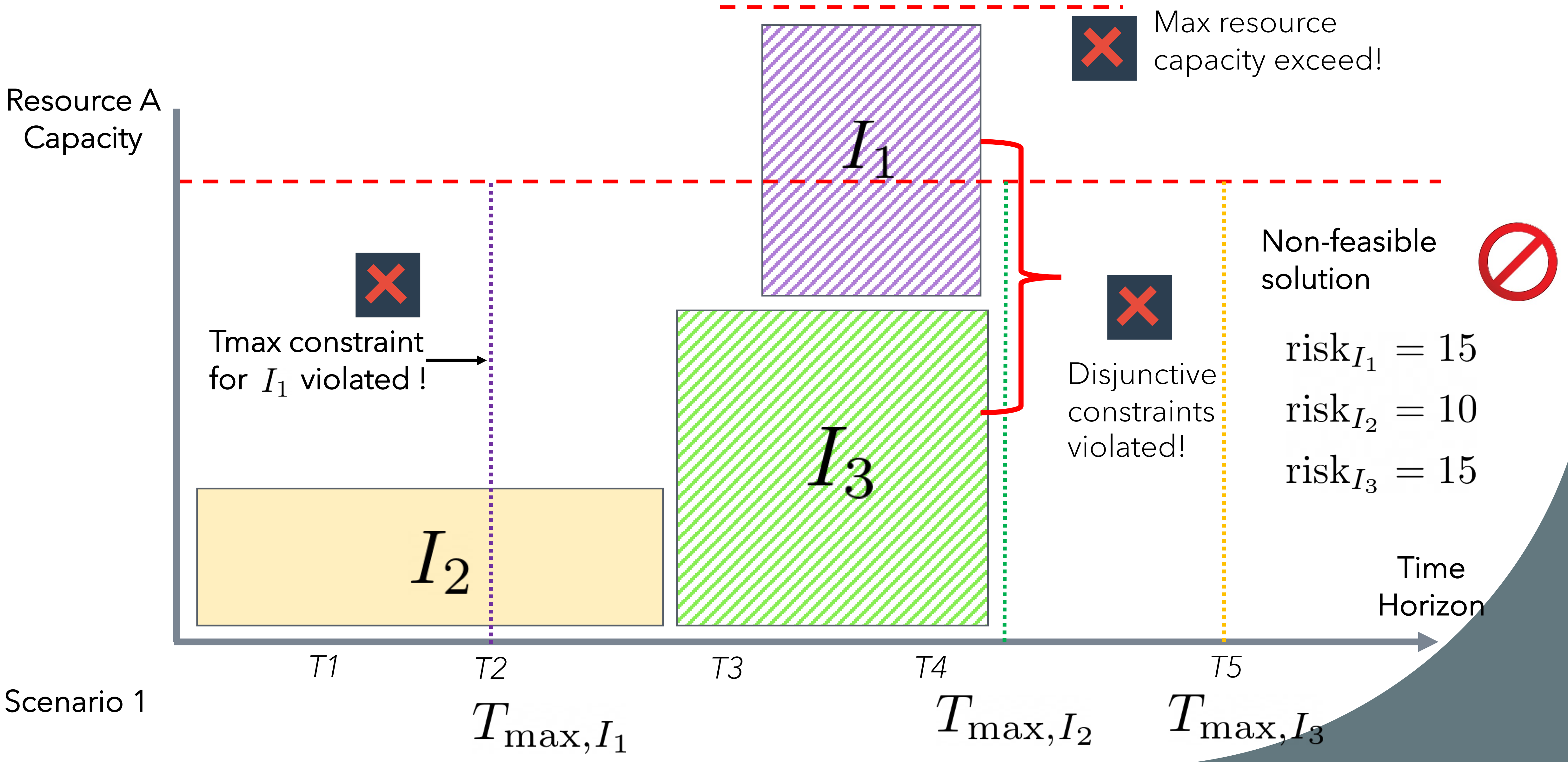
Problem Definition

- Extension of Resource Constrained Project Scheduling Problem (RPCPSP)



Scenario 1

Problem Definition



Model Variables & Assumptions

- Time horizon, $H = \{1, ..., T\}$
  For instance T = 365 for a day by day schedule and T = 53 for a week by week one
- Resources, $C = \{c_1, c_2, ..., c_N\}$
  Equipment, manpower of different skill sets
- (u_t^c) Upper bound of resources at different time step
- (l_t^c) Lower bound of resources at different time step (prevent workforce from not being utilised on some days)
- Interventions, $I = \{I_1, I_2, ..., I_N\}$
- Time duration, $\Delta_{i,t}$
- Resource workload, $r_{i,t'}^{c,t}$
- Intervention time and start time $t \in H, t' \in H$

Model Variables & Assumptions

- Risks, $\text{risk}_{i,t'}^{s,t}$

  e.g the risk level of performing certain interventions during winter is higher as compared to summer time

- Scenario, $s \in S_t$

  e.g different load demand or extreme weather events

Decision Variable

$$L - \text{a list of L of pairs}\,(i, t) \in I \times H, \text{where}\,t\,\text{is the starting time of intervention}\,i.$$

$$[(I_1 : 5), (I_2 : 2), (I_3 : 32), (I_4 : 23), (I_5 : 36), ..., (I_N : 87)]$$

Constraints

- Once an intervention starts, it cannot be interrupted

$$t_{i,\mathrm{end}} = t'_i + \Delta_{i,t}$$

- All interventions must be scheduled, and must not exceed the maximum time allowed

$$1 \leq t'_i + \Delta_{i,t'} \leq T_{\max,i} \ \ \forall i, t'$$

- Resource constraints

$$l_t^c \leq \sum_{i \in I_t} r_{i,t'_i}^{c,t} \leq u_t^c \quad \forall\, c \in C, t \in H$$

- Disjunctive constraints

$$i_1 \in I_t \implies i_2 \notin I_t \ \ \forall (i_1, i_2, t) \in \Phi_{\mathrm{Exclusion}}$$

$I_t \subseteq$ I the set of interventions in process at time t $\in$ H

Objective Function

- Objective 1: Minimise total risks of all interventions averaged over all scenarios and time horizon

$$\boxed{obj_1 = \frac{1}{T}\frac{1}{|S_t|}\sum_{s\in S_t, t\in H, i\in I_t} \mathrm{risk}_{i,t_i'}^{s,t}}$$

- Objective 2: Minimise the excess of risks (indirect measure of variance)

$$\boxed{obj_2 = \frac{1}{T}\sum_{t\in H} \max(0, Q_\tau^t - \overline{\mathrm{risk}}^t)}$$

where Q_τ^t is the quantile of risk values among the scenarios at time t. For example, $Q_{0.5}^t$ is the median.

- Total objective (Minimize)

$$\boxed{obj(\tau) = \alpha \cdot obj_1 + (1-\alpha)\cdot obj_2(\tau)}$$

Problem Solving

- Finding a feasible solution is non-trivial
- Naive Attempt: Mathematical Programming using Docplex
- Wise Attempt: Lagrangian Relaxation with Heuristics Searches in Python
  - Random Iterated Local Search (RILS)
  - Ant Colony Optimization (ACO)
  - Local Beam Search (LBS)
  - Simulated Annealing (SA)
  - Genetic Algorithm (GA)

Mathematical Programming using Docplex

Major difficulties: nested dictionary, cannot use decision variables as dict.keys()

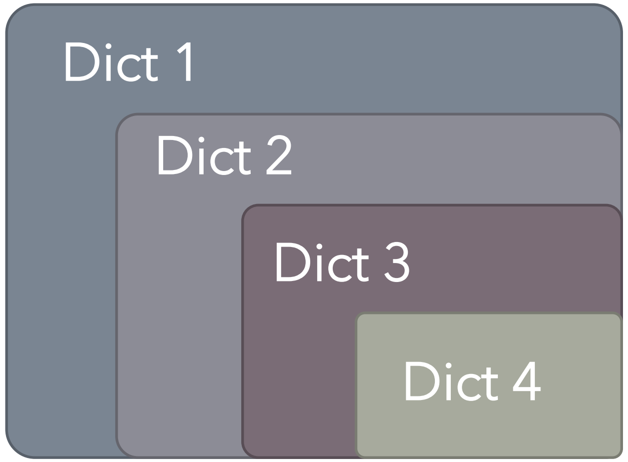
- Solved by defining more decision variables and adding constrains

Objective Function: cannot sort inside decision variables

- Can only compute part of the objective, leading to non-optimal solutions

Suitable for small-size problems

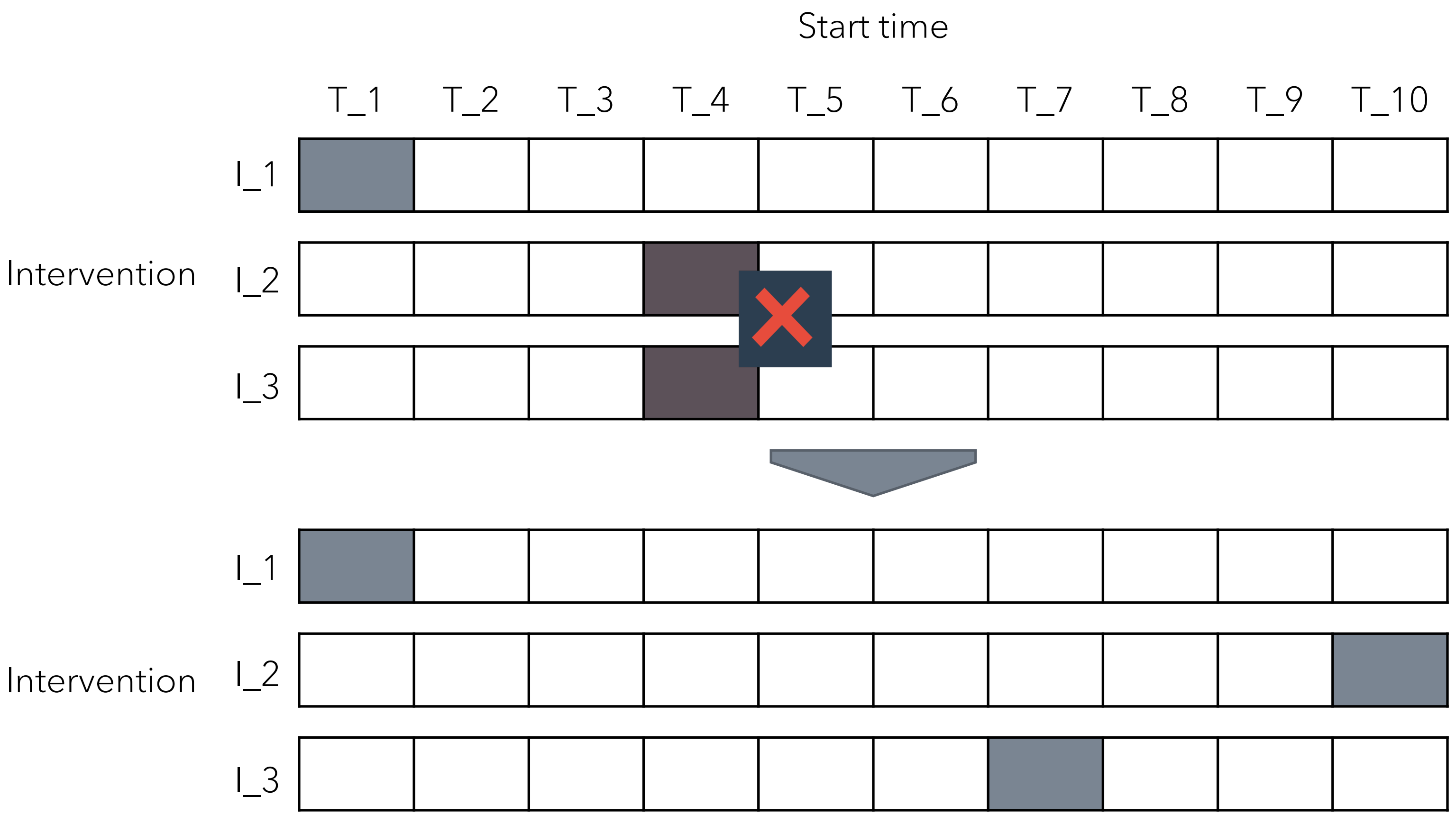
- As the problem size increase, the convergence process is VERY slow
- 5 min -> 90% mipgap with horrible solutions
- SUPER slow to load data with size > 100 MB



Risk_1 > Risk_2 ?


```
Elapsed time = 125.92 sec. (197579.25 ticks, tree = 0.09 MB, solutions = 15)
*    39+    9                    45068.9118      51.6525          99.89%
*    41+    9                    45065.0249      51.6525          99.89%
     45    29      198.3433  2631    45065.0249      51.6525    49480   99.89%
```

Random Iterated Local Search (RILS)



Penalty =1
penalty_set = {I2,I3}

Randomly select a start time for I_2 and I_3 until no more conflicts

Penalty =0
penalty_set = {}

Feasible Solution!

Fix a good start_time, change the conflicting start_time.

Select the combinations with the **lowest penalty count** and repeat

Random Iterated Local Search (RILS)

Min-conflict approach

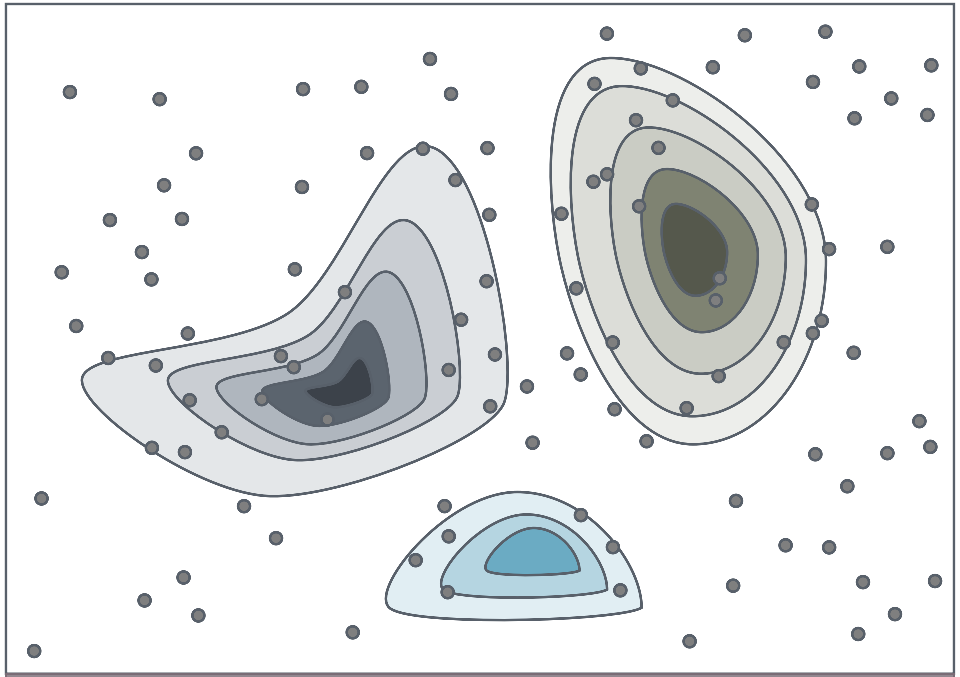
- Always guarantee a feasible solution

≈ Constraint Programming

- Find as many feasible solutions as possible
- Select the solution with the best objective value

Cannot search neighborhood

- Stop when the penalty = 0
- No notion of gradient descent toward optimality



Random but have no tendency to move toward optimal

Lagrangian Relaxation

Relax the hard constraints and include a penalty term to the objective function

- Resource constraints
- Disjunctive constraints

$y_{i,t,c} = 1$ for every constraint violated

$$obj(\tau) = \alpha \cdot obj_1 + (1 - \alpha) \cdot obj_2 + \boxed{P \sum_{i,t,c} y_{i,t,c}}$$

where P is the cost of penalty.

Ant Colony Optimization (ACO)

Pheromone Trail

Weights for Lagrangian Relaxation	
1000	1

Interventions

Start time

	I_1	I_2	…	…	…	I_79	I_80	Penalty	Objective	Weighted-objective
solution 1	23	1	…	…	…	13	76	3	4444	7444
solution 2	45	22	…	…	…	43	62	5	3333	8333
solution 3	12	35	…	…	…	73	47	0	4444	4444
			…	…	…			…	…	…
solution 49	42	3	…	…	…	55	38	20	4444	24444
solution 50	29	2	…	…	…	13	67	8	2222	12222

Applying pheromone on top3 solutions!

start_time probability for Intervention 1

(Assuming max start time for I_1 is 60)

1	…	12	…	23	…	45	…	61	…	80
$\frac{1}{60}$	$\frac{1}{60}$	$\frac{1}{60}+\frac{100}{7444}$	$\frac{1}{60}$	$\frac{1}{60}+\frac{100}{7444}$	$\frac{1}{60}$	$\frac{1}{60}+\frac{100}{7444}$	$\frac{1}{60}$	0	0	0

Weighting:100

Ant Colony Optimization (ACO)

Local Heuristic

Start time →

Intervention ↓

1/60	1/60	…	…	…	0	0
1/80	1/80	…	…	…	1/80	1/80
1/79	1/79	…	…	…	1/79	0

Probability for intervention i to choose start time t

originally $\frac{1}{\text{max start\_time}}$

But for some cases which feasible solutions are hard to find…

Provide some hints for the searching process!

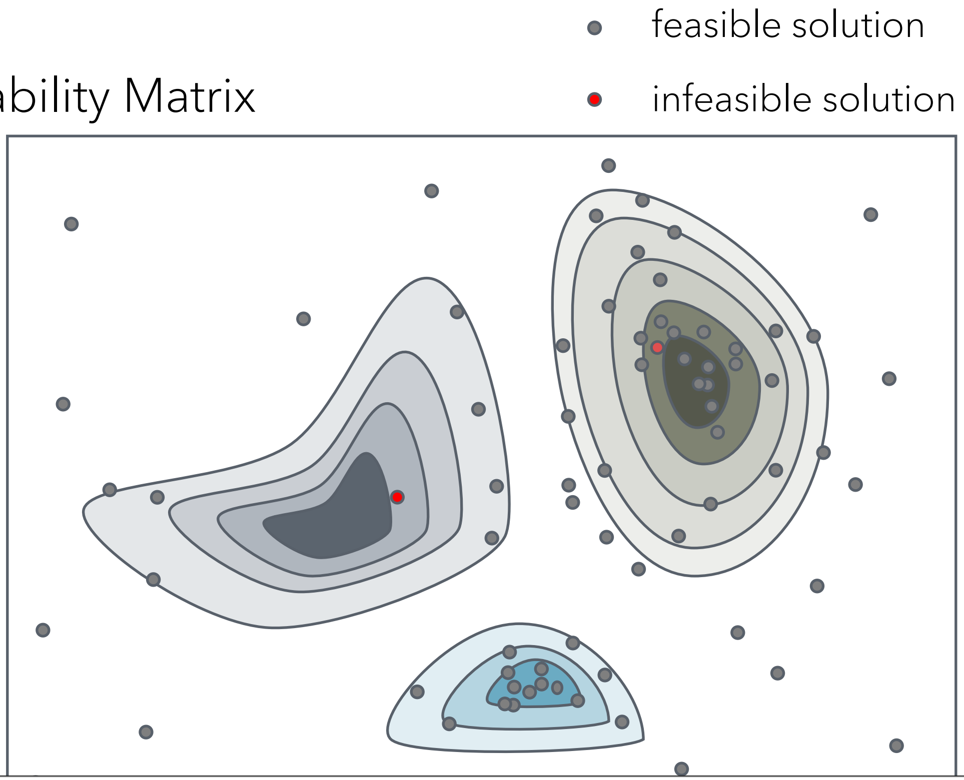
Start time →

Intervention ↓

1/60	1/20	…	…	…	0	0
1/80	1/80	…	…	…	1/50	1/80
1/30	1/79	…	…	…	1/79	0

Feasible solution from
Random Iterated Local Search (RILS)

Ant Colony Optimization (ACO)



Local heuristic + pheromone trail → Normalization → Probability Matrix

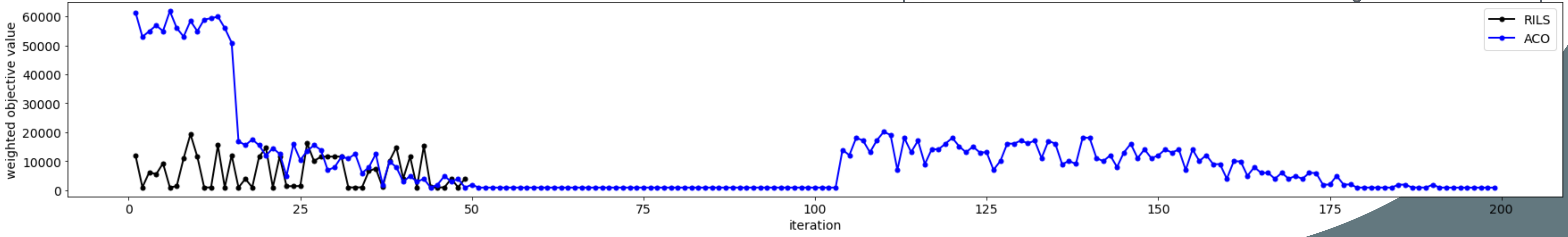
- Ants explore the good solutions
- Repeat until satisfied

Exploration

- Assign some probability to explore alternative solutions
- Increase the probability if good

Dependent on the first feasible solution

- When stuck , reset the pheromone trail and local heuristic matrix
- Could be worse, but worth trying



Local Beam Search

- Size of neighborhood = 50, initialize with random solutions
  - Select top k = 5 best solutions and apply perturbation to generate 10 child solutions for each parent
  - repeat the process until convergence
- Mechanism of perturbation
  - Pick interventions with *p* probability (e.g 0.05) and randomly shift the start time by *W* (e.g 5)
  - Check and impose time constraints for each intervention

Selected parents	I_1	I_2	I_3	…	…	…	I_78	I_79	I_80
Solution 1	23	1	45	…	…	…	22	13	76
Solution 2	45	22	26	…	…	…	3	43	62
Solution 50	12	35	36	…	…	…	39	73	47

Apply perturbation to generate new child

Solution 1_1	25	6	45	…	…	…	22	13	76
Solution 1_2	23	1	42	…	…	…	25	18	76
Solution 1_3	23	5	45	…	…	…	22	13	70
Solution 1_4	23	1	41	…	…	…	20	13	76

Simulated Annealing

- Select solution with p = exp(-E/kT)
  - E = obj (new) – obj (old)
- Neighborhood generation
  - One solution at a time
  - Apply perturbations interventions with *p (T)* probability and randomly shift the start time by *W(T)*
  - More perturbation at the start (higher temperature), and less perturbation as it cools down
- Temperature cooldown every N iterations

$$T' = T_{old} \cdot 0.7$$

- Python library from **scikit-opt**

Genetic Algorithm (GA)

Why choose GA?

- The structure of solution is easy perform cross-overs and mutations.
- More radical in exploring the solution space

How to perform GA? (using scikit-opt)

- Generate initial chromosomes
- For each iteration, do:
  - Ranking
  - Selection
  - Cross-over
  - Mutation

	I_1	I_2	I_3	I_4	I_5	I_6
Parents 1	23	1	45	17	11	10
Parents 2	45	22	26	8	11	16
Cross-over	23	1	26	8	11	10
Mutation	23	1	42	8	11	10

Genetic Algorithm (GA)

In general, GA's convergence is hard to predict and often stuck at local optima when there are many constraints.

How to tune GA?

- A larger mutation probability can help to escape from local optima but made it harder to reserve desirable genes.
- A larger population size can help to escape from local optima as it contains more possibility, but increase the time of iterations

Results

Problem Set	Size (interventions x periods x scenario)	Characteristics
A_01	181 x 90 x 1	Small size with few constraints
A_05	180 x 182 x 120	Large size but fewer constraints
A_06	180 x 182 x 1	Medium size with moderate number of constraints
A_15	108 x 53 x 347	Medium size with large number constraints

Measure **optimality gap** at t = 60s, 300s, 600s, 900s

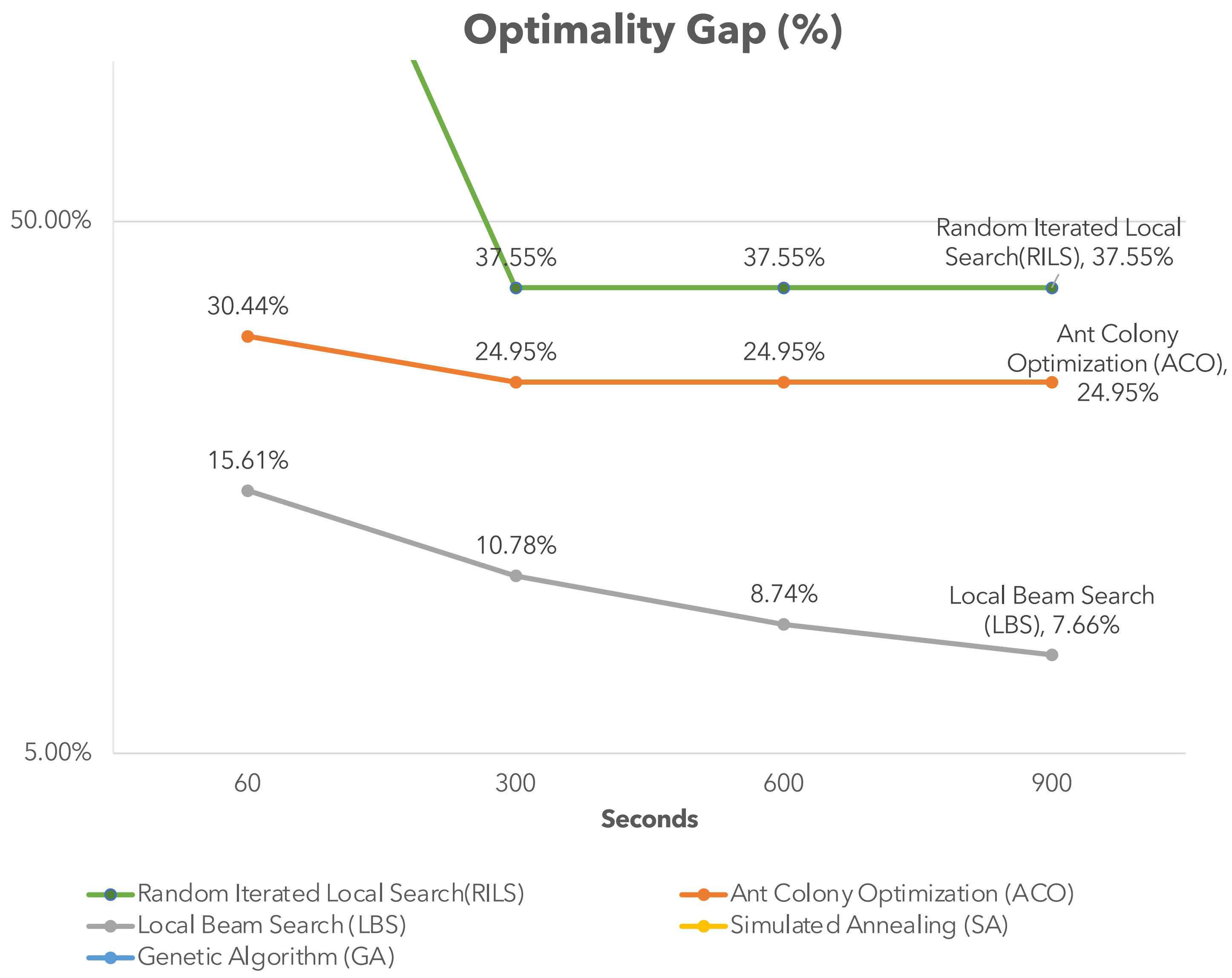
- * Optimal value = best known solution from competition results

Run benchmark for each algo

- Performance varies for each algo slightly from run to run depending on initial random solution
- Ideally should run for ***N*** tests for every algo and measure the mean/spread

Results

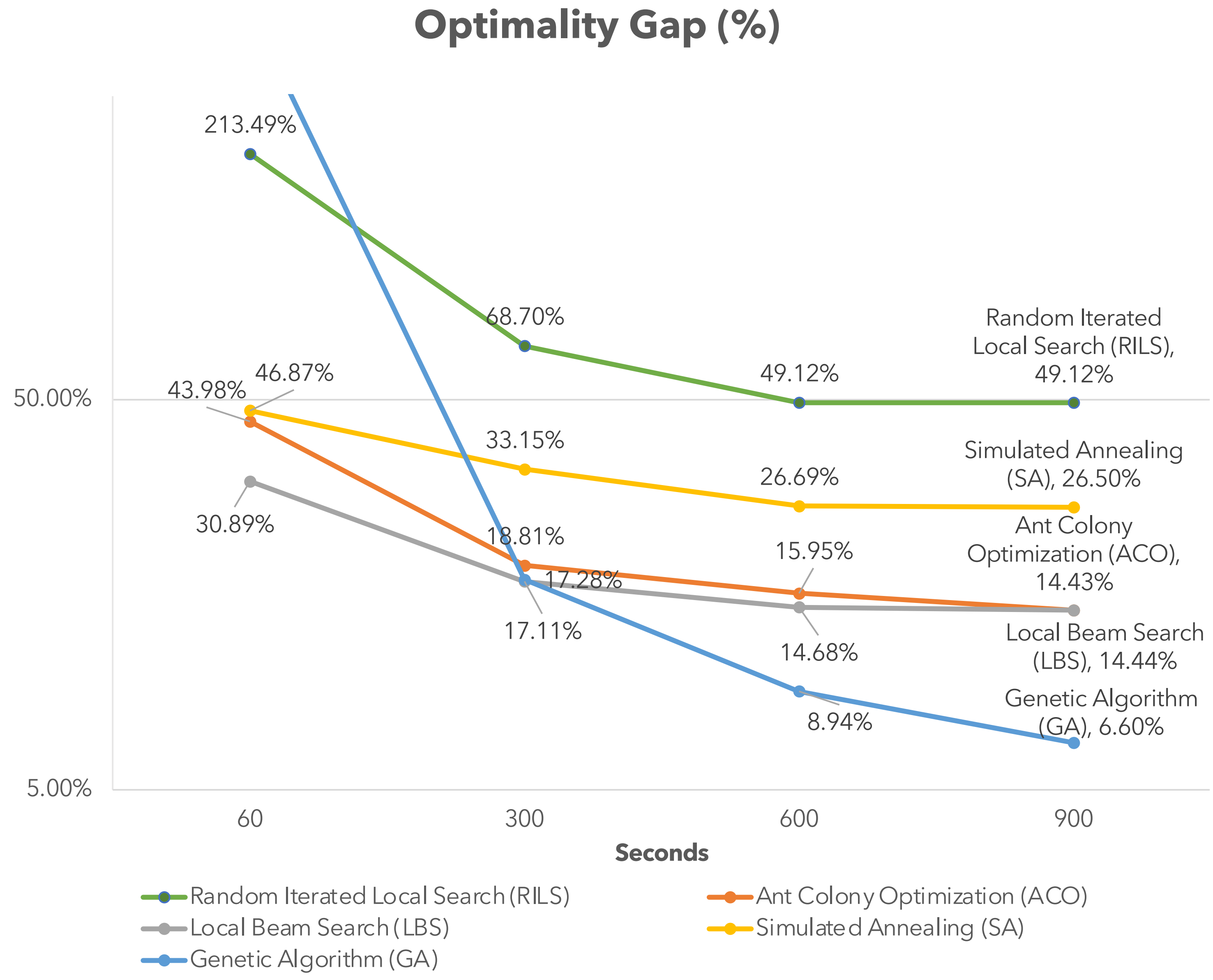
- A_01 – 181 (interventions) x 90 (periods) x 1 (scenario)



Optimality Gap	60s	Violation	300s	Violation	600s	Violation	900s	Violation
RILS	15656.87%	0	37.55%	0	37.55%	0	37.55%	0
ACO	30.44%	0	24.95%	0	24.95%	0	24.95%	0
LBS	15.61%	0	10.78%	0	8.74%	0	7.66%	0
SA	24.60%	1	17.56%	1	17.56%	1	17.56%	1
GA	19.15%	4	6.71%	4	4.12%	4	3.18%	3

Results

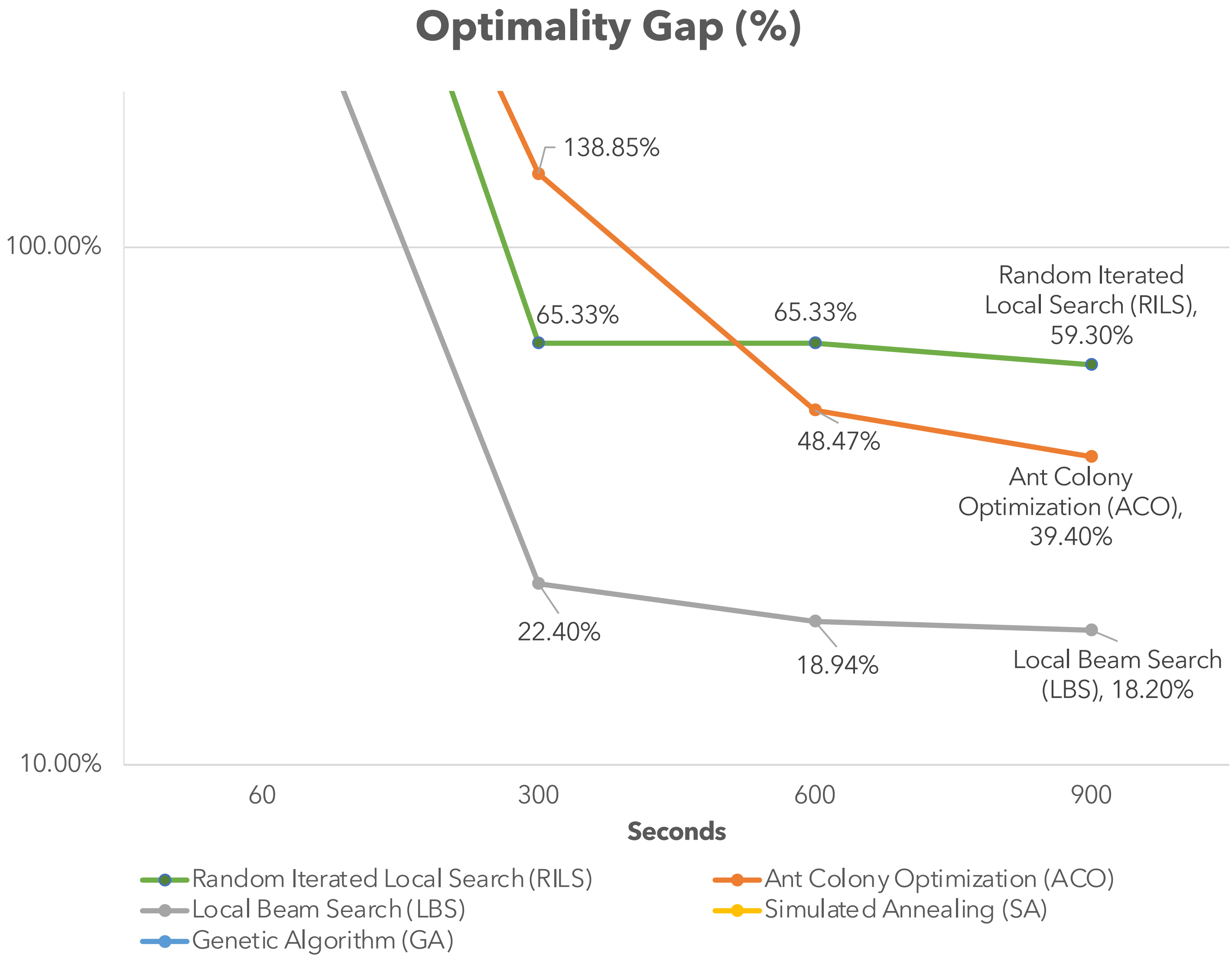
A_05 - 180 interventions x 182 periods x 120 scenario



Optimality Gap	60s	Violation	300s	Violation	600s	Violation	900s	Violation
Random Iterated Local Search	213.49%	0	68.7%	0	49.12%	0	49.12%	0
Ant Colony Optimization (ACO)	43.98%	0	18.81%	0	15.95%	0	14.43%	0
Local Beam Search (LBS)	30.89%	0	17.11%	0	14.68%	0	14.44%	0
Simulated Annealing (SA)	46.87%	0	33.15%	0	26.69%	0	26.5%	0
Genetic Algorithm (GA)	46.24%	1	17.28%	0	8.94%	0	6.59%	0

Results

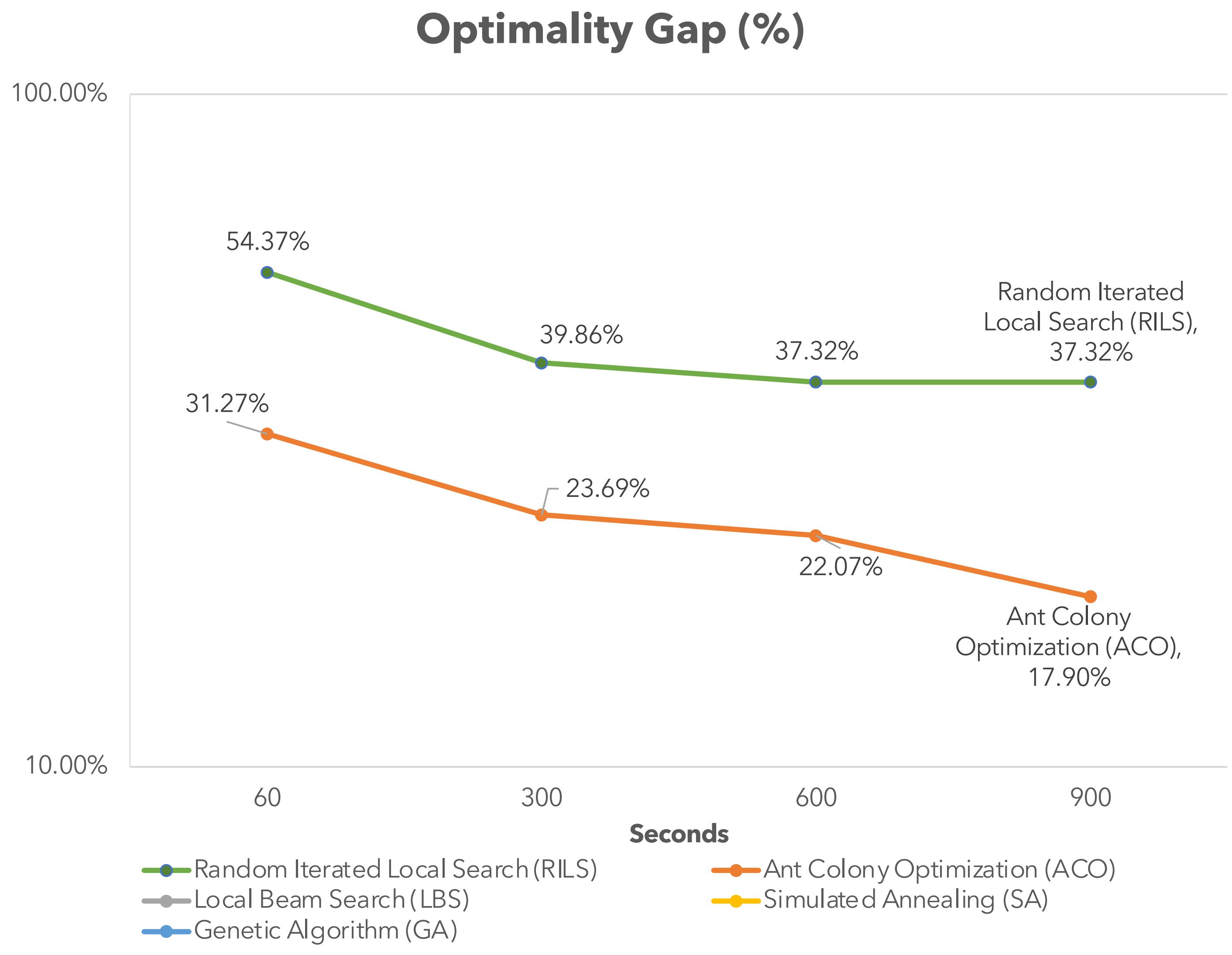
A_06 - 180 interventions x 182 periods x 1 scenario



Optimality Gap	60s	Violation	300s	Violation	600s	Violation	900s	Violation
Random Iterated Local Search	2318.42 %	0	65.33%	0	65.33%	0	59.3%	0
Ant Colony Optimization (ACO)	1938.4 %	0	138.85%	0	48.47%	0	39.4%	0
Local Beam Search (LBS)	59.11 %	36	22.4%	0	18.94%	0	18.2%	0
Simulated Annealing (SA)	56.7%	23	28.8%	16	28.86%	16	28.8%	16
Genetic Algorithm (GA)	56.9%	17	27.4%	10	21.35%	9	19.5%	9

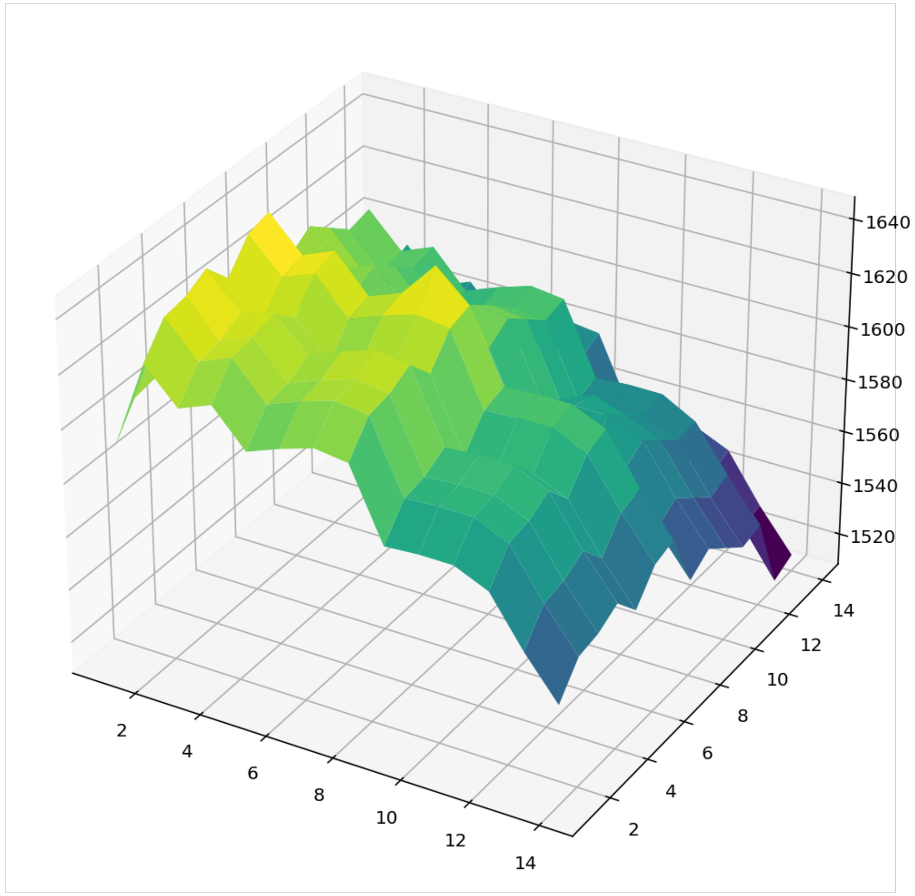
Results

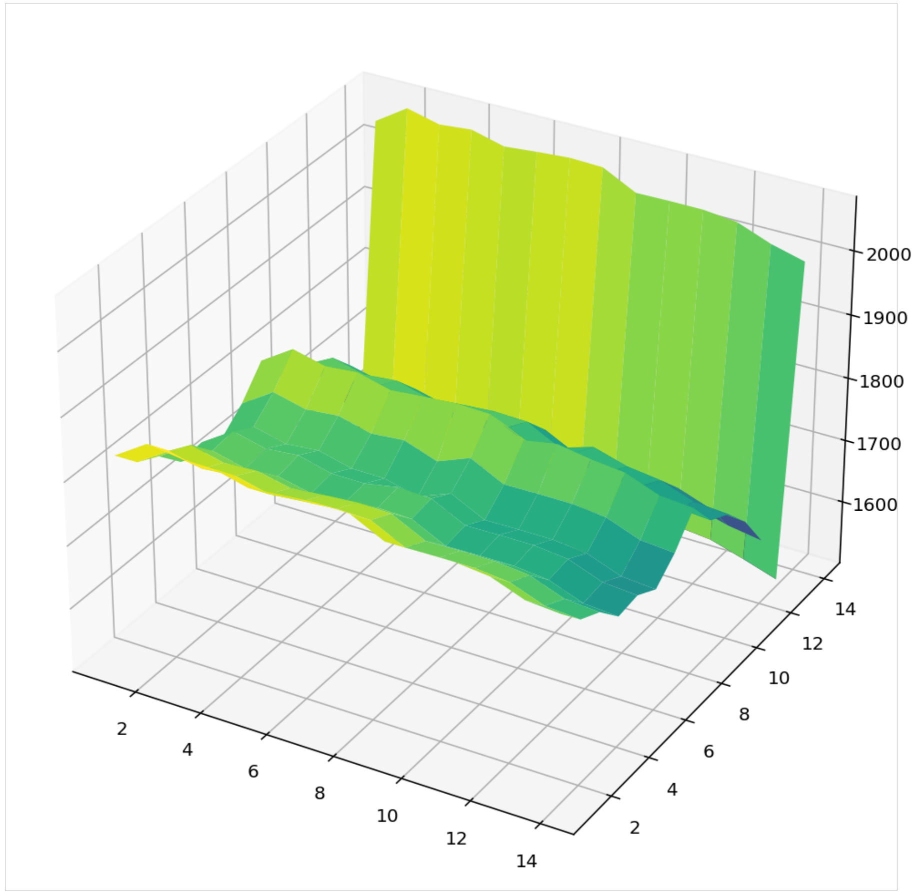
A_15 - 108 interventions x 53 periods x 347 scenario



Optimality Gap	60s	Violation	300s	Violation	600s	Violation	900s	Violation
Random Iterated Local Search	54.37%	0	39.86%	0	37.32%	0	37.32%	0
Ant Colony Optimization (ACO)	31.27%	0	23.69%	0	22.07%	0	17.9%	0
Local Beam Search (LBS)	11.60%	9	7.85%	7	7.85%	7	7.85%	7
Simulated Annealing (SA)	12.37%	7	1.61%	7	1.61%	7	1.61%	7
Genetic Algorithm (GA)	22.66%	1	14.61%	1	10.40%	1	10.14%	1

Key Learning Points – Objective Function





Key Learning Points

- Neighborhood generation scheme

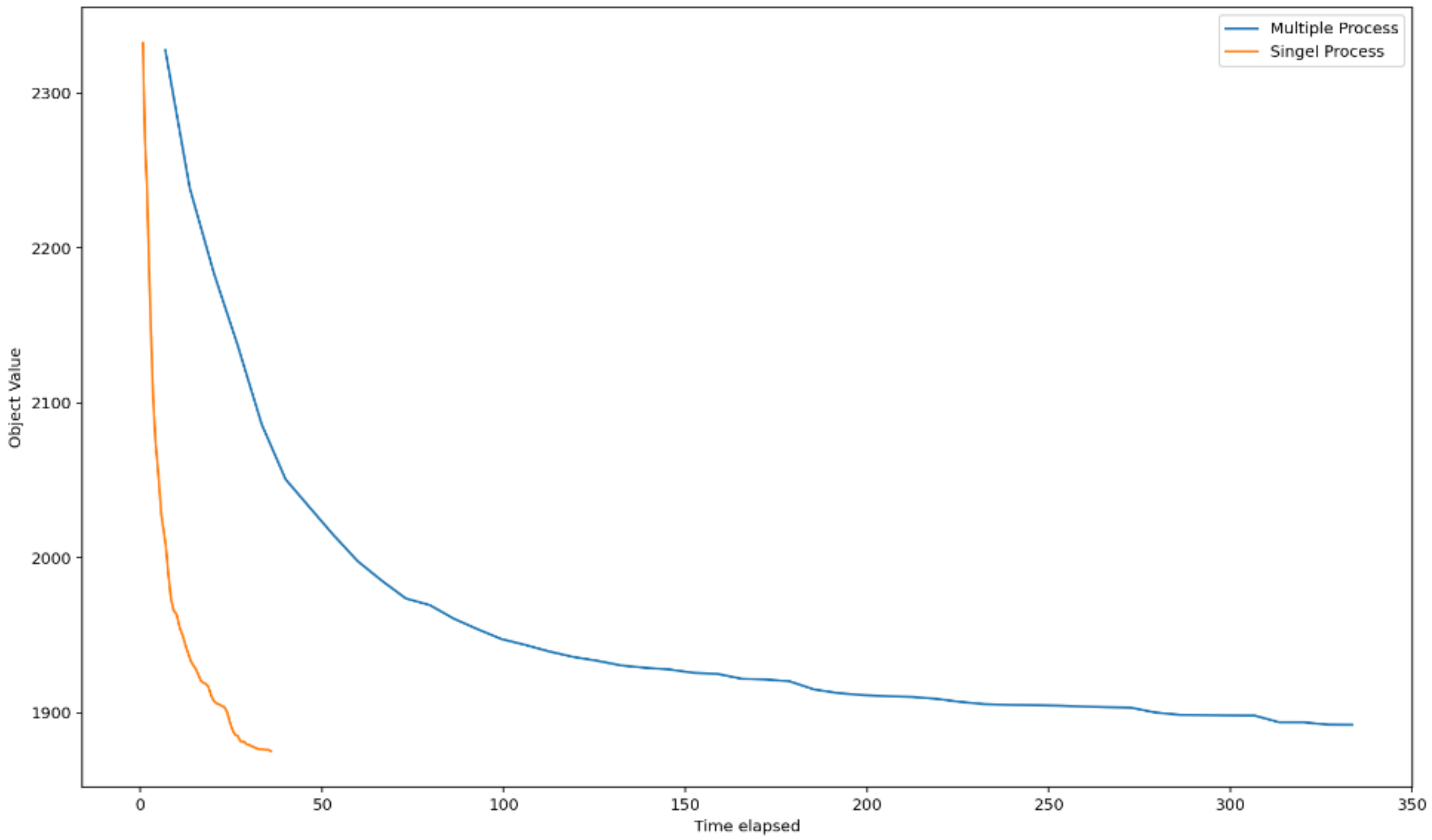
- Tuning of hyperparameters (penalty cost, probability threshold, time window …)
  - Convergence speed
  - Adaptive design
  - Ability to find feasible solution
  - Exploitation vs exploration

- Sensitivity to initial solution
  - How to be robust
  - Usage of warm start

- How to escape from local optima
  - Restart after *N* iterations with no improvement

- Time complexity and multiprocessing techniques

Key Learning Points – Multiple Processes



Summary

- Successful application of Langrange relaxation and several heuristics in solving an extended RPCPSP problem

- In general, ACO most robust but LBS most optimal

- Strategies to tune heuristics search to balance between exploration vs exploitation at different junctures