eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

NP-hard problems still persist…

… despite Deep Learning

Josu Ceberio, May 2024

NP-Hard

This is about complexity

Given a NP-hard optimization problems,

<u>we do not know an algorithm</u> that is able to solve <u>all the instances of the problem</u> <u>in polynomial time</u>.

In other words, it is the set of most challenging problems.

Historical record

Optimization approaches

Minimal distance (Euclid, 300 BC)

Secretary problem (Kepler, 1615)

Problem of minimal surfaces (Lagrange, 1754)

Transportation problem (Monge, 1784)

Gradient method (Cauchy, 1847)

Travelling Salesman Problem (Menger, 1932) WW II

Linear Programming and Simplex (Dantzig, 1947)

Theory of duality (von Neumann, 1947)

Quadratic Optimization (Markowitz, 1951)

Dynamic Programming (Bellman, 1953)

Hungarian algorithm (Kuhn, 1955)

Dijkstra's algorithm (Dijkstra, 1956)

Optimality principle (Bellman, 1957)

A* algorithm (Hart, 1968)

Genetic Algorithms (Holland, 1975)

Scatter Search (Glover, 1977)

Simulated Annealing (Kirkpatrick et al., 1983)

Genetic Programming (Koza, 1988)

No Free Lunch (Wolpert & MacReady, 1997)

Factorized Distribution Algorithm (Muhlenbein, 1999)

Iterated Local Search (Lourenço, 2001)

NSGA-II (Deb, 2002)

…

…

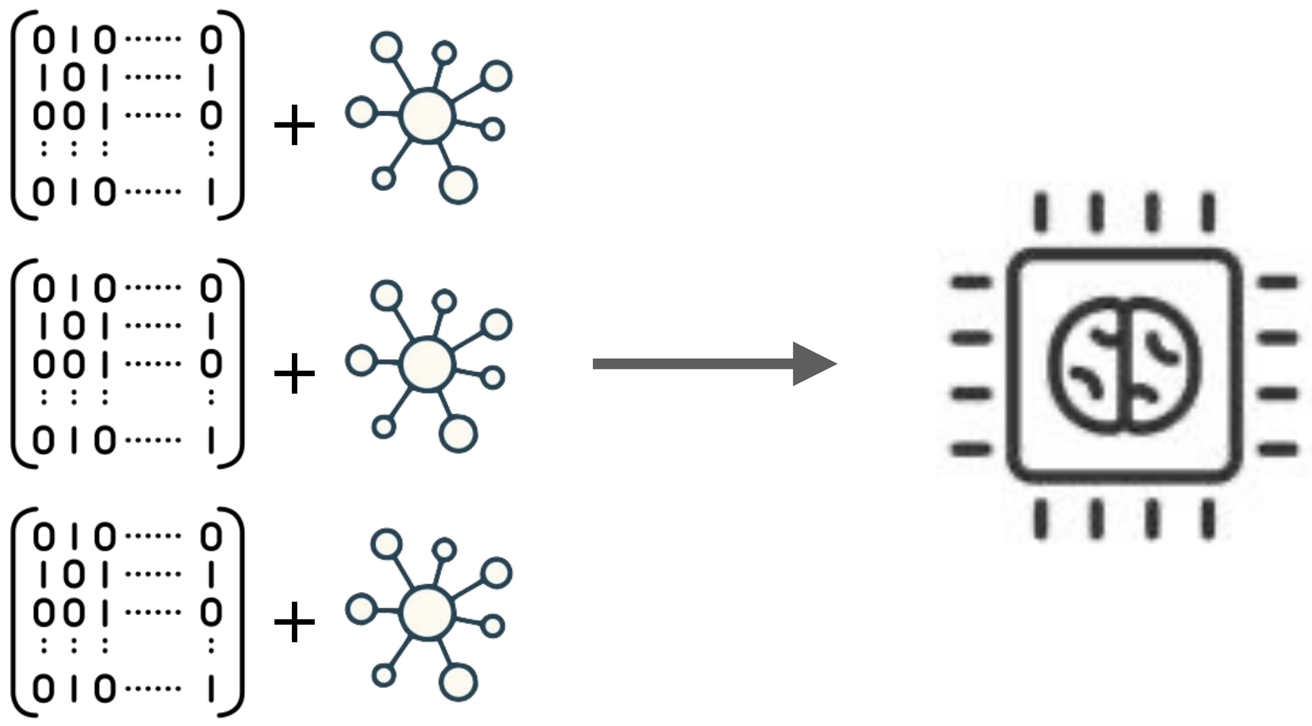
Pointer-network model for TSP (Vinyals, 2015)

Neural Combinatorial Optimization (Bello, 2016)

Source: http://www.mitrikitti.fi/opthist.html

Deep Learning
But… what's really in it?

Supervised Training



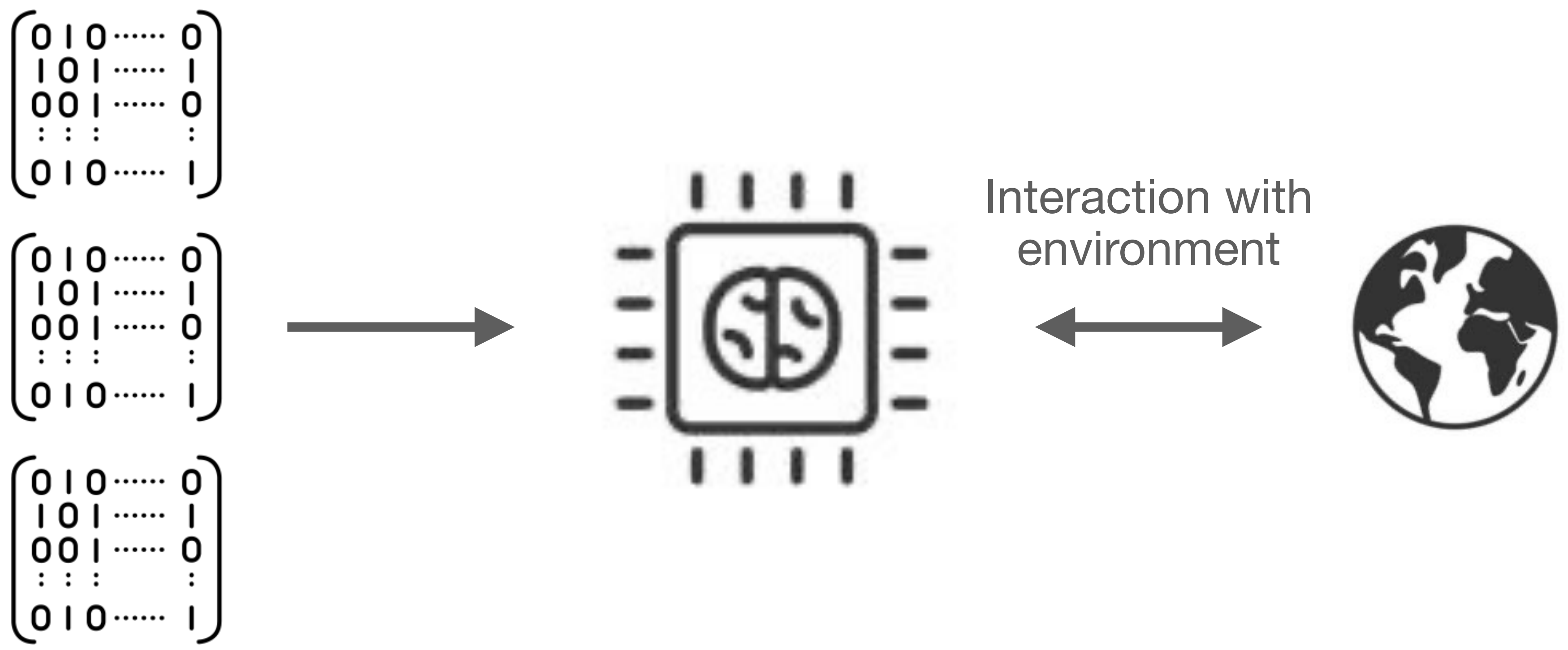
Infer (construct) a solution

Vinyals O., Fortunato M., and Jaitly N. (2015). **Pointer Networks**. Advances in Neural Information Processing Systems 28.

Deep Learning

But… what’s really in it?

Reinforcement Learning



Infer (construct) a solution

Bello I., Pham H., Quoc V. L., Norouzi M., and Bengio S. (2017). **Neural Combinatorial Optimization with Reinforcement Learning**. ICLR 2017.

Deep Learning

But… what's really in it?

	Method	$n=20$ Obj.	Gap	Time	$n=50$ Obj.	Gap	Time	$n=100$ Obj.	Gap	Time
TSP	Concorde	3.84	0.00%	(1m)	5.70	0.00%	(2m)	7.76	0.00%	(3m)
	LKH3	3.84	0.00%	(18s)	5.70	0.00%	(5m)	7.76	0.00%	(21m)
	Gurobi	3.84	0.00%	(7s)	5.70	0.00%	(2m)	7.76	0.00%	(17m)
	Gurobi (1s)	3.84	0.00%	(8s)	5.70	0.00%	(2m)		-	
	Nearest Insertion	4.33	12.91%	(1s)	6.78	19.03%	(2s)	9.46	21.82%	(6s)
	Random Insertion	4.00	4.36%	(0s)	6.13	7.65%	(1s)	8.52	9.69%	(3s)
	Farthest Insertion	3.93	2.36%	(1s)	6.01	5.53%	(2s)	8.35	7.59%	(7s)
	Nearest Neighbor	4.50	17.23%	(0s)	7.00	22.94%	(0s)	9.68	24.73%	(0s)
	Vinyals et al. (gr.)	3.88	1.15%		7.66	34.48%			-	
	Bello et al. (gr.)	3.89	1.42%		5.95	4.46%		8.30	6.90%	
	Dai et al.	3.89	1.42%		5.99	5.16%		8.31	7.03%	
	Nowak et al.	3.93	2.46%			-			-	
	EAN (greedy)	3.86	0.66%	(2m)	5.92	3.98%	(5m)	8.42	8.41%	(8m)
	AM (greedy)	**3.85**	**0.34%**	(0s)	**5.80**	**1.76%**	(2s)	**8.12**	**4.53%**	(6s)
	OR Tools	3.85	0.37%		5.80	1.83%		7.99	2.90%	
	Chr.f. + 2OPT	3.85	0.37%		5.79	1.65%			-	
	Bello et al. (s.)		-		5.75	0.95%		8.00	3.03%	
	EAN (gr. + 2OPT)	3.85	0.42%	(4m)	5.85	2.77%	(26m)	8.17	5.21%	(3h)
	EAN (sampling)	3.84	0.11%	(5m)	5.77	1.28%	(17m)	8.75	12.70%	(56m)
	EAN (s. + 2OPT)	3.84	0.09%	(6m)	5.75	1.00%	(32m)	8.12	4.64%	(5h)
	AM (sampling)	**3.84**	**0.08%**	(5m)	**5.73**	**0.52%**	(24m)	**7.94**	**2.26%**	(1h)
OP (distance)	Gurobi	5.39	0.00%	(16m)		-			-	
	Gurobi (1s)	4.62	14.22%	(4m)	1.29	92.03%	(6m)	0.58	98.25%	(7m)
	Gurobi (10s)	5.37	0.33%	(12m)	10.96	32.20%	(51m)	1.34	95.97%	(53m)
	Gurobi (30s)	5.38	0.05%	(14m)	13.57	16.09%	(2h)	3.23	90.28%	(3h)
	Compass	5.37	0.36%	(2m)	16.17	0.00%	(5m)	33.19	0.00%	(15m)
	Tsili (greedy)	4.08	24.25%	(4s)	12.46	22.94%	(4s)	25.69	22.59%	(5s)
	AM (greedy)	**5.19**	**3.64%**	(0s)	**15.64**	**3.23%**	(1s)	**31.62**	**4.75%**	(5s)
	GA (Python)	5.12	4.88%	(10m)	10.90	32.59%	(1h)	14.91	55.08%	(5h)
	OR Tools (10s)	4.09	24.05%	(52m)		-			-	
	Tsili (sampling)	5.30	1.62%	(28s)	15.50	4.14%	(2m)	30.52	8.05%	(6m)
	AM (sampling)	**5.30**	**1.56%**	(4m)	**16.07**	**0.60%**	(16m)	**32.68**	**1.55%**	(53m)

Kool W., van Hoof H., and Welling M. (2019). **Attention, Learn to Solve Routing Problems!**. ICLR 2019.

Published as a conference paper at ICLR 2019

Table 1: Attention Model (AM) vs baselines. The gap % is w.r.t. the best value across all methods.

	Method	$n=20$ Obj.	Gap	Time	$n=50$ Obj.	Gap	Time	$n=100$ Obj.	Gap	Time
TSP	Concorde	3.84	0.00%	(1m)	5.70	0.00%	(2m)	7.76	0.00%	(3m)
	LKH3	3.84	0.00%	(18s)	5.70	0.00%	(5m)	7.76	0.00%	(21m)
	Gurobi	3.84	0.00%	(7s)	5.70	0.00%	(2m)	7.76	0.00%	(17m)
	Gurobi (1s)	3.84	0.00%	(8s)	5.70	0.00%	(2m)		-	
	Nearest Insertion	4.33	12.91%	(1s)	6.78	19.03%	(2s)	9.46	21.82%	(6s)
	Random Insertion	4.00	4.36%	(0s)	6.13	7.65%	(1s)	8.52	9.69%	(3s)
	Farthest Insertion	3.93	2.36%	(1s)	6.01	5.53%	(2s)	8.35	7.59%	(7s)
	Nearest Neighbor	4.50	17.23%	(0s)	7.00	22.94%	(0s)	9.68	24.73%	(0s)
	Vinyals et al. (gr.)	3.88	1.15%		7.66	34.48%			-	
	Bello et al. (gr.)	3.89	1.42%		5.95	4.46%		8.30	6.90%	
	Dai et al.	3.89	1.42%		5.99	5.16%		8.31	7.03%	
	Nowak et al.	3.93	2.46%			-			-	
	EAN (greedy)	3.86	0.66%	(2m)	5.92	3.98%	(5m)	8.42	8.41%	(8m)
	AM (greedy)	**3.85**	**0.34%**	(0s)	**5.80**	**1.76%**	(2s)	**8.12**	**4.53%**	(6s)
	OR Tools	3.85	0.37%		5.80	1.83%		7.99	2.90%	
	Chr.f. + 2OPT	3.85	0.37%		5.79	1.65%			-	
	Bello et al. (s.)		-		5.75	0.95%		8.00	3.03%	
	EAN (gr. + 2OPT)	3.85	0.42%	(4m)	5.85	2.77%	(26m)	8.17	5.21%	(3h)
	EAN (sampling)	3.84	0.11%	(5m)	5.77	1.28%	(17m)	8.75	12.70%	(56m)
	EAN (s. + 2OPT)	3.84	0.09%	(6m)	5.75	1.00%	(32m)	8.12	4.64%	(5h)
	AM (sampling)	**3.84**	**0.08%**	(5m)	**5.73**	**0.52%**	(24m)	**7.94**	**2.26%**	(1h)
CVRP	Gurobi	6.10	0.00%			-			-	
	LKH3	6.14	0.58%	(2h)	10.38	0.00%	(7h)	15.65	0.00%	(13h)
	RL (greedy)	6.59	8.03%		11.39	9.78%		17.23	10.12%	
	AM (greedy)	**6.40**	**4.97%**	(1s)	**10.98**	**5.86%**	(3s)	**16.80**	**7.34%**	(8s)
	RL (beam 10)	6.40	4.92%		11.15	7.46%		16.96	8.39%	
	Random CW	6.81	11.64%		12.25	18.07%		18.96	21.18%	
	Random Sweep	7.08	16.07%		12.96	24.91%		20.33	29.93%	
	OR Tools	6.43	5.41%		11.31	9.01%		17.16	9.67%	
	AM (sampling)	**6.25**	**2.49%**	(6m)	**10.62**	**2.40%**	(28m)	**16.23**	**3.72%**	(2h)
SDVRP	RL (greedy)	6.51	4.19%		11.32	6.88%		17.12	5.23%	
	AM (greedy)	**6.39**	**2.34%**	(1s)	**10.92**	**3.08%**	(4s)	**16.83**	**3.42%**	(11s)
	RL (beam 10)	6.34	1.47%		11.08	4.61%		16.86	3.63%	
	AM (sampling)	**6.25**	**0.00%**	(9m)	**10.59**	**0.00%**	(42m)	**16.27**	**0.00%**	(3h)
OP (distance)	Gurobi	5.39	0.00%	(16m)		-			-	
	Gurobi (1s)	4.62	14.22%	(4m)	1.29	92.03%	(6m)	0.58	98.25%	(7m)
	Gurobi (10s)	5.37	0.33%	(12m)	10.96	32.20%	(51m)	1.34	95.97%	(53m)
	Gurobi (30s)	5.38	0.05%	(14m)	13.57	16.09%	(2h)	3.23	90.28%	(3h)
	Compass	5.37	0.36%	(2m)	16.17	0.00%	(5m)	33.19	0.00%	(15m)
	Tsili (greedy)	4.08	24.25%	(4s)	12.46	22.94%	(4s)	25.69	22.59%	(5s)
	AM (greedy)	**5.19**	**3.64%**	(0s)	**15.64**	**3.23%**	(1s)	**31.62**	**4.75%**	(5s)
	GA (Python)	5.12	4.88%	(10m)	10.90	32.59%	(1h)	14.91	55.08%	(5h)
	OR Tools (10s)	4.09	24.05%	(52m)		-			-	
	Tsili (sampling)	5.30	1.62%	(28s)	15.50	4.14%	(2m)	30.52	8.05%	(6m)
	AM (sampling)	**5.30**	**1.56%**	(4m)	**16.07**	**0.60%**	(16m)	**32.68**	**1.55%**	(53m)
PCTSP	Gurobi	3.13	0.00%	(2m)		-			-	
	Gurobi (1s)	3.14	0.07%	(1m)		-			-	
	Gurobi (10s)	3.13	0.00%	(2m)	4.54	1.36%	(32m)		-	
	Gurobi (30s)	3.13	0.00%	(2m)	4.48	0.03%	(54m)		-	
	AM (greedy)	**3.18**	**1.62%**	(0s)	**4.60**	**2.66%**	(2s)	**6.25**	**4.46%**	(5s)
	ILS (C++)	3.16	0.77%	(16m)	4.50	0.36%	(2h)	**5.98**	**0.00%**	(12h)
	OR Tools (10s)	3.14	0.05%	(52m)	4.51	0.70%	(52m)	6.35	6.21%	(52m)
	OR Tools (60s)	**3.13**	**0.01%**	(5h)	**4.48**	**0.00%**	(5h)	6.07	1.56%	(5h)
	ILS (Python 10x)	5.21	66.19%	(4m)	12.51	179.05%	(3m)	23.98	300.95%	(3m)
	AM (sampling)	3.15	0.45%	(5m)	4.52	0.74%	(19m)	6.08	1.67%	(1h)
SPCTSP	REOPT (all)	3.34	2.38%	(17m)	4.68	1.04%	(2h)	6.22	1.10%	(12h)
	REOPT (half)	3.31	1.38%	(25m)	**4.64**	**0.00%**	(3h)	**6.16**	**0.00%**	(16h)
	REOPT (first)	3.31	1.60%	(1h)	4.66	0.44%	(22h)		-	
	AM (greedy)	**3.26**	**0.00%**	(0s)	4.65	0.33%	(2s)	6.32	2.69%	(5s)

Deep Learning

But… what's really in it?

Competitive performance.

Low inference times.

Questionable comparisons.

All are constructive.

PROS

CONS

The Challenge

Prove me wrong!

“Do NCO models outperform metaheuristics ?”

Combinatorial Optimization Problems

Formal definition

Finite search space of solutions Ω

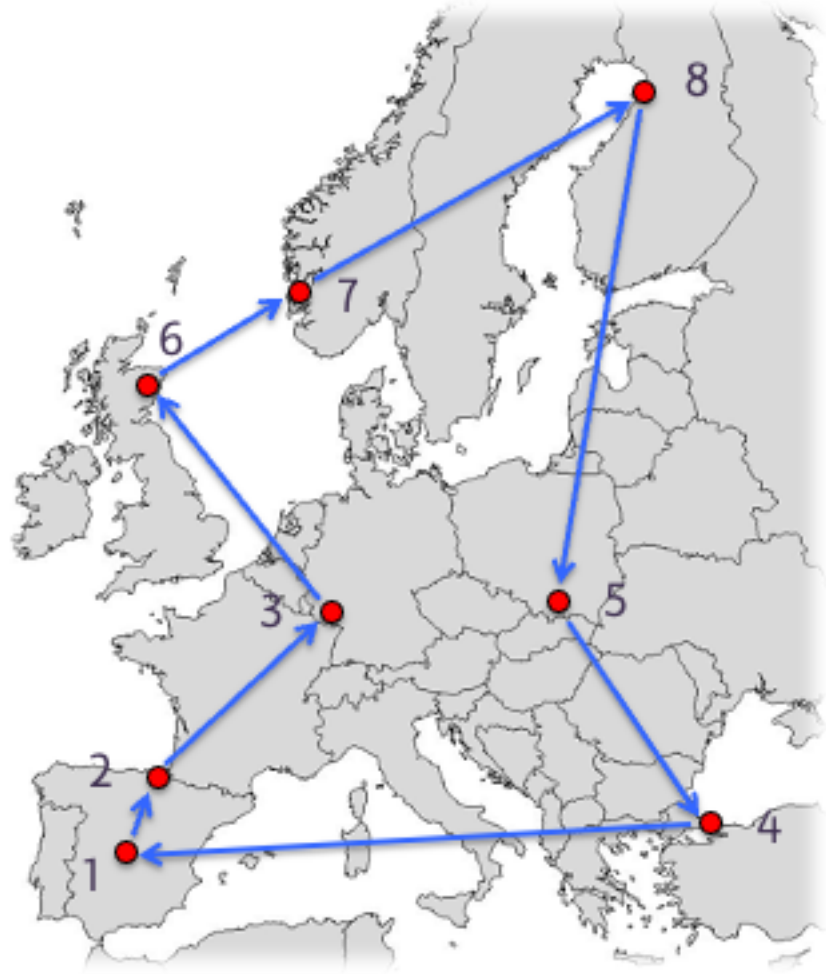
Objective function $f : \Omega \to \mathbb{R}$

The aim:

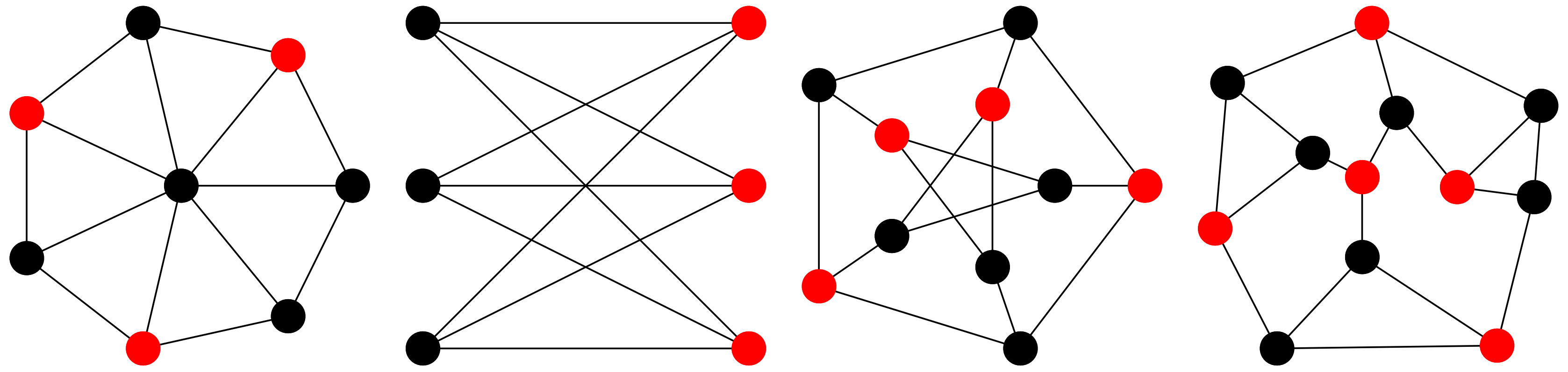
$$s^* = \arg\max_{s \in \Omega} f(s)$$

Combinatorial Optimization Problems

Some examples



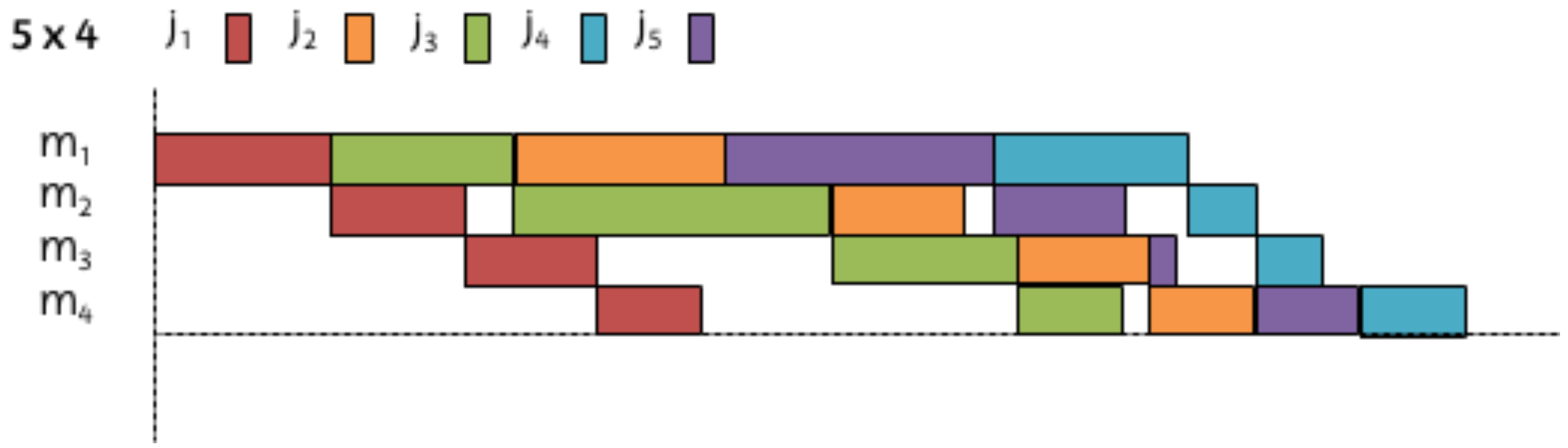
Travelling Salesman Problem



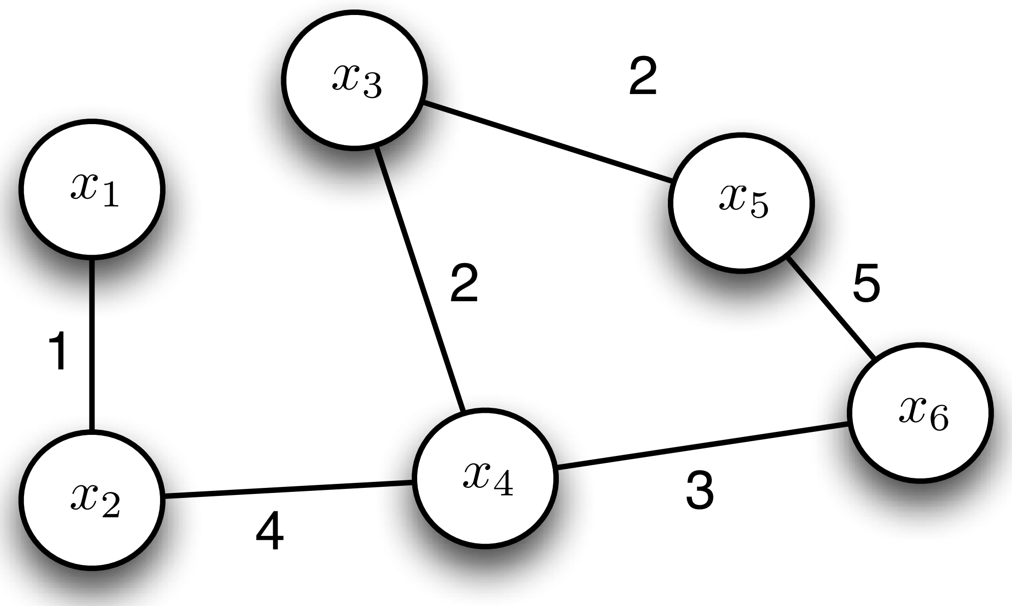
Maximum Independent Set

	1	2	3	4	5
1	0	16	11	15	7
2	21	0	14	15	9
3	26	23	0	26	12
4	22	22	11	0	13
5	30	28	25	24	0

Linear Ordering Problem



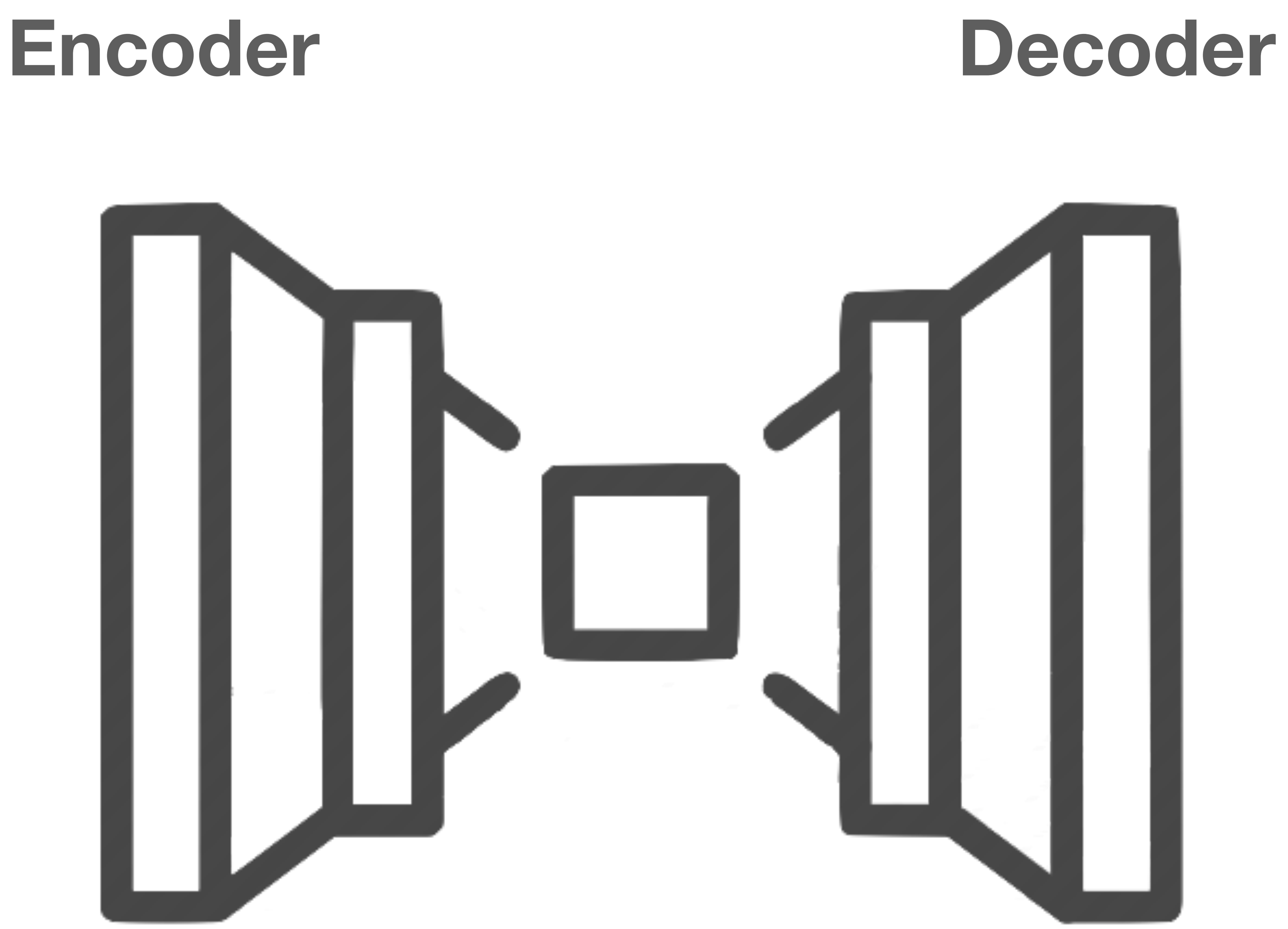
Flowshop Scheduling Problem



Max-Cut Problem

The constructive

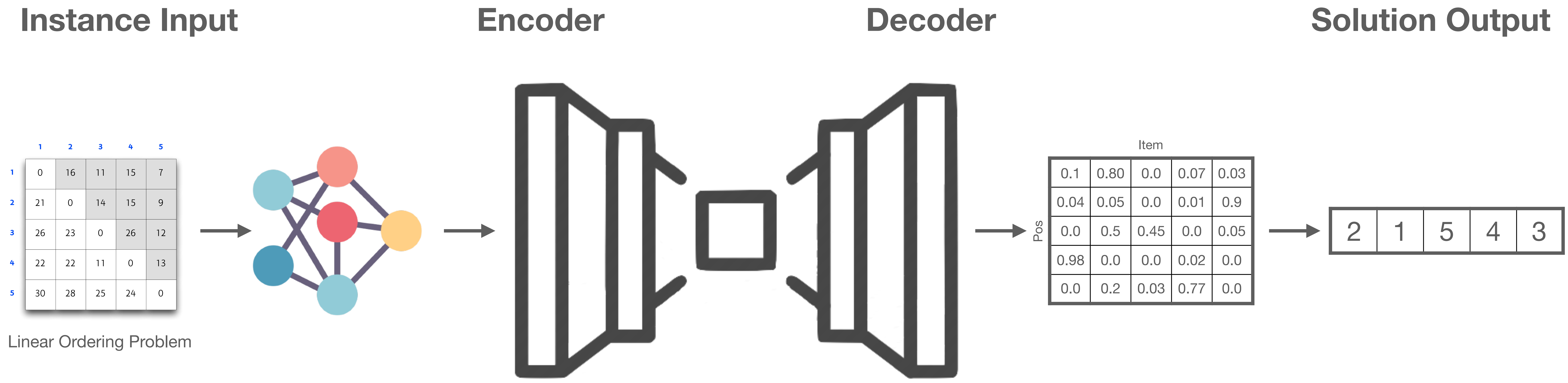
A Neural Combinatorial Optimization approach



Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2024). **Applicability of Neural Combinatorial Optimization: A Critical View**. ACM Trans. on Evolutionary Learning and Optimization.

The constructive

A Neural Combinatorial Optimization approach



Relevant aspects:

- Model the problem as a GNN or similar.
- Keep a rich representation.

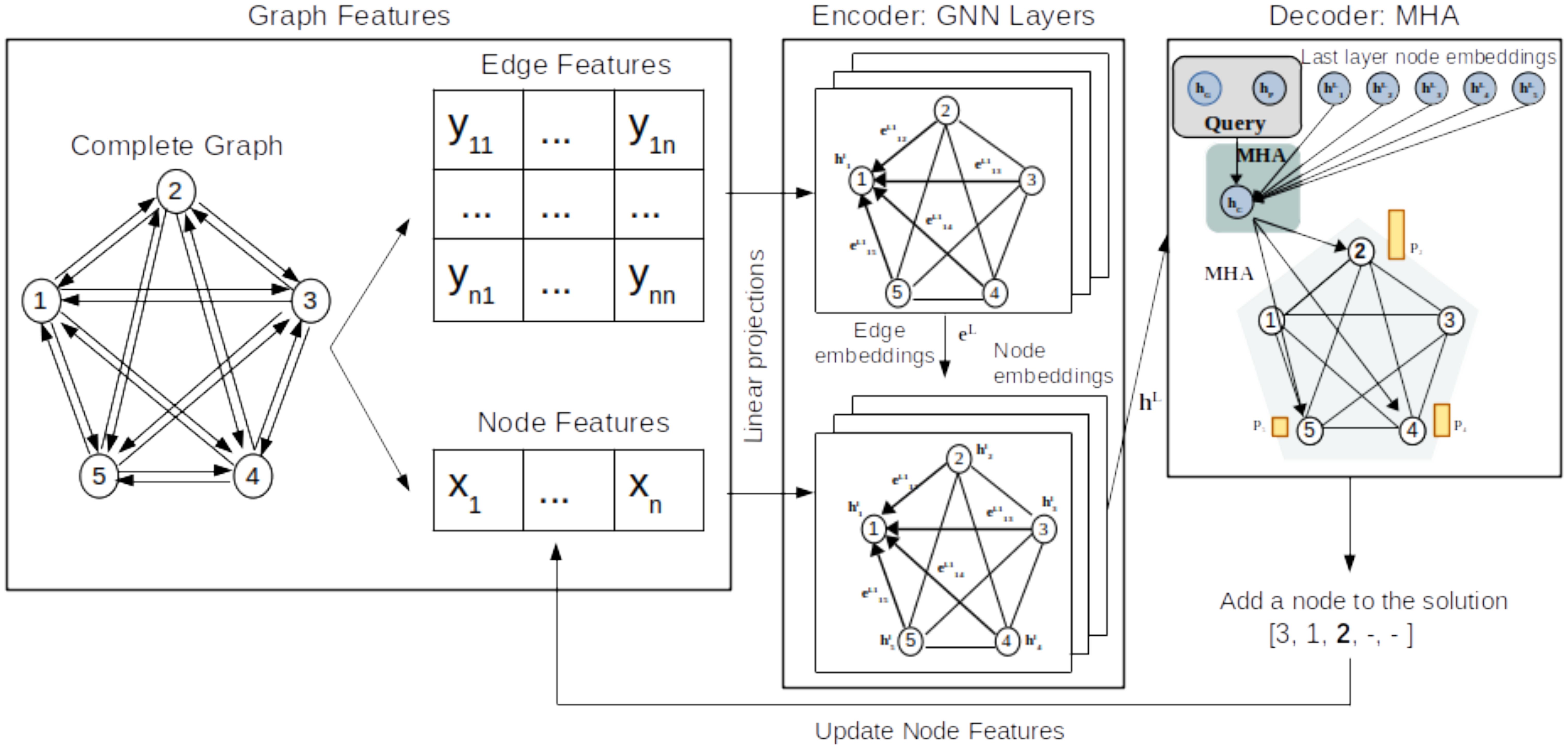
Procedure:

- Autoregressive.
- End-to-end

Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2024). **Applicability of Neural Combinatorial Optimization: A Critical View**. ACM Trans. on Evolutionary Learning and Optimization.

The constructive

A Neural Combinatorial Optimization approach



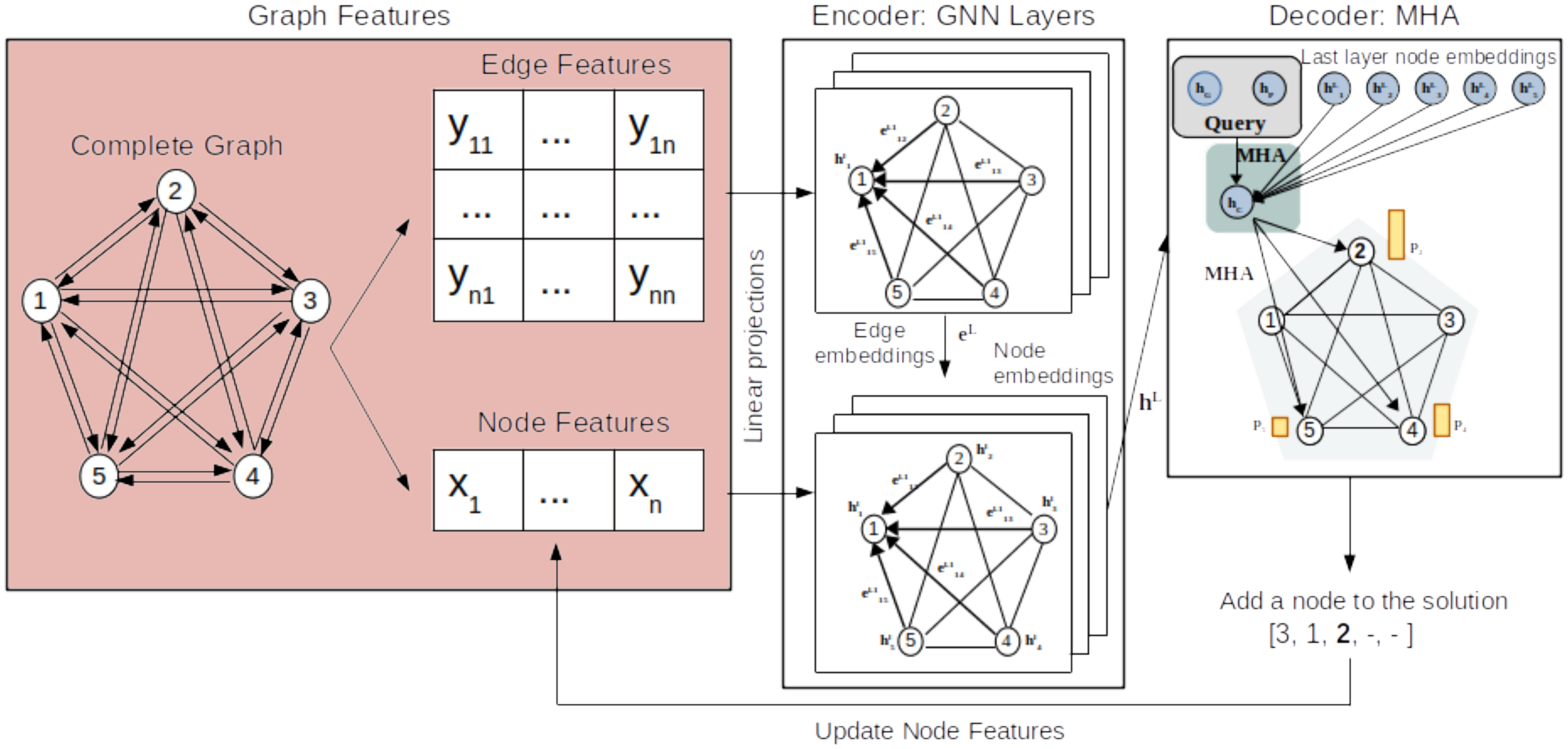
Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2024). **Applicability of Neural Combinatorial Optimization: A Critical View**. ACM Trans. on Evolutionary Learning and Optimization.

The constructive

A Neural Combinatorial Optimization approach

Graph features:

- Edge features y_{ij} taken from the instance matrix.
- Node features x_i, a priori, meaningless.
- This information feeds the GNN encoder.



Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2024). **Applicability of Neural Combinatorial Optimization: A Critical View**. ACM Trans. on Evolutionary Learning and Optimization.

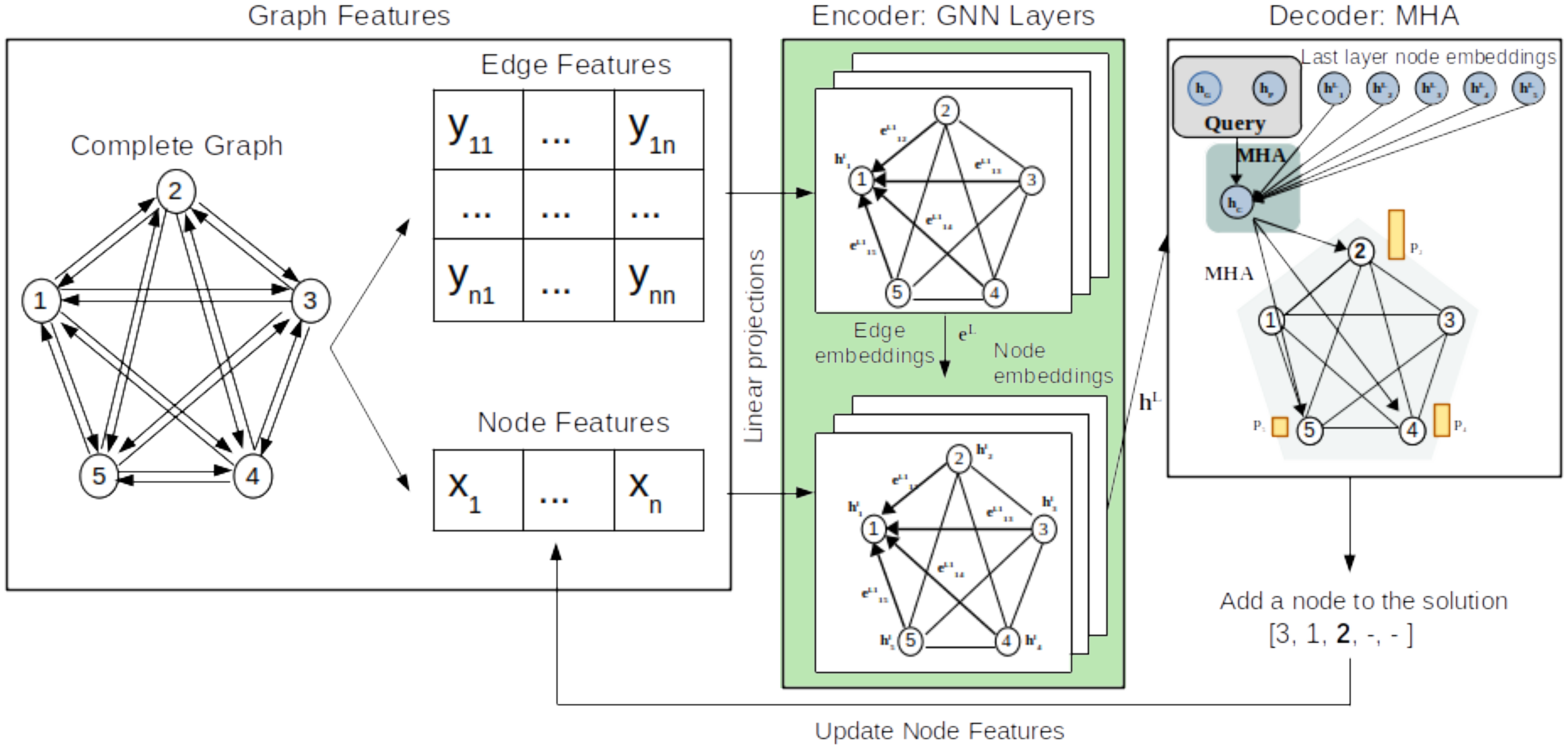
The constructive

A Neural Combinatorial Optimization approach

Encoder: GNN layers

- From features to node h_i^l and edge e_{ij}^l embeddings.
- Embeddings linear initialization:

$$h_i^{l=1} = x_i^T * A_x + B_x$$
$$e_{ij}^{l=1} = y_{ij}^T * A_y + B_y$$



Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2024). **Applicability of Neural Combinatorial Optimization: A Critical View**. ACM Trans. on Evolutionary Learning and Optimization.

The constructive

A Neural Combinatorial Optimization approach

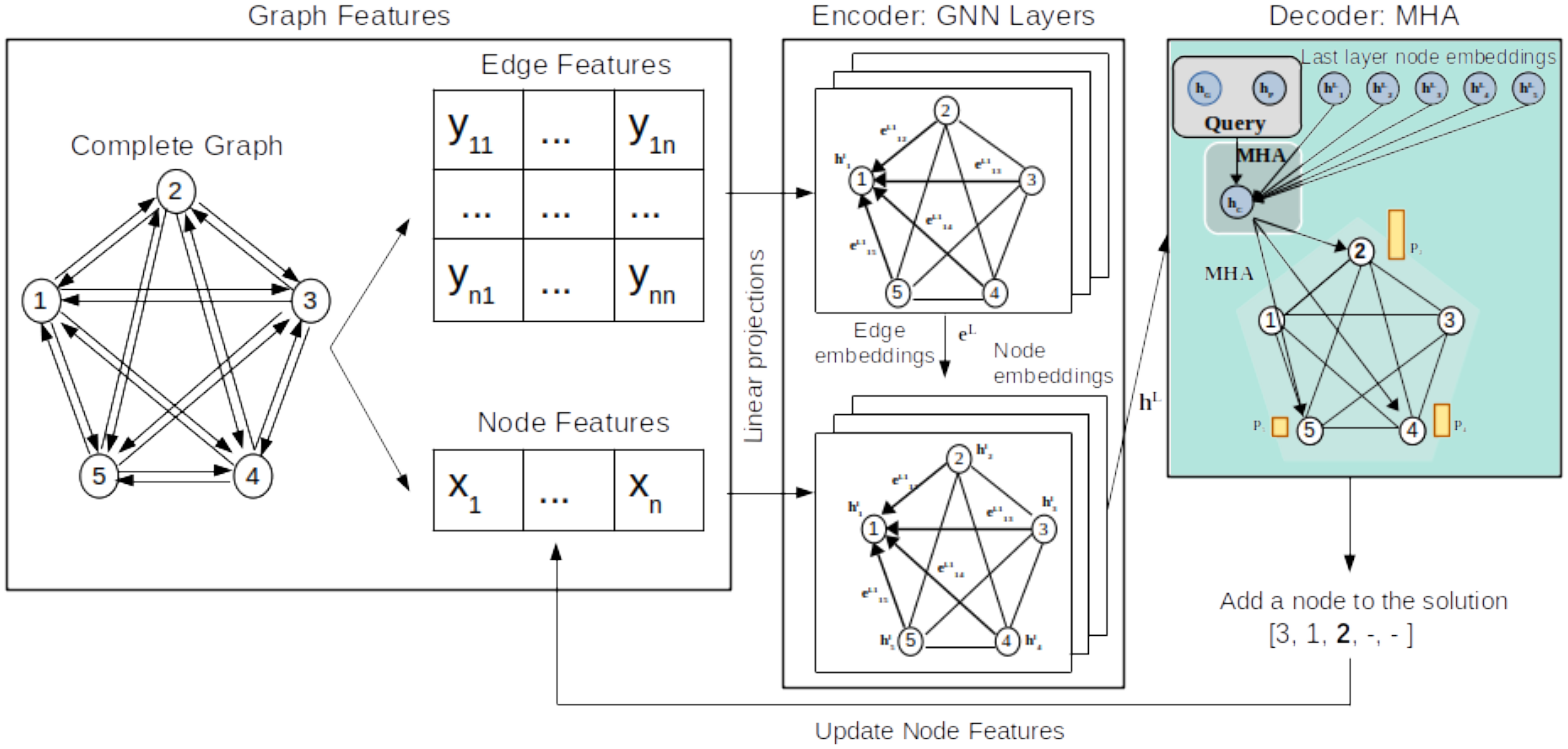
Decoder:

- Multi-Head Attention was used.
- Later, tests showed MLP performed equally.

Learning:

- REINFORCE algorithm.
- Fundamentals:

$$\mathcal{L}(\theta\,|\,s) = \mathbb{E}_{p_\theta(\pi|s)}[-(R(\pi) - b(s))\log p_\theta(\pi\,|\,s)]$$



Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2024). **Applicability of Neural Combinatorial Optimization: A Critical View**. ACM Trans. on Evolutionary Learning and Optimization.

The constructive

A Neural Combinatorial Optimization approach

Performance results

Table 2. LOP. Analysis of the performance using instance sizes the model has been trained with. The given value is the average and standard deviation gap (%) to the best known value for 1000 instances over 5 different executions. Lower is better. Non-optimal results from the exact method are marked with *.

Method	n=20	n=30	n=40	n=50
Exact (SCIP)	**0.00** ± 0.00%	**0.00** ± 0.00%	**0.00** ± 0.00%	1.11 ± 0.50%*
MA	**0.00** ± 0.00%	**0.00** ± 0.00%	**0.00** ± 0.00%	**0.00** ± 0.00%
Becker	3.38 ± 0.00%	3.44 ± 0.00%	3.35 ± 0.00%	3.27 ± 0.00%
GNN	0.24 ± 0.00%	0.29 ± 0.00%	0.41 ± 0.01%	0.48 ± 0.01%
GNN-Pop	0.14 ± 0.00%	0.18 ± 0.00%	0.28 ± 0.00%	0.34 ± 0.00%

Computational cost

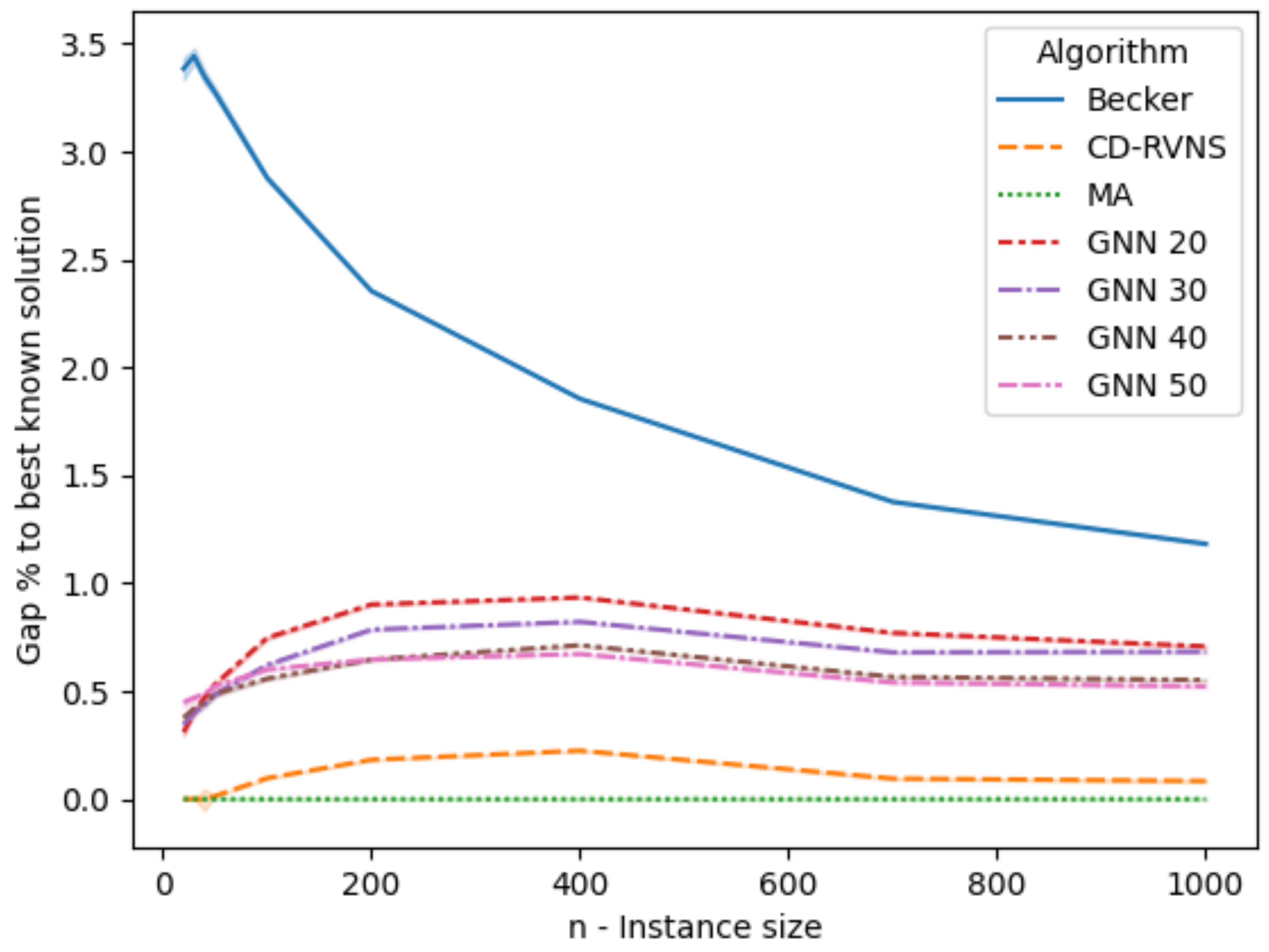
Table 4. LOP. Execution times. The term max denotes that at least one of the executions of the exact algorithm has reached the maximum time (h, m, s refer to hours, minutes and seconds, respectively).

Method	n=20	n=30	n=40	n=50	n=100	n=200	n=1000
Exact (SCIP)	0.52s	13.4s	5.3m	max	max	max	max
MA	0.10s	0.18s	0.29s	0.43s	2.5s	19.6s	20.2m
Becker	0.001s	0.002s	0.004s	0.006s	0.02s	0.10s	3.40s
GNN	0.07s	0.11s	0.16s	0.19s	0.36s	0.74s	1.1m
GNN-training	20h	41h	73h	94h	-	-	-

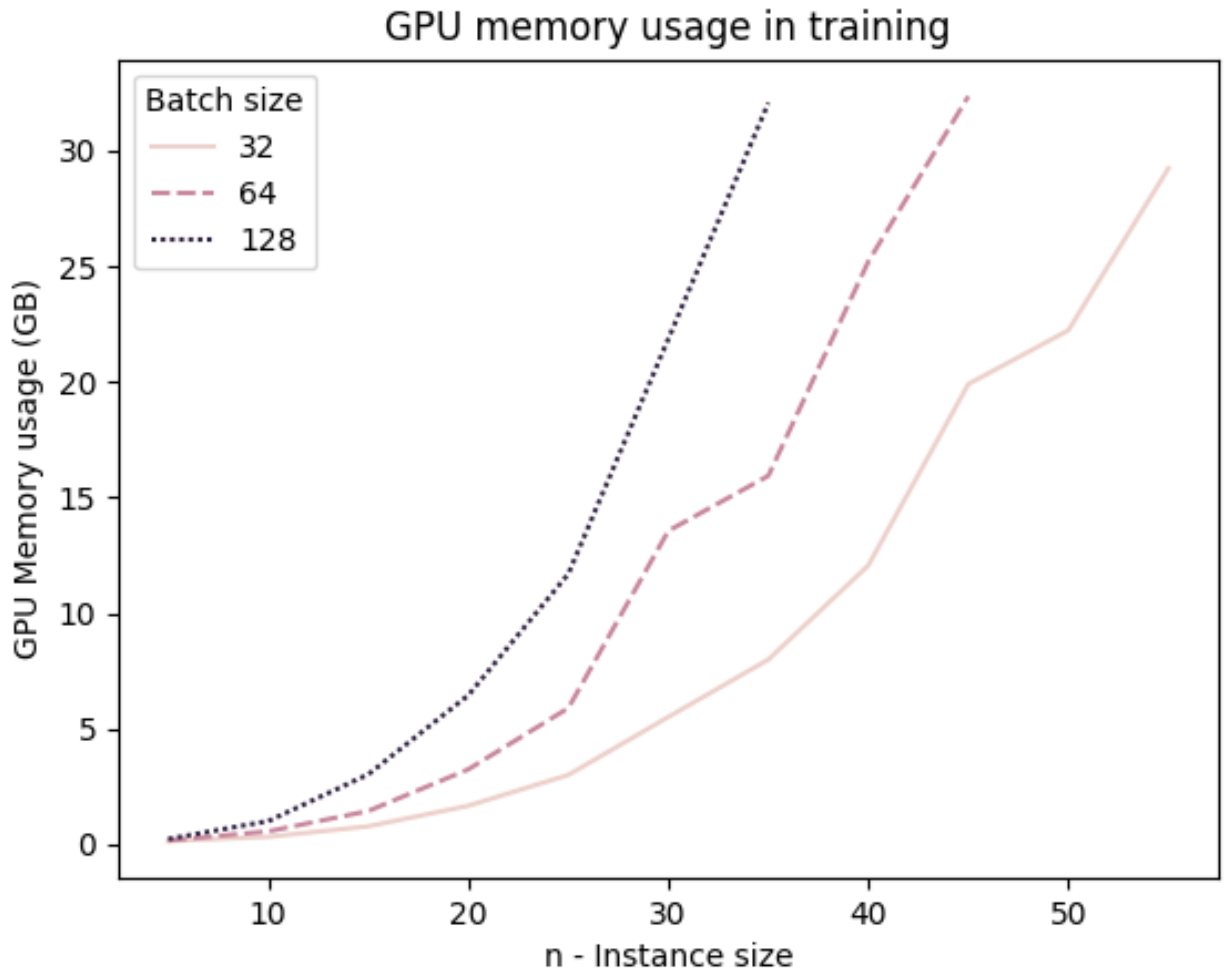
The constructive

A Neural Combinatorial Optimization approach

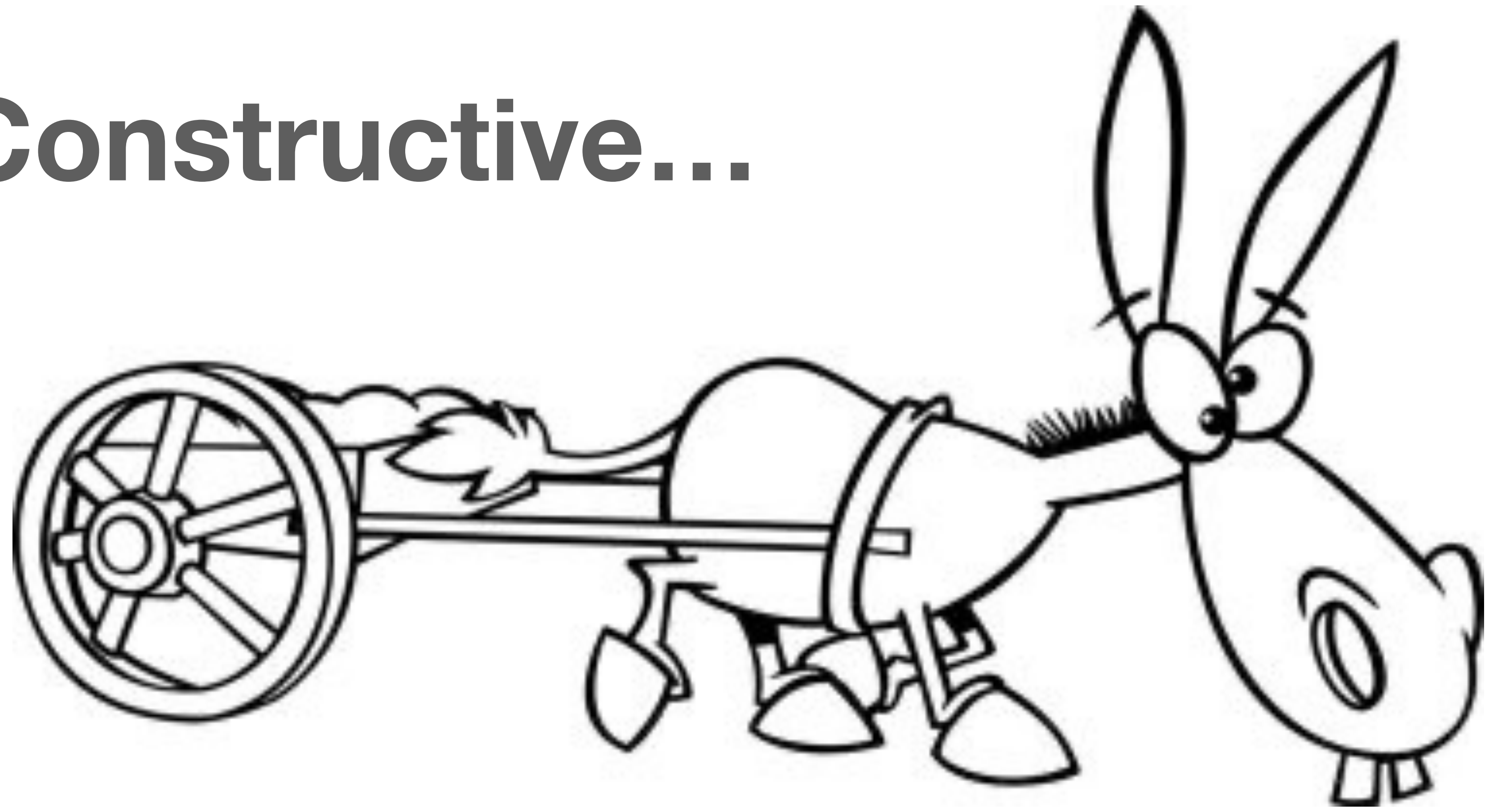
Performance results



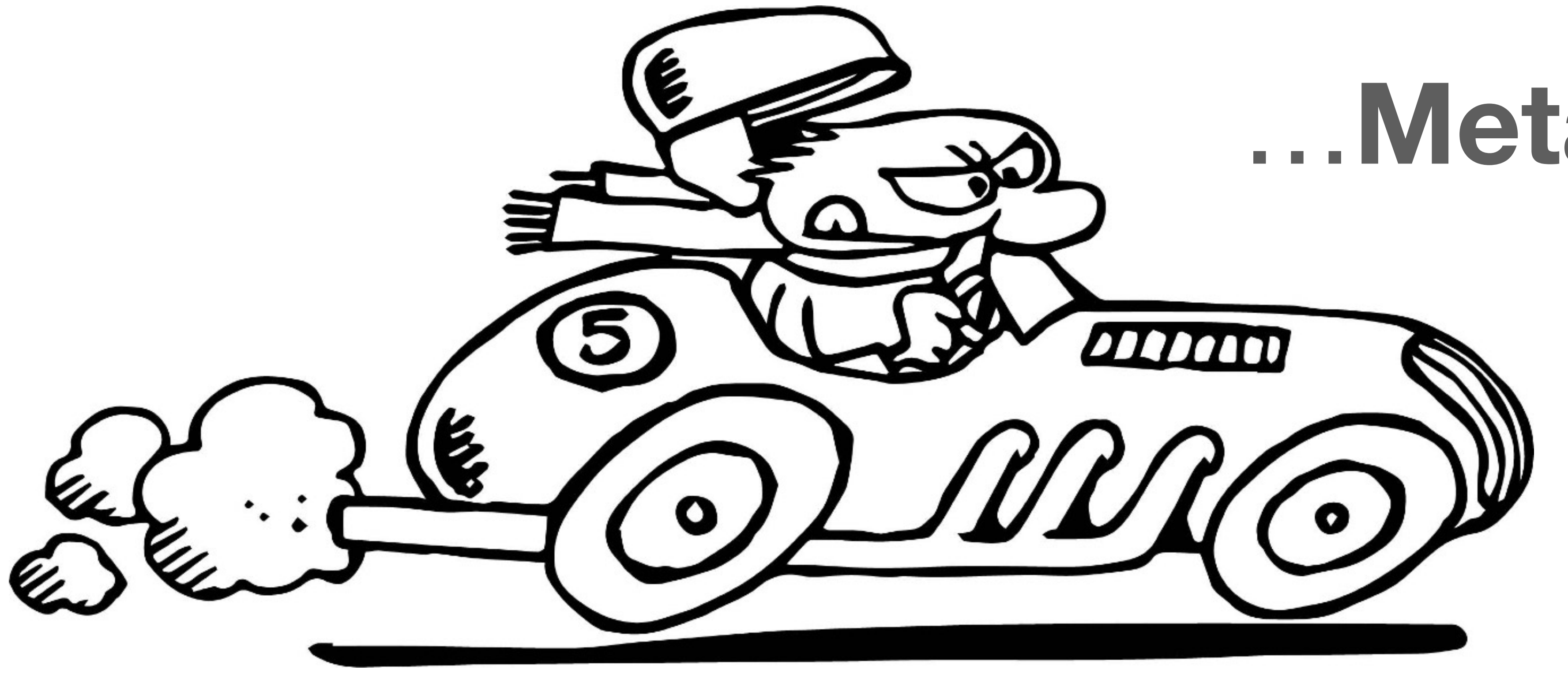
Computational cost



Constructive…

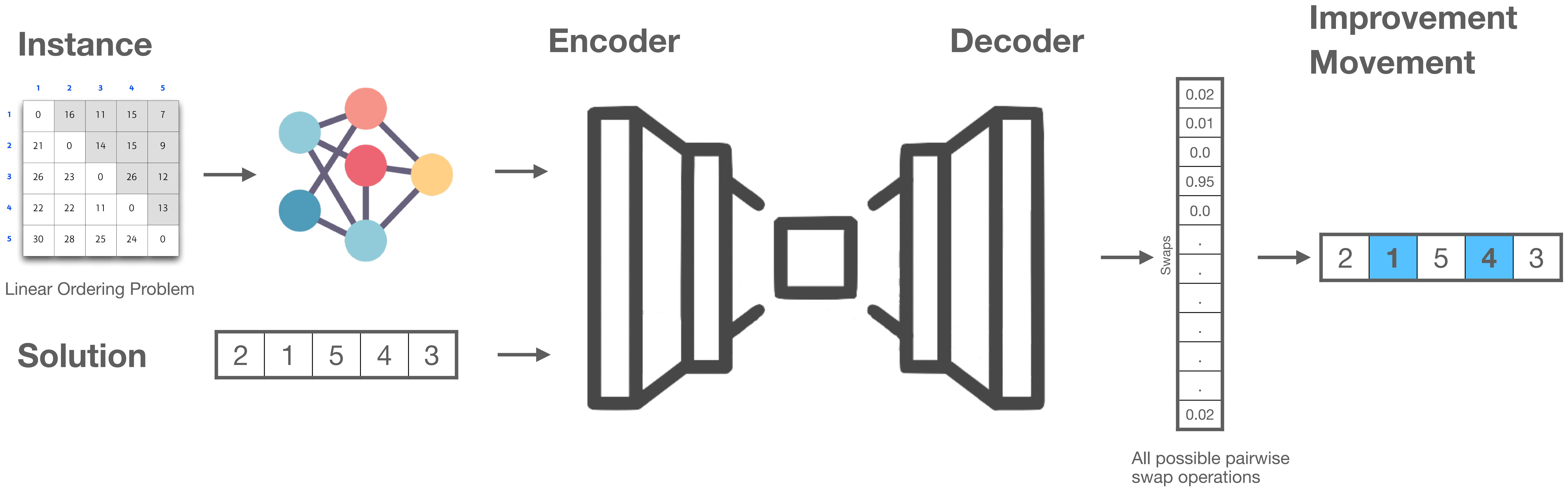


…Metaheuristics



Why not improving?

Don’t want to construct!

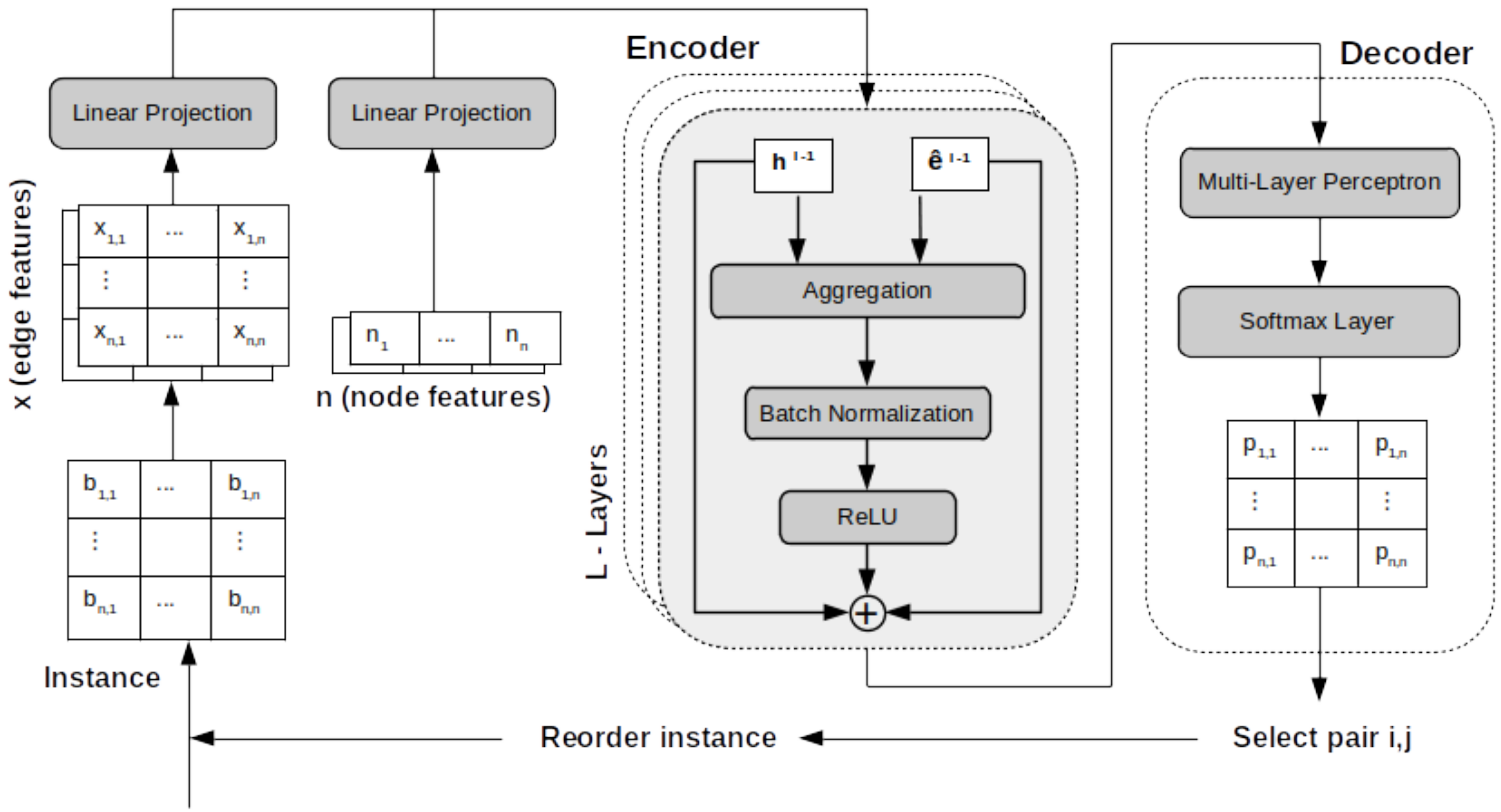


The cost of revising the neighborhood greedily is $O(n^2)$!!

Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2023). **Neural Improvement Heuristics for Graph Combinatorial Optimization Problems**. IEEE Trans. on Neural Networks and Learning Systems.

Why not improving?

Modifications on the architecture



Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2023). **Neural Improvement Heuristics for Graph Combinatorial Optimization Problems**. IEEE Trans. on Neural Networks and Learning Systems.

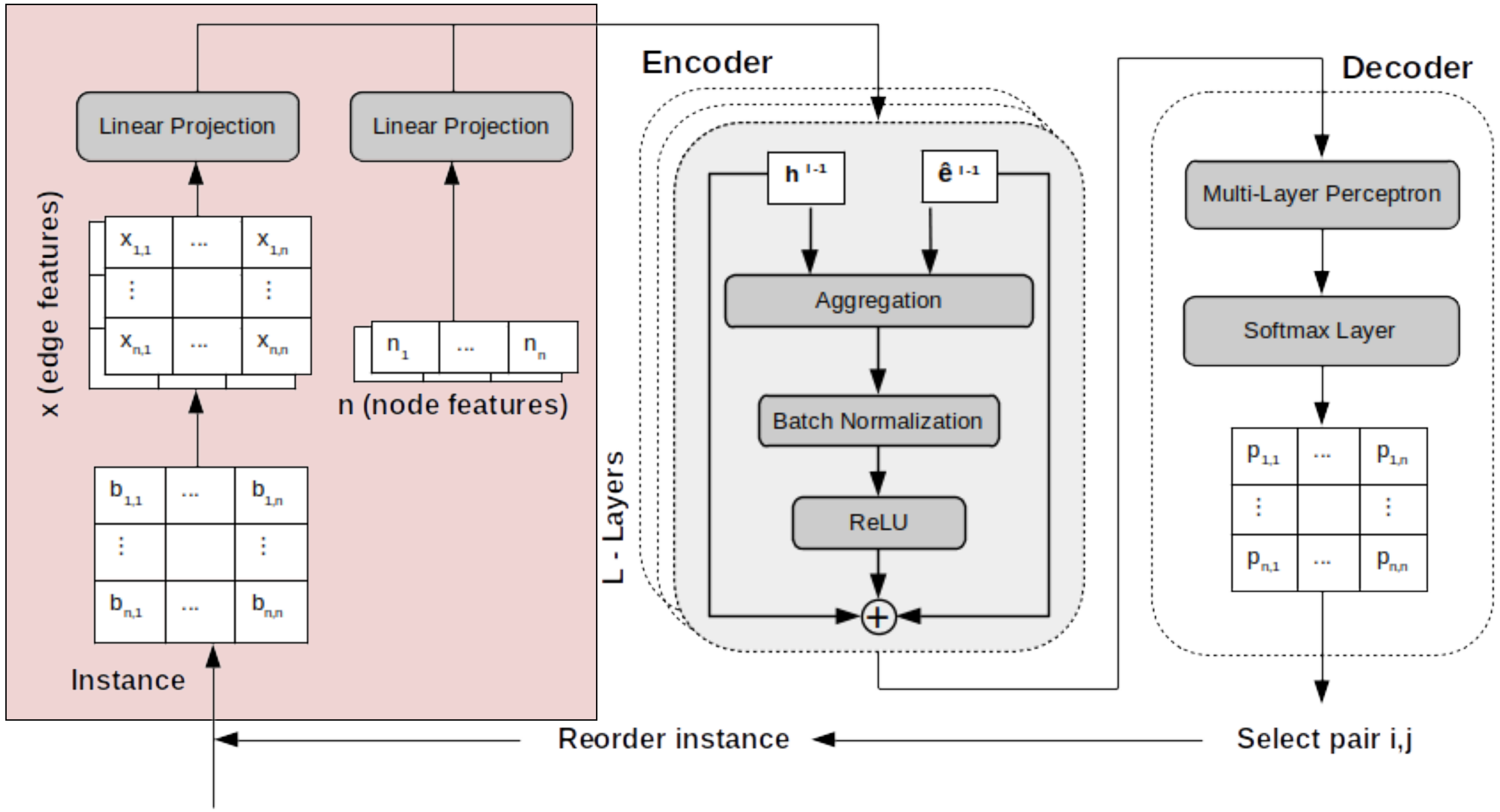
Why not improving?

Modifications on the architecture

Graph features:

- Edge features $x_{ij} \in \mathbb{R}^2$ taken from the instance matrix.
- Node features $\mathbf{n}$, random vector from $\mathbb{R}^N$.
- Embeddings linear initialization:

$$h_i^{l=1} = n_i * V_h + U_h$$
$$e_{ij}^{l=1} = x_{ij} * V_e + U_e$$



Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2023). **Neural Improvement Heuristics for Graph Combinatorial Optimization Problems**. IEEE Trans. on Neural Networks and Learning Systems.

Why not improving?

Modifications on the architecture

Graph features:

- Edge features $x_{ij} \in \mathbb{R}^2$ taken from the instance matrix.
- Node features $\mathbf{n}$, random vector from $\mathbb{R}^N$.
- Embeddings linear initialization:

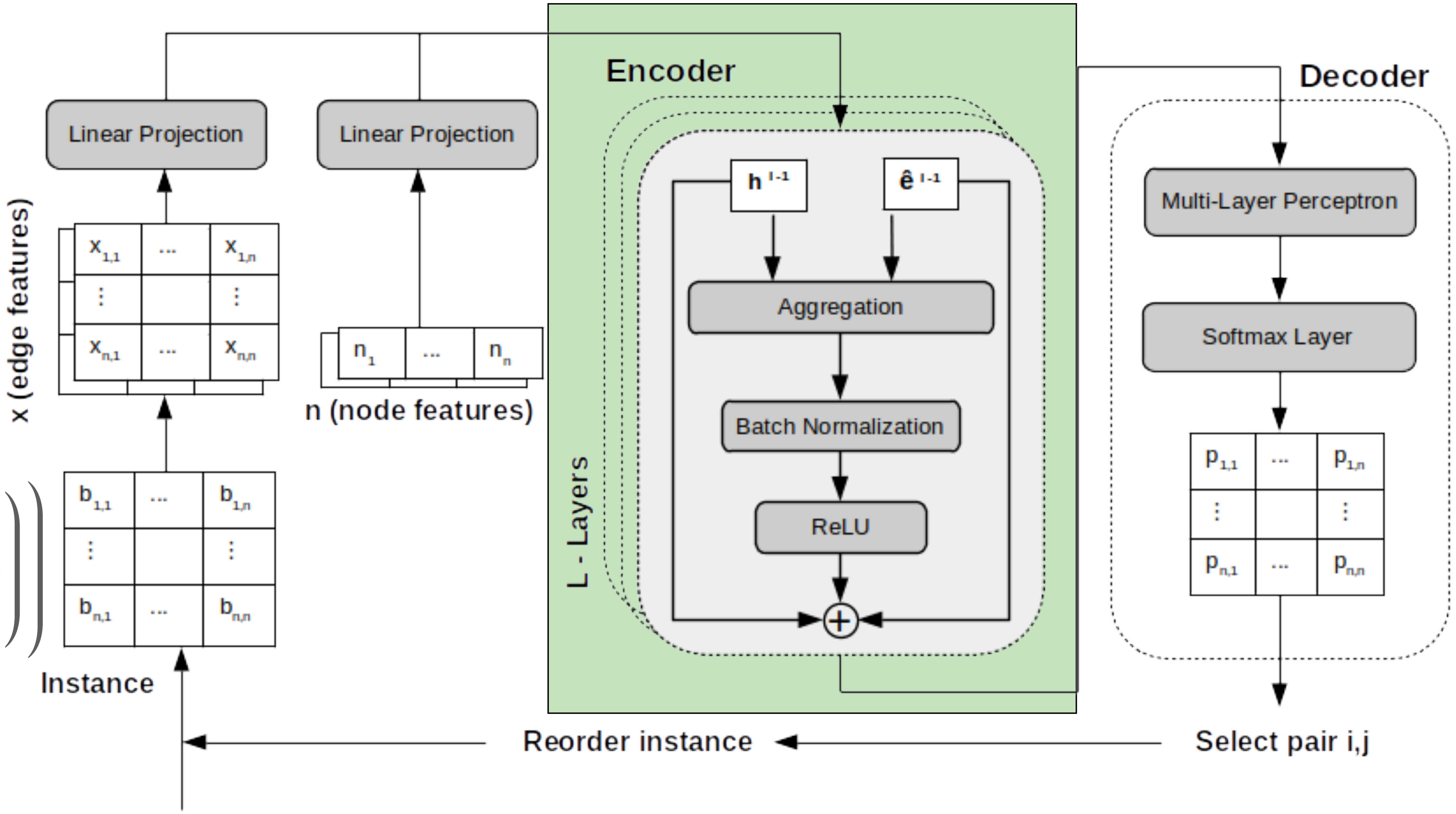
$$h_i^{l=1} = n_i * V_h + U_h$$
$$e_{ij}^{l=1} = x_{ij} * V_e + U_e$$

Encoder: GNN layers

- Message passing:

$$h_i^{l+1} = h_i^l + Relu\left(BN\left(W_1^l h_i^l + \sum_{j=1}^{N} (\sigma(e_{ij}^l) \odot W_2^l h_j^l)\right)\right)$$
$$e_{ij}^{l+1} = e_{ij}^l + Relu\left(BN\left(W_3^l e_{ij}^l + W_4^l h_i^l + W_5^l h_j^l\right)\right)$$

- Result of the last layer edge embeddings: $\mathbf{e}_{ij}^L$



Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2023). **Neural Improvement Heuristics for Graph Combinatorial Optimization Problems**. IEEE Trans. on Neural Networks and Learning Systems.

Why not improving?
Modifications on the architecture

Graph features:

- Edge features $x_{ij} \in \mathbb{R}^2$ taken from the instance matrix.
- Node features $\mathbf{n}$, random vector from $\mathbb{R}^N$.
- Embeddings linear initialization:

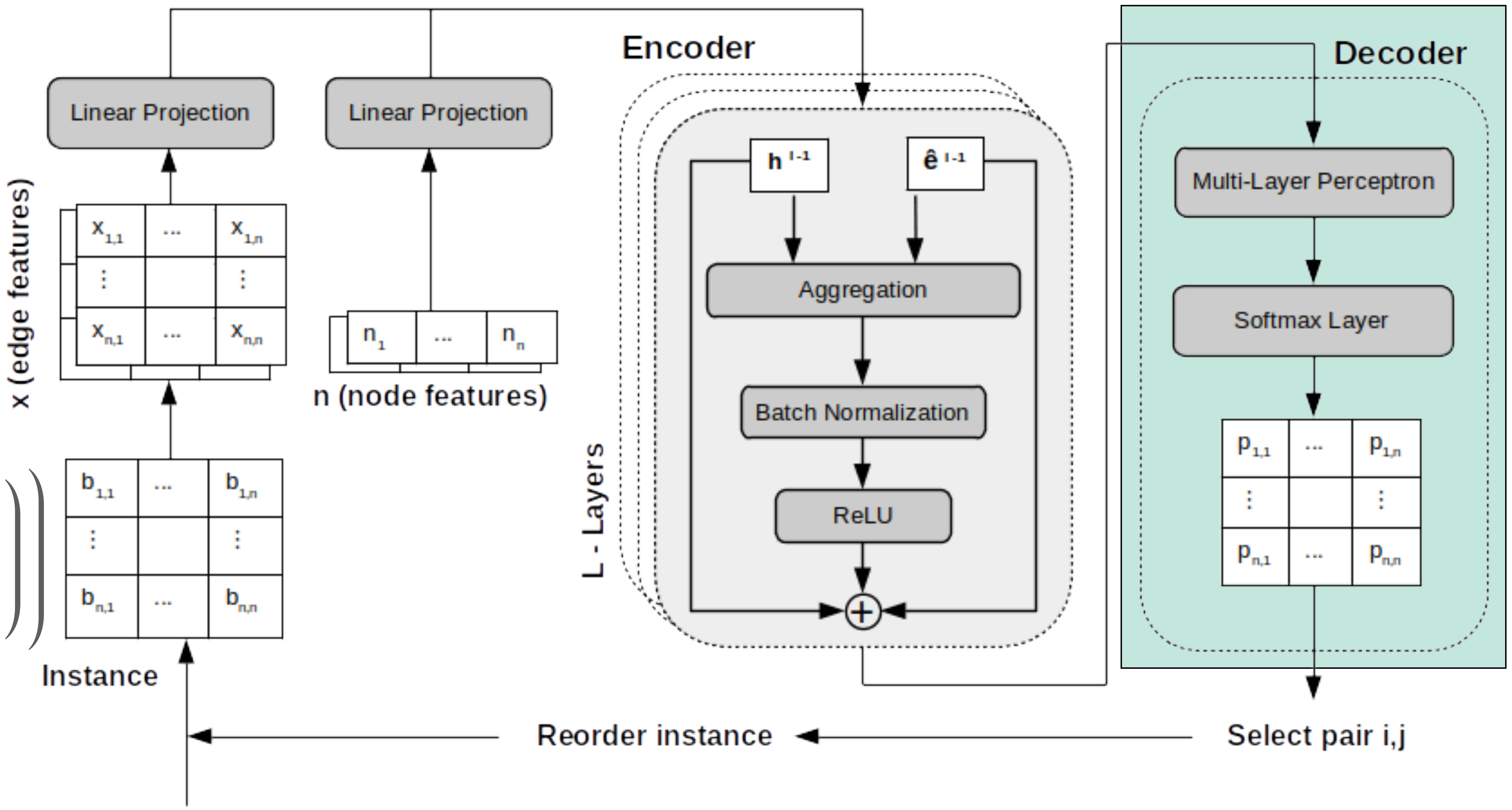
$$h_i^{l=1} = n_i * V_h + U_h$$
$$e_{ij}^{l=1} = x_{ij} * V_e + U_e$$

Encoder: GNN layers

- Message passing:

$$h_i^{l+1} = h_i^l + Relu\left(BN\left(W_1^l h_i^l + \sum_{j=1}^{N} (\sigma(e_{ij}^l) \odot W_2^l h_j^l)\right)\right)$$
$$e_{ij}^{l+1} = e_{ij}^l + Relu\left(BN\left(W_3^l e_{ij}^l + W_4^l h_i^l + W_5^l h_j^l\right)\right)$$

- Result of the last layer edge embeddings: $\mathbf{e}_{ij}^L$



Decoder:

- Multi-Layer Perceptron and softmax layer.

Learning:

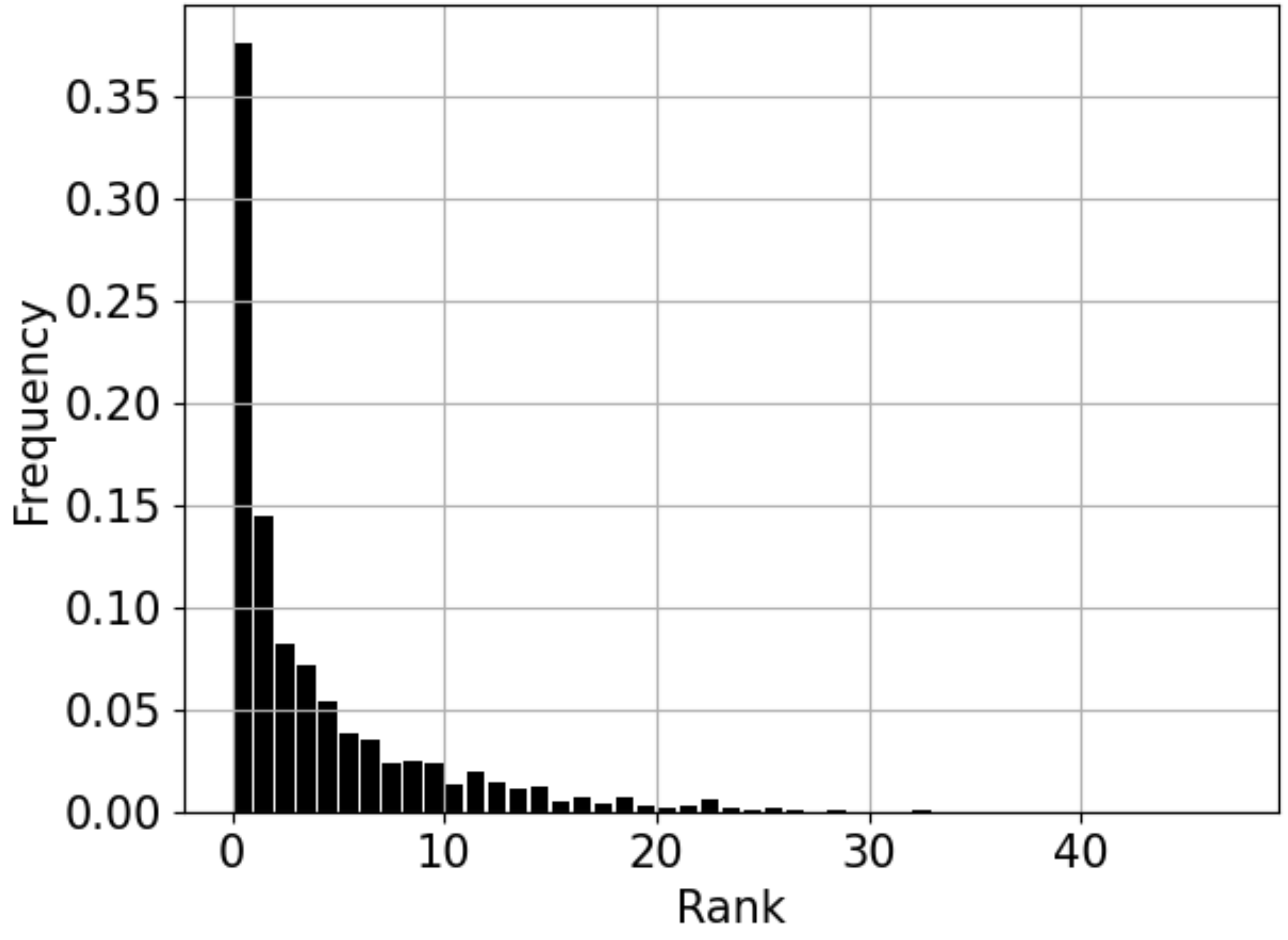
- REINFORCE algorithm: $\mathscr{L}(\theta \mid s) = \mathbb{E}_{p_\theta(s,\omega_t)}[-R_t \log p_\theta(s, \omega_t)]$

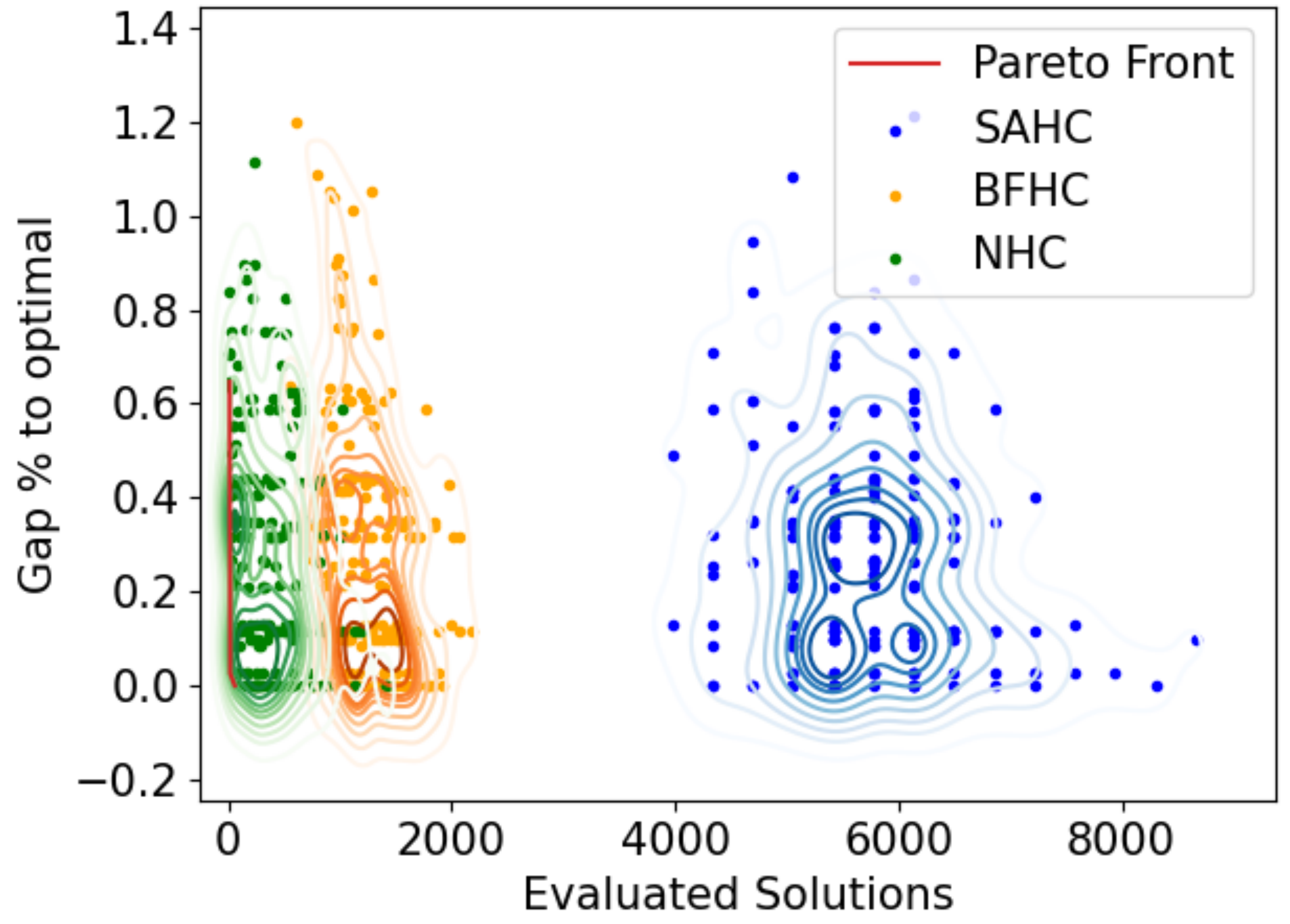
Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2023). **Neural Improvement Heuristics for Graph Combinatorial Optimization Problems**. IEEE Trans. on Neural Networks and Learning Systems.

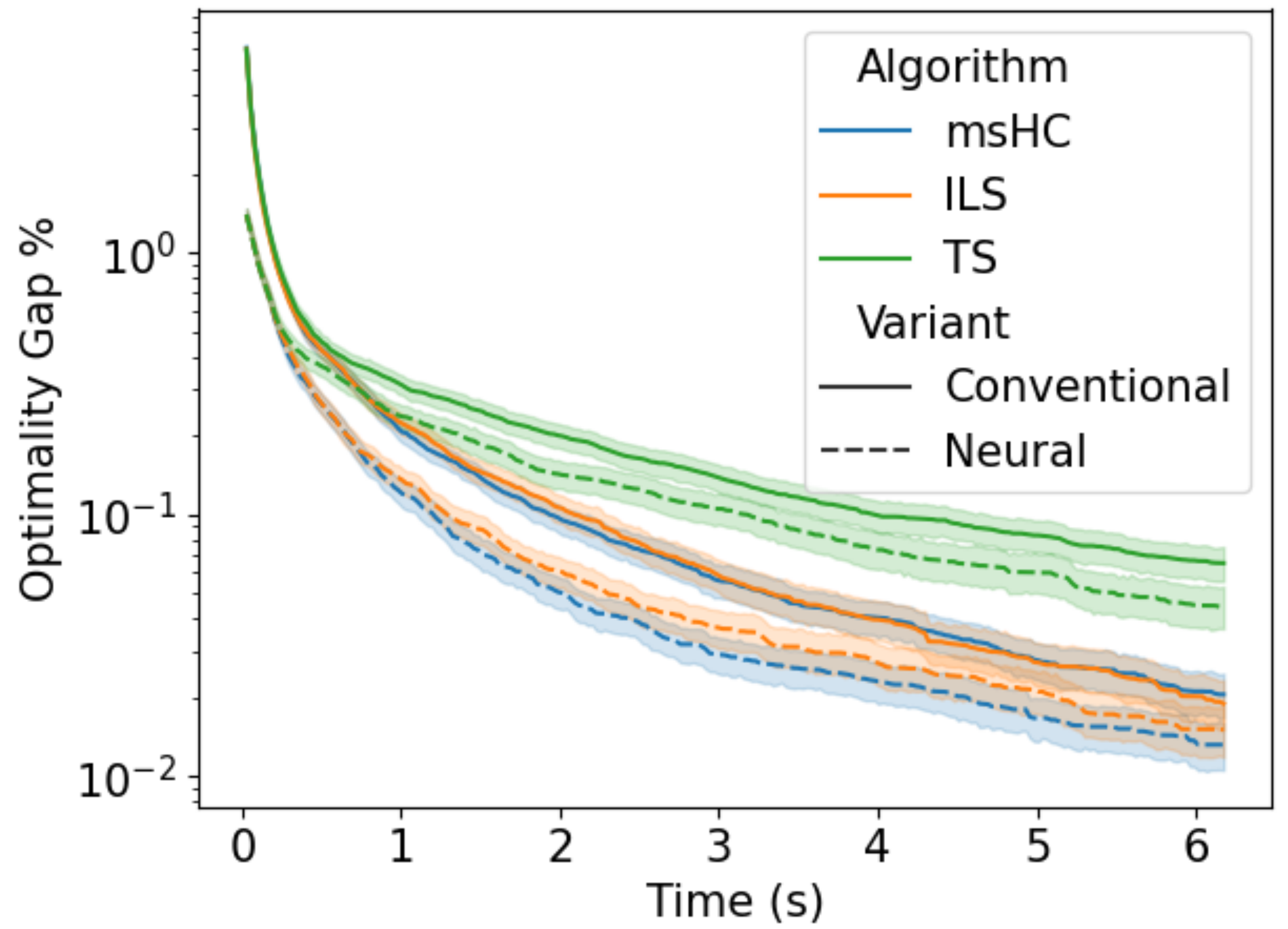
Improvement strategies

Low-complexity local search

Some results:







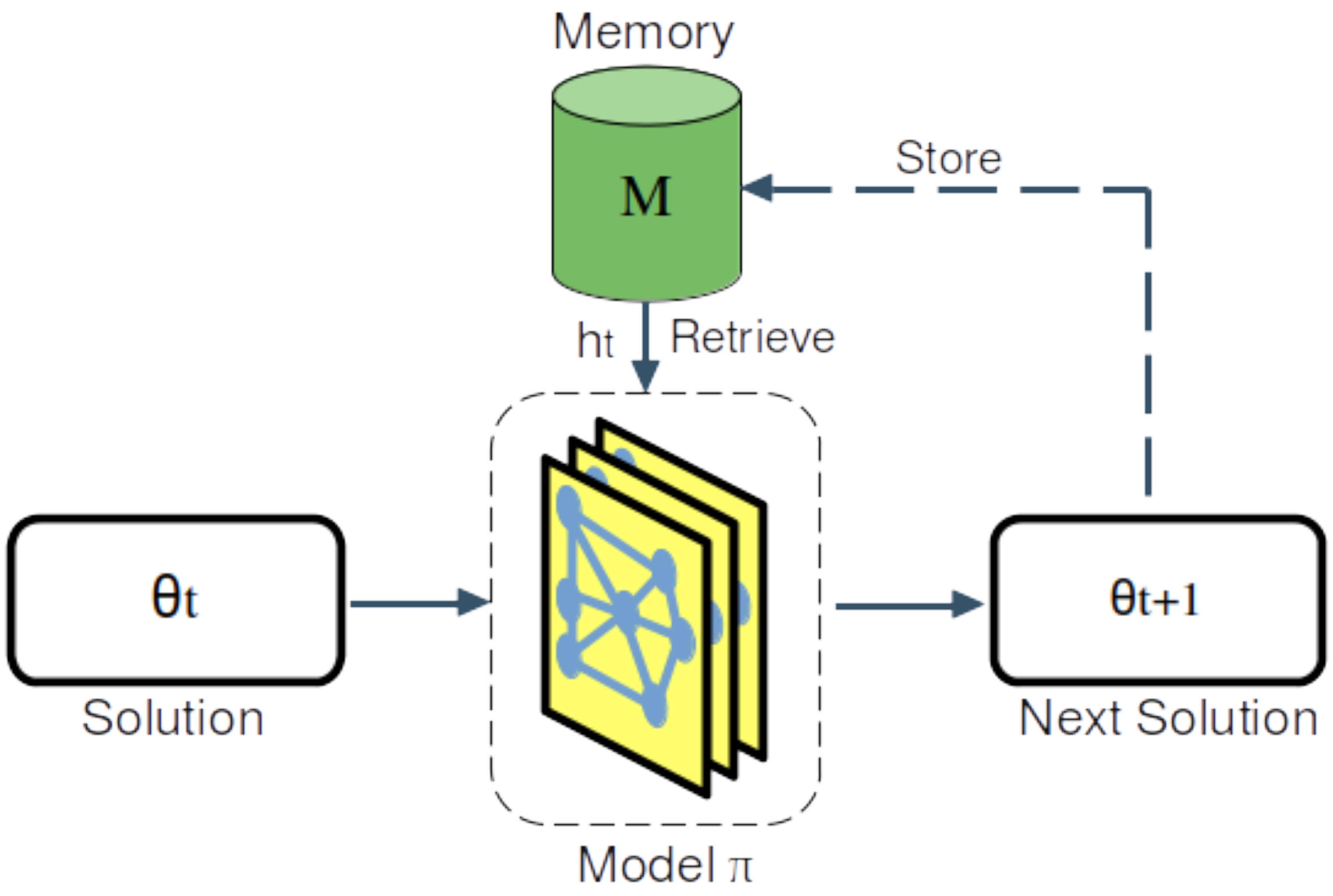
Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2023). **Neural Improvement Heuristics for Graph Combinatorial Optimization Problems**. IEEE Trans. on Neural Networks and Learning Systems.

Downside

Don't want to repeat!

Avoid revisiting

- In the ideal scenario, no solutions would be revisited.
- An internal memory?
- Avoid tabu search (external memory).



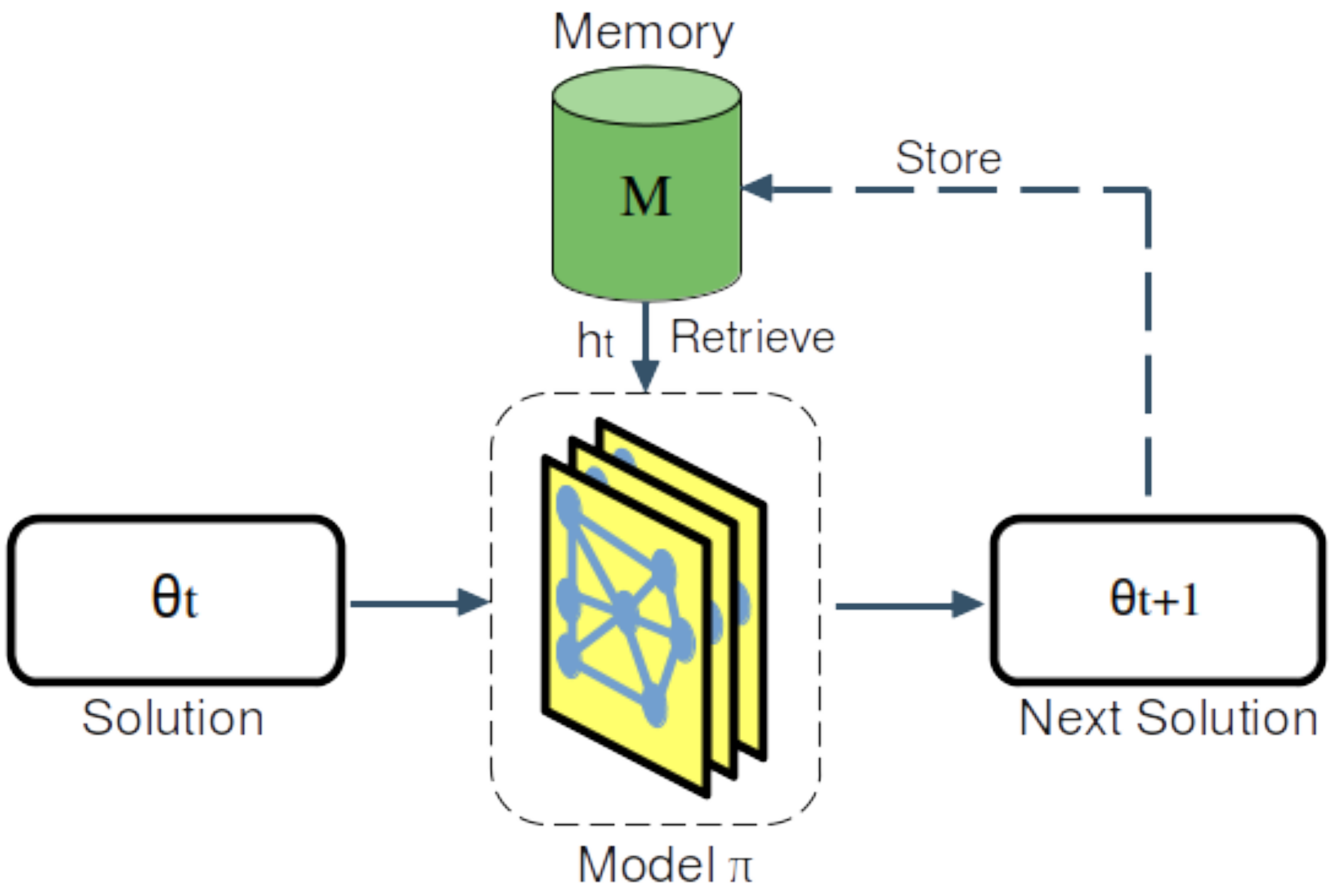
Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2024). **MARCO: A Memory-Augmented Reinforcement Framework for Combinatorial Optimization**. International Joint Conference on Artificial Intelligence.

Incorporating memory

Avoid revisiting

- Memory design dependent on the scheme and the problem.

- Similarity-based search mechanism to retrieve past relevant information.



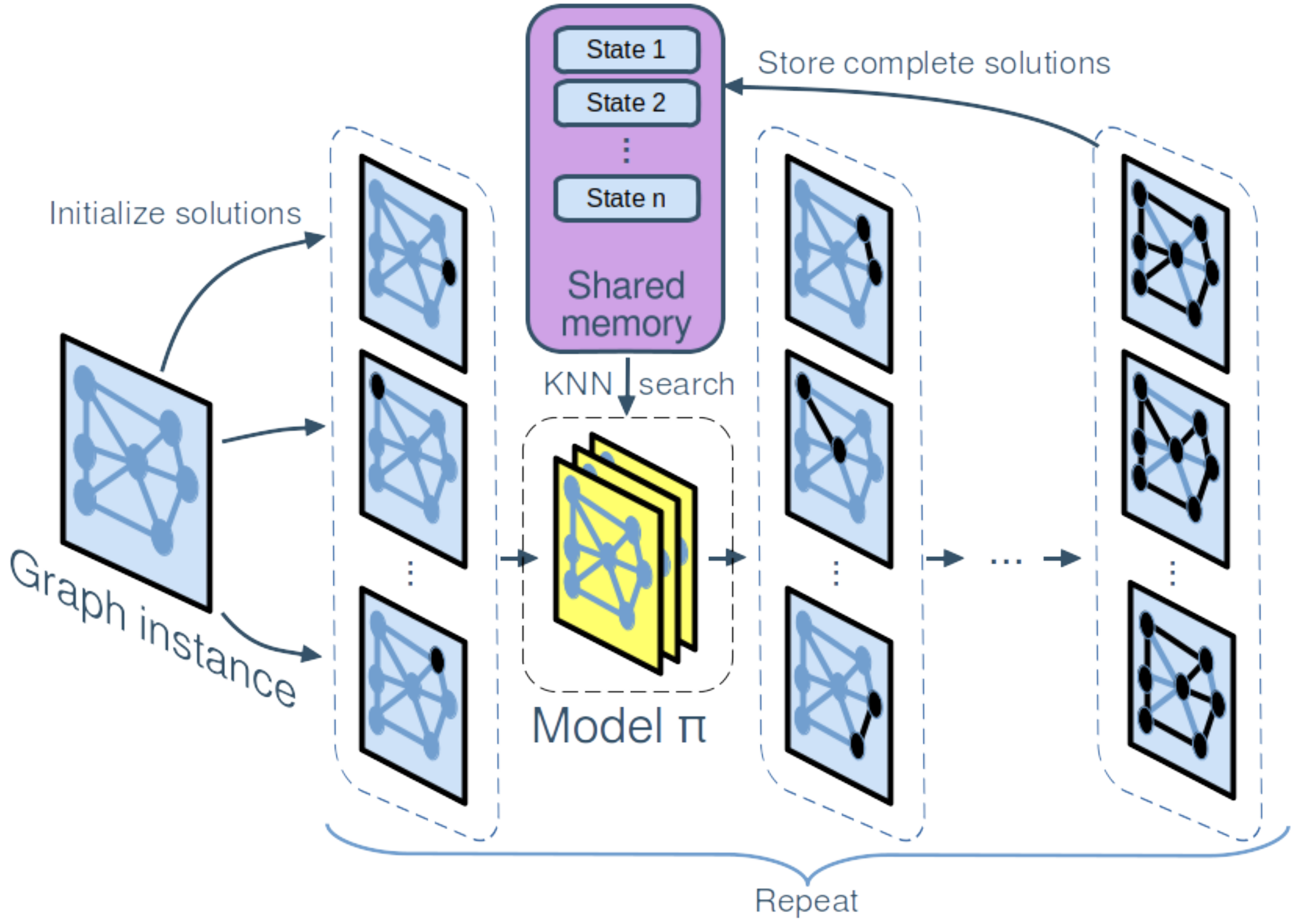
Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2024). **MARCO: A Memory-Augmented Reinforcement Framework for Combinatorial Optimization**. International Joint Conference on Artificial Intelligence.

Incorporating memory

Avoid revisiting

Constructive scheme for permutations problems

- **Records:** visited solutions θ_t.
- For every partial solution, retrieve the allocation of the items to positions in similar solutions.
- **Result** h_t : weighted average of the remaining items that were placed in the k most similar solutions.

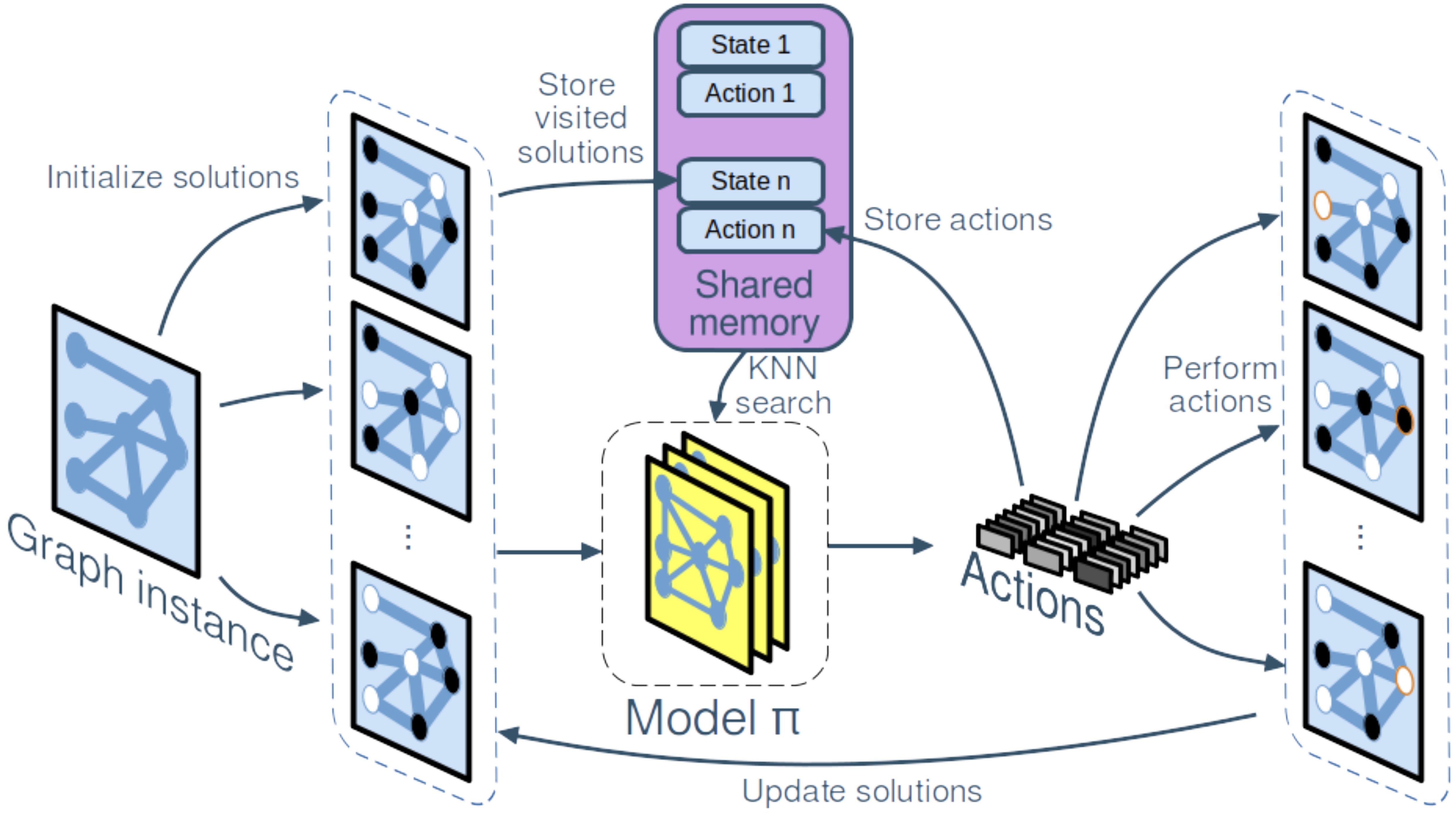


Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2024). **MARCO: A Memory-Augmented Reinforcement Framework for Combinatorial Optimization**. International Joint Conference on Artificial Intelligence.

Incorporating memory
Avoid revisiting

Improvement scheme for binary problems

- **Records**: visited solutions θ_t and adopted action (bit-flip)
- Retrieve the actions performed in similar solutions.
- **Result** h_t : weighted average of the actions that were executed in the k most similar solutions.



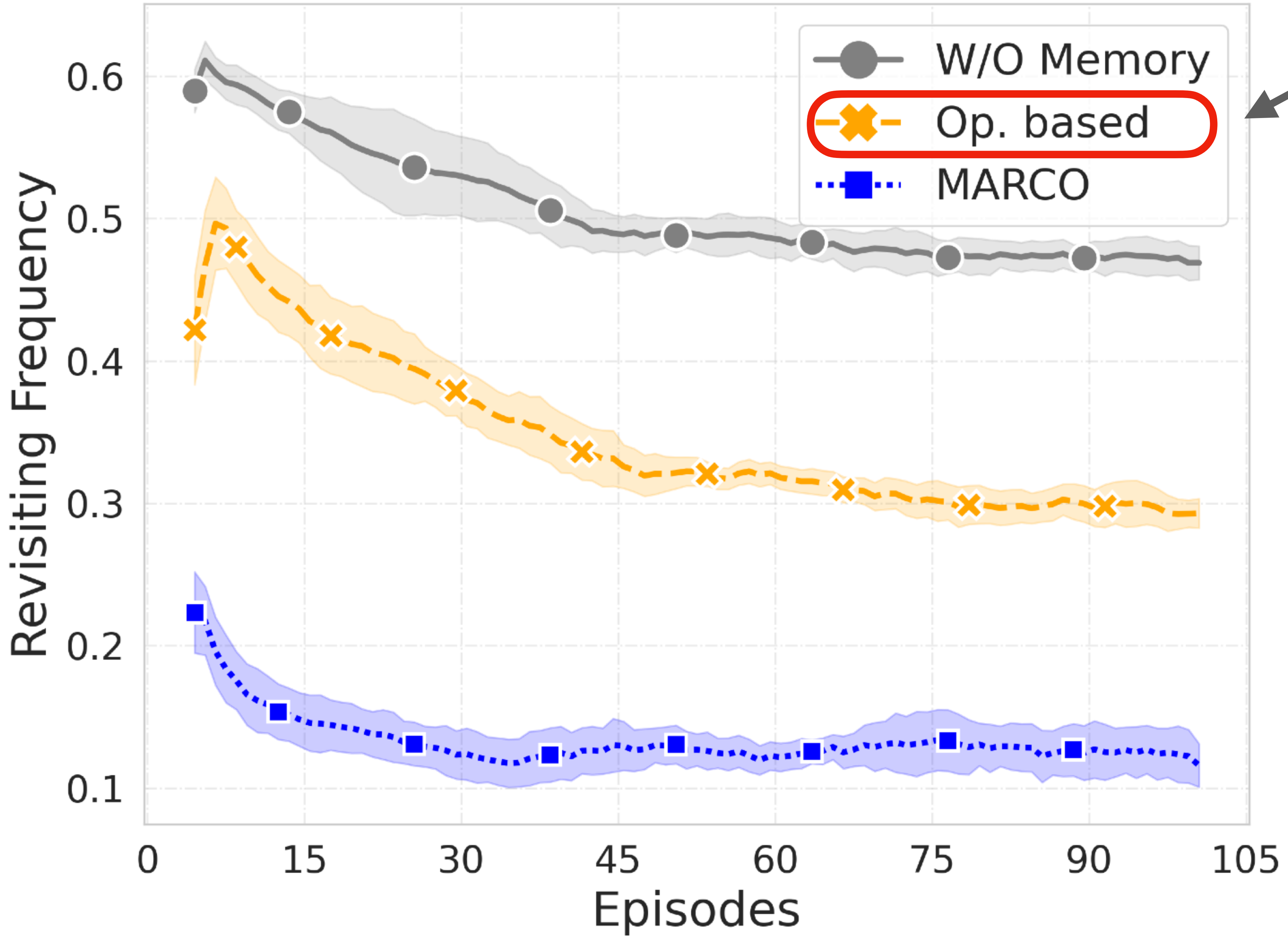
Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2024). **MARCO: A Memory-Augmented Reinforcement Framework for Combinatorial Optimization**. International Joint Conference on Artificial Intelligence.

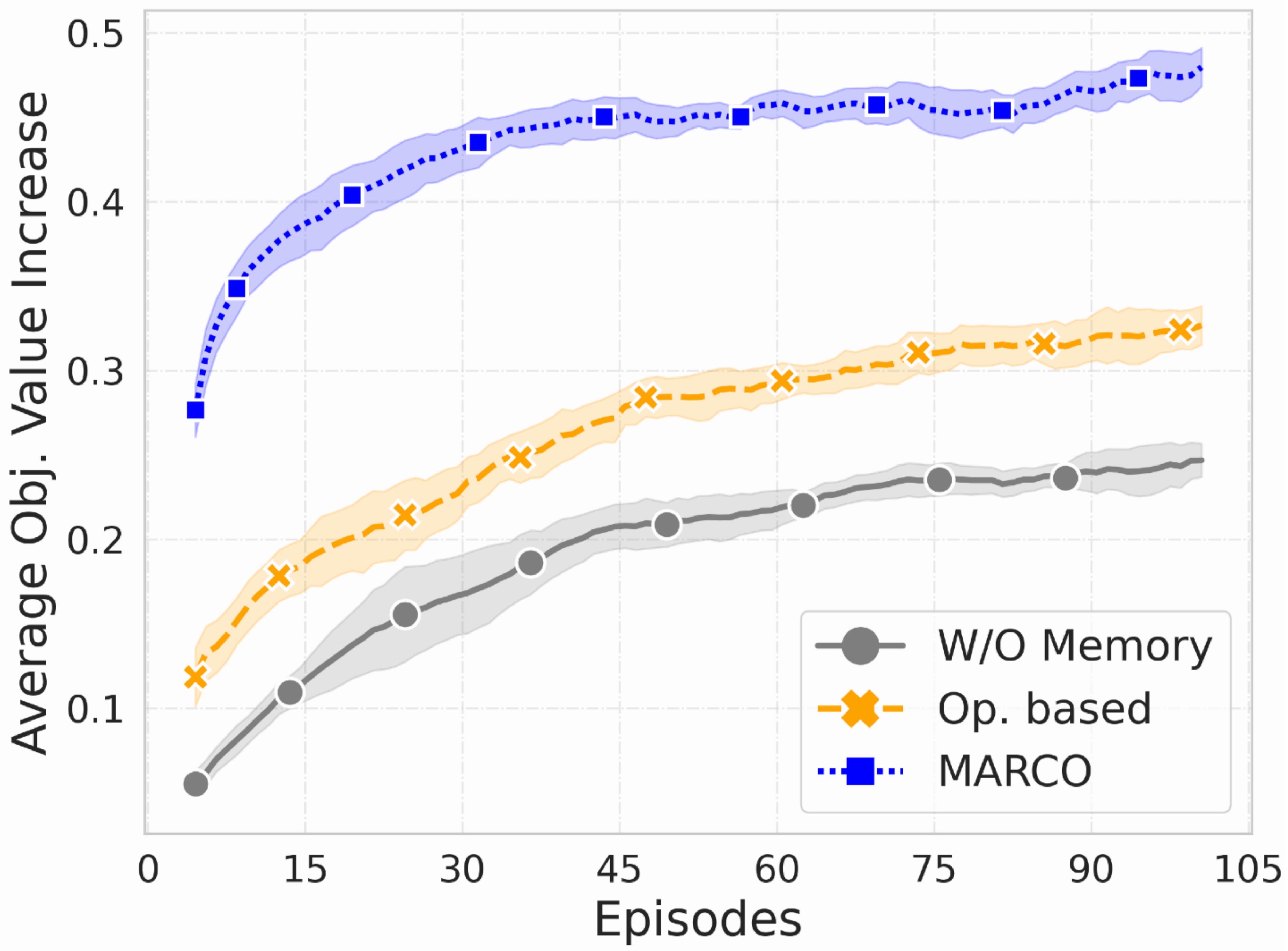
Incorporating memory

Avoid revisiting - did we succeed?

Some results during training:

Avoid previous action



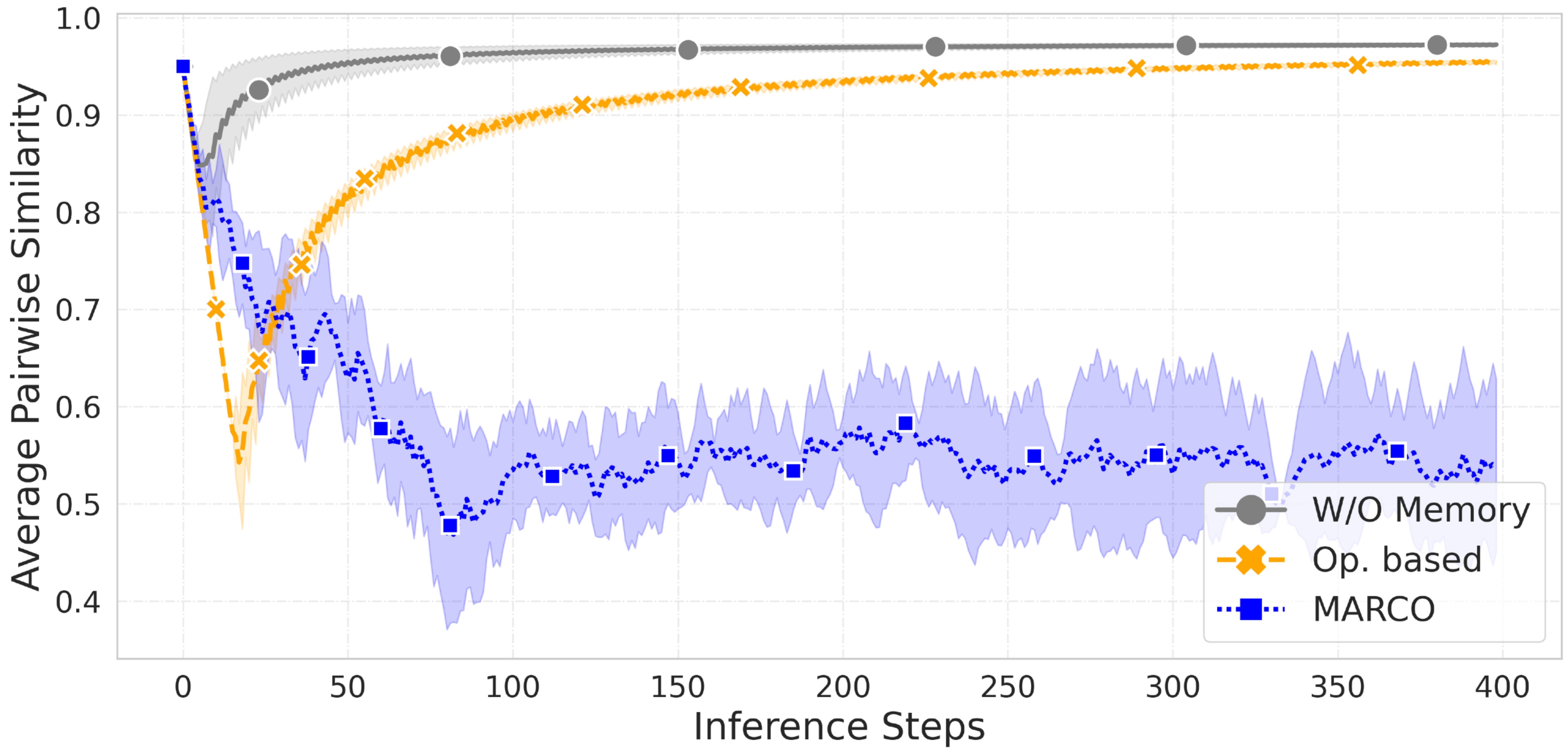


Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2024). **MARCO: A Memory-Augmented Reinforcement Framework for Combinatorial Optimization**. International Joint Conference on Artificial Intelligence.

Incorporating memory

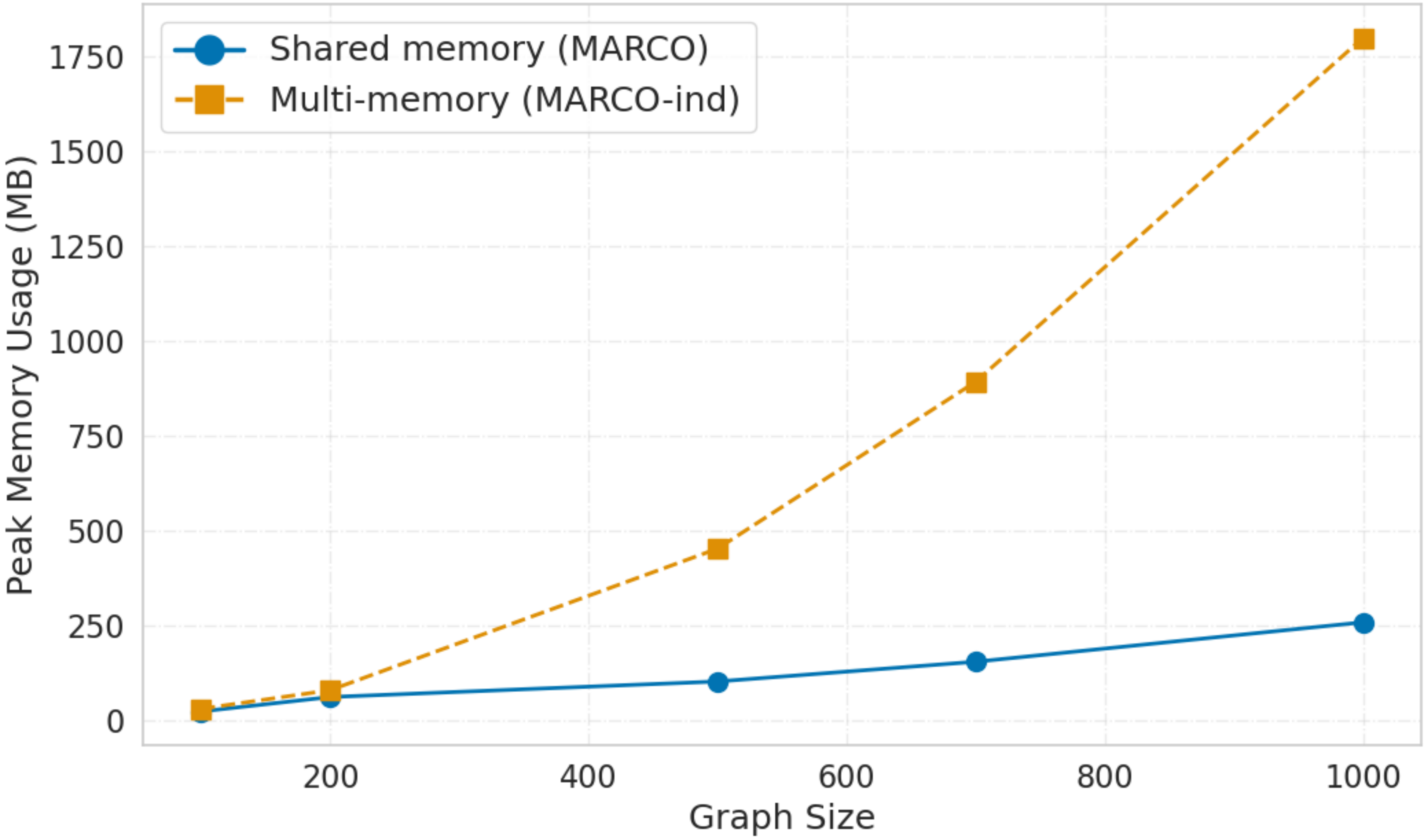
Avoid revisiting - did we succeed?

Some results during inference:



Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2024). **MARCO: A Memory-Augmented Reinforcement Framework for Combinatorial Optimization**. International Joint Conference on Artificial Intelligence.

Incorporating memory
Memory complexity



Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2024). **MARCO: A Memory-Augmented Reinforcement Framework for Combinatorial Optimization**. International Joint Conference on Artificial Intelligence.

Incorporating memory

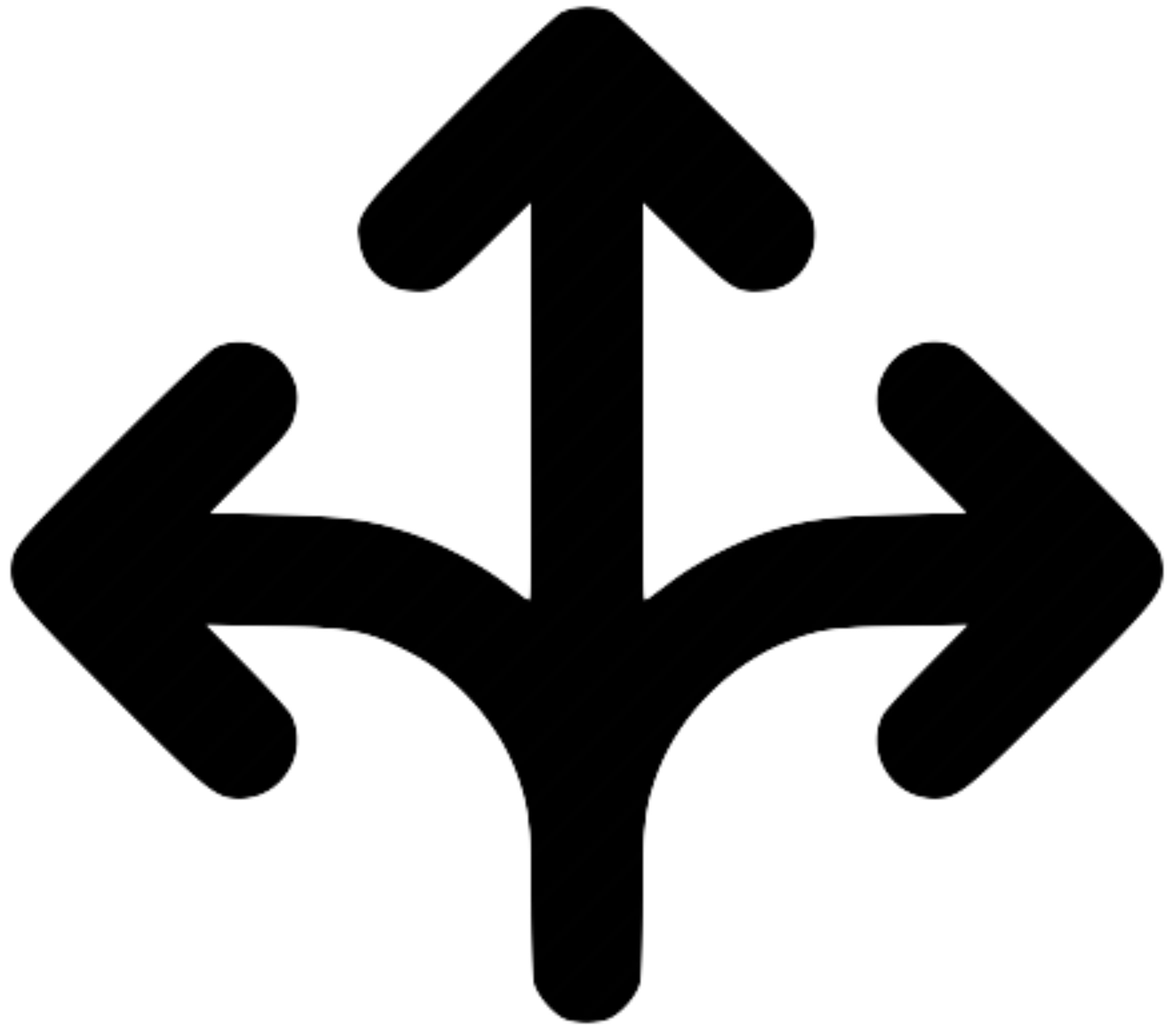
Implicitly population-based?

"In the improvement scheme, <u>multiple threads were run, simultaneously, sharing the same memory".</u>

Garmendia, A.I., Ceberio, J., and Mendiburu, A. (2024). **MARCO: A Memory-Augmented Reinforcement Framework for Combinatorial Optimization**. International Joint Conference on Artificial Intelligence.

Food for thought

- Close to metaheuristics' performance.
- Population-based approaches look the next step.
- What if the problem cannot be represented as a graph?Which encoder should we use?
- Get closer to the real-world practitioners. At this point we are even further.
- Greener algorithms. Prohibitive energy consumption.



Universidad del País Vasco Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

NP-hard problems still persist…

… despite Deep Learning don't you think?

Josu Ceberio, May 2024