ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ
SCHOOL OF BUSINESS

ΤΜΗΜΑ ΔΙΟΙΚΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ & ΤΕΧΝΟΛΟΓΙΑΣ
DEPARTMENT OF MANAGEMENT SCIENCE & TECHNOLOGY

An adaptive memory matheuristic for the SOP

Dontas Michael, mdontas@aueb.gr
Sideris Georgios, geosideris@aueb.gr
Manousakis Eleftherios, lmanousakis@aueb.gr
Zachariadis Emmanouil, ezach@aueb.gr

Department of Management Science and Technology,
Athens University of Economics and Business

EURO 2022, Espoo Finland, 3 - 6 July

Presentation Outline

01

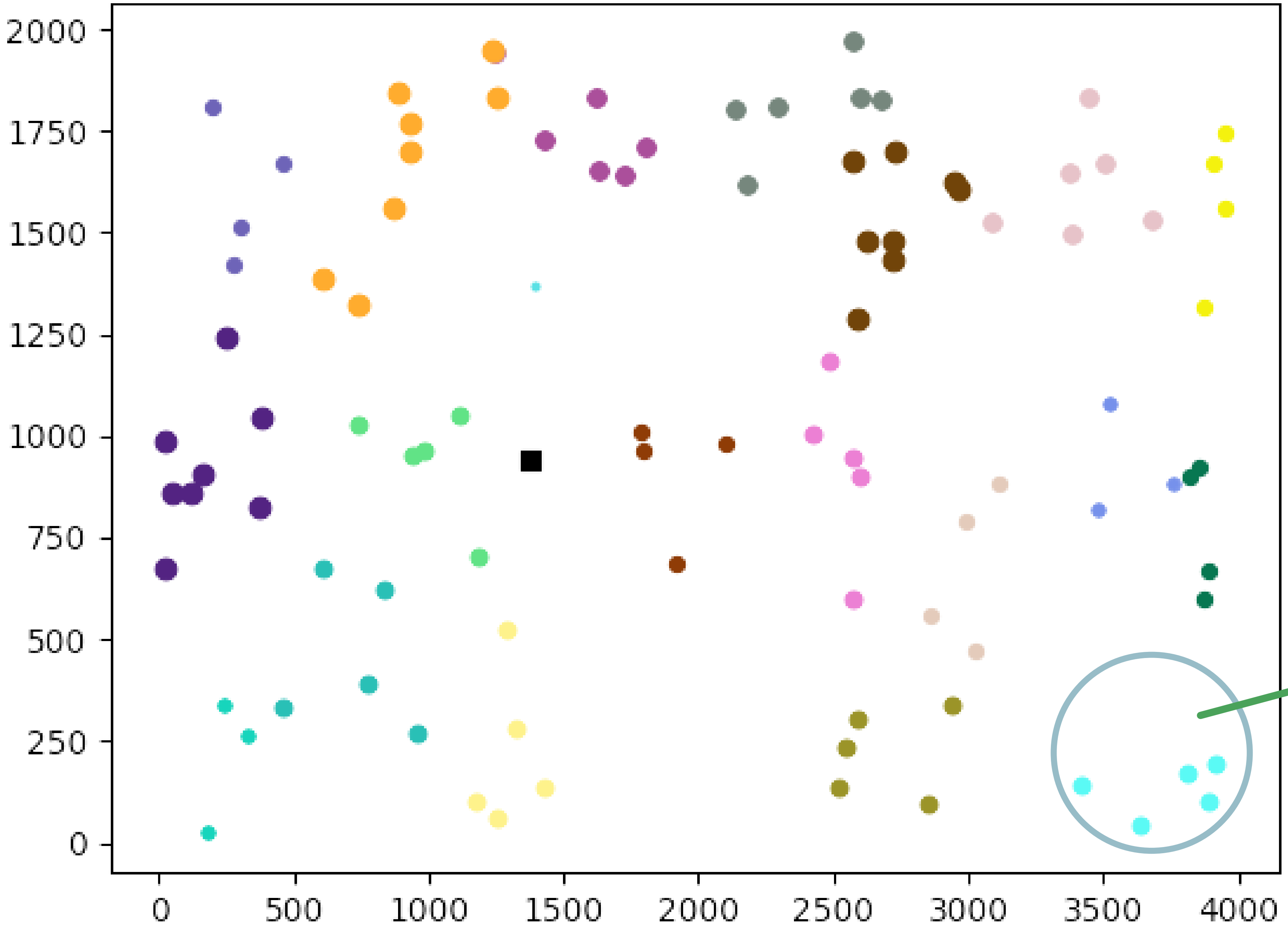
The Set Orienteering Problem

Problem Description

Objective: Maximize the profit of visited clusters

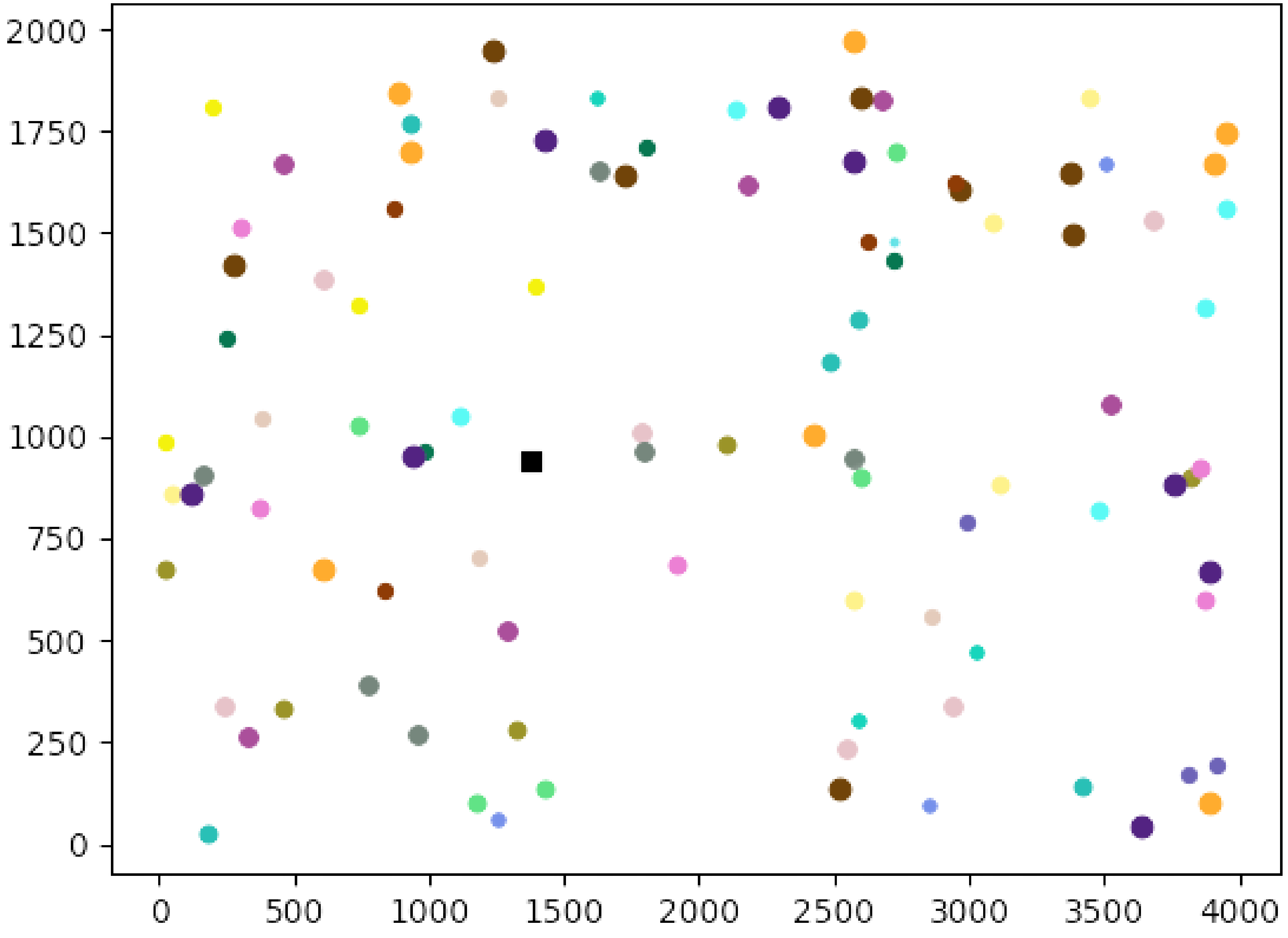
- **Single** vehicle routing.
- Each customer is associated with a **profit** value and is part of a set of customers (cluster).
- Route length cannot exceed a specified **time limit** (Tmax).
- It takes **only one customer** visit to collect the **total profit** associated with the **cluster**.
- Route must start from **depot** and return back to it.

Problem Visualization

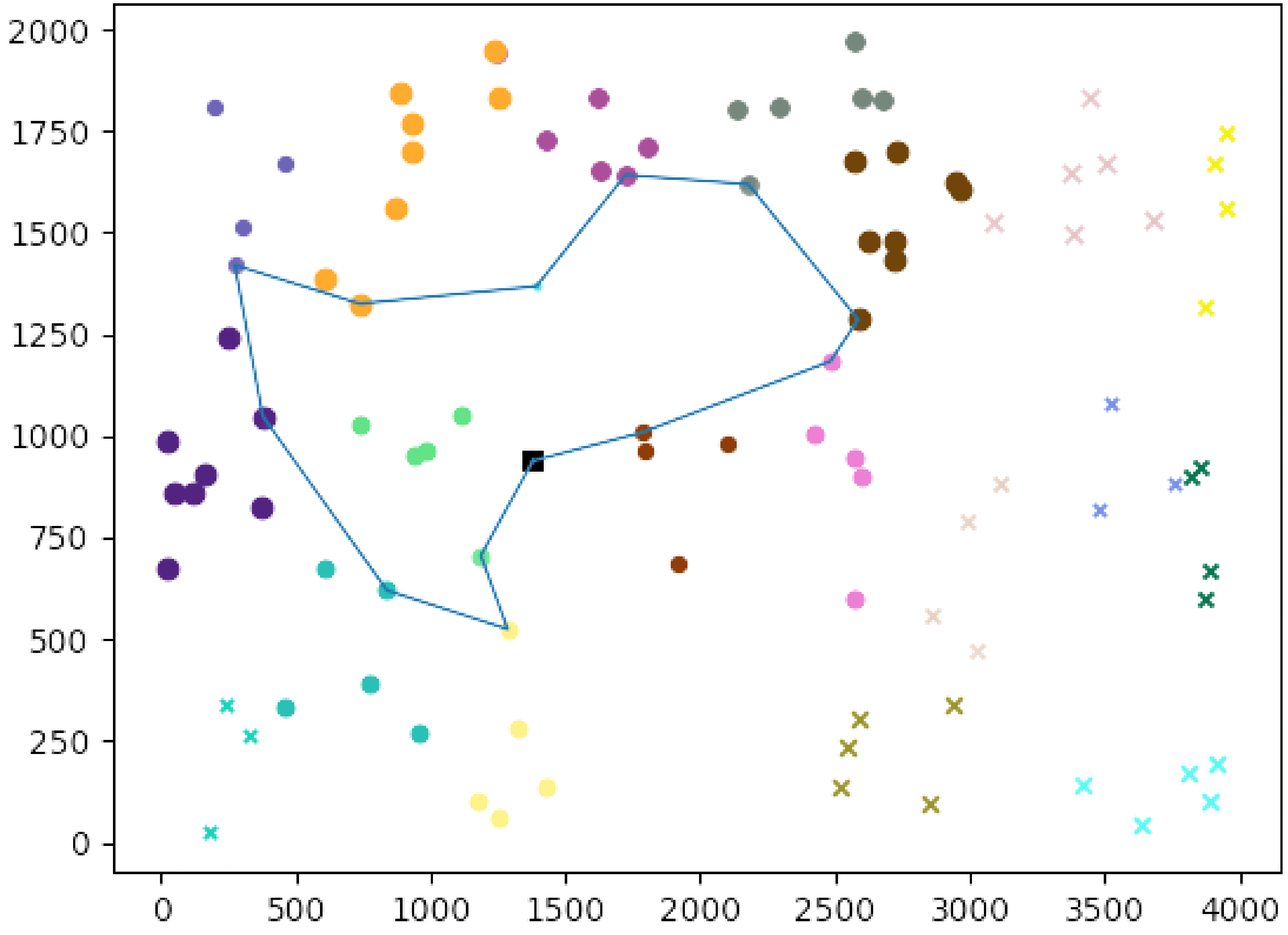


Customer cluster

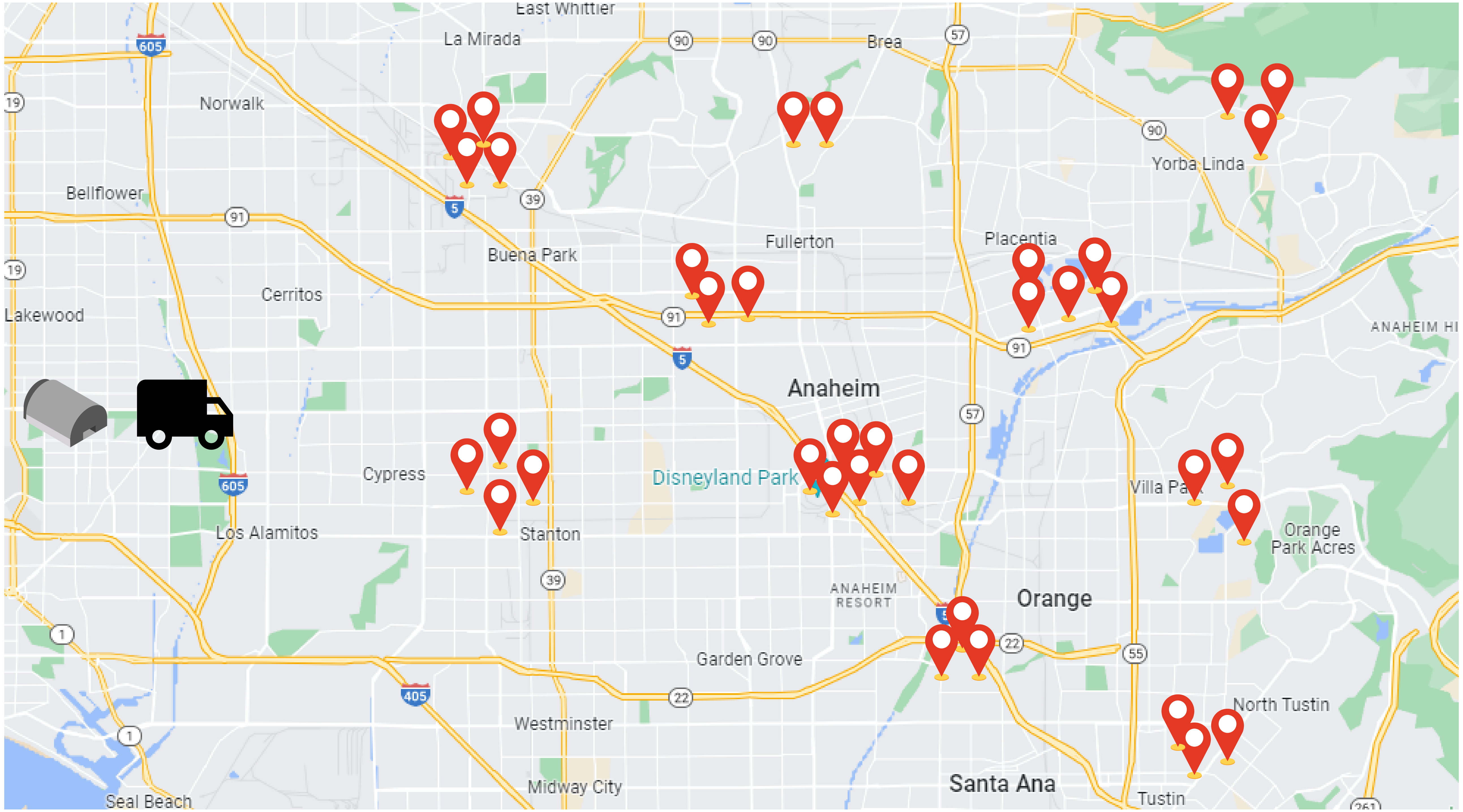
Visualization - Random Dataset



Feasible Solution



Practical Application



02

The AMMH Algorithm

Overall Scheme

Algorithm 1 *Overall Scheme*

$S \leftarrow ConstructInitialSol()$, $S^* \leftarrow \emptyset$ → Initial Solution Construction
for $r \leftarrow 1$ to $RESTARTS$ **do**
    $S \leftarrow LocalSearch(S)$ → Neighborhood Exploration
    if $Z(S) > Z(S^*)$ **then**
        $S^* \leftarrow S$
    end if
    $S \leftarrow SR\text{-}2()$ → Solution Reconstruction
end for
return S^*

Solution Construction Algorithm

Solution Construction

Algorithm 2 *ConstructInitialSol*

```
S_unvisited ← P, S ← [depot, depot]
do
    feasible_insertions ← []
    for set in S_unvisited do
        for node in set do
            for pos ← 1 to |S| − 1 do
                c ← CalculateCost(node, S, pos)
                if C(S) + c <= t_max then
                    feasible_insertions.Insert([node, set, pos, c])
                end if
            end for
        end for
    end for
    if |feasible_insertions| > 0 then
        OrderDescending(feasible_insertions, Z)
        selected_move ← SelectRandom(feasible_insertions, l)
        UpdateSolution(S, selected_move)
        S_unvisited.Remove(selected_move[1])
    end if
while |feasible_insertions| > 0
return S
```

Minimum Inertions logic

Criterion: added_profit/added_cost

Use of **Restricted Candidate List**

Neighborhood Exploration

01

Operators

02

Tabu Policy

03

Intensification procedure

Neighborhood Exploration

01

Operators

02

Tabu Policy

03

Intensification procedure

Local Search Operators

Outer Insertion

Introduce a node into the solution

In-Out Swap

Interchange between an outer and an inner node

Inner Relocation

Move an inner node to a new position in the solution.

Customer Removal

Remove a node from the solution

Selection criterion:

M*profit + cost

Neighborhood Exploration

01

Operators

02

Tabu Policy

03

Intensification procedure

Tabu policy - Promises

Promises-based policy is a
variation of the Tabu Search

Whenever a **set** is removed from the solution, a new promise is defined - equal to the solution's **objective**.

Restriction:
The set will be re-accessible to the operators **only** if current solution's objective is greater than the promise.

Promises are **reinitialized** after a certain number of iterations.

Neighborhood Exploration

01

Operators

02

Tabu Policy

03

Intensification procedure

Intensification Procedure

Basic Intuition

In depth exploration of high quality regions, by applying **exact** and **powerful heuristic** methods.

Aim

Improvement of elite or promising solutions.

Intensification Procedure - Outline

Customer Insertion-Deletion (CID)

Optimal insertions-deletions

Profit Increase

Shortest Path Problem

Optimal node selection given set sequence

TSP Heuristic

Optimal set sequence given node selection

Cost Reduction

Customer Insertion-Deletion (CID)

Parameters

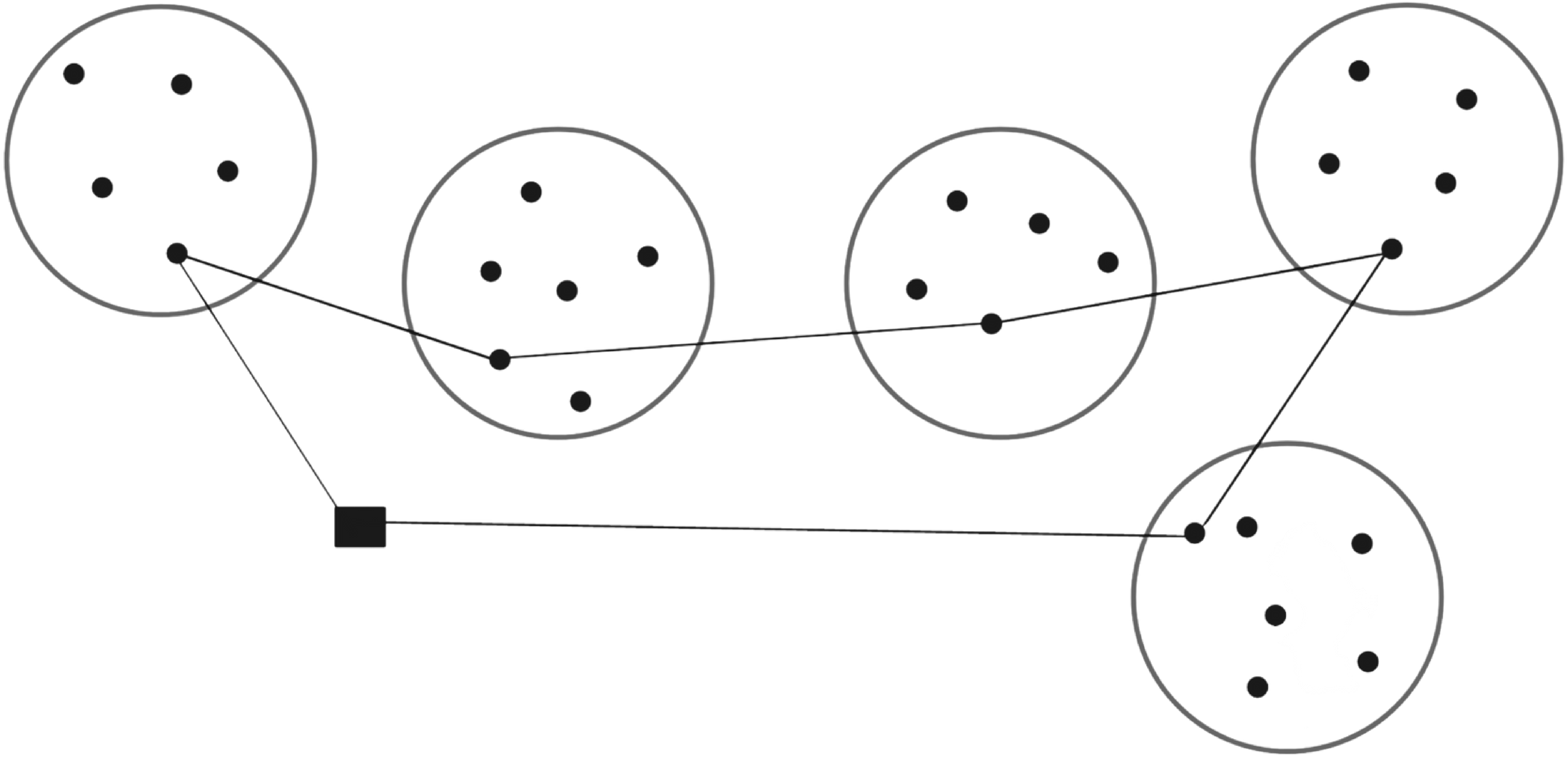
i: max number of insertions
d: max number of deletions

Matheuristic Method

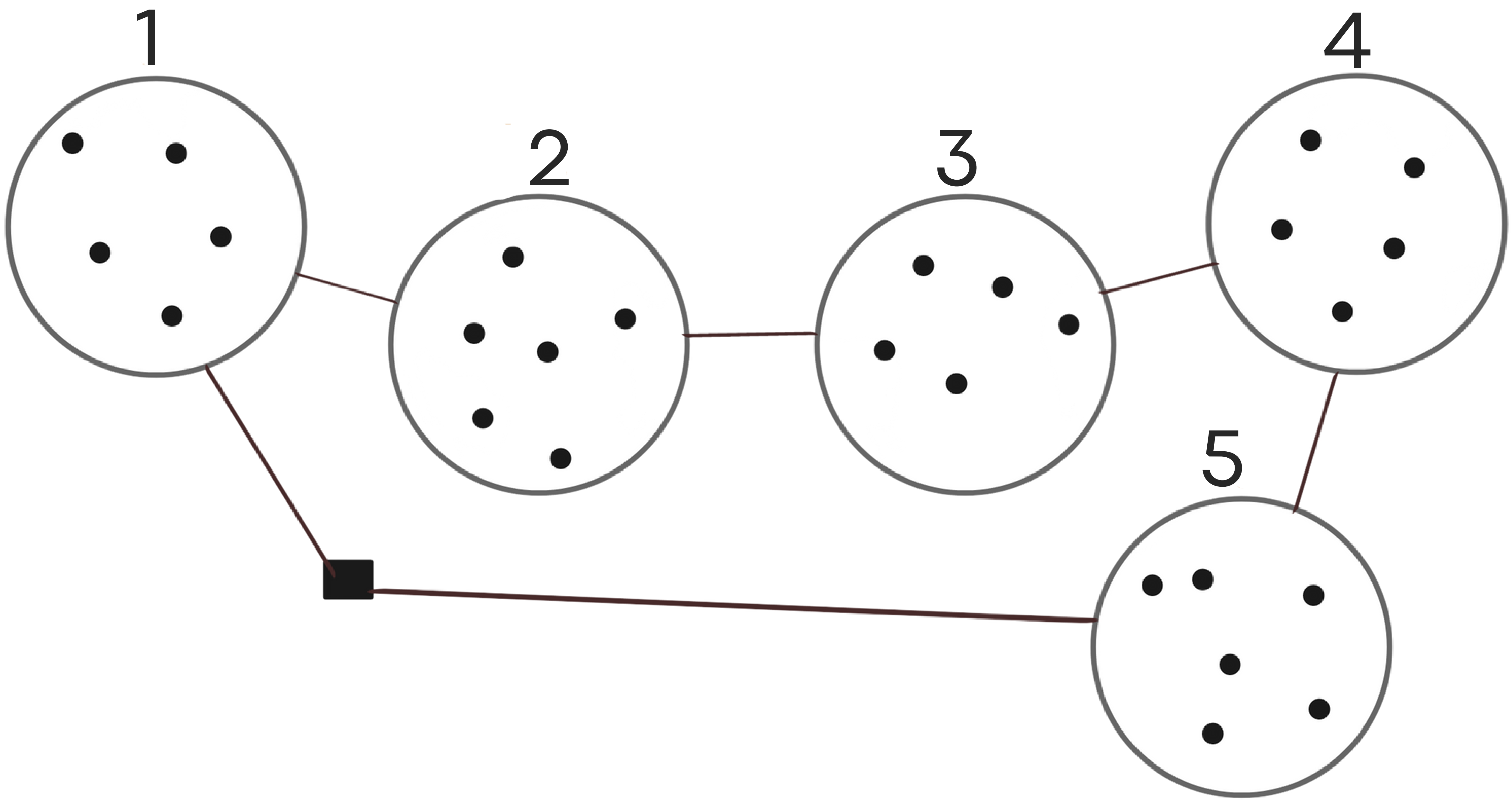
Process

Solve the problem of **simultaneous** insertions and deletions to **optimality**, aiming at **profit maximization**.

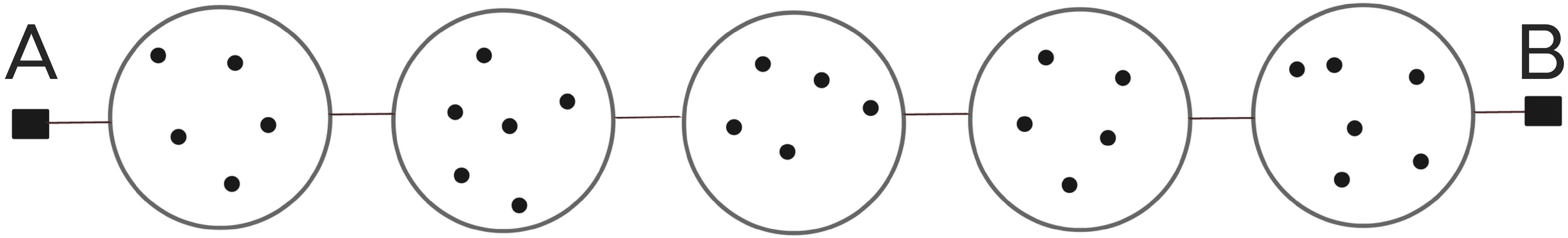
Shortest Path Problem



Shortest Path Problem



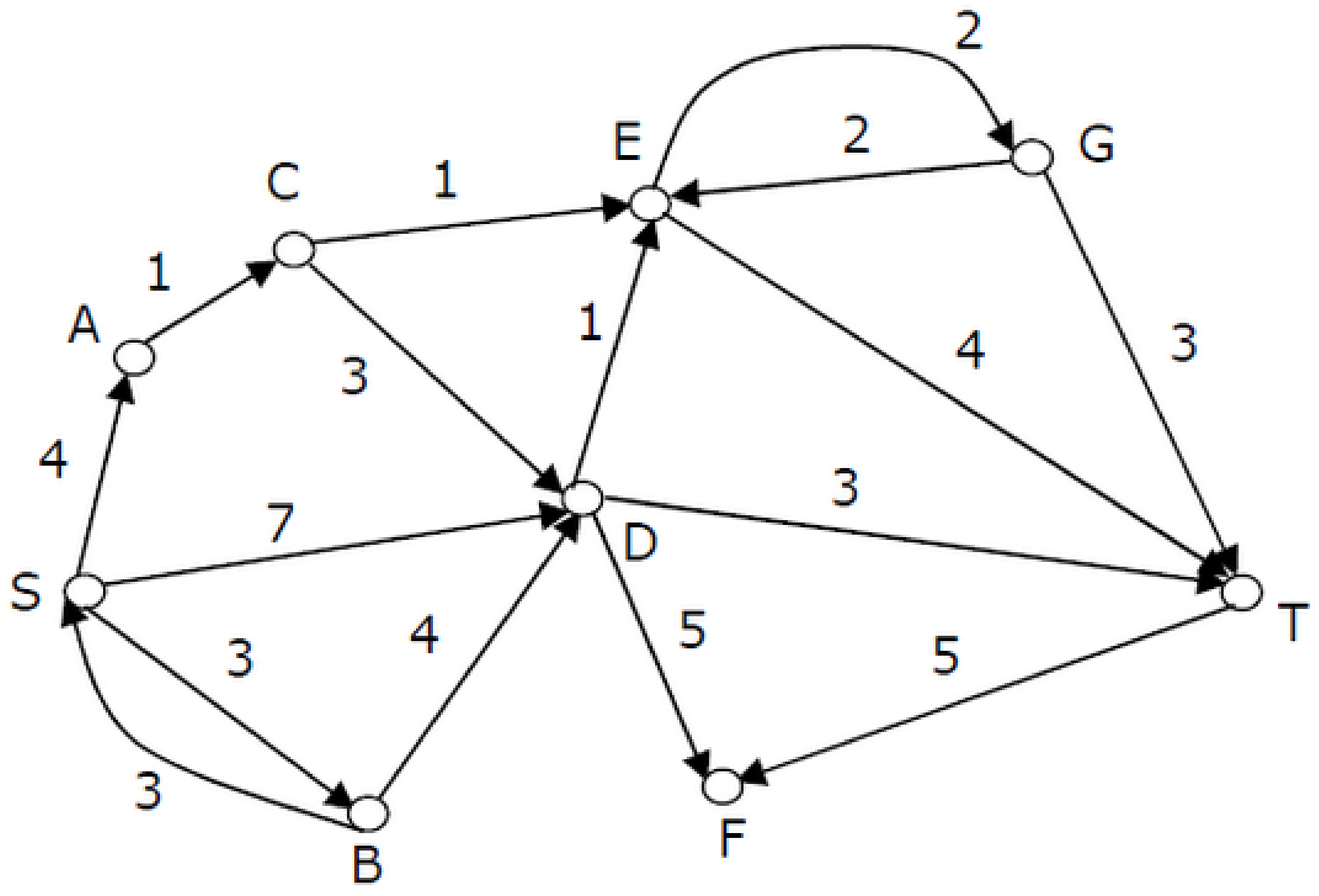
Shortest Path Problem



Shortest Path Problem

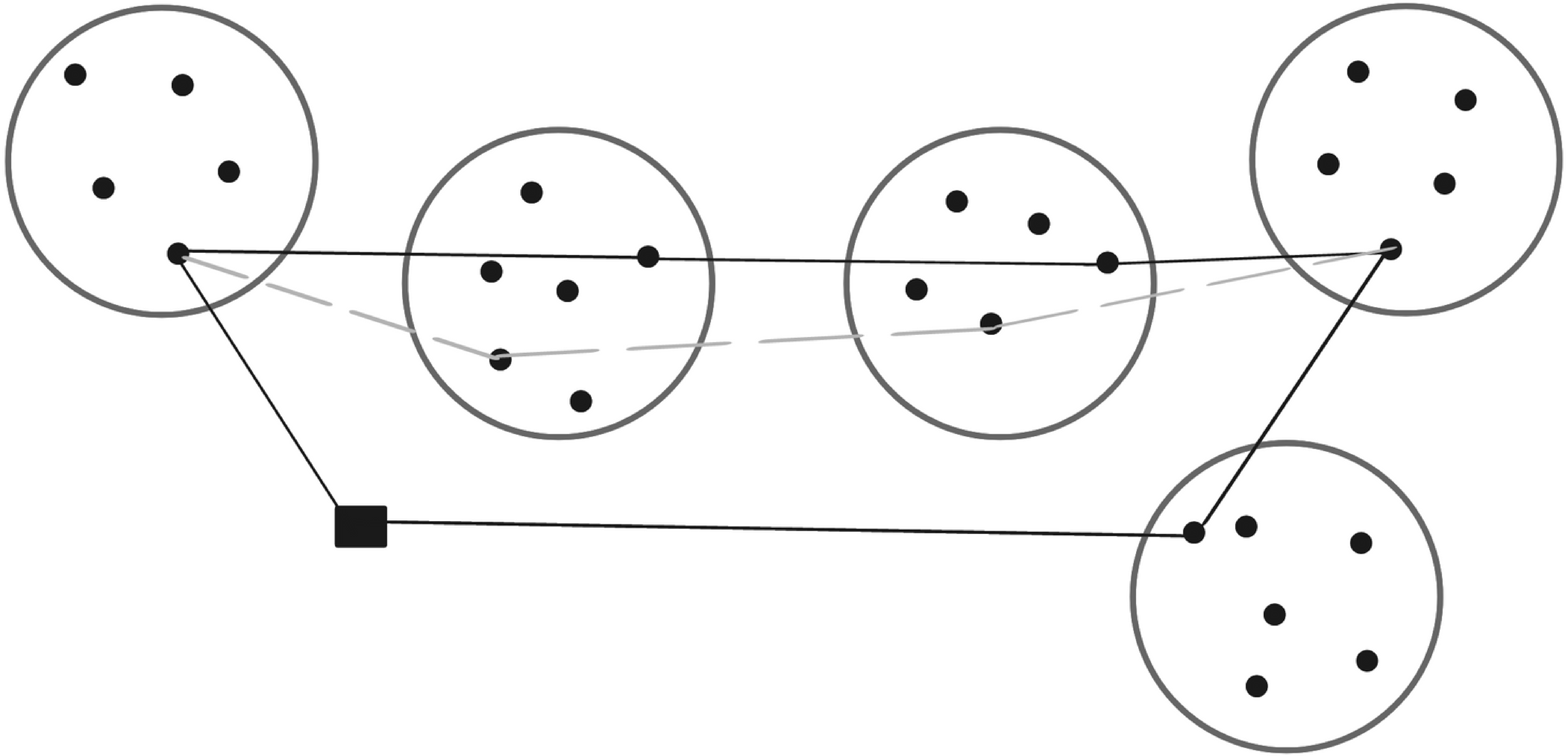
Suitable Algorithms

- **Bellman-Ford** - O(VE)
- **Dijkstra** - O(E+VlogV)
- **Directed Acyclic Graph Shortest Path** (using Topological Sorting) - O(V+E)

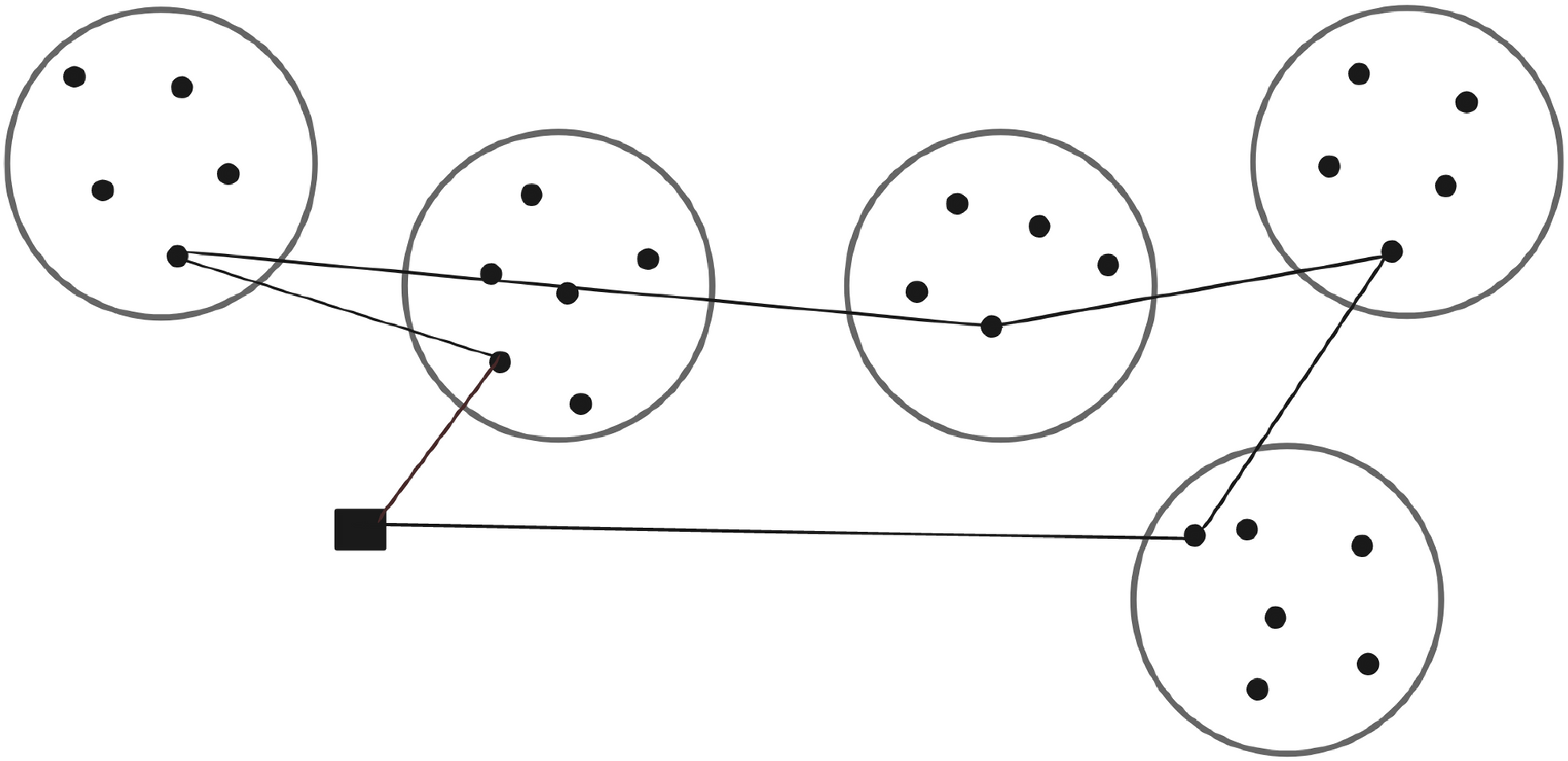


Shortest Path Problem

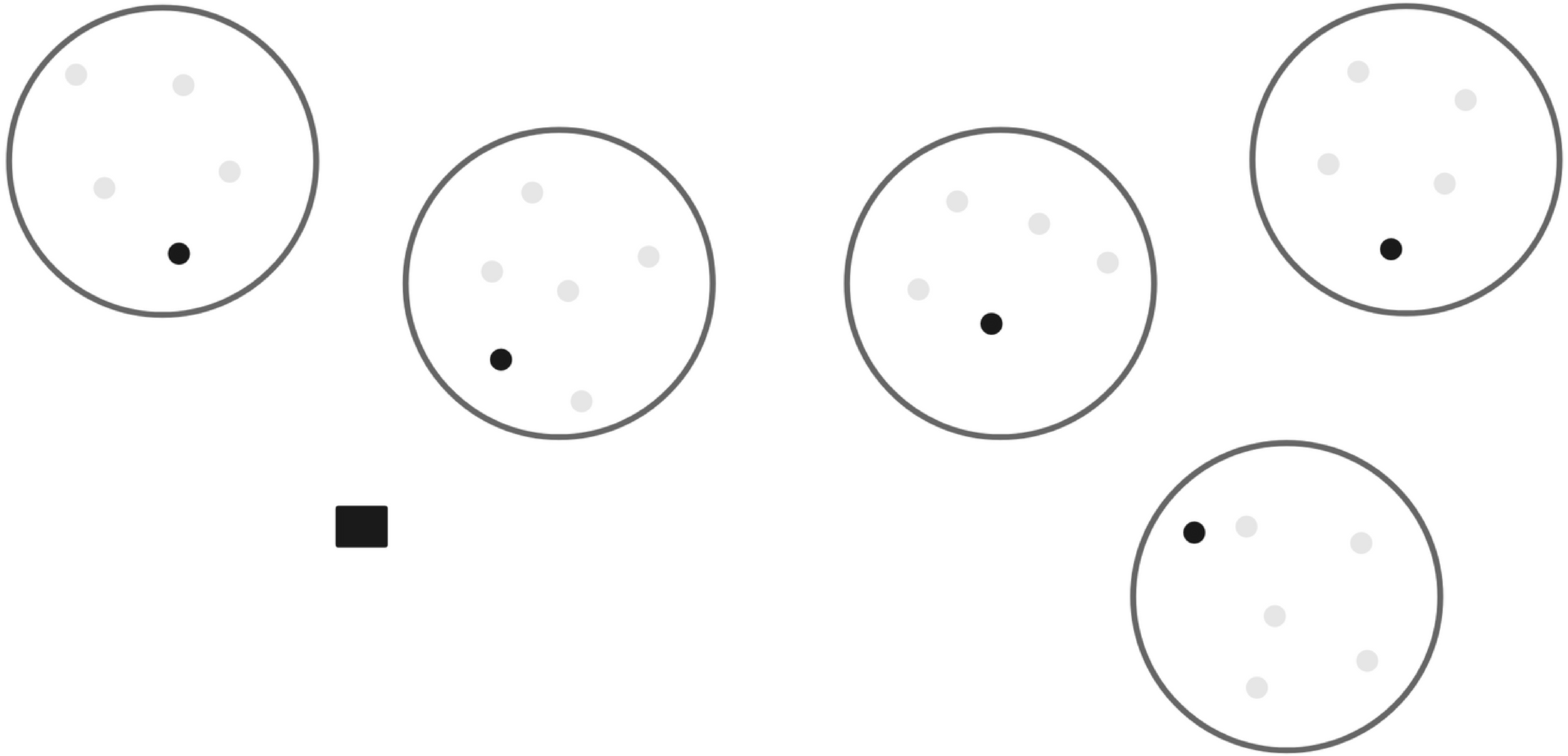
Enhanced Solution



Traveling Salesman Problem



Traveling Salesman Problem



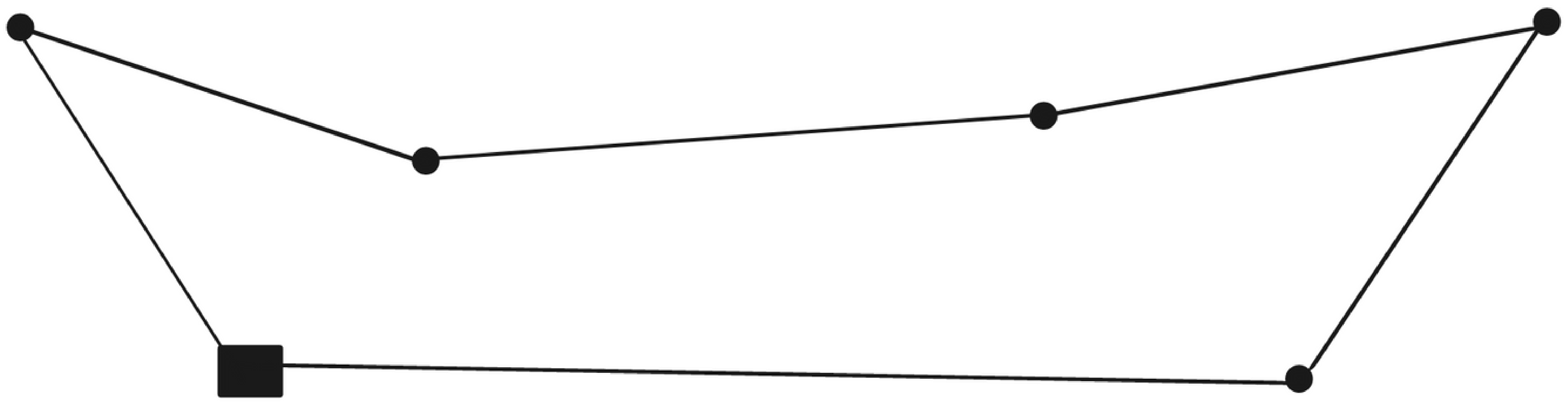
Traveling Salesman Problem

Traveling Salesman Problem

Heuristic: **Lin-Kernighan-Helsgaun Solver**

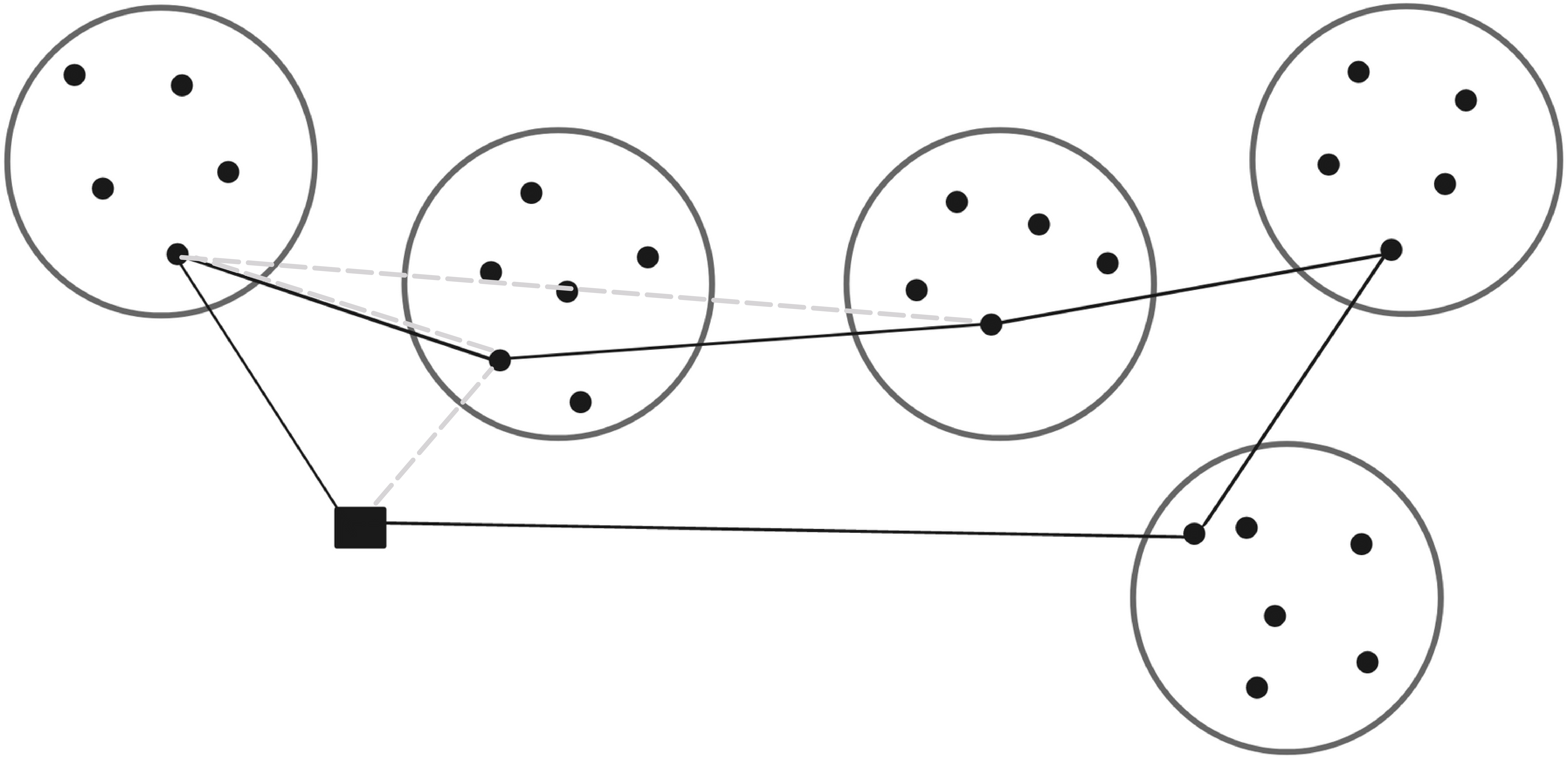
- Very effective heuristic capable of solving to **optimality** problems with up to a million nodes
- For the problem scale of our test cases - a few hundreds of nodes - LKH usually obtains the optimal solutions **very fast** (less than a CPU second)

TSP heuristic



TSP heuristic

Enhanced Solution



Intensification Procedure - Outline

Customer Insertion-Deletion (CID)

Optimal insertions-deletions

Shortest Path Problem

Optimal node selection given set sequence

TSP Heuristic

Optimal set sequence given node selection

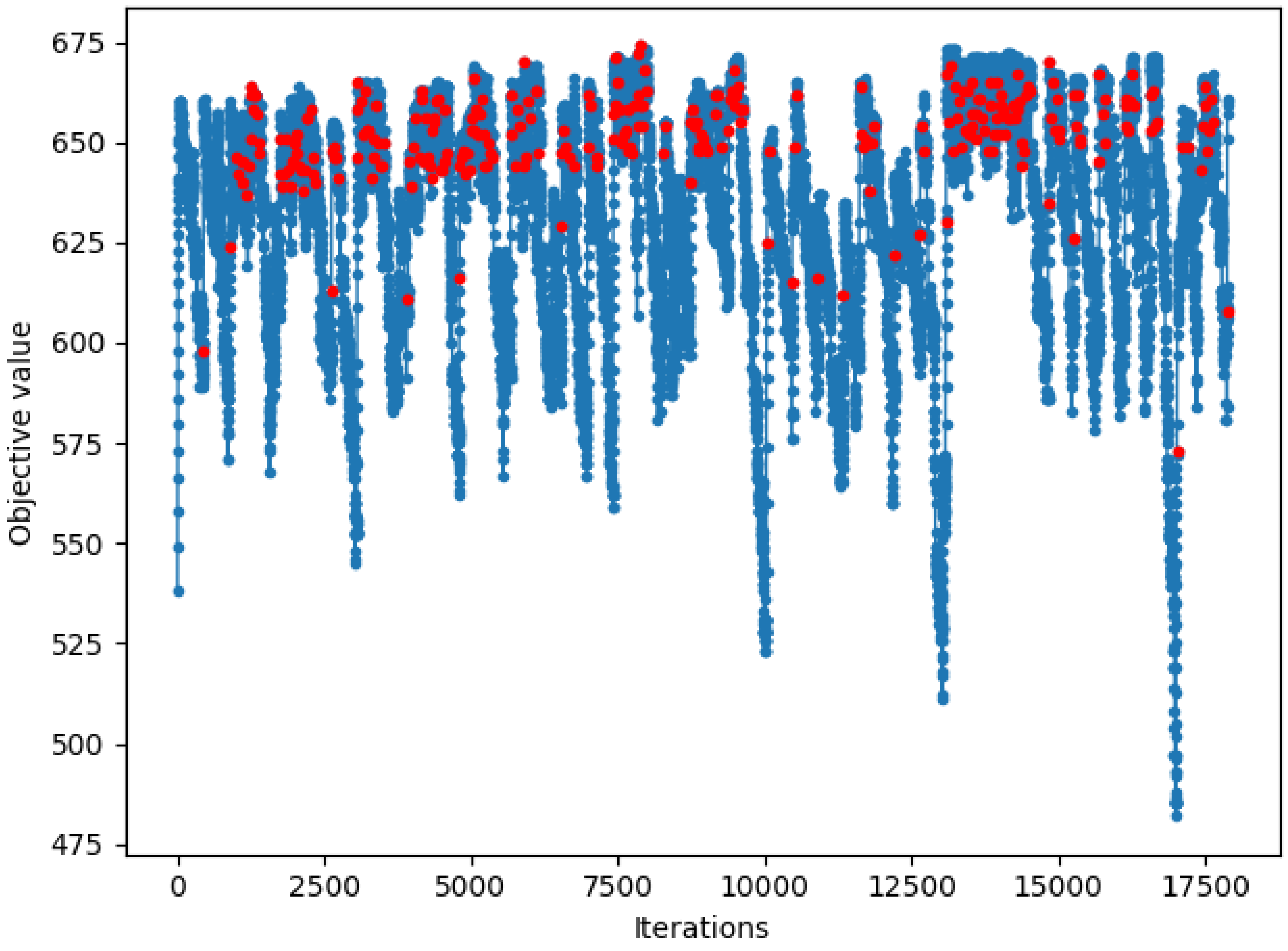
Profit Increase

Cost Reduction

Intensification Procedure

When is it executed?

- When a **new best** solution is found.
- When the **gap** between the current and the best solution is **smaller than a threshold g**.
- When **promises** are **reinitialized**.



Overall Scheme

Algorithm 1 *Overall Scheme*

$S \leftarrow ConstructInitialSol()$, $S^* \leftarrow \emptyset$ → Initial Solution Construction

for $r \leftarrow 1$ **to** $RESTARTS$ **do**

  $S \leftarrow LocalSearch(S)$ → Neighborhood Exploration

  if $Z(S) > Z(S^*)$ **then**

    $S^* \leftarrow S$

  end if

  $S \leftarrow SR\text{-}2()$ → Solution Reconstruction

end for

return S^*

Solution Reconstruction

Adaptive Logic

- The algorithm uses a **memory-based** structure that takes advantage of **previous Local Search states**.
- This structure is a **pool of high quality solutions**.
- These solutions are stored during the Local Search iterations and are used in the **Solution Reconstrucion** stage.

Adaptive Logic

Solution Pool: A collection of elite solutions, sorted by their value of the objective function.

The **decision** of whether a solution will be inserted to the Pool depends on:

- The **objective** of the solution
- The non-violation of an **upper similarity bound** with the solutions currently present in the Pool.

Goal: Discovery of features that distinguish high quality solutions.

Solution Reconstruction

- The exploitation of Pool solutions begins with the extraction of **frequent chains of nodes**.
- **Chains**: sequences of 3 to 6 nodes
- The most frequently appearing node chains are encountered applying a **pattern matching** procedure.

Chains Collection Procedure

Solution Pool

```
Sol 1: 0 -> 49 -> 89 -> 30 -> 103 -> 67 -> 22 -> 66 -> 42 ->
96 -> 109 -> 48 -> 32 -> 130 -> 16 -> 84 -> 45 -> 81 -> 31 ->
104 -> 82 -> 90 -> 6 -> 105 -> 72 -> 76 -> 37 -> 40 -> 97 ->
15 -> 54 -> 108 -> 64 -> 41 -> 123 -> 63 -> 17 -> 99 -> 29 ->
73 -> 125 -> 115 -> 10 -> 92 -> 0

Sol 2: 0 -> 5 -> 89 -> 30 -> 103 -> 67 -> 22 -> 66 -> 42 ->
96 -> 109 -> 48 -> 32 -> 130 -> 16 -> 127 -> 50 -> 55 -> 82 ->
27 -> 90 -> 6 -> 86 -> 43 -> 105 -> 72 -> 85 -> 19 -> 113 ->
40 -> 97 -> 15 -> 54 -> 108 -> 64 -> 41 -> 123 -> 17 -> 99 ->
73 -> 125 -> 115 -> 10 -> 92 -> 0

Sol 3: 0 -> 49 -> 89 -> 30 -> 103 -> 67 -> 22 -> 66 -> 42 ->
96 -> 109 -> 48 -> 32 -> 130 -> 16 -> 127 -> 50 -> 55 -> 82 ->
27 -> 90 -> 6 -> 105 -> 72 -> 76 -> 37 -> 113 -> 40 -> 97 ->
15 -> 54 -> 108 -> 64 -> 41 -> 123 -> 63 -> 17 -> 99 -> 29 ->
125 -> 73 -> 115 -> 10 -> 92 -> 0
```

Chains Collection Procedure

Solution Pool

```
Sol 1: 0 -> 49 -> 89 -> 30 -> 103 -> 67 -> 22 -> 66 -> 42 ->
96 -> 109 -> 48 -> 32 -> 130 -> 16 -> 84 -> 45 -> 81 -> 31 ->
104 -> 82 -> 90 -> 6 -> 105 -> 72 -> 76 -> 37 -> 40 -> 97 ->
15 -> 54 -> 108 -> 64 -> 41 -> 123 -> 63 -> 17 -> 99 -> 29 ->
73 -> 125 -> 115 -> 10 -> 92 -> 0

Sol 2: 0 -> 5 -> 89 -> 30 -> 103 -> 67 -> 22 -> 66 -> 42 ->
96 -> 109 -> 48 -> 32 -> 130 -> 16 -> 127 -> 50 -> 55 -> 82 ->
27 -> 90 -> 6 -> 86 -> 43 -> 105 -> 72 -> 85 -> 19 -> 113 ->
40 -> 97 -> 15 -> 54 -> 108 -> 64 -> 41 -> 123 -> 17 -> 99 ->
73 -> 125 -> 115 -> 10 -> 92 -> 0

Sol 3: 0 -> 49 -> 89 -> 30 -> 103 -> 67 -> 22 -> 66 -> 42 ->
96 -> 109 -> 48 -> 32 -> 130 -> 16 -> 127 -> 50 -> 55 -> 82 ->
27 -> 90 -> 6 -> 105 -> 72 -> 76 -> 37 -> 113 -> 40 -> 97 ->
15 -> 54 -> 108 -> 64 -> 41 -> 123 -> 63 -> 17 -> 99 -> 29 ->
125 -> 73 -> 115 -> 10 -> 92 -> 0
```

Chains Collection Procedure

```
Sol 1: 0 -> 49 -> 89 -> 30 -> 103 -> 67 -> 22 -> 66 -> 42 ->
96 -> 109 -> 48 -> 32 -> 130 -> 16 -> 84 -> 45 -> 81 -> 31 ->
104 -> 82 -> 90 -> 6 -> 105 -> 72 -> 76 -> 37 -> 40 -> 97 ->
15 -> 54 -> 108 -> 64 -> 41 -> 123 -> 63 -> 17 -> 99 -> 29 ->
73 -> 125 -> 115 -> 10 -> 92 -> 0

Sol 2: 0 -> 5 -> 89 -> 30 -> 103 -> 67 -> 22 -> 66 -> 42 ->
96 -> 109 -> 48 -> 32 -> 130 -> 16 -> 127 -> 50 -> 55 -> 82 ->
27 -> 90 -> 6 -> 86 -> 43 -> 105 -> 72 -> 85 -> 19 -> 113 ->
40 -> 97 -> 15 -> 54 -> 108 -> 64 -> 41 -> 123 -> 17 -> 99 ->
73 -> 125 -> 115 -> 10 -> 92 -> 0

Sol 3: 0 -> 49 -> 89 -> 30 -> 103 -> 67 -> 22 -> 66 -> 42 ->
96 -> 109 -> 48 -> 32 -> 130 -> 16 -> 127 -> 50 -> 55 -> 82 ->
27 -> 90 -> 6 -> 105 -> 72 -> 76 -> 37 -> 113 -> 40 -> 97 ->
15 -> 54 -> 108 -> 64 -> 41 -> 123 -> 63 -> 17 -> 99 -> 29 ->
125 -> 73 -> 115 -> 10 -> 92 -> 0
```

Chain frequency: 3

Chains Collection Procedure

Solution Pool

```
Sol 1: 0 -> 49 -> 89 -> 30 -> 103 -> 67 -> 22 -> 66 -> 42 ->
96 -> 109 -> 48 -> 32 -> 130 -> 16 -> 84 -> 45 -> 81 -> 31 ->
104 -> 82 -> 90 -> 6 -> 105 -> 72 -> 76 -> 37 -> 40 -> 97 ->
15 -> 54 -> 108 -> 64 -> 41 -> 123 -> 63 -> 17 -> 99 -> 29 ->
73 -> 125 -> 115 -> 10 -> 92 -> 0

Sol 2: 0 -> 5 -> 89 -> 30 -> 103 -> 67 -> 22 -> 66 -> 42 ->
96 -> 109 -> 48 -> 32 -> 130 -> 16 -> 127 -> 50 -> 55 -> 82 ->
27 -> 90 -> 6 -> 86 -> 43 -> 105 -> 72 -> 85 -> 19 -> 113 ->
40 -> 97 -> 15 -> 54 -> 108 -> 64 -> 41 -> 123 -> 17 -> 99 ->
73 -> 125 -> 115 -> 10 -> 92 -> 0

Sol 3: 0 -> 49 -> 89 -> 30 -> 103 -> 67 -> 22 -> 66 -> 42 ->
96 -> 109 -> 48 -> 32 -> 130 -> 16 -> 127 -> 50 -> 55 -> 82 ->
27 -> 90 -> 6 -> 105 -> 72 -> 76 -> 37 -> 113 -> 40 -> 97 ->
15 -> 54 -> 108 -> 64 -> 41 -> 123 -> 63 -> 17 -> 99 -> 29 ->
125 -> 73 -> 115 -> 10 -> 92 -> 0
```

Text Representation

```
0-49-89-30-103-67-22-66-42-96-109-48-32-130-16-84-45-81-31-104
-82-90-6-105-72-76-37-40-97-15-54-108-64-41-123-63-17-99-29-73
-125-115-10-92-0-0-5-89-30-103-67-22-66-42-96-109-48-32-130-16
-127-50-55-82-27-90-6-86-43-105-72-85-19-113-40-97-15-54-108-6
4-41-123-17-99-73-125-115-10-92-0-0-49-89-30-103-67-22-66-42-9
6-109-48-32-130-16-127-50-55-82-27-90-6-105-72-76-37-113-40-97
-15-54-108-64-41-123-63-17-99-29-125-73-115-10-92-0
```

Chains Collection Procedure

Input 67-22-66 (string to match)

```
0-49-89-30-103-67-22-66-42-96-109-48-32-130-16-84-45-81-31-104
-82-90-6-105-72-76-37-40-97-15-54-108-64-41-123-63-17-99-29-73
-125-115-10-92-0-0-5-89-30-103-67-22-66-42-96-109-48-32-130-16
-127-50-55-82-27-90-6-86-43-105-72-85-19-113-40-97-15-54-108-6
4-41-123-17-99-73-125-115-10-92-0-0-49-89-30-103-67-22-66-42-9
6-109-48-32-130-16-127-50-55-82-27-90-6-105-72-76-37-113-40-97
-15-54-108-64-41-123-63-17-99-29-125-73-115-10-92-0
```

(Text)

Process String matching algorithm (Boyer-Moore-Horspool)

Output Frequency: 3

Chains Collection Procedure

- This procedure is executed for every **unique** chain of nodes, and those that appear **at least twice** are **collected**.
- Out of all accumulated chains, the **50% most frequent** sequences are qualified and used at the next stage.
- At the final step, the chosen chains will be employed by the **Reconstruction Algorithms**.

Reconstrucion Algorithms

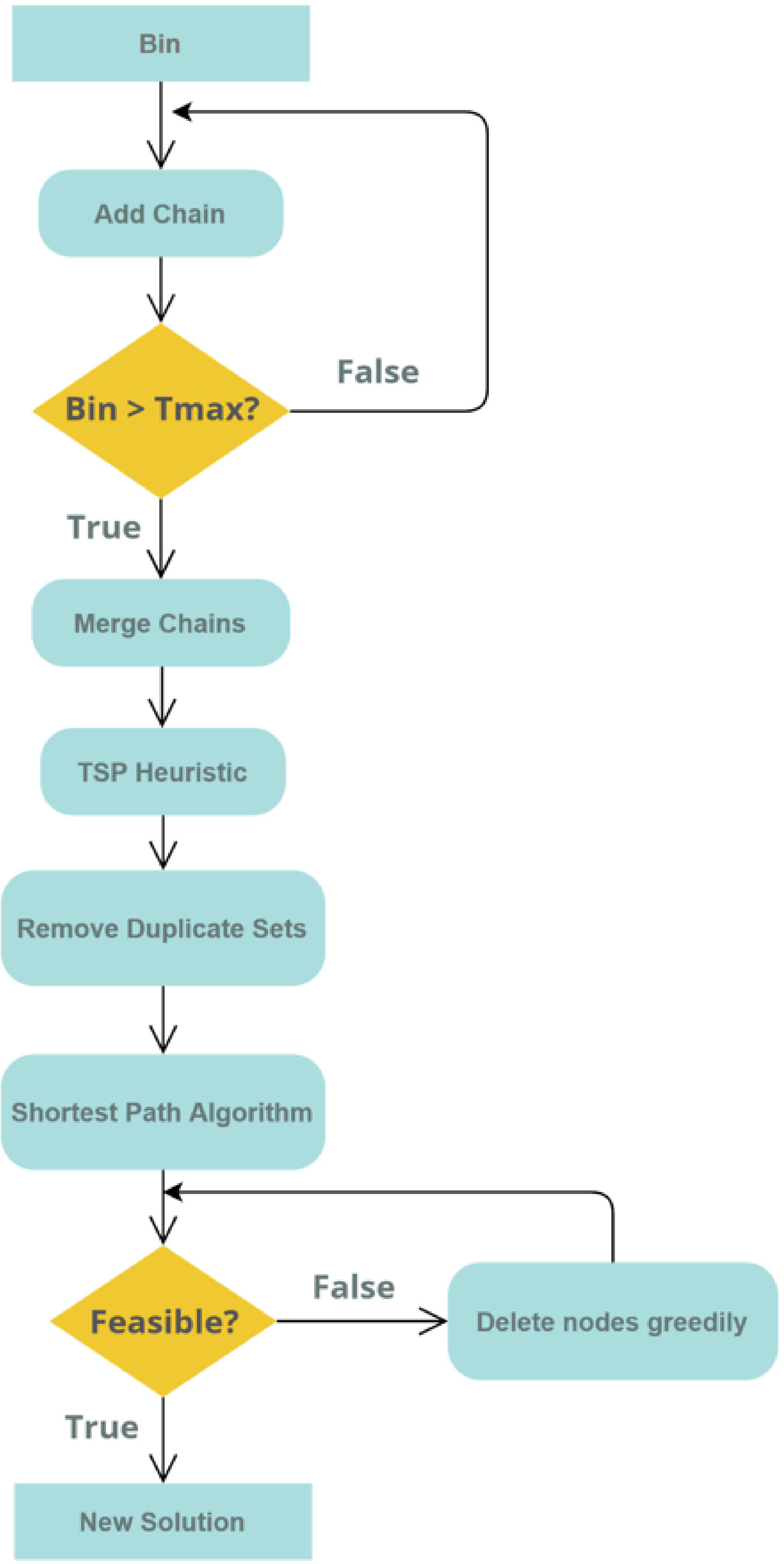
- SR-1
- SR-2
- SR-3

Reconstruction Algorithm SR-1

- Exactly same logic as the construction algorithm.
- But, it only takes into account the nodes that appear in the top 50% of the most frequent chains.

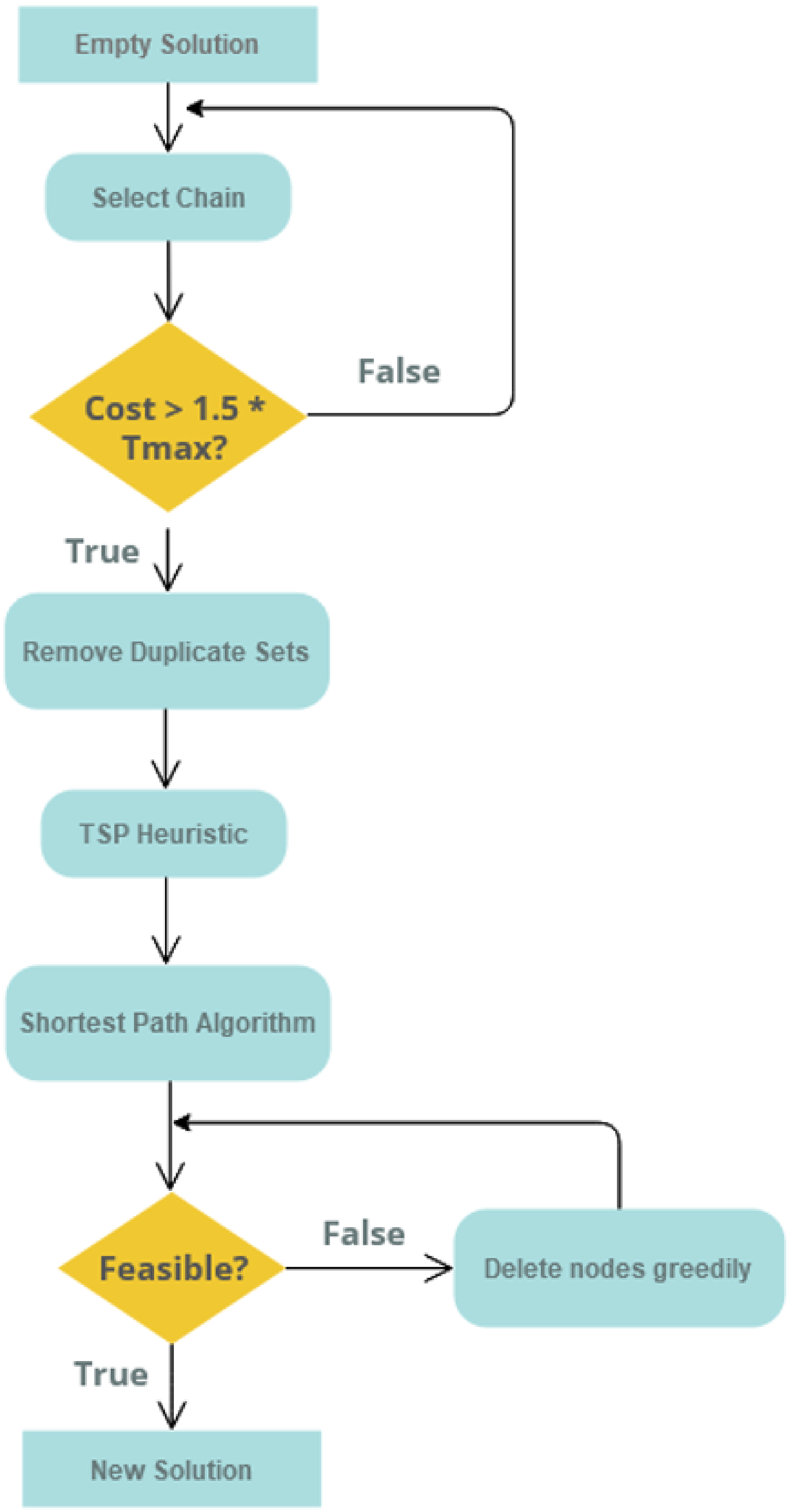
Reconstruction Algorithm SR-2

- **Bin** of Tmax capacity.
- Chains are **inserted** into the bin until it is full.
- The chains are **connected** to a single sequence.
- **TSP** heuristic is applied to this sequence.
- Nodes of **duplicate sets** are serially **removed**.
- **Shortest path** algorithm is applied, reducing the cost of the created solution.
- In case of **infeasibility**, nodes are deleted following a **greedy** criterion.



Reconstruction Algorithm SR-3

- **Minimum Insertions logic**
- Insertions are performed at a **chain level**.
- To do that, a **center** is assigned to each chain.
- Upper cost bound for the solution is set to **Tmax' = 1.5 * Tmax**.
- A single node of each set is stochastically selected, to be kept into the solution.
- **TSP** and **Shortest Path** algorithms are applied.
- In case of **infeasibility**, nodes are deleted following a **greedy** criterion.



03

AMMH Results

Intensification Procedure value

Instance	n	ω	pg	Diff_{sol}	$\text{Diff}_{sol.avg}$	Diff_{time}
115rat575_RND	575	0.4	g1	0.47	0.24	1.18
115rat575_RND	575	0.6	g1	0.00	0.55	3.25
115u574	574	0.6	g2	0.69	1.82	1.80
132d657_RND	657	0.4	g2	0.00	0.65	1.70
132d657	657	0.4	g2	0.44	1.27	0.35
132d657	657	0.8	g1	1.35	1.36	4.18
145u724	724	0.6	g2	1.20	3.62	0.96
157rat783_RND	783	0.4	g2	1.10	1.37	1.20
157rat783	783	0.6	g2	1.14	3.55	0.62
157rat783	783	0.8	g1	1.35	1.97	1.33
201pr1002_RND	1002	0.6	g1	0.40	0.71	4.00
201pr1002	1002	0.6	g2	1.52	3.86	0.56
212u1060_RND	1060	0.4	g1	0.10	0.62	1.95
212u1060_RND	1060	0.6	g2	0.15	0.16	3.55
212u1060	1060	0.4	g1	-0.37	2.08	0.25
212u1060	1060	0.8	g1	0.90	1.51	2.04
217vm1084	1084	0.4	g1	0.45	2.99	-0.08
217vm1084	1084	0.6	g2	0.92	4.67	0.81
217vm1084	1084	0.8	g2	1.46	1.62	3.08
53gil262_RND	262	0.6	g2	0.00	0.07	12.75
80rd400_RND	400	0.4	g2	0.00	0.54	2.79
80rd400	400	0.8	g1	0.82	0.83	3.97
89pcb442_RND	442	0.4	g1	-0.26	0.00	2.95
89pcb442_RND	442	0.4	g2	0.00	0.26	3.29
99d493	493	0.8	g2	0.63	0.85	7.50
Average				0.58	1.49	2.64

Enhancement of best solution: 0.58%

Enhancement of average solution: 1.49%

Increase in computational time: 2.64%

SR algorithms comparison

Instance	Best	No pool	SR1	SR2	SR3
		Gap	Gap	Gap	Gap
1187rl5934_T40_p1	2953	0.038	0.058	0.000	0.018
200E1k.0_T80_p1	911	0.001	0.002	0.000	0.001
235pcb1173_RND_T60_p2	58877	0.028	0.000	0.072	0.009
259d1291_T40_p2	29520	0.103	0.061	0.111	0.000
261rl1304_T60_p2	48198	0.005	0.057	0.012	0.000
265rl1323_T80_p2	61850	0.000	0.139	0.084	0.043
276nrw1379_RND_T60_p1	1367	0.001	0.001	0.001	0.000
280fl1400_T40_p1	884	0.000	0.000	0.000	0.000
287u1432_T60_p1	1052	0.000	0.002	0.001	0.001
316fl1577_T80_p1	1398	0.001	0.001	0.001	0.000
331d1655_RND_T40_p2	64684	0.141	0.195	0.259	0.000
350vm1748_RND_T60_p2	88203	0.000	0.000	0.000	0.000
364u1817_T80_p2	82443	0.067	0.178	0.109	0.000
378rl1889_T40_p2	49869	0.821	0.646	0.000	0.335
421d2103_T60_p1	1410	0.002	0.001	0.000	0.001
431u2152_T80_p2	96430	0.600	0.587	0.000	0.098
464u2319_RND_T40_p1	1569	0.007	0.007	0.001	0.000
479pr2392_RND_T60_p2	120627	0.255	0.125	0.034	0.000
49usa1097_RND_T40_p1	1096	0.000	0.000	0.000	0.000
608pcb3038_T80_p1	2709	0.012	0.003	0.000	0.000
633E3k.0_T40_p1	1578	0.015	0.023	0.000	0.008
759fl3795_T60_p2	140880	1.908	1.640	0.000	0.222
893fnl4461_T80_p1	3804	0.001	0.006	0.000	0.001
Average	-	0.174	0.162	$\boxed{0.030}$	0.032

SR2 and **SR3** algorithms produced the best results with **SR2** being the best.

Summarized final results

Table 3: Algorithms comparison on data set

Clustering Type	ω	pg	MASOP			VNS			BRKGA			AMMH			
			#Bst	t	Gap	#Bst	t	Gap	#Bst	t	Gap	#New	# Bst	t	Gap
Original	0.4	g1	41	73.04	0.18	41	20.9	0.7	41	12.39	0.23	5	**48**	10.38	0.09
		g2	36	89.98	0.54	39	20.4	0.8	40	12.57	0.24	7	**49**	13.61	0.07
	0.6	g1	37	45.32	0.24	35	35.2	1.0	36	20.60	0.44	12	**51**	20.91	0.00
		g2	37	51.04	0.15	34	35.2	0.9	39	20.67	0.32	11	**50**	21.08	0.00
	0.8	g1	37	35.66	0.20	34	52.0	0.7	35	28.29	0.64	10	**51**	22.95	0.00
		g2	37	39.27	0.12	36	52.5	0.6	33	28.30	0.61	11	**49**	29.85	0.01
Random	0.4	g1	31	97.83	1.29	33	85.8	0.7	34	46.04	1.05	13	**51**	19.72	0.00
		g2	31	121.85	0.86	33	87.4	0.8	32	45.78	0.80	16	**51**	29.90	0.00
	0.6	g1	42	84.48	0.13	35	147.6	0.5	31	64.96	0.76	9	**51**	27.52	0.00
		g2	39	84.96	0.17	31	145.3	0.4	31	63.30	0.52	11	**50**	32.58	0.00
	0.8	g1	51	136.21	0.00	51	230.8	0.0	43	69.24	0.07	0	51	21.56	0.00
		g2	51	148.92	0.00	51	233.4	0.0	44	67.78	0.06	0	51	21.36	0.00
Total/Average			470	84.05	0.32	453	95.52	0.60	439	39.99	0.48	105	**603**	22.62	0.01

603/612 (98%) best

102/612 (17%) new best

Large Datasets

- The available SOP datasets were **relatively small**.
- The **high efficiency of AMMH** even at the largest of them, led us to create **new and even bigger datasets**.
- The new instances contain up to **3038 nodes** and **633 sets**.

References

Angelelli E., Archetti C., Vindigni M. The Clustered Orienteering Problem // European Journal of Operational Research. 2014. 238, 2. 404–414.

Applegate David, Bixby Ribert, Chvatal Vasek, Cook William. Concorde TSP solver. 2006.

Archetti Claudia, Carrabs Francesco, Cerulli Raffaele. The Set Orienteering Problem // European Journal of Operational Research. 2018. 267, 1. 264–272.

Bernardino Raquel, Paias Ana. Solving the family traveling salesman problem // European Journal of Operational Research. 2018. 267, 2. 453–466.

Boyer Robert S., Moore J. Strother. A Fast String Searching Algorithm // Commun. ACM. X 1977. 20, 10. 762–772.

Carrabs Francesco. A biased random-key genetic algorithm for the set orienteering problem // European Journal of Operational Research. 2020. 292, 3. 830–854.

Chao I. Ming, Golden Bruce L., Wasil Edward A. A fast and effective heuristic for the orienteering problem // European Journal of Operational Research. 1996. 88, 3. 475–489.

Chisman James A. The clustered traveling salesman problem // Computers Operations Research. 1975. 2, 2. 115–119.

Fischetti Matted, Gonzalez Juan José Salazar, Toth Paolo. Solving the Orienteering Problem through branch-and-cut // INFORMS Journal on Computing. may 1998. 10, 2. 133–148.

Fischetti Matteo, Salazar González Juan José, Toth Paolo. A Branch-and-Cut Algorithm for the Symmetric Generalized Traveling Salesman Problem // Operations Research. 1997. 45, 3. 378–394.

Helsgaun Keld. An effective implementation of the Lin–Kernighan traveling salesman heuristic // European Journal of Operational Research. 2000. 126, 1. 106–130.

Jongens Kees, Volgenant Ton. The symmetric clustered traveling salesman problem // European Journal of Operational Research. 1985. 19, 1. 68–75.

Ke Liangjun, Zhai Laipeng, Li Jing, Chan Felix T.S. Pareto mimic algorithm: An approach to the team orienteering problem // Omega (United Kingdom). 2016. 61. 155–166.

Keshtkaran Morteza, Ziarati Koorush. A novel GRASP solution approach for the Orienteering Problem // Journal of Heuristics. 2016. 22.

Laporte Gilbert, Asef-Vaziri Ardavan, Sriskandarajah Chelliah. Some Applications of the Generalized Travelling Salesman Problem // The Journal of the Operational Research Society. 12 1996. 47.

Laporte Gilbert, Nobert Yves. Generalized Travelling Salesman Problem Through n Sets of Nodes: An Integer Programming Approach // INFOR Journal. 1983. 21.

Lin S., Kernighan B. An Effective Heuristic Algorithm for the Traveling-Salesman Problem // Oper. Res. 1973. 21. 498–516.

Lokin F.C.J. Procedures for travelling salesman problems with additional constraints // European Journal of Operational Research. 1979. 3, 2. 135–141.

Morán-Mirabal L.F., González-Velarde J.L., Resende M.G.C. Randomized heuristics for the family traveling salesperson problem // International Transactions in Operational Research. 2014. 21, 1. 41–57.

Pěnička Robert, Faigl Jan, Saska Martin. Variable Neighborhood Search for the Set Orienteering Problem and its application to other Orienteering Problem variants // European Journal of Operational Research. 2019. 276, 3. 816–825.

Pop Petrică, Matei Oliviu, Pintea Camelia. A Two-Level Diploid Genetic Based Algorithm for Solving the Family Traveling Salesman Problem // Proceedings of the Genetic and Evolutionary Computation Conference. New York, NY, USA: Association for Computing Machinery, 2018. 340–346. (GECCO '18).

Repoussis P. P., Tarantilis C. D. Solving the fleet size and mix vehicle routing problem with time windows via adaptive memory programming // Transportation Research Part C: Emerging Technologies. 2010. 18, 5. 695–712.

Smith Stephen L., Imeson Frank. GLNS: An effective large neighborhood search heuristic for the Generalized Traveling Salesman Problem // Computers and Operations Research. 11 2017. 87. 1–19.

Snyder Lawrence V., Daskin Mark S. A random-key genetic algorithm for the generalized traveling salesman problem // European Journal of Operational Research. 2006. 174.

Taillard Éric D., Gambardella Luca M., Gendreau Michel, Potvin Jean Yves. Adaptive memory programming: A unified view of metaheuristics // European Journal of Operational Research. 2001. 135, 1. 1–16.

Taillard Éric D., Helsgaun Keld. POPMUSIC for the travelling salesman problem // European Journal of Operational Research. 2019. 272, 2. 420–429.

Tarantilis C. D. Solving the vehicle routing problem with adaptive memory programming methodology // Computers and Operations Research. 2005. 32, 9. 2309–2327.

Tsiligirides T. Heuristic methods applied to orienteering // Journal of the Operational Research Society. 1984. 35, 9. 797–809.

Vansteenwegen Pieter, Souffriau Wouter, Oudheusden Dirk Van. The orienteering problem: A survey // European Journal of Operational Research. 2011. 209, 1. 1–10.

Zachariadis Emmanouil E., Tarantilis Christos D., Kiranoudis Chris T. The load-dependent vehicle routing problem and its pick-up and delivery extension // Transportation Research Part B: Methodological. 2015. 71. 158–181.

Thank you for your attention!

Dontas Michael
mdontas@aueb.gr
Michael Dontas

Sideris Georgios
geosideris@aueb.gr
George Sideris

Manousakis Eleftherios
lmanousakis@aueb.gr
Eleftherios Manousakis

Zachariadis Emmanouil
ezach@aueb.gr
Emmanouil Zachariadis

Department of Management Science and Technology,
Athens University of Economics and Business

EURO 2022, Espoo Finland, 3 - 6 July

H.F.R.I.
Hellenic Foundation for Research & Innovation

The research project was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "2nd Call for H.F.R.I. Research Projects to support Faculty Members & Researchers" (Project Number: 02562).