Handling uncertainty in routing and scheduling problems

David Pisinger, DTU Management, Technical University of Denmark

PATAT-2026, Nottingham, 25-28 August 2026

PATAT
Conference
2026

THE 15TH CONFERENCE ON THE
PRACTICE AND THEORY OF
AUTOMATED TIMETABLING

Abstract

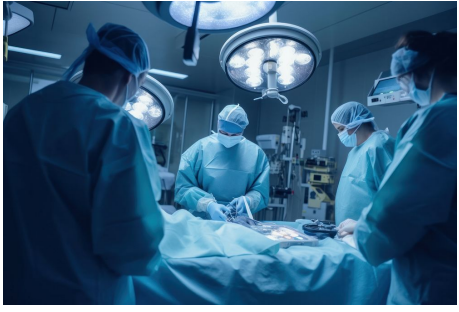
Many routing and scheduling problems have an inherent uncertainty, that make it difficult to find the best possible plan. The talk will present a general local search framework for scenario-based two-stage combinatorial stochastic programming problems. The framework is based on Adaptive Large Neighborhood Search (ALNS), where an upper-level local search method operates on the first-stage decision variables, while a number of lower-level local search methods optimize the individual scenarios. As a demonstration of the Stochastic ALNS algorithm, the framework is tested on the two-stage stochastic Prize-collecting Vehicle Routing Problem, and stochastic Team Orienteering Problem. In all problems, the customers, demands and travel times can be stochastic as long as they can be represented by a number of scenarios. Computational experiments show that the algorithm scales well with the number of scenarios, making it possible to solve instances with up to 1000 customers in short time.

Furthermore, the talk will present a deep generative scenario-search algorithm as an alternative solution method. Using deep generative models that are trained to replicate empirical scenarios, we propose a local search algorithm that intelligently explores the scenario space, repeatedly solving smaller optimization problems. The goal is to use a few synthetic scenarios to reach a heuristic solution that is close to what could be found by solving the SOP on the full set of empirical scenarios. Finally, we address explainability of the found SOP, since we do not only find optimal first-stage decisions, but also identify a small set of realistic scenarios that justify the choices.

How to handle uncertainty in timetabling and routing

Optimization problems that involve uncertainty

- Schedule busses, but driving time is stochastic
- Schedule surgery rooms, but surgery time is stochastic



Reuse existing solution methods

- Plenitude of good algorithms / models
- Avoid clean-sheet implementation (costly, errors)

Fast solution methods

- In a stochastic world: the earlier you can act the more possibilities

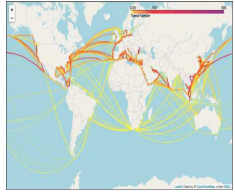
Two-stage stochastic optimization

A framework that separates decision-making into two sequential phases—an initial "here-and-now" decision before uncertainties are known, and a subsequent "recourse" action taken after random events unfold—aiming to minimize the expected cost or maximize the expected return across all possible future scenarios.

Examples:

- **stochastic timetabling**: Announce course-time, then students sign up, allocate rooms
- **stochastic liner shipping**: Construct routes for each vessel, then demand at ports, flow through network

42101	Operations Research	Monday AM
42112	Mathematical Programming Modeling	Friday PM
42114	Integer Programming	Tuesday PM
42115	Network Optimization	Friday AM
42136	Advanced Operations Research Methods	Thursday AM
42137	Optimization Using Metaheuristics	Monday PM
4214?	Recent Research Results	Thursday PM



Scenario-based stochastic optimization

- Represent future by a (large) set S of scenarios (Shapiro et al. 2014)
- *Sample average approximation method*
- maximize expected value

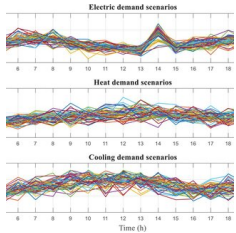
$$z_{\mathbb{E}} = \max_{x \in N_1} \frac{1}{|S|} \sum_{s \in S} z_s$$

  where z_s is the optimal value of the second-stage problem for scenarios s
- The problem becomes a deterministic problem (however it may be large)
- We are solving an approximation

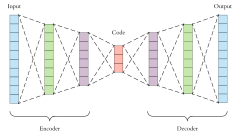
Solve **optimally** for small set S $\Leftrightarrow$ **heuristically** for large set S

Two solution methods

Stochastic Adaptive Large Neighborhood Search (SALNS)
"brute-force", working on all scenarios

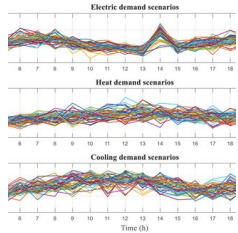


Deep Generative Scenario Search (DGSS)
"learning based", generates a few synthetic scenarios

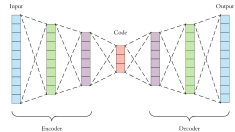


Two solution methods - when to use

Stochastic Adaptive Large Neighborhood Search (SALNS)
Chaotic system, cannot learn pattern in scenarios
Low data quality (generative model will learn errors)
Rare but important scenarios cannot be ignored



Deep Generative Scenario Search (DGSS)
Complex system, generative model learns pattern in scenarios
Improved explainability (small set of scenarios - certificate)



Stochastic ALNS

Computers & Operations Research 188 (2026) 107365

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Distributed or integrated stochastic ALNS? — Two-stage stochastic team orienteering problem

David Pisinger

DTU Management, Technical University of Denmark, Denmark

ARTICLE INFO

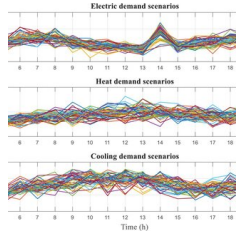
Keywords:
Two-stage stochastic optimization
Metaheuristics
Consensus fixing
Adaptive large neighborhood search
Orienteering problem

ABSTRACT

The team orienteering problem is a generalization of the selective traveling salesman problem. A fleet of vehicles is available to visit a subset of customers within a given time limit. Each customer has an associated profit, and the profit can be collected at most once. In the two-stage stochastic version of the problem, the customers are split into deterministic subscription customers, that (if selected) must be visited in all scenarios, and stochastic on-demand customers that are optional in each scenario. The task is to select a subset of the subscription customers, such that the expected profit of selected subscription customers and on-demand customers is maximized. We present a distributed and an integrated *Stochastic Adaptive Large Neighborhood Search* (SALNS) algorithm for the problem. In the distributed SALNS, an upper-level local search method operates on the first-stage decision variables, while a number of lower-level local search methods optimize the individual scenarios. In the integrated SALNS a single local search method operates on both the first-stage and second-stage decision variables. Computational experiments comparing the two approaches show that the distributed algorithm is able to find better solutions in shorter time. Furthermore, the distributed algorithm scales well with the number of scenarios, making it possible to solve instances with up to 400 first-stage and 400 second-stage customers for 128 scenarios. In many cases, we see an improvement of more than 20% compared to the initial solution.

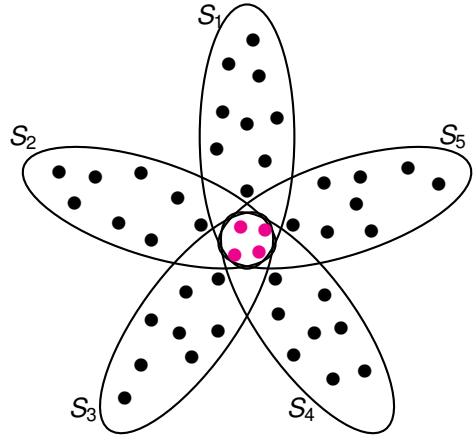
Two-stage stochastic optimization problem

- Assume that future can be represented by a finite set of scenarios $S = \{S_1, S_2, \ldots, S_k\}$
  - Historic data
  - Forecasted data
  - Sampling from distribution



- First-stage variables x that need to be decided now
- Second-stage variables y^s for scenario $s \in S$
- Maximize expected value $\mathbb{E}$ (i.e. “average” objective over all future scenarios)

Two-stage version



- First-stage variables x same in all scenarios
- Second-stage variables y^s for scenario $s \in S$

ALNS

Algorithm 1: Adaptive Large Neighborhood Search

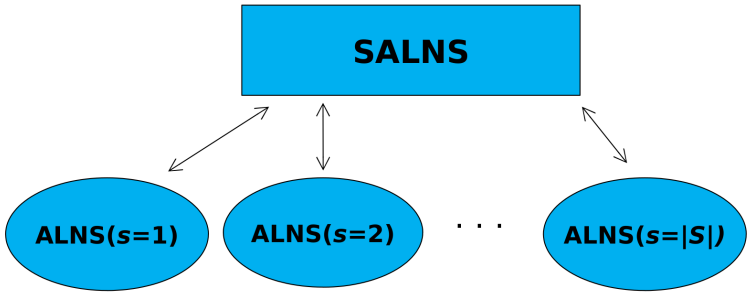
Input w, a feasible solution;
$w_b = w$;
while *stop criterion not met* **do**
    select destroy and repair methods $\mathbf{d} \in \Omega^-$ and $\mathbf{r} \in \Omega^+$;
    $\hat{w} = \mathbf{r}(\mathbf{d}(w))$;
    if $z(\hat{w}) > z(w_b)$ **then** $w_b = \hat{w}$;
    if accept$(\hat{w}, w)$ **then** $w = \hat{w}$;
end
Output w_b, the best found solution;

- Neighborhoods defined by destroy and repair methods
- Learning mechanism to select neighborhoods based on past performance

Stochastic ALNS

- Extends ALNS to two-stage stochastic optimization problems
- Highly parallel
- Second-stage problem can be solved with existing methods

Stochastic ALNS (SALNS)



Sets and Variables

- First-stage decisions N_1
- Second-stage decisions N_2^s for $s \in S$

Sets of variables

- $x_j \in \{0, 1\}$ first-stage variables $j \in N_1$
- $y_j^s \in \{0, 1\}$ second-stage variables $j \in N_2^s$ for $s \in S$

SALNS

Duplicate first-stage variables in each scenario $s \in S$

$$\overline{y}_j^s \equiv x_j$$

Now, split N_1 into two sets

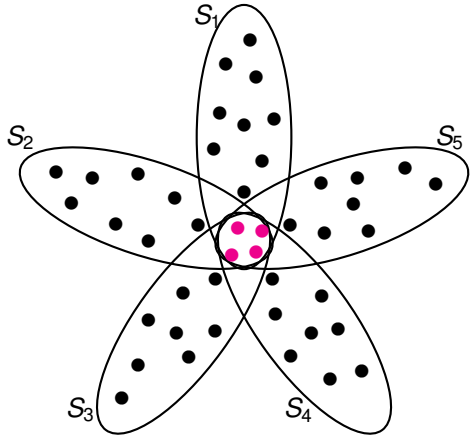
- Fixed variables F (magenta)

$$\overline{y}_j^s = x_j \qquad j \in F$$

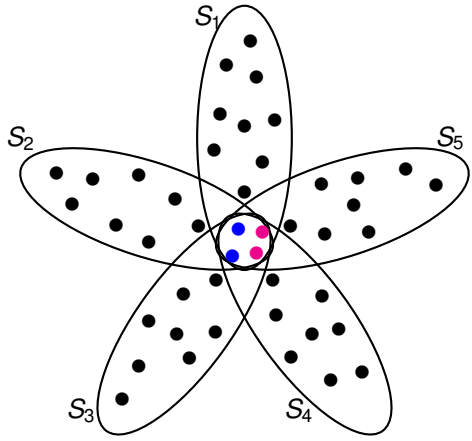
- Unlocked variables U (blue)

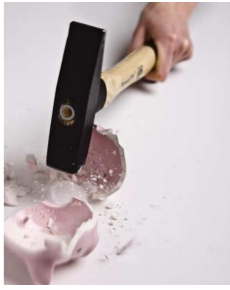
Second-stage variables N_2^s (black)

SALNS, feasible solution

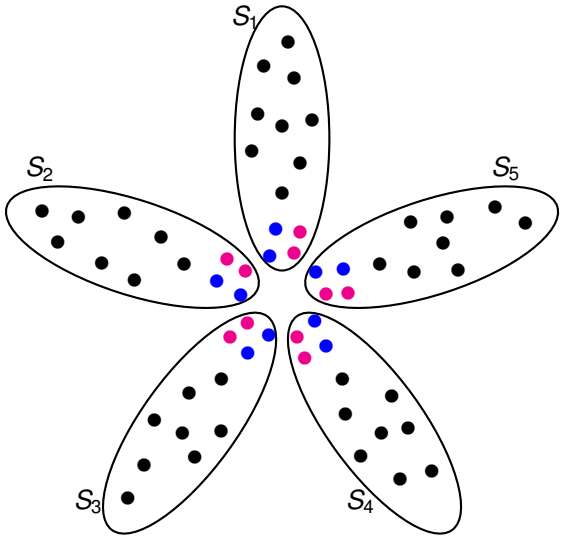


SALNS, free some first-stage variables

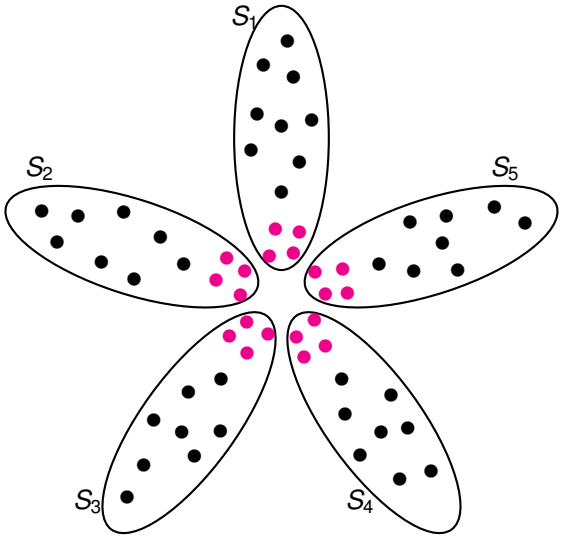




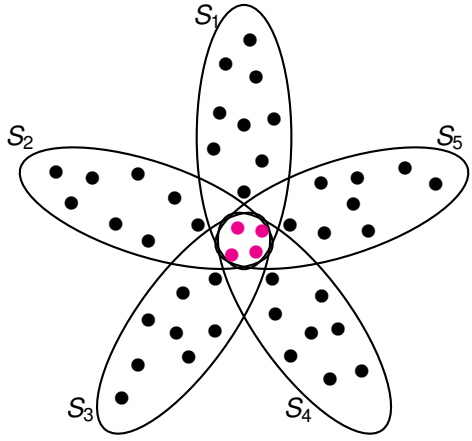
SALNS, run ALNS for each scenario (in parallel)



SALNS, try to reach consensus



SALNS, feasible solution (update best solution)



SALNS

Algorithm 2: Stochastic Adaptive Large Neighborhood Search

Input (x, y), a feasible solution to first-stage and all second-stage problems;
$(x_b, y_b) = (x, y)$; $F = N_1$;
while *stop criterion not met* **do**
    select destroy and repair methods $\mathbf{d} \in \Omega^-$ and $\mathbf{r} \in \Omega^+$;
    destroy the solution;
    for each $s \in S$ solve the respective problem;
    repair the solution;
    for each $s \in S$ improve solution;
    if $z(\hat{x}, \hat{y}) > z(x_b, y_b)$ **then** $(x_b, y_b) = (\hat{x}, \hat{y})$;
    if accept$(z(\hat{x}, \hat{y}), z(x, y))$ **then** $(x, y) = (\hat{x}, \hat{y})$;
end
Output (x_b, y_b), the best found solution;

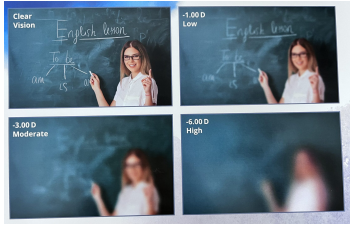
Destroy Neighborhoods (upper level)

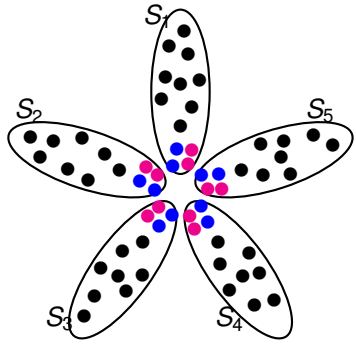
Free a set of first-stage variables

- Nodes located close to each other
- Nodes same demand
- Nodes performing worst
- Least consensus nodes
- Random nodes

Repair Neighborhoods (upper level)

Reaching consensus (Pisinger 2026)
Greedy method: Most obvious now, blurry later





D Pisinger “Consensus fixing for two-stage stochastic optimization – applied to stochastic prize collecting TSP” Transportation Research Part E (2026)

Repair Neighborhoods

Reaching consensus by **voting**

- How important is the decision for you? (δ_{si})
- Who should vote, and how much? (f_s)

In our case the voters are the scenarios $s \in S$

Calculate **polarity**

$$\Pi_i = \left| \sum_{s \in S} f_s \delta_{si0} - \sum_{s \in S} f_s \delta_{si1} \right|$$

The variable i with largest value of Π_i is then fixed to value of x_i having the cheapest cost

Repair Neighborhoods

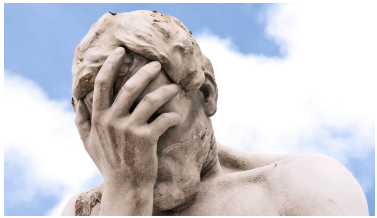
Reaching consensus by voting

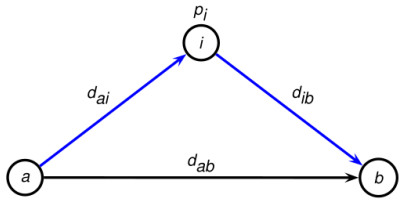
- **How important is the decision for you?** (δ_{si})
- Who should vote, and how much? (f_s)

Regret fixing

Difference in objective function ($x_j = 1$ vs $x_j = 0$)

$$\delta^r_{si0} = z_{si0}$$
$$\delta^r_{si1} = z_{si1} \tag{1}$$





Majority-fixing

$$\delta^m_{si0} = \begin{cases} 1 & \text{if } x_{si} = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\delta^m_{si1} = \begin{cases} 1 & \text{if } x_{si} = 1 \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

Threshold-fixing

$$\delta^t_{si0} = \begin{cases} 1 & \text{if } x_{si} = 0 \text{ and } |z_{si1} - z_{si0}| > T_1 \\ 0 & \text{otherwise} \end{cases}$$

$$\delta^t_{si1} = \begin{cases} 1 & \text{if } x_{si} = 1 \text{ and } |z_{si1} - z_{si0}| > T_2 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where T_1 and T_2 are a thresholds for considering the change in objective function to be important. In most applications it will be natural to choose $T_1 = T_2$, but in principle the two threshold values can be different.

Reaching consensus

- How important is the decision for you? (δ_{si})
- **Who should vote, and how much?** (f_s)

Uniform

Scenario	6	4	7	3	1	8	5	2
z	15.7	13.2	10.5	9.2	7.3	5.1	4.8	3.6
Scale	1	1	1	1	1	1	1	1

Random

Scenario	6	4	7	3	1	8	5	2
z	15.7	13.2	10.5	9.2	7.3	5.1	4.8	3.6
Scale	0	0	0	0	0	0	1	0

Proletarian

Scenario	6	4	7	3	1	8	5	2
z	15.7	13.2	10.5	9.2	7.3	5.1	4.8	3.6
Scale	0	0	0	0	0	0	0	1

Proletarian, linear descent

$$\frac{1}{n}$$

Scenario	6	4	7	3	1	8	5	2
z	15.7	13.2	10.5	9.2	7.3	5.1	4.8	3.6
Scale	$\frac{1}{8}$	$\frac{1}{7}$	$\frac{1}{6}$	$\frac{1}{5}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{2}$	1

Proletarian, quadratic descent

$$\frac{1}{n^2}$$

Scenario	6	4	7	3	1	8	5	2
z	15.7	13.2	10.5	9.2	7.3	5.1	4.8	3.6
Scale	$\frac{1}{8^2}$	$\frac{1}{7^2}$	$\frac{1}{6^2}$	$\frac{1}{5^2}$	$\frac{1}{4^2}$	$\frac{1}{3^2}$	$\frac{1}{2^2}$	1

Elitarian

Scenario	6	4	7	3	1	8	5	2
z	15.7	13.2	10.5	9.2	7.3	5.1	4.8	3.6
Scale	1	0	0	0	0	0	0	0

Elitarian, linear descent

$$\frac{1}{n}$$

Scenario	6	4	7	3	1	8	5	2
z	15.7	13.2	10.5	9.2	7.3	5.1	4.8	3.6
Scale	1	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{1}{6}$	$\frac{1}{7}$	$\frac{1}{8}$

Elitarian, quadratic descent

$$\frac{1}{n^2}$$

Scenario	6	4	7	3	1	8	5	2
z	15.7	13.2	10.5	9.2	7.3	5.1	4.8	3.6
Scale	1	$\frac{1}{2^2}$	$\frac{1}{3^2}$	$\frac{1}{4^2}$	$\frac{1}{5^2}$	$\frac{1}{6^2}$	$\frac{1}{7^2}$	$\frac{1}{8^2}$

Reaching consensus

- Each strategy results in a **repair neighborhood** in SALNS
- We fix variables one by one
- After fixing one variable, we run a few iterations of ALNS
- Until all first-stage variables have reached consensus

Consensus fixing

Algorithm 3: Pseudo-code for consensus-fixing heuristic

1 Let $x = \{x^1, \ldots, x^{|S|}\}$ be the current solutions for each scenario $s \in S$;
2 Let the set of currently fixed first-stage variables be F;
3 **while** $F \neq N_1$ **do**
4 For all scenarios $s \in S$ solve independant problems with x_{si} fixed for $i \in F$;
5 Calculate polarity Π_i for all variables $i \in N_1 \setminus F$;
6 Select the variable $h \in N_1 \setminus F$ with largest polarity Π_h and fix x_h to the best value;
7 Set $F := F \cup \{h\}$;
8 **end**

Notice that line 4 and 5 can be executed in parallel for each scenario $s \in S$.

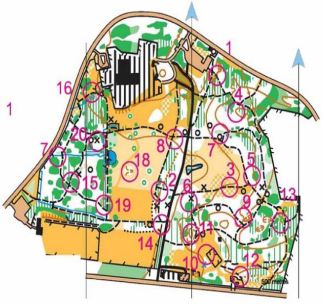
SALNS - all pieces put together

Algorithm 4: Stochastic Adaptive Large Neighborhood Search

Input (x, y), a feasible solution to first-stage and all second-stage problems;
$(x_b, y_b) = (x, y)$; $F = N_1$;
while *stop criterion not met* **do**
    select destroy and repair methods $\mathbf{d} \in \Omega^-$ and $\mathbf{r} \in \Omega^+$;
    destroy the solution;
    for each $s \in S$ solve the respective problem;
    repair the solution;
    for each $s \in S$ improve solution;
    if $z(\hat{x}, \hat{y}) > z(x_b, y_b)$ **then** $(x_b, y_b) = (\hat{x}, \hat{y})$;
    if accept$(z(\hat{x}, \hat{y}), z(x, y))$ **then** $(x, y) = (\hat{x}, \hat{y})$;
end
Output (x_b, y_b), the best found solution;

- Learning mechanism to select neighborhoods based on past performance

Team orienteering problem (generalized prize-collecting TSP)



- Team of runners/vehicles, each having a time limit
- Each prize can be collected at most once
- Each node j has profit $p_j^s \geq 0$

Two-stage stochastic version, scenarios (days) $S_1, \ldots, S_k$

- First-stage (subscription customers) N_1
- Second-stage (on-demand customers) N_2^s

Team orienteering problem

Deterministic version

$$\boxed{\text{selection} + \text{assignment} + \text{sequencing}}$$

Stochastic version

$$\boxed{\text{selection1} + \text{selection2} + \text{assignment} + \text{sequencing}}$$

$$\underbrace{\qquad}_{\textit{first stage}} \qquad \underbrace{\qquad\qquad\qquad}_{\textit{second stage}}$$

Team orienteering problem

Correspondence between TOP and scheduling concepts

Team Orienteering Problem	**Scheduling interpretation**
Location	Activity, job, lecture, task, or operation
Prize	Profit, priority, utility, or importance
Runner/vehicle	Employee, machine, classroom, or production line
Travel time	Setup time, transition time, or changeover time
Route	Schedule assigned to a resource
Route duration	Available working or processing time
Unvisited location	Unscheduled activity
Time window	Activity availability interval
Depot	Start/end of a planning horizon or shift
prioritize activities	all activities are mandatory

Team orienteering problem

- x_j for $j \in N_1$ indicates which first-stage (subscription customers) are selected

The objective is to maximize the expected profit $z_{\mathbb{E}}$ over all scenarios $s \in S$

$$z_{\mathbb{E}} = \max_x \frac{1}{|S|} \sum_{s \in S} z^s(x) \tag{4}$$

Team orienteering problem, scenario $s \in S$

- $\bar{y}_j^{sk}$ for $j \in N_2$, which second-stage (on-demand customers) are selected, vehicle k
- y_j^{sk} routing variables

$$\max \quad z^s(x) = \sum_{k \in K} \sum_{j \in N^s} p_j^s \bar{y}_j^{sk} \tag{5}$$

$$\sum_{k \in K} \bar{y}_j^{sk} \leq 1 \qquad j \in N^s \tag{6}$$

$$\sum_{i \in N^s \setminus \{j\}} y_{ij}^{sk} = \bar{y}_j^{sk} \qquad j \in N^s, k \in K \tag{7}$$

$$\sum_{i \in N^s \setminus \{j\}} y_{ji}^{sk} = \bar{y}_j^{sk} \qquad j \in N^s, k \in K \tag{8}$$

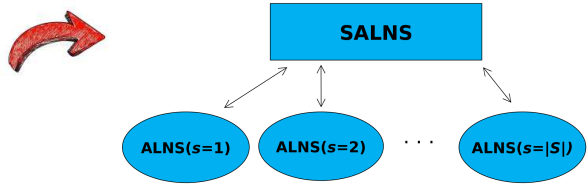
$$\sum_{(i,j) \in E^s} \bar{y}_{ij}^{sk} \leq T \qquad k \in K \tag{9}$$

$$\sum_{k \in K} \bar{y}_j^{sk} = x_j \qquad j \in N_1 \tag{10}$$

Let $G^{sk}(y)$ is the subgraph of G defined by the selected arcs y^{sk}

$$G^{sk}(y) \text{ is a cycle} \tag{11}$$

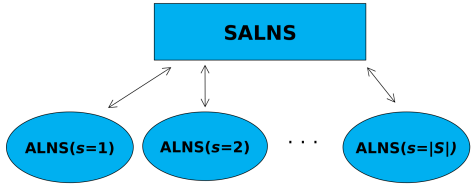
Team orienteering problem - SALNS



Destroy methods – freeing first-stage variables

- Free random
- Free worst
- Free near
- Free similar time
- Free near and similar time
- Free least consensus

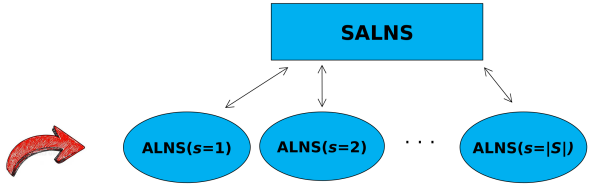
Team orienteering problem - SALNS



Repair methods (ensuring non-anticipativity)
Regret-fixing with different voting schemes:

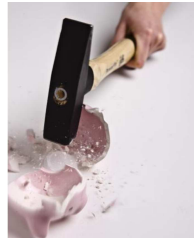
- Uniform
- Elitairian
- Elitairian, linear decrease
- Elitairian, quadratic decrease
- Proletarian
- Proletarian, linear decrease
- Proletarian, quadratic decrease
- Random selection

Team orienteering problem - ALNS

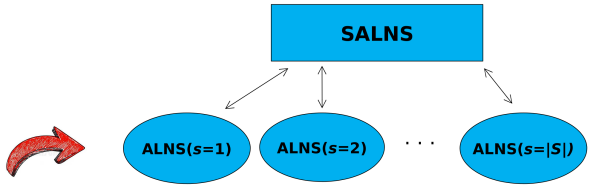


Destroy methods

- Remove random
- Remove worst
- Remove near
- Remove similar time
- Remove near and similar time



Team orienteering problem - ALNS



Repair methods

- Insert most profitable
- Insert most profitable / time
- Insert full

Test setup

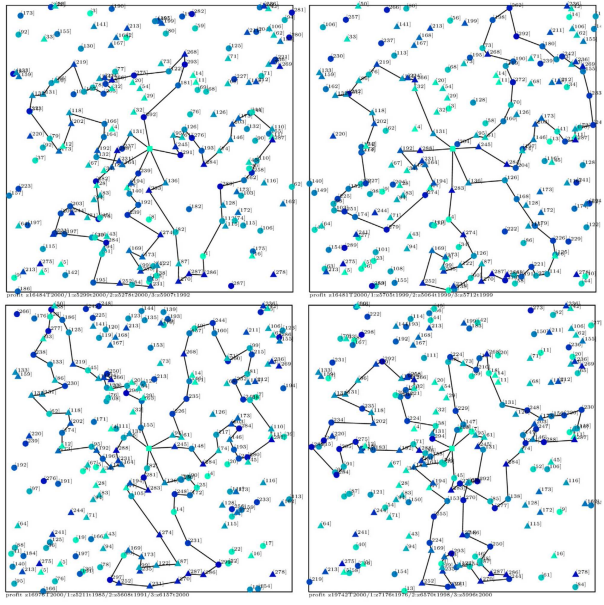
Random generated data of size n_{max}

- $S = \{1, \ldots, |S|\}$ scenarios, having same probability
- $[\frac{3}{4}n_{max}, n_{max}]$ customers located in rectangle $W \times H = 1000 \times 1000$
- Profits randomly distributed in the interval $[1, 300]$
- Euclidean distances rounded to nearest integer (driving time)
- $|K| = 3$ homogeneous vehicles
- Time limit $Q = 2000$

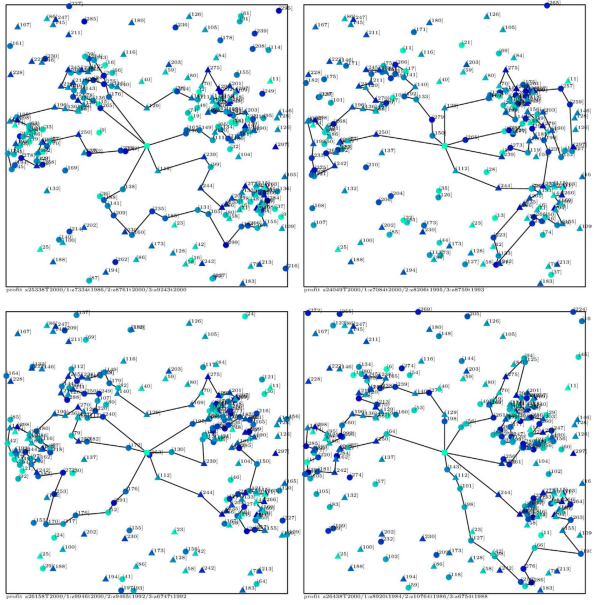
Average of 10 instances

Tests run on:
AMD Ryzen 9, 9750X CPU, with 16 cores, 4.5 GHz base clock, 5.7 GHz turbo clock.
AMD Threadripper 7980X with 64 cores, 3.2 GHz base clock, 5.1 GHz turbo clock.

Uniform data



Clustered data



Greedy Heuristic

Consensus fixing

- For each $s \in S$ solve individual problem using ALNS
- Fix one variable using consensus fixing
- Repeat until all variables are fixed

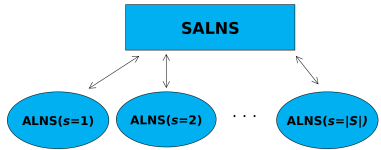
About 25% of CPU time for greedy

Iterations in SALNS

Tuning iterations, SALNS ($n_{\max} = 128$, $|S| = 16$ scenarios)

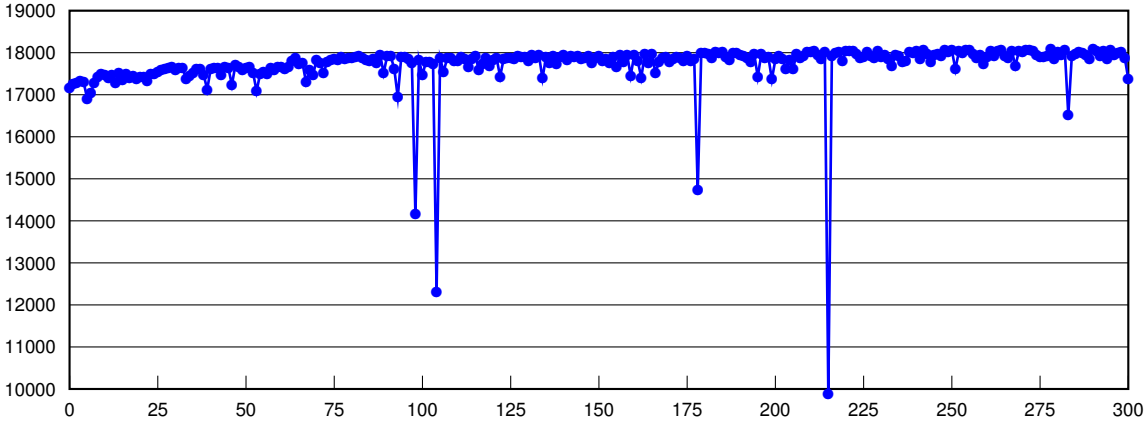
iterations I_{max}	objective $z_{\mathbb{E}}$	absolute improvement	relative improvement	computational time
0	17948			26.02
3	18153	205	1.14%	30.70
10	18356	408	2.27%	43.14
30	18516	568	3.16%	76.33
100	18725	777	4.33%	197.92
300	**18967**	1019	5.68%	560.52
1000	19024	1076	5.99%	1809.49
3000	19038	1090	6.07%	5344.00

$I_{alns} = 300$
Average of 10 instances.
$I_{max} = 300$ selected



SALNS search progress

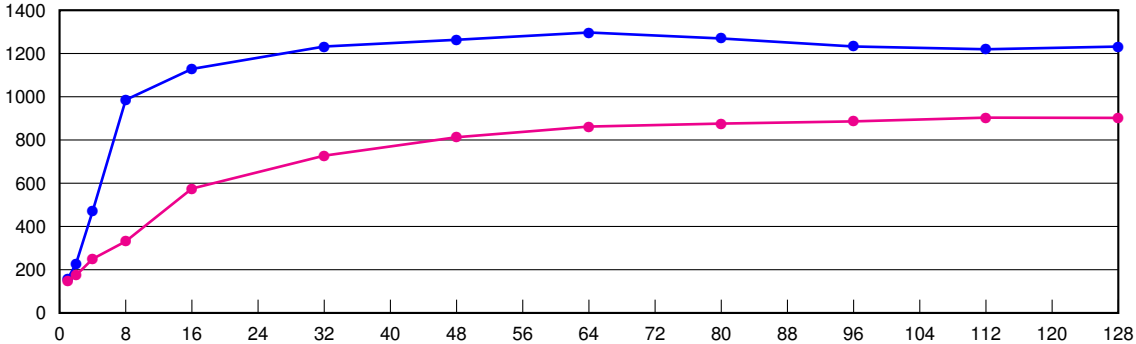
Development of objective $z_{\mathbb{E}}$ during upper-level SALNS algorithm.



Uniform data, $|S| = 16$ scenarios, $n_{\max} = 128$. (AMD Ryzen)

SALNS - many scenarios

Computational time (wall time) in seconds as function of number of scenarios $|S|$, $n_{\max} = 128$



Depicted for 1 to 128 scenarios, for **uniform** data (blue) and **clustered** data (magenta). (AMD Threadripper)

SALNS - improvement of objective

Objective $z_{\mathbb{E}}$ for CF heuristic, versus SALNS final solution.

	uniform					**clustered**				
$n_{\max}$	100	150	200	300	400	100	150	200	300	400
CF heuristic	14794	18414	21576	26272	31205	18262	23671	27377	34382	38047
SALNS final	16081	19924	23540	28852	33430	19697	26349	32457	41088	47955
improvement (%)	8.7	8.2	9.1	9.8	7.1	7.9	11.3	18.6	19.5	26.0

$|S| = 128$ scenarios. Average of 10 instances. (AMD Threadripper)

How good are solutions?

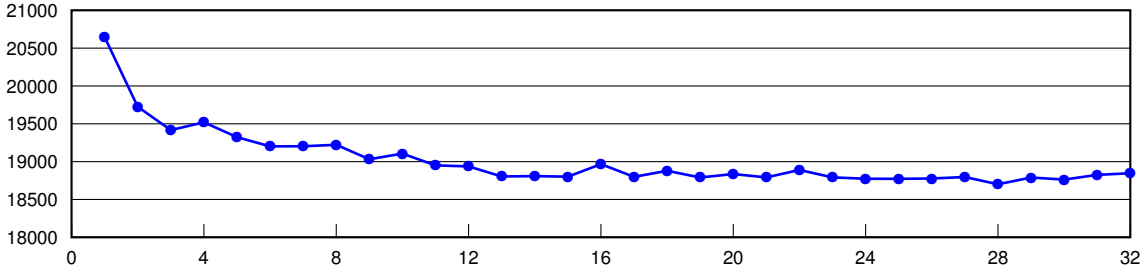
SALNS, objective

n_{max}	100	150	200	250	300	350	400	450	500
independant	16906	21261	25449	28588	31471	33730	36865	38595	40243
SALNS	16257	20364	24395	26955	29908	32195	34725	37120	38571
deviation (%)	4.0	4.4	4.3	6.1	5.2	4.8	6.2	4.0	4.3

$|S| = 16$ scenarios. Average of 10 instances.

Increasing scenarios SALNS

SALNS, solution quality. Expected value $z_{\mathbb{E}}$ as function of $|S|$



SALNS - cookbook

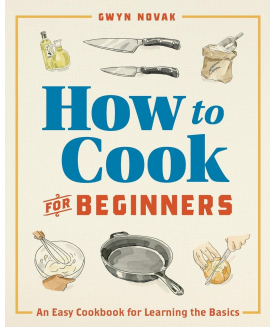
- If you have an ALNS algorithm for deterministic problem
- ... want to generalize to two-stage stochastic problem

Greedy heuristic

- Use ALNS for each scenario $s \in S$ (in parallel)
- Consensus-fixing to agree on first-stage variables

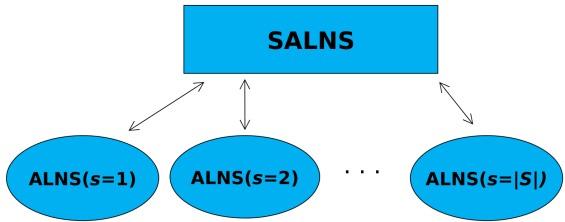
Stochastic ALNS

- Destroy neighborhoods for first-stage variables
- Use ALNS for each scenario $s \in S$ (in parallel)
- Consensus-fixing to agree on first-stage variables



SALNS - cookbook

- Lower-level algorithm can be a MIP-model
- Lower-level algorithm can be greedy heuristic
- Lower-level algorithm can be any metaheuristic
- Objective function can be robust optimization $z_{\mathbb{R}}$ instead of expected value $z_{\mathbb{E}}$



Conclusion

- Easy to move from deterministic to two-stage stochastic model
- Scales well with number of scenarios
- Very good performance
- Highly parallel

Deep Generative Scenario Search (DGSS)

European Journal of Operational Research 335 (2026) 1008–1038

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/eor

Interfaces with Other Disciplines

Deep generative scenario search for stochastic optimization

David Pisinger *, Atefeh Hemmati Golsefidi

DTU Management, Technical University of Denmark, 2800 Kgs.Lyngby, Denmark

ARTICLE INFO

Dataset link: 10.11583/DTU.32414469

Keywords:
Combinatorial optimization
Stochastic optimization
Deep generative models
Scenario search

ABSTRACT

We study two-stage stochastic optimization problems (SOP) where the task is to make first-stage decisions based on a finite set of future scenarios, denoted by the empirical set. The goal is to minimize the expected cost of the first and future second-stage decisions. Using deep generative models that are trained to replicate empirical scenarios, we propose a local search algorithm that intelligently explores the scenario space, repeatedly solving smaller optimization problems. The goal is to use a few synthetic scenarios to reach a heuristic solution close to what could be found by solving the SOP on the complete set of empirical scenarios. The framework is tested on three NP-hard two-stage SOPs: A capacitated facility location problem, a prize-collecting vehicle routing problem, and an EVs' charging station location problem.

Computational experiments comparing the local search method with 22 sampling-based methods and 3 deep learning methods show that the local search consistently finds high-quality first-stage solutions when evaluated on the full set of empirical scenarios. Furthermore, it generally evades infeasible second-stage solutions and has a modest variation in the solution quality. Since it is an iterative method, the local search returns a whole trajectory of candidate solutions that can be further assessed with respect to various soft criteria.

The algorithm not only finds optimal first-stage decisions, but also identifies a small set of realistic scenarios that justify the choices. We call this set a *certificate* for the found solution, and it may improve explainability.

Deep Generative Scenario Search (DGSS) for Two-Stage Stochastic Optimization

Atefeh Hemmati Golsefidi & David Pisinger

Funded by ERC-AdG, DECIDE project

Motivation: Decision-Making Under Uncertainty

Two-Stage Stochastic Optimization

$$\min \; \boldsymbol{cx} + \boldsymbol{\Sigma_s \pi_s q_s y_s}$$

$$\text{s.t. } x \in X, \; y_s \in Y_s(x) \; \forall s \in S$$

$\boldsymbol{x}$	First-stage decisions (made now)
$\boldsymbol{y_s}$	Second-stage decisions (per scenario)
$\boldsymbol{\lvert S \rvert}$	Empirical scenario set — large!
$\boldsymbol{\pi_s}$	Probability of scenario s

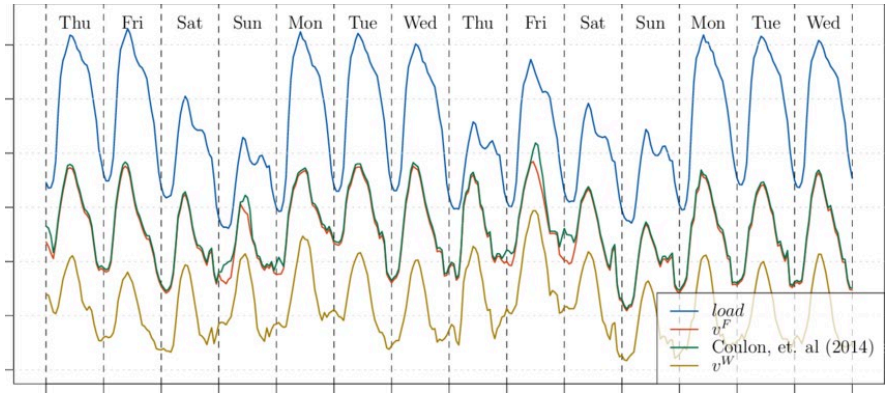
Core Challenge

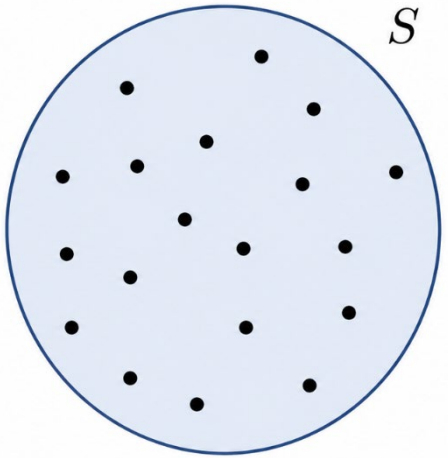
Represent **uncertainty** accurately,

without making the **optimization model** too **large**.

- Capture risk and variability
- Balance accuracy and tractability
- Enable scalable decision-making

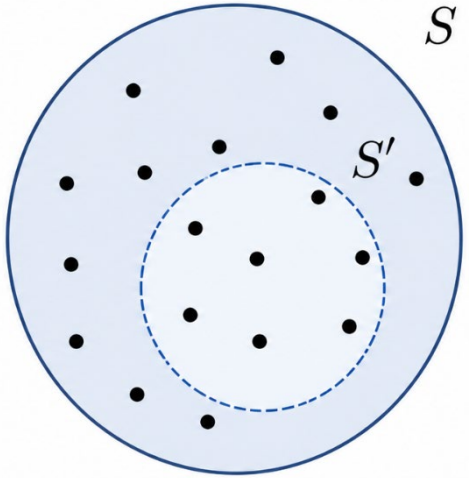
Scenario reduction





SOP

First-stage decisions



SOP

First-stage decisions

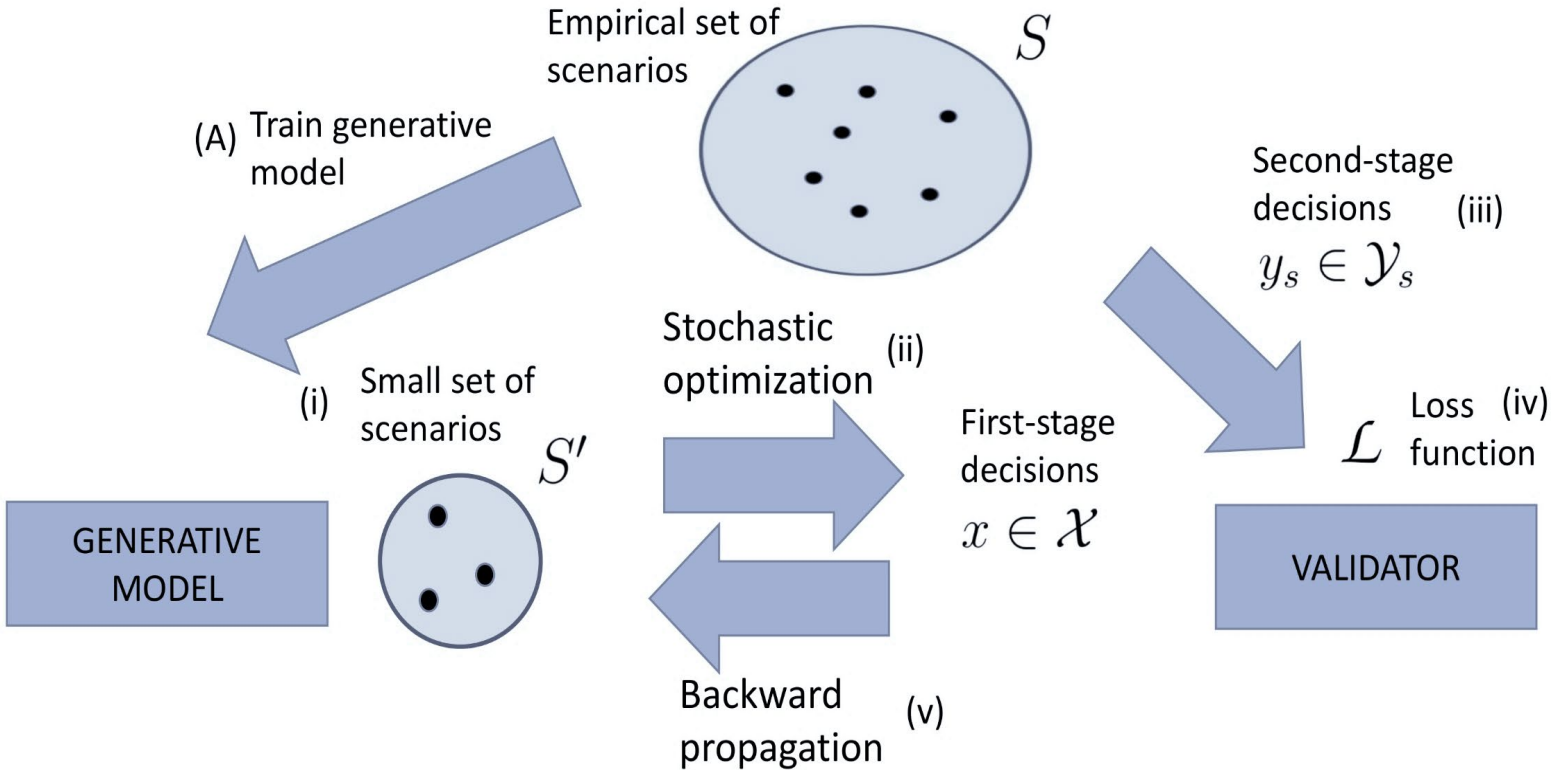
Find a small subset of scenarios to reduce numerical complexity while keeping the error at an acceptable level

Research question(s)

1. Can we generate a small set of synthetic scenarios that leads to high-quality first-stage decisions?

2. Can a deep generative model be used as a local search algorithm in scenario space?

DGSS: Deep Generative Scenario Search — Overview



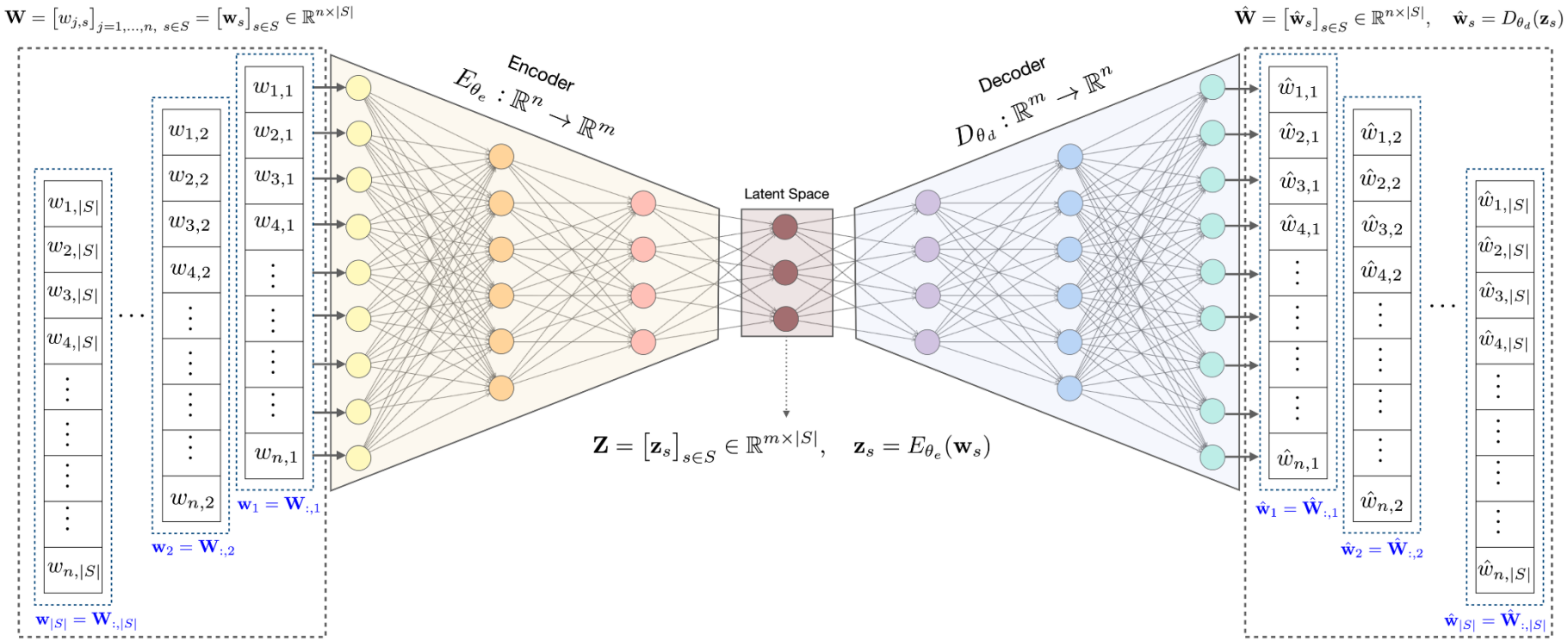
Objective-driven feedback loop

The generator receives feedback from the true full-set evaluation, not only from reconstruction quality.

Realism–cost trade-off

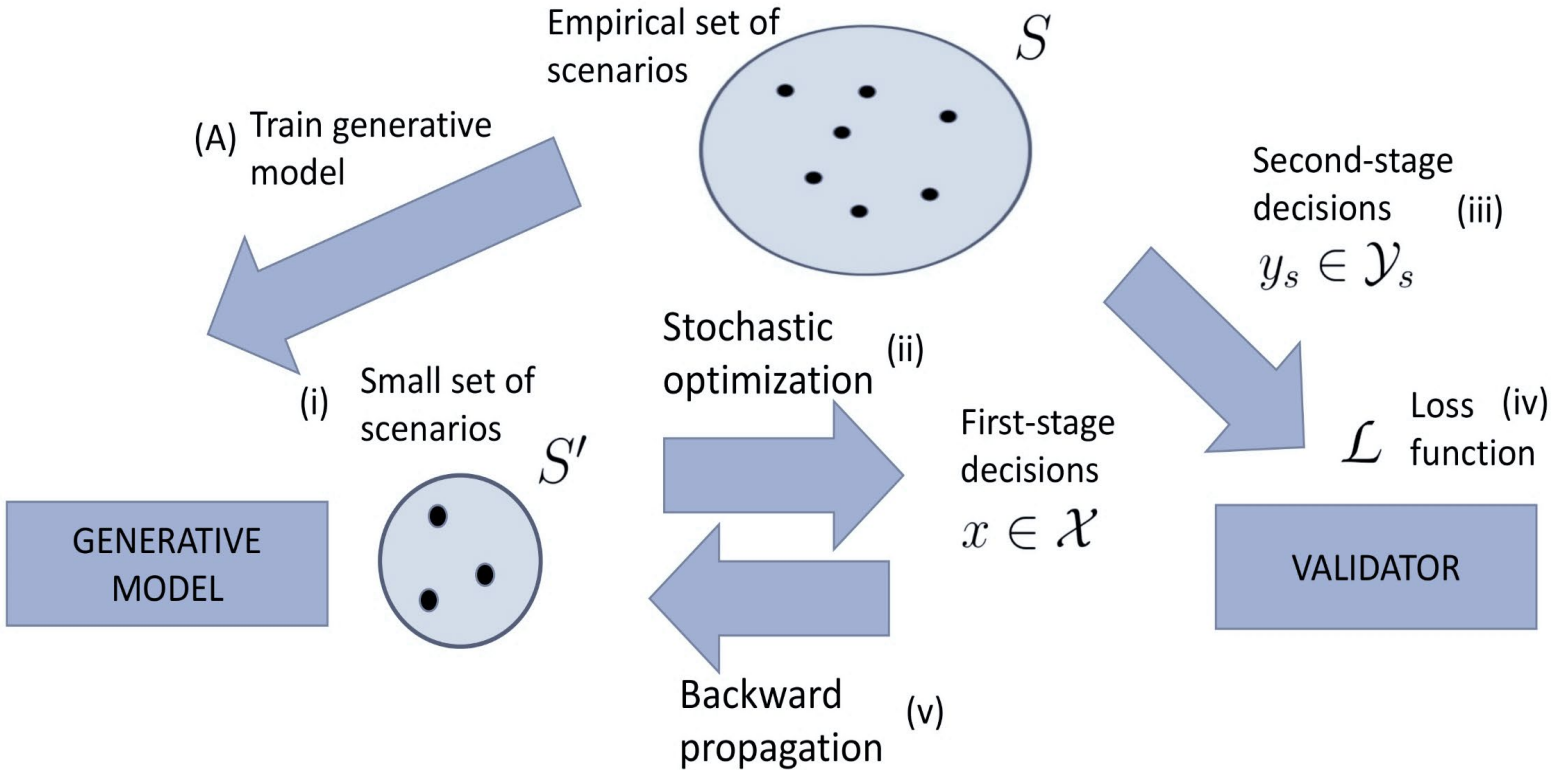
Keep generated scenarios realistic with reconstruction loss while guiding them toward decisions that minimize recourse cost.

Step A: reconstruct the empirical scenario distribution



- Learn **empirical patterns** in the data
- **Pre-train the generator** before optimization feedback
- Keep generated scenarios **realistic and data-grounded**

DGSS: Deep Generative Scenario Search — Step B



Loss function

$$\mathcal{L} = \gamma_1 \mathcal{L}_{MSE} + \gamma_2 \mathcal{L}_{OBJ}$$

SPSA gradient

Two-point finite difference: perturb Z by $\pm \varepsilon\Delta$, $\Delta \in \{-1, +1\}$; only 2 extra SOP solves per step

Step B: Objective-Guided Local Search via SPSA

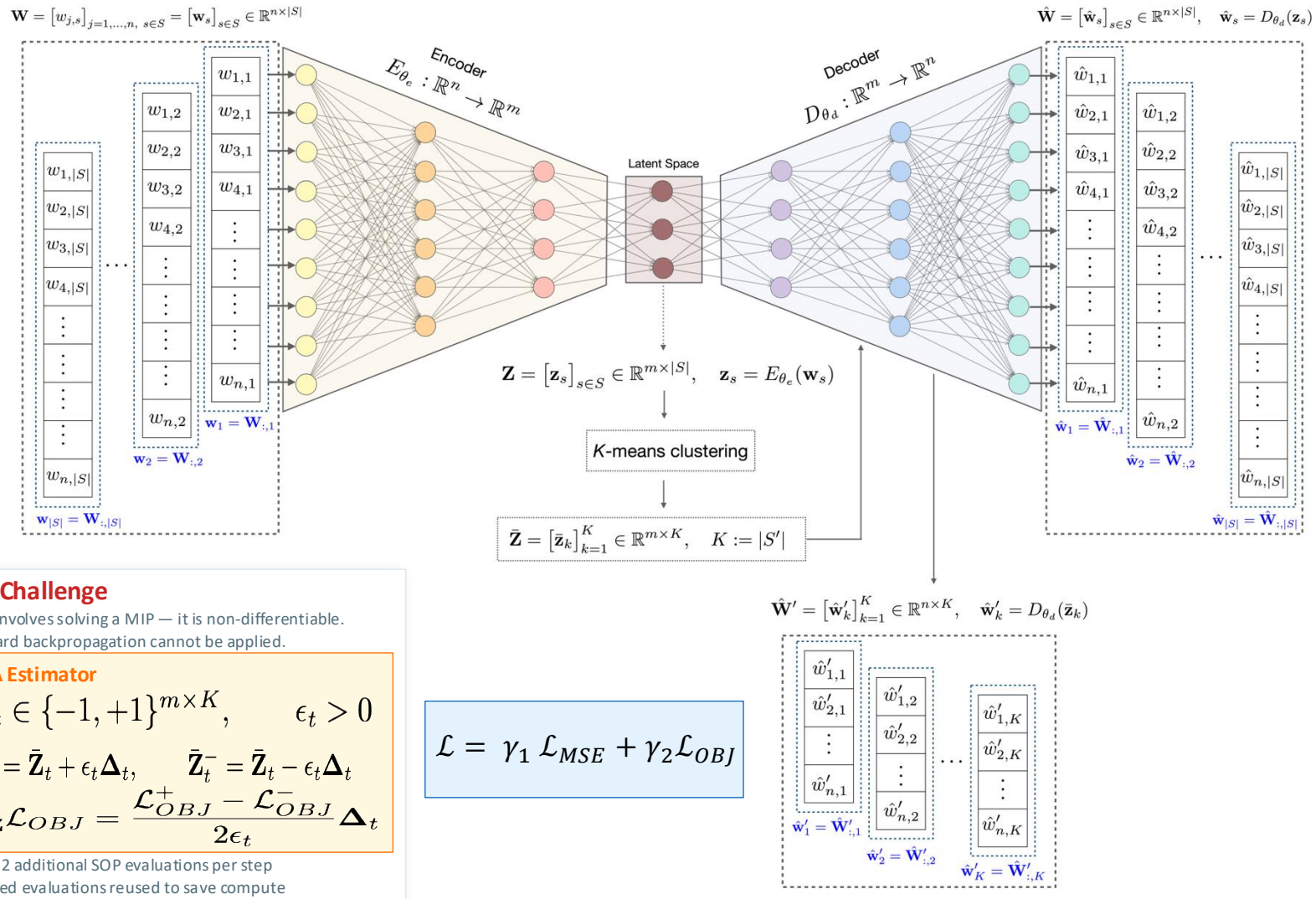
(i) Generate S′
Encode all $s \in S$; apply K-means on Z to get K centroids; decode → synthetic candidate set S'

(ii) Solve SOP(S′)
Solve the reduced stochastic program on S' → first-stage decision $x \in X$

(iii) Evaluate on full S
Fix x; solve recourse for every $s \in S$ → full-set evaluation cost $\mathcal{L}_{OBJ} = cx + \Sigma_s \pi_s \tilde{Q}_s(x)$

(iv) SPSA gradient
Two-point finite difference: perturb Z by $\pm \varepsilon\Delta, \Delta \in \{-1, +1\}$; only 2 extra SOP solves per step

(v) Update model
Backpropagate $\mathcal{L} = \mathcal{L}_{MSE} + \mathcal{L}_{OBJ}$ through encoder-decoder; cache evaluated solutions



The Challenge

$\mathcal{L}_{OBJ}$ involves solving a MIP — it is non-differentiable. Standard backpropagation cannot be applied.

SPSA Estimator

$$\boldsymbol{\Delta}_t \in \{-1, +1\}^{m\times K}, \qquad \epsilon_t > 0$$

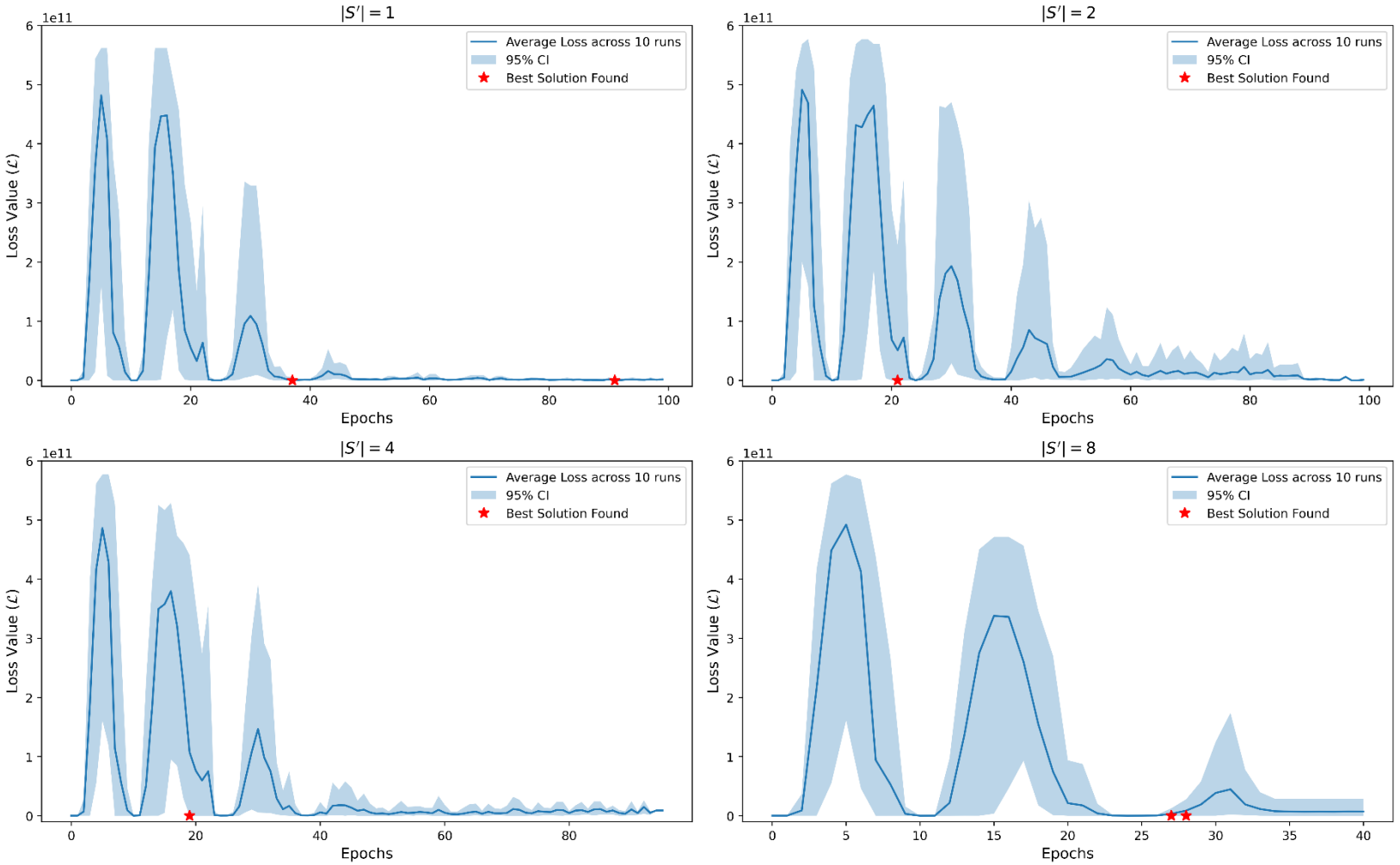
$$\bar{\mathbf{Z}}_t^+ = \bar{\mathbf{Z}}_t + \epsilon_t \boldsymbol{\Delta}_t, \qquad \bar{\mathbf{Z}}_t^- = \bar{\mathbf{Z}}_t - \epsilon_t \boldsymbol{\Delta}_t$$

$$\widehat{\nabla}_{\bar{\mathbf{Z}}} \mathcal{L}_{OBJ} = \frac{\mathcal{L}_{OBJ}^+ - \mathcal{L}_{OBJ}^-}{2\epsilon_t} \boldsymbol{\Delta}_t$$

- Only 2 additional SOP evaluations per step
- Cached evaluations reused to save compute
- Enables backprop into the autoencoder

$$\mathcal{L} = \gamma_1 \mathcal{L}_{MSE} + \gamma_2 \mathcal{L}_{OBJ}$$

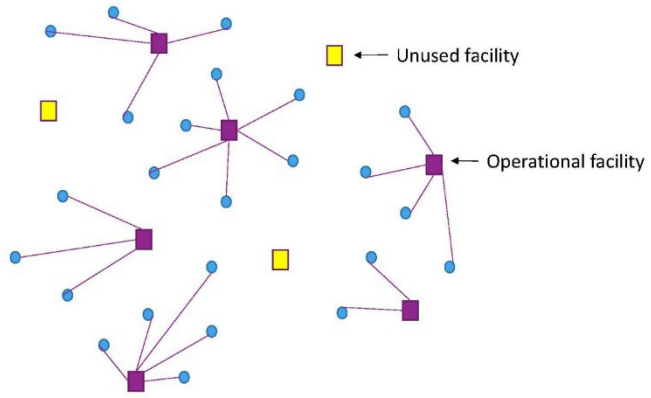
Convergence of local search algorithm



Three NP-Hard Test Problems

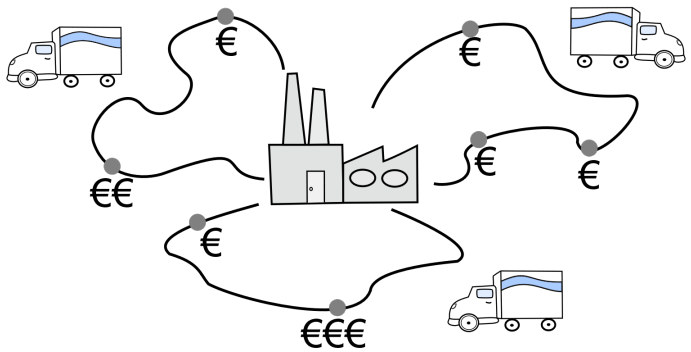
CFLP
Stochastic Capacitated Facility Location

- Open facilities to serve customer demand
- Each facility: opening cost & capacity
- Uncertain: customer preferences & demand
- Data and formulation from Wu et al. (2022, 2025)



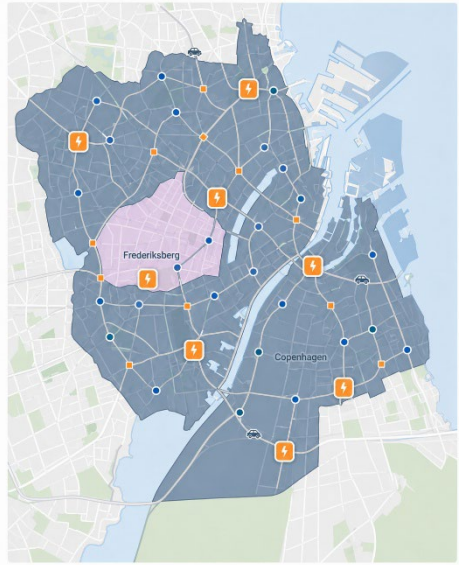
SPVRP
Stochastic Prize-Collecting Vehicle Routing

- Route vehicles to collect prizes
- Subscription (first-stage) & on-demand (second-stage) customers
- 70% subscribers, 30% on-demand
- Kaggle Drone Delivery data, 50 nodes
- Formulation form Pisinger (2026).



CSLP
Stochastic EV Charging Station Location

- Place EV chargers to serve demand
- 15,000 EVs/day in Copenhagen area
- Data and formulation from Golsefidi et.al. (2025)
- 200 simulations resulting in 200 scenarios
- 400 locations clustered into 50 nodes (k-means)



All problems: 20 test instances, 200 empirical scenarios · MIP-solved with Gurobi · K = 1, 2, 4, 8 candidate scenarios · 5 seeds per instance

Exact solution

- 200 base scenarios
- Beyond limits of Gurobi
- Time-limit 3600 seconds

Comprehensive Comparison: 25 Baseline Methods

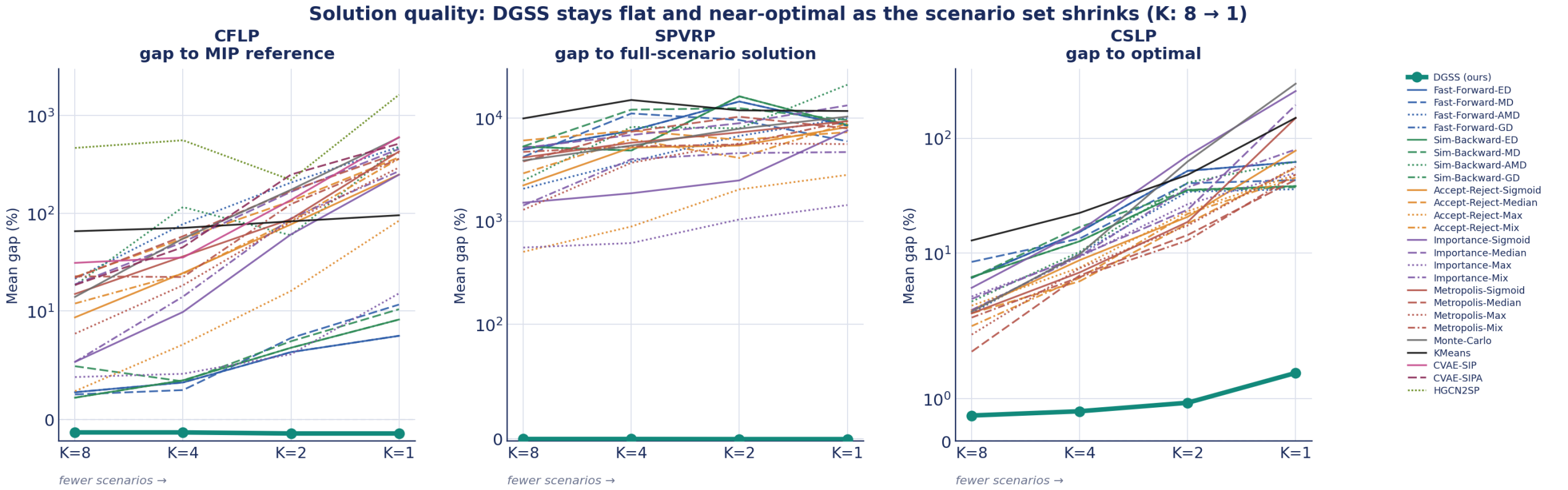
Sampling-Based (22 methods)

- Fast Forward Selection — 4 distance variants (Heitsch & Römisch 2003)
- Simultaneous Backward Selection — 4 distance variants
- Importance Sampling — 4 target distribution variants
- Metropolis-Hastings — 4 target distribution variants
- Accept-Reject — 4 target distribution variants
- K-Means, Monte Carlo

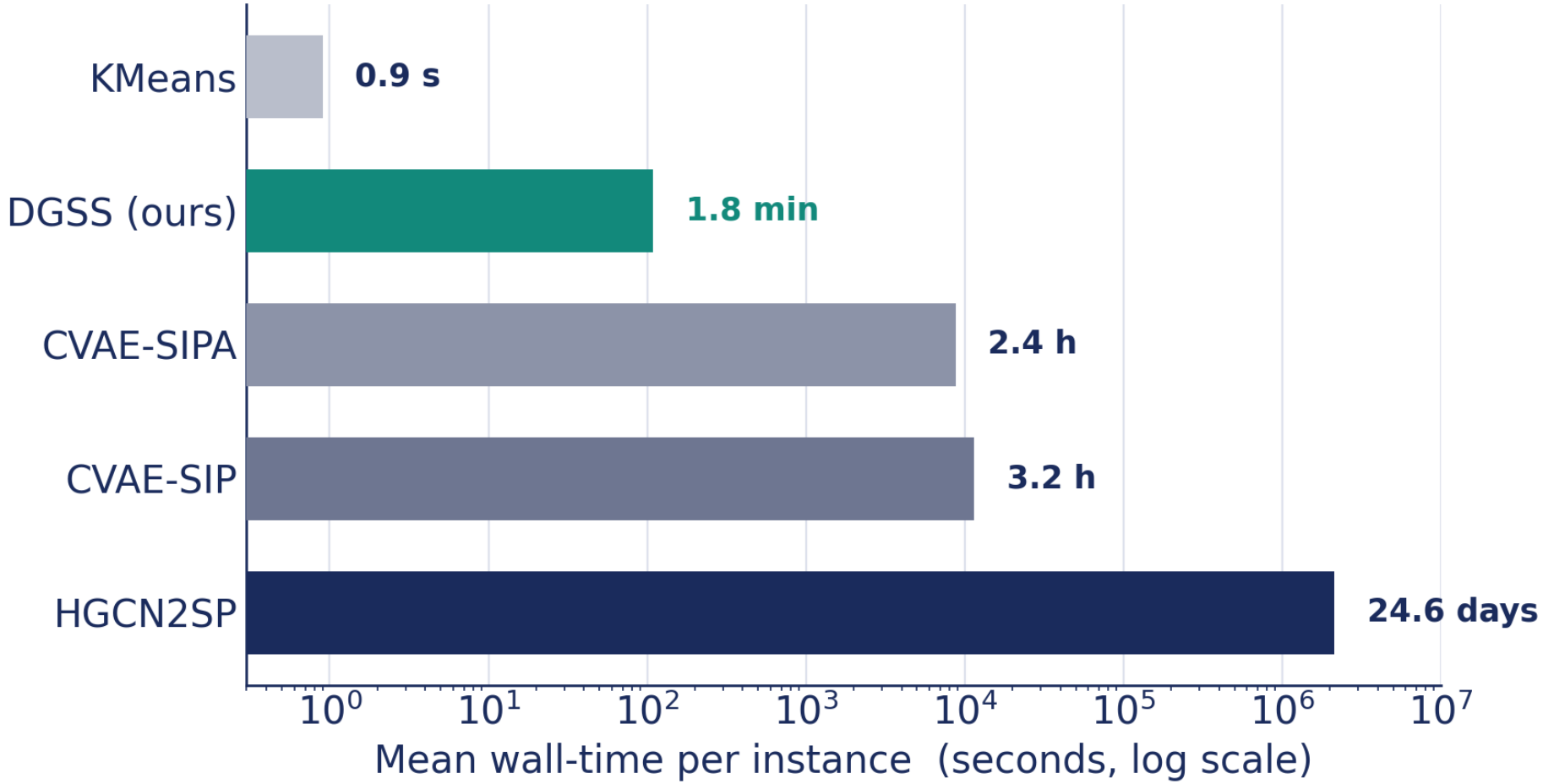
Deep Learning-Based (3 methods)

- CVAE-SIP (Wu et al. 2022): conditional VAE trained offline on 200 instances
- CVAE-SIPA (Wu et al. 2022): adds semi-supervised objective prediction
- HGCN2SP (Wu et al. 2025): hierarchical GCN policy via RL + 3hr/instance solver labels

Results: Solution quality — all methods, all three problems



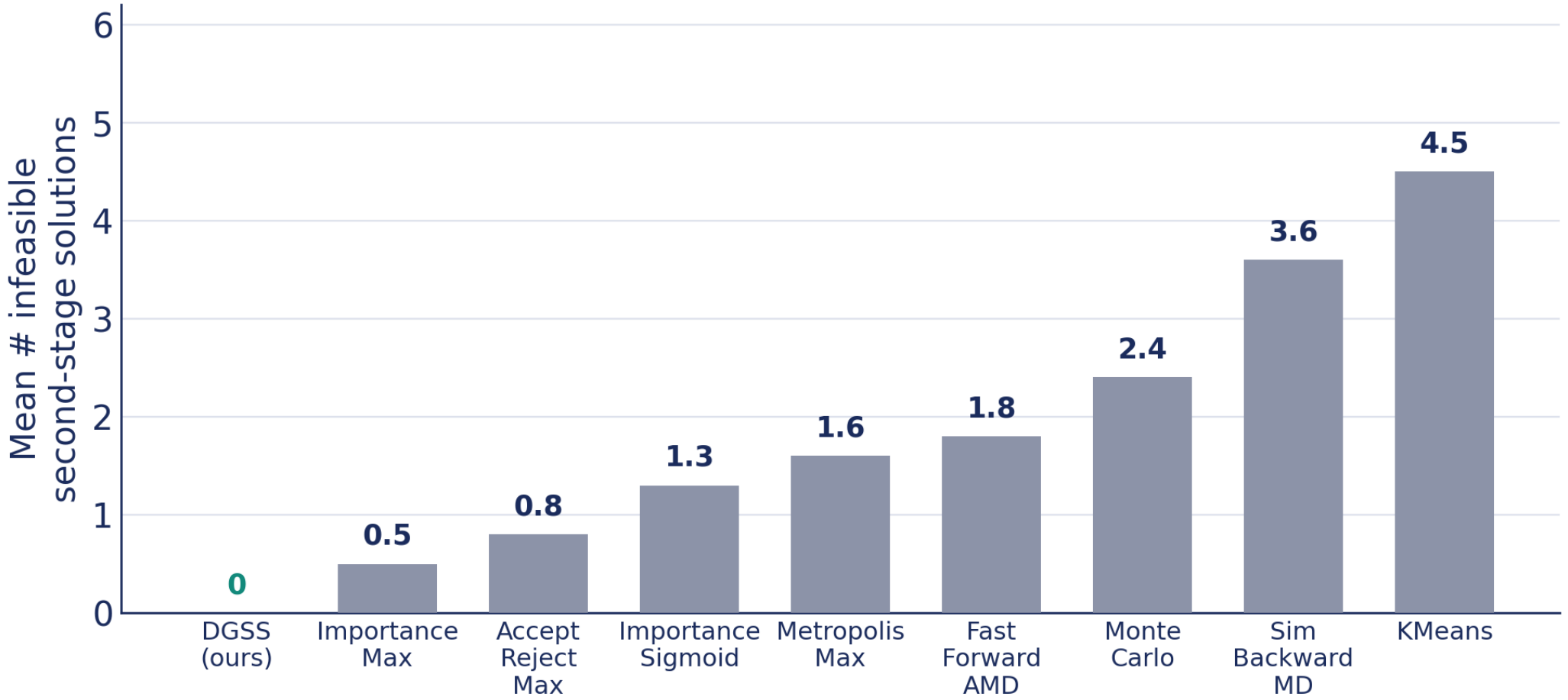
Results: Runtime



Results: Feasibility — SPVRP

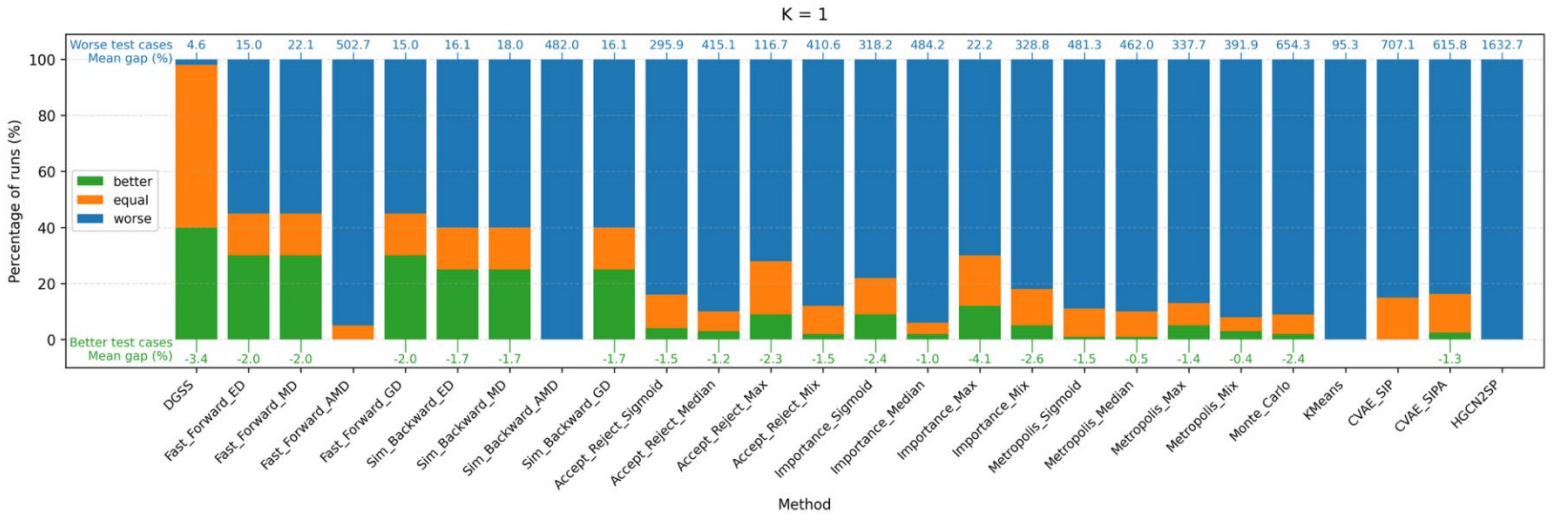
DGSS also stays near-optimal (gap ≈ −0.3%) and is feasible in all 100 runs

DGSS is the only method with zero infeasible solutions

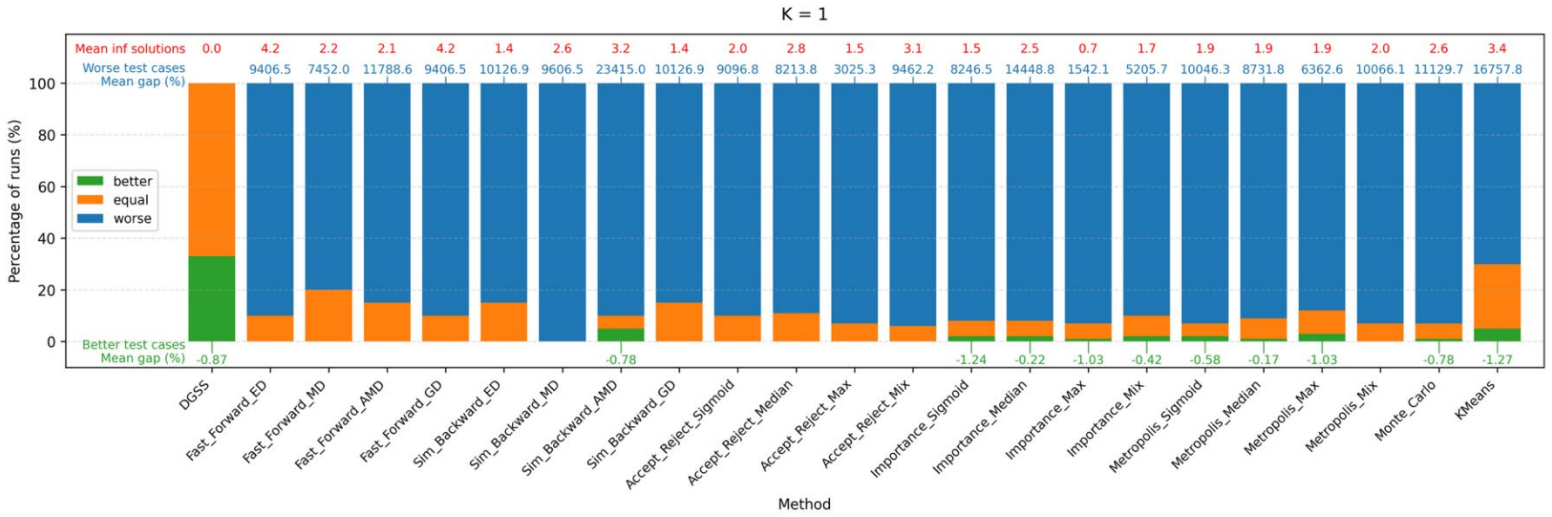


Results: Robustness — better / equal / worse than the reference

CFLP



SPVRP



Conclusion

What DGSS offers

Instance-Wise

Solves the specific SOP at hand, without pre-solved instances or offline training.

Black-Box Compatible

Works with any two-stage problem, including simulation-based or non-differentiable settings.

Iterative & Adaptive

Refines progressively synthetic scenarios through a multi-step local search. Returns a full trajectory of candidates for post-hoc analysis.

Explainable

Returns a compact and realistic certificate scenario S* that explains the selected first-stage solution x*.

Distribution-Free

Uses empirical scenarios directly, without assuming a parametric probability distribution.

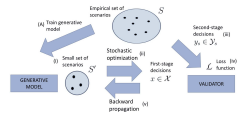
Infeasibility-Aware

Evaluates candidates on the full scenario set and penalizes infeasible second-stage solutions. Nearly zero infeasibility rate.

Two solution methods - summing up

Stochastic Adaptive Large Neighborhood Search (SALNS)
Chaotic system, cannot learn pattern in scenarios
Low data quality (generative model will learn errors)
Rare but important scenarios cannot be ignored
Robust optimization

Deep Generative Scenario Search (DGSS)
Complex system, generative model learns pattern in scenarios
Improved explainability (small set of scenarios - certificate)



References

1. Chao, Golden Wasil, *The team orienteering problem*, European Journal of Operational Research, 88 464-474 (1996).
2. Ropke, Pisinger, *An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows.* Transportation Science 40, 4 (2006), 455–472.
3. Pisinger, *Consensus fixing for two-stage stochastic optimization problems.* Transportation Research part E, 210 104770 (2026).
4. Pisinger, *Stochastic ALNS for the two-stage prize-collecting vehicle routing problem.* Transportation Research Part C, 179 105293 (2025).
5. Pisinger, *Distributed versus Integrated Stochastic ALNS – two-stage stochastic team orienteering problem*, Computers & Operations Research, 188 107365 (2026).
6. Pisinger, Golsefidi, *Deep Generative Scenario Search for stochastic optimization*, European Journal of Operational Research, 335 1008-1038 (2026).