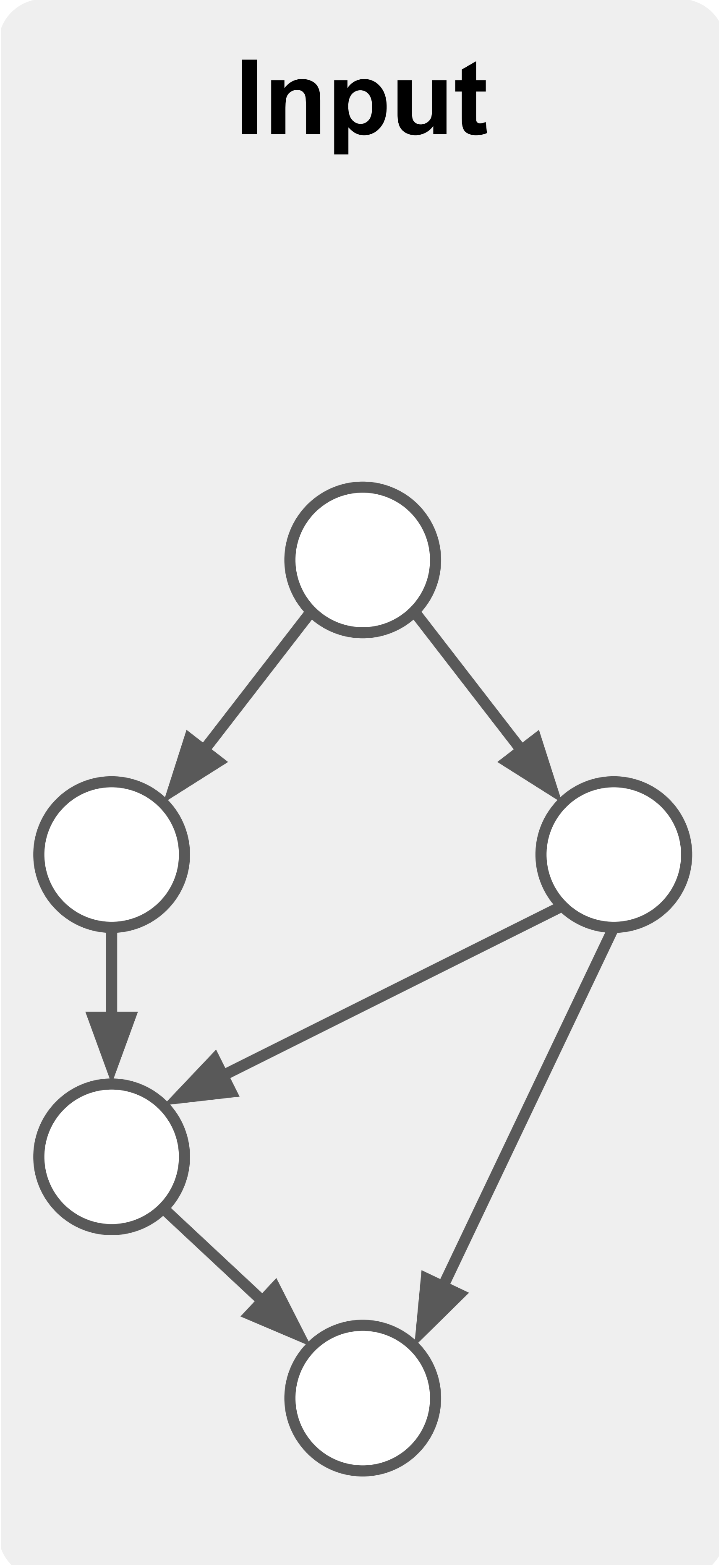
REGAL: Transfer Learning For Fast Optimization of Computation Graphs

Aditya Paliwal,Felix Gimeno,Vinod Nair,Yujia Li,Miles Lubin,Pushmeet Kohli,Oriol Vinyals

Shuntian Liu 23/11/2022

Optimising Computation Graphs

- Device placement
- Scheduling
- NP-hard

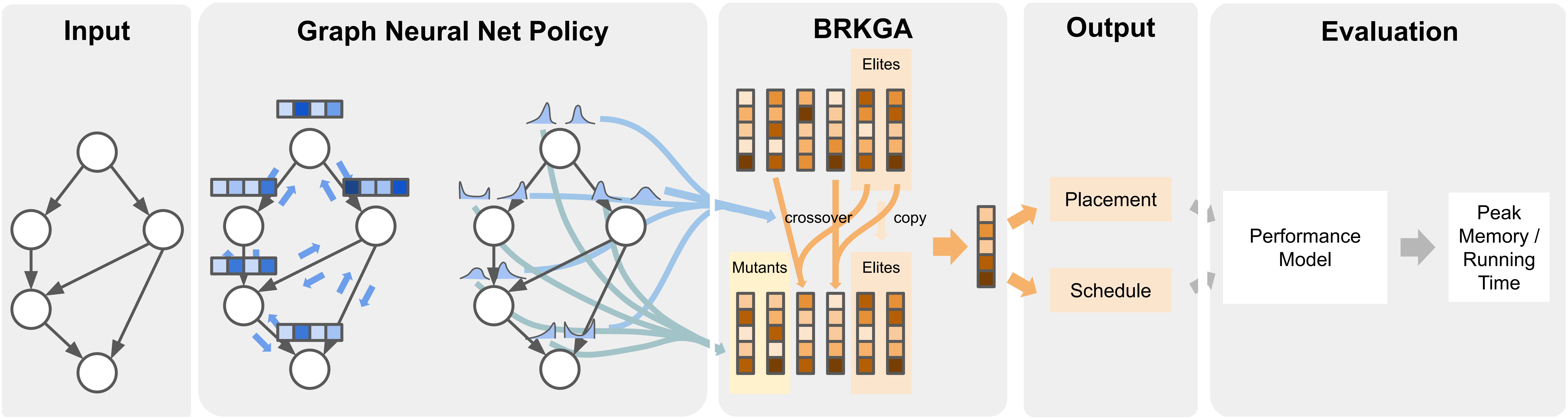


Motivation & Related Work

- AutoTVM
  - No transfer across models
- Learning to super optimise programs
  - Handcrafted instance & small graphs
- Parallel task scheduling
  - Traditionally not learning-based
- Little attempt to learn to transfer to new graphs on a large scale

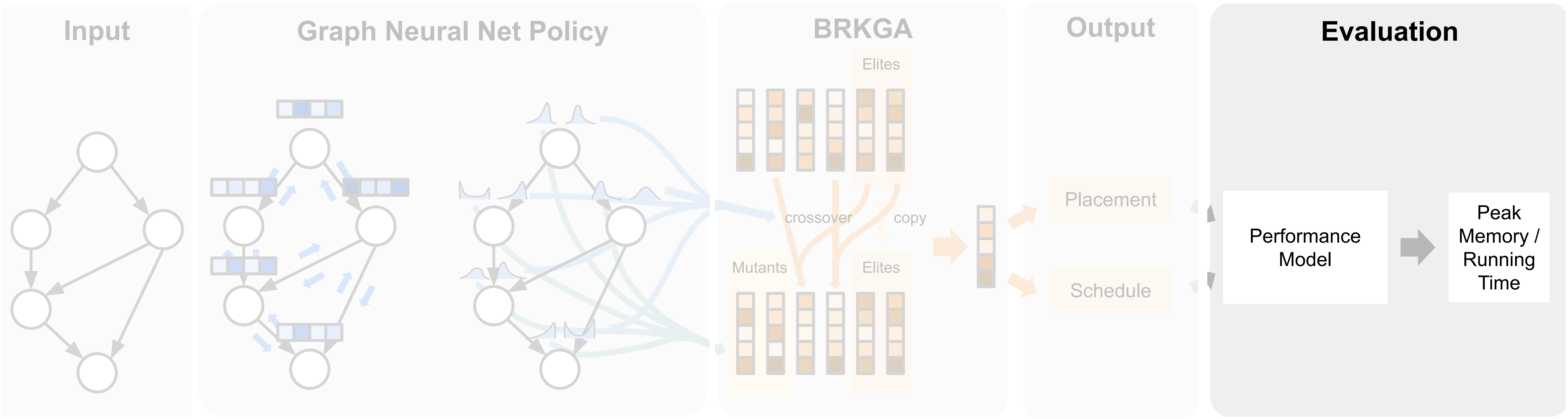
REGAL: Transfer Learning For Fast Optimization of Computation Graphs

Pipeline


Input
Graph Neural Net Policy
BRKGA
Elites
crossover
copy
Mutants
Elites
Output
Placement
Schedule
Evaluation
Performance Model
Peak Memory / Running Time

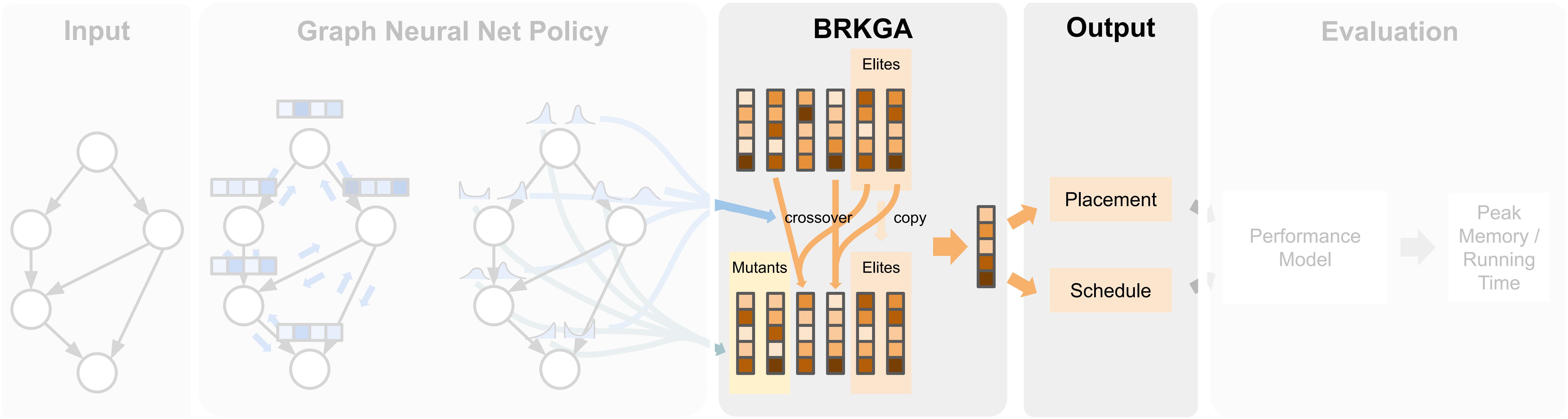
Objective

Peak memory minimisation



BRKGA

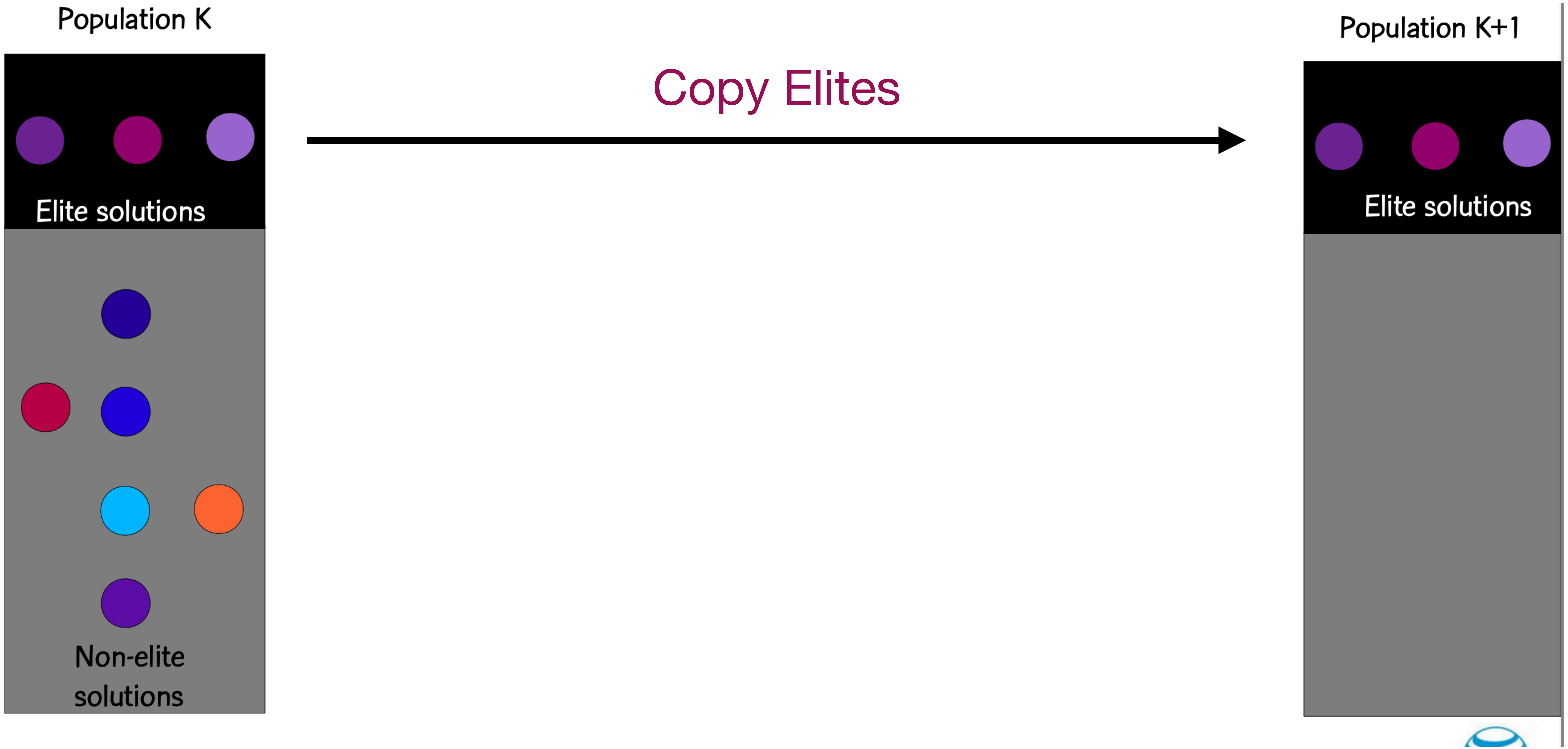
Biased random key genetic algorithm



BRKGA

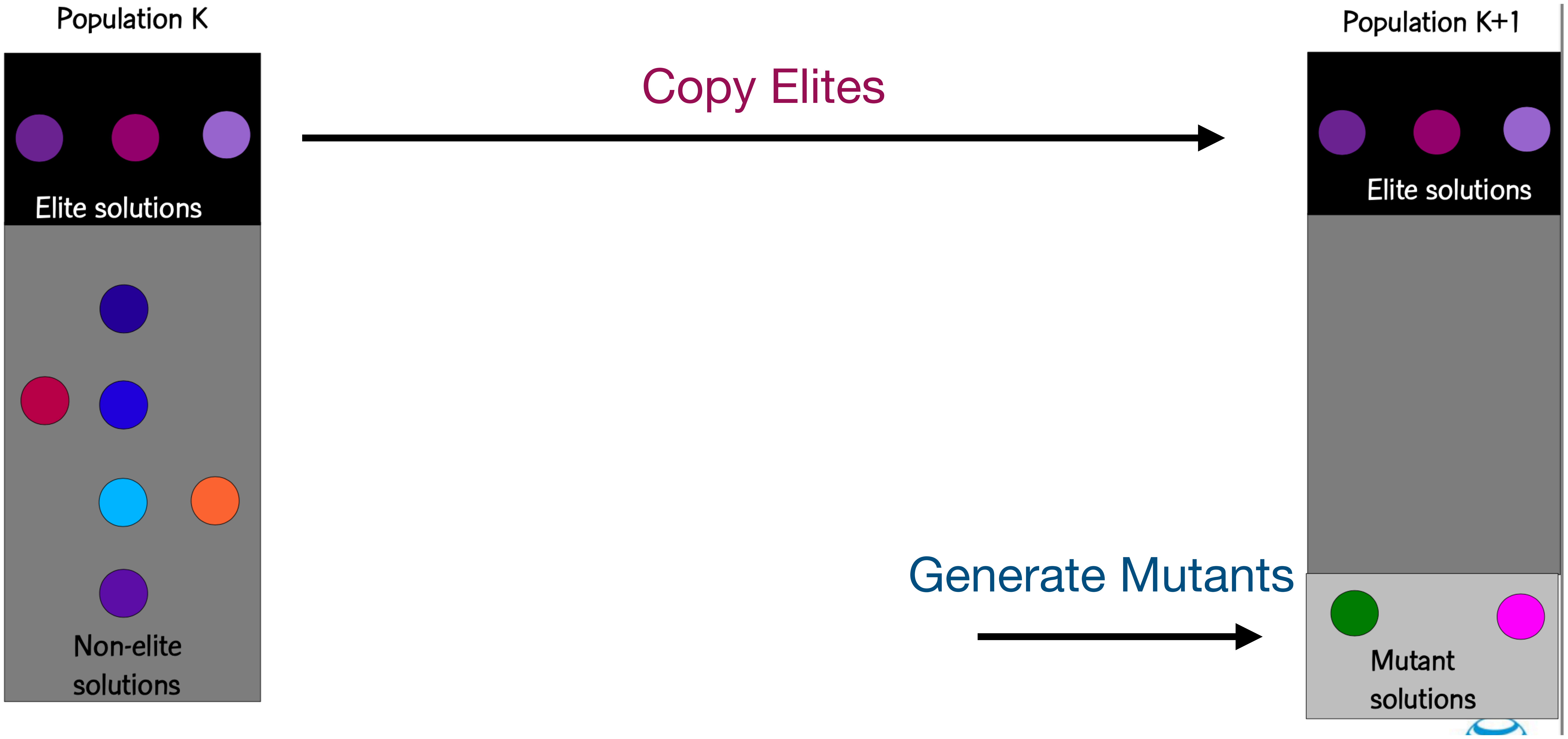
Evolution

BRKGA
Evolution

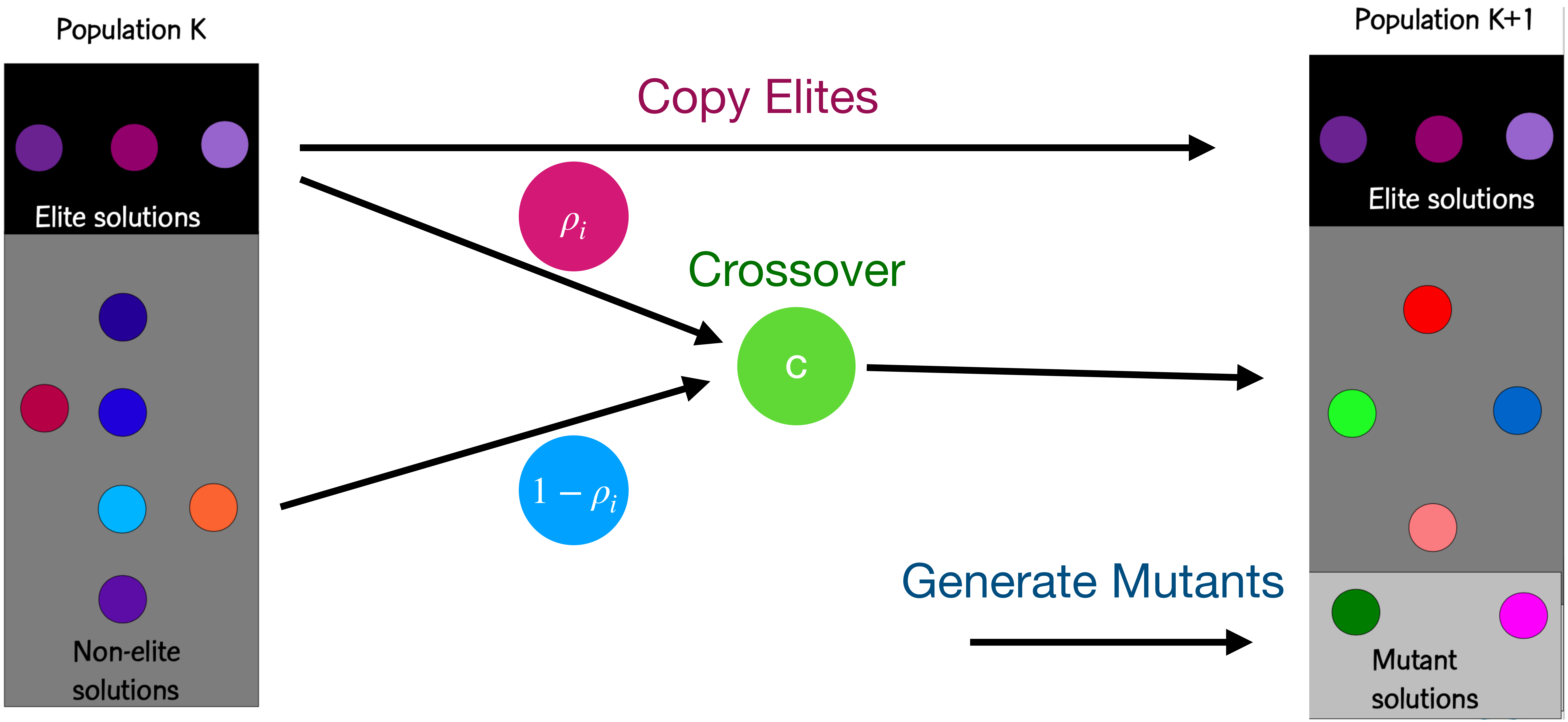


BRKGA

Evolution



BRKGA
Evolution



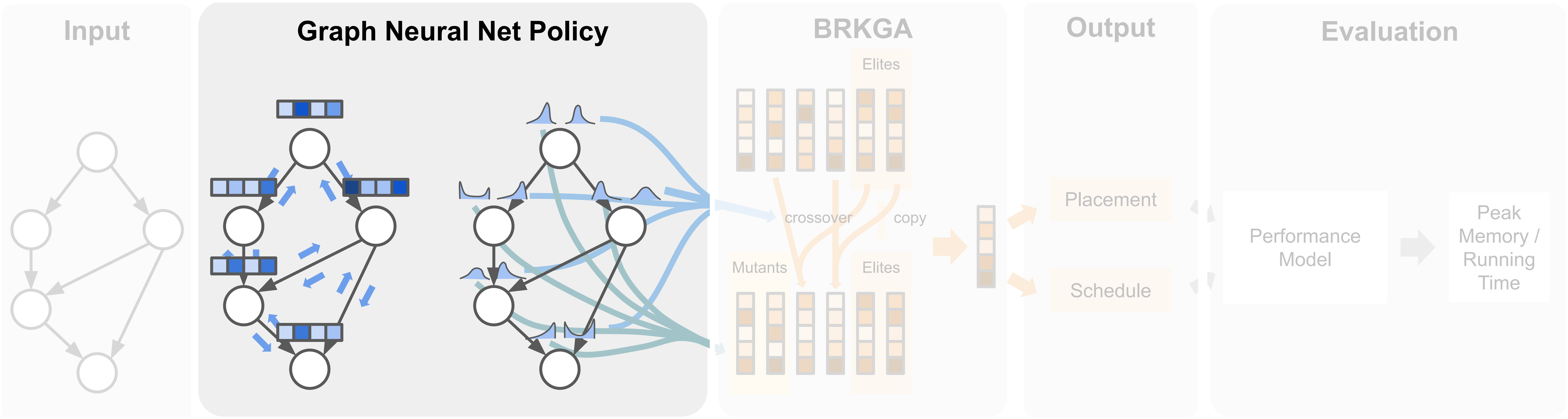
BRKGA

Encoding & Decoding

- Chromosome

  - n-d vector $[0,1]^n$
  - Ops-device affinity
  - Scheduling priorities
  - Tensor transfer priorities
- Fitness function
  - $f : [0,1]^n \rightarrow \mathbb{R}$

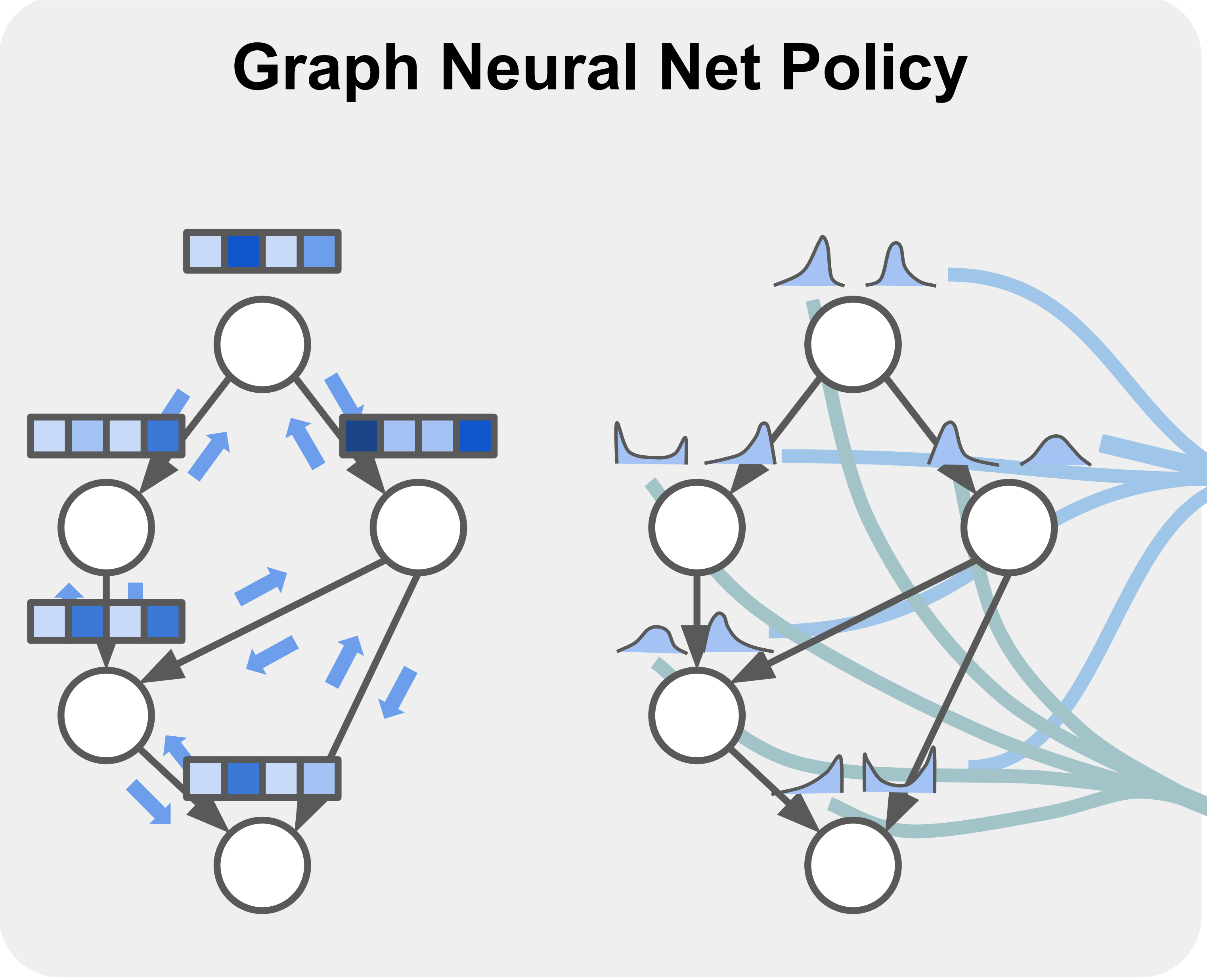
GNN policy



Input
Graph Neural Net Policy
BRKGA
Elites
crossover
copy
Mutants
Elites
Output
Placement
Schedule
Evaluation
Performance Model
Peak Memory / Running Time

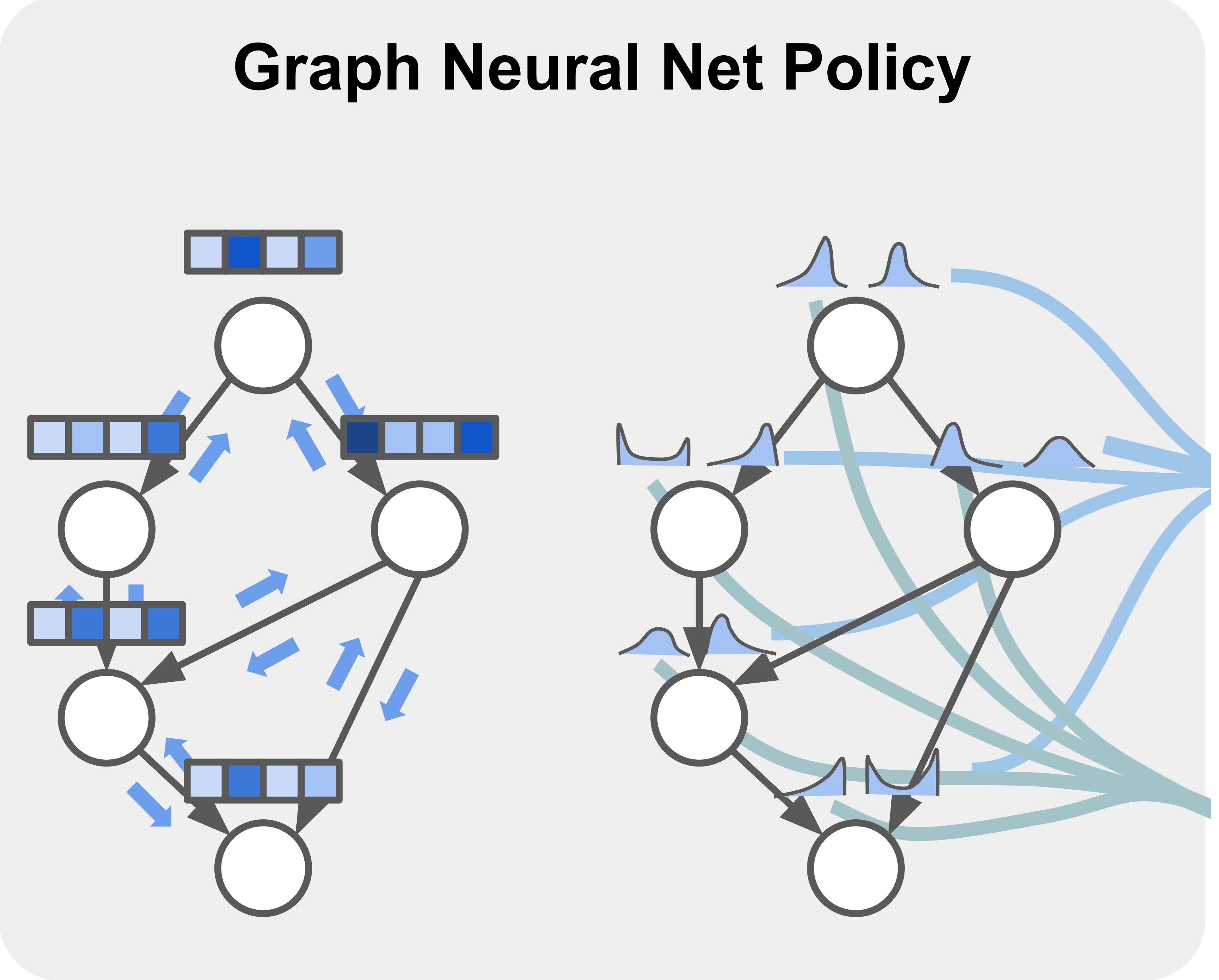
GNN policy

- Aim to generate
  - Parameters of chromosome generation distribution $\mathscr{D}$
  - Elite biases (ρ_i)
  - As a vector $\mathbf{y}_v$ for each node v



GNN policy

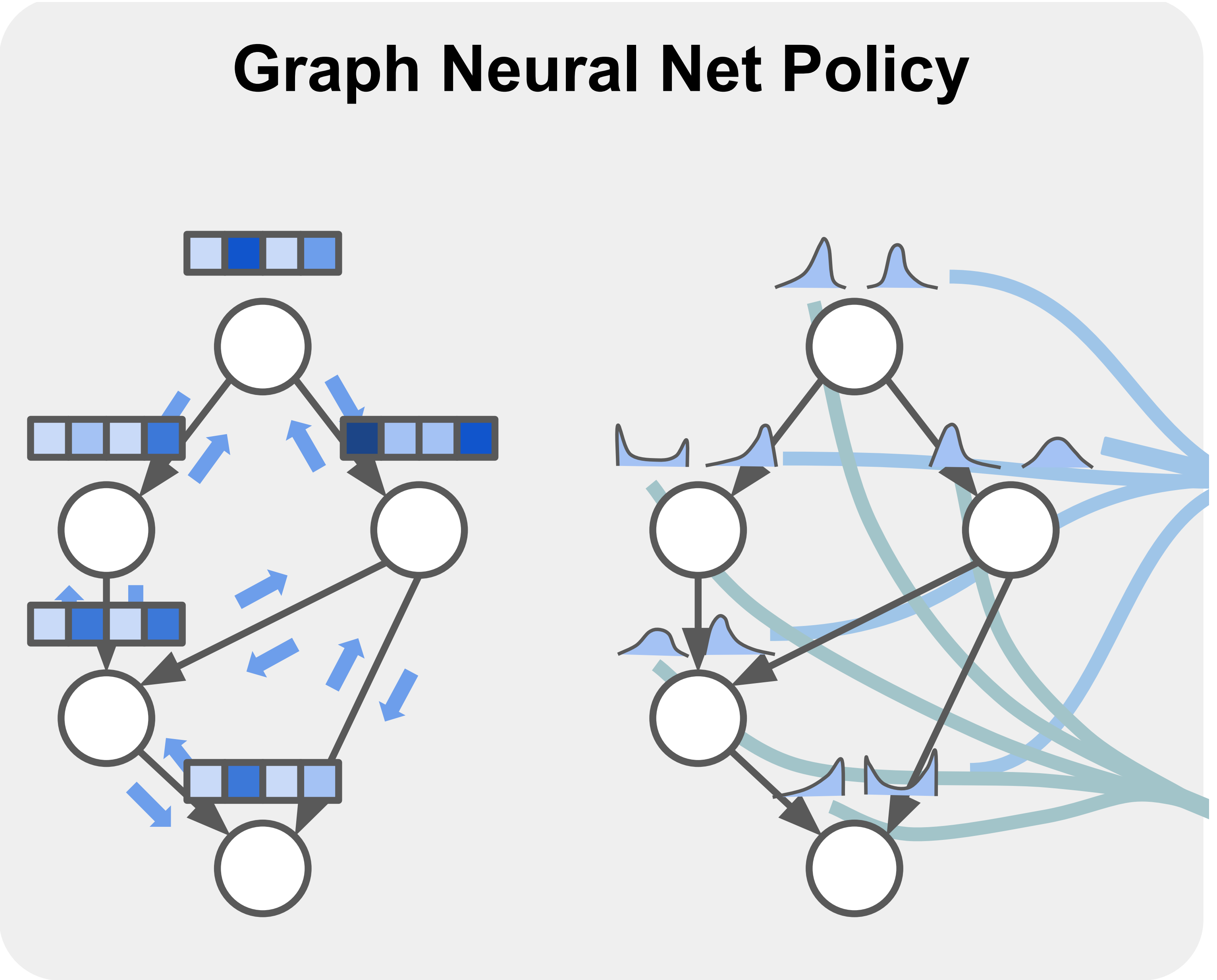
- Aim to generate
  - Parameters of chromosome generation distribution $\mathscr{D}$
  - Elite biases (ρ_i)
  - As a vector $\mathbf{y}_v$ for each node v
- GNN
  - Representation vectors $\mathbf{h}_v$ for each node v
  - Structural information of the graph



GNN policy

How do we go from $\mathbf{h}_v$ to $\mathbf{y}_v$?

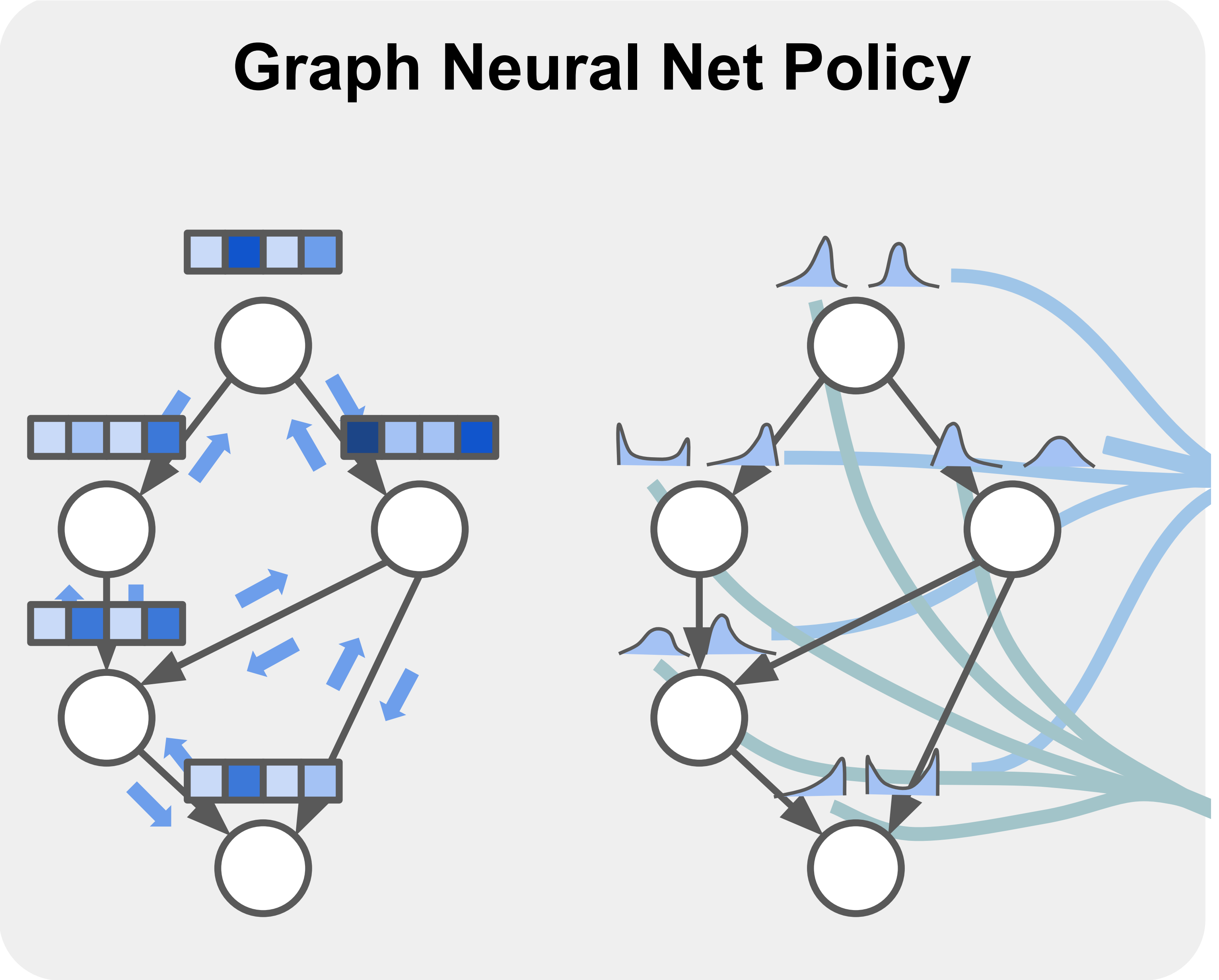
- Conditionally independent predictions
- Autoregressive predictions
- Actions & Rewards (Aka RL)



GNN policy

REINFORCE

- Sample action vector $\mathbf{y}$ from $p(\mathbf{y} \mid G)$
- Reward $r = -\dfrac{o_a(G)}{o_s(G)}$
- Maximise
- $$L = \mathbb{E}_G \left[\sum_{\mathbf{y}} p(\mathbf{y} \mid G) r(\mathbf{y}, G) \right]$$



Results

Vs Baselines

- Constraint programming
- Graph partition
- Local search (greedy)
- Graph-As-Sequence

Table 1: Performance for all methods, averaged over the graphs in the test set of the TensorFlow and XLA datasets.

Algorithm	TensorFlow dataset (test)		XLA dataset	
	% Improv. over BRKGA5K	% Gap from best	% Improv. over BRKGA5K	% Gap from best
CP SAT	-1.77%	13.89%	-47.14%	71.35%
GP + DFS	-6.51%	16.63%	-21.43%	39.86%
Local Search	0.63%	8.65%	-6.69%	21.98%
BRKGA 5K	0%	9.65%	0%	14.04%
Tuned BRKGA	0.8%	8.54%	0.452%	13.52%
GAS	0.16%	9.33%	-1.1%	15.36%
REGAL	**3.56%**	**4.44%**	**3.74%**	**9.40%**

Discussion

Ablation analysis

Table 3: Performance of REGAL with various subsets of actions.

Placement	Scheduling	Elite Bias	Valid	Test	XLA
Yes	No	No	-0.4%	-0.2%	-0.4%
No	Yes	No	4.4%	**3.65%**	1%
Yes	Yes	No	**4.67%**	3.56%	**3.74%**
Yes	No	Yes	-1.53%	-1.1%	-2.2%
No	Yes	Yes	2.47%	1.4%	-0.4%
Yes	Yes	Yes	2.58%	1.88%	-0.7%

Comments

- Extensive evaluation and impressive results
- Transfer learning through policy network
- Objectives other than peak memory minimisation
- Too many optimisation layers, very complex system
- Justification of BRKGA

Conclusions

- Optimisation *all the way down*
- Input -> GNN -> REINFORCE -> BRKGA -> Decision
- Transfers well

References

- A. Paliwal, F. Gimeno, V. Nair, et al., REGAL: Reinforced genetic algorithm learning for optimizing computation graphs, 2020. arXiv: 1905.02494 [cs.LG].
- Mauricio G. C. Resende: Biased random-key genetic algorithms: A tutorial, 2012
- Zak Singh, R244 Paper Presentation on REGAL, 2021