Learn to Optimize – An Overview

Ke TANG

Department of Computer Science and Engineering

Southern University of Science and Technology (SUSTech)

tangk3@sustech.edu.cn

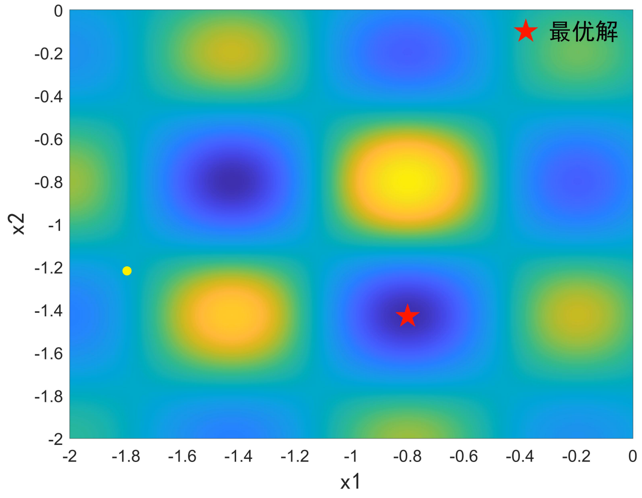
https://faculty.sustech.edu.cn/tangk3/

Outline

- Introduction
- L2O On-The-Fly
- L2O for Algorithm Design
- Summary

Introduction

- **Optimization problems are ubiquitous.**
- **Analytical solutions usually do not exist for important optimization problems.**
- **Such problems could only be solved in an iterative trial-and-error way.**
  - Simplex method
  - Gradient descent method
  - Trust-Region method
  - Branch-and-Bound method
  - Simulated Annealing
  - Evolutionary Algorithms
  - Bayesian Optimization
  - ...



$$\begin{aligned} &\text{maximize} \quad f(x) \\ &\text{subject to: } g_i(x) \leq 0, \quad i = 1 \ldots m \\ &\qquad\qquad\quad h_j(x) = 0, \quad j = 1 \ldots p \end{aligned}$$

Introduction

In a nutshell, a trial-and-error procedure maps one candidate solution to another.

$$\phi:\ \mathrm{x} \rightarrow \mathrm{x}', \forall \mathrm{x}, \mathrm{x}' \in \mathcal{X}$$

The mapping function appears under different names in the literature

- Heuristic function
- Search bias
- Gradient
- Acquisition function
- ...

Introduction

It is non-trivial to design a good search bias for many optimization problems.

- The problem is complex (e.g., non-convex, not differentiable, no explicit formulation...)

Numerical Optimization

Convex Optimization

Linear Programming

Integer Linear Programming

Mixed Integer Programming

X

Simplex

Interior Point

Gradient Descent

Trust-Region

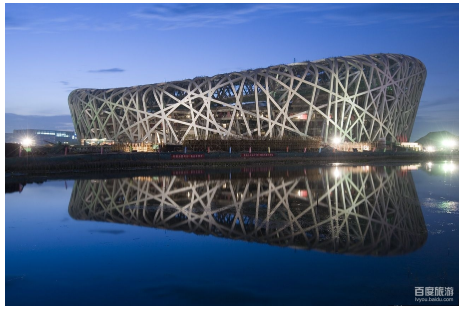
Cutting Plane

Branch-and-Bound

......

? ? ?

$$\begin{aligned} \text{maximize} \quad & f(x) \\ \text{subject to:} \quad & g_i(x) \le 0, \quad i = 1 \ldots m \\ & h_j(x) = 0, \quad j = 1 \ldots p \end{aligned}$$



Truss Design Problem

Toolbox accumulated over 50 years

What is the mathematical formulation of these functions?

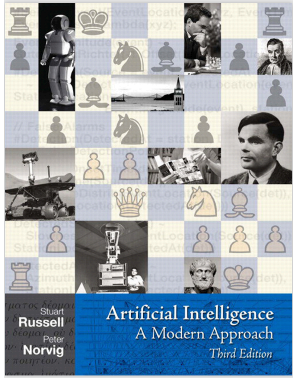
Introduction

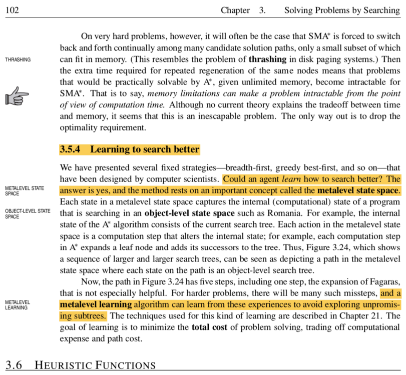
It is non-trivial to design a good search bias for many optimization problems.

- The problem is complex (e.g., non-convex, not differentiable, no explicit formulation...)
- Requires heavy domain expertise and prior knowledge to design
  - A *surrogate* objective function with good properties
  - Good *heuristic function* for heuristic search (such as A*)
  - Good *sub-problem* for branch-and-bound
  - ...

Can we acquire prior knowledge from data? – Learn to Optimize (L2O)

- A general methodology
- Not brand new





102 Chapter 3. Solving Problems by Searching

On very hard problems, however, it will often be the case that SMA* is forced to switch back and forth continually among many candidate solution paths, only a small subset of which can fit in memory. (This resembles the problem of **thrashing** in disk paging systems.) Then the extra time required for repeated regeneration of the same nodes means that problems that would be practically solvable by A*, given unlimited memory, become intractable for SMA*. That is to say, *memory limitations can make a problem intractable from the point of view of computation time.* Although no current theory explains the tradeoff between time and memory, it seems that this is an inescapable problem. The only way out is to drop the optimality requirement.

THRASHING

3.5.4 Learning to search better

We have presented several fixed strategies—breadth-first, greedy best-first, and so on—that have been designed by computer scientists. Could an agent *learn* how to search better? The answer is yes, and the method rests on an important concept called the **metalevel state space**. Each state in a metalevel state space captures the internal (computational) state of a program that is searching in an **object-level state space** such as Romania. For example, the internal state of the A* algorithm consists of the current search tree. Each action in the metalevel state space is a computation step that alters the internal state; for example, each computation step in A* expands a leaf node and adds its successors to the tree. Thus, Figure 3.24, which shows a sequence of larger and larger search trees, can be seen as depicting a path in the metalevel state space where each state on the path is an object-level search tree.

METALEVEL STATE SPACE

OBJECT-LEVEL STATE SPACE

Now, the path in Figure 3.24 has five steps, including one step, the expansion of Fagaras, that is not especially helpful. For harder problems, there will be many such missteps, and a **metalevel learning** algorithm can learn from these experiences to avoid exploring unpromising subtrees. The techniques used for this kind of learning are described in Chapter 21. The goal of learning is to minimize the **total cost** of problem solving, trading off computational expense and path cost.

METALEVEL LEARNING

3.6 HEURISTIC FUNCTIONS

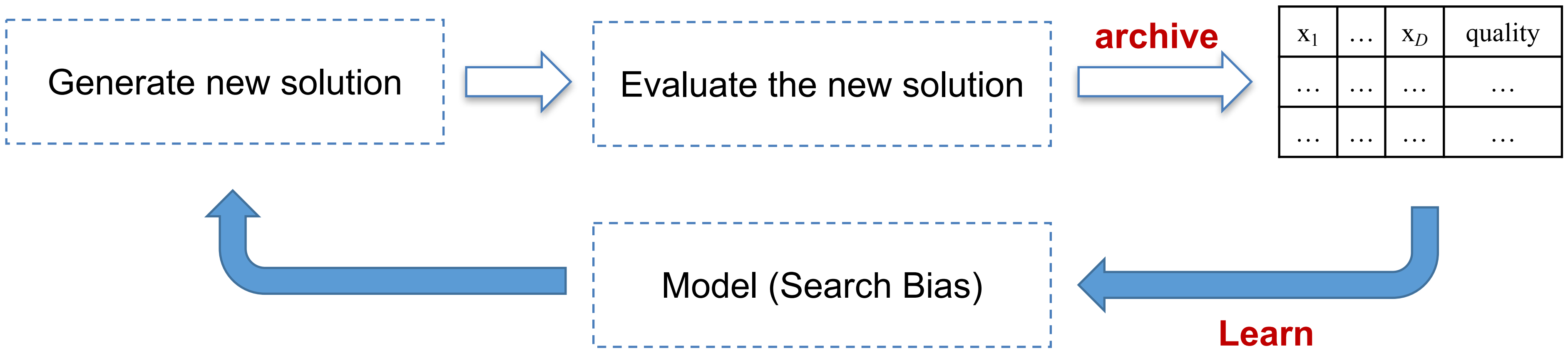
Artificial Intelligence – A Modern Approach
Third Edition, pp. 102, 2010

Outline

- Introduction
- L2O On-The-Fly
- L2O for Algorithm Design
- Summary

A Unified Framework

Many instantiations of the L2O idea share a unified framework.

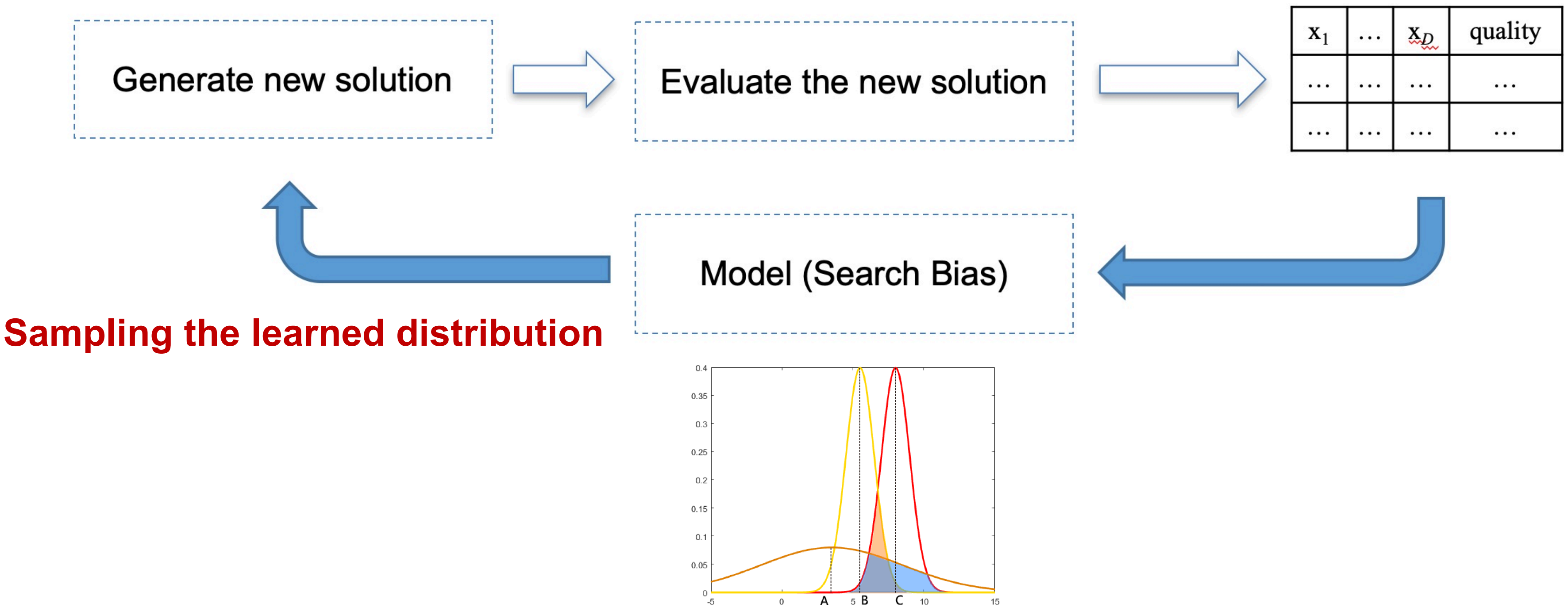


In machine learning language: What is the model learned for?

Learn for What?

To generate **(sampling from a probability distribution) new candidate solutions.**

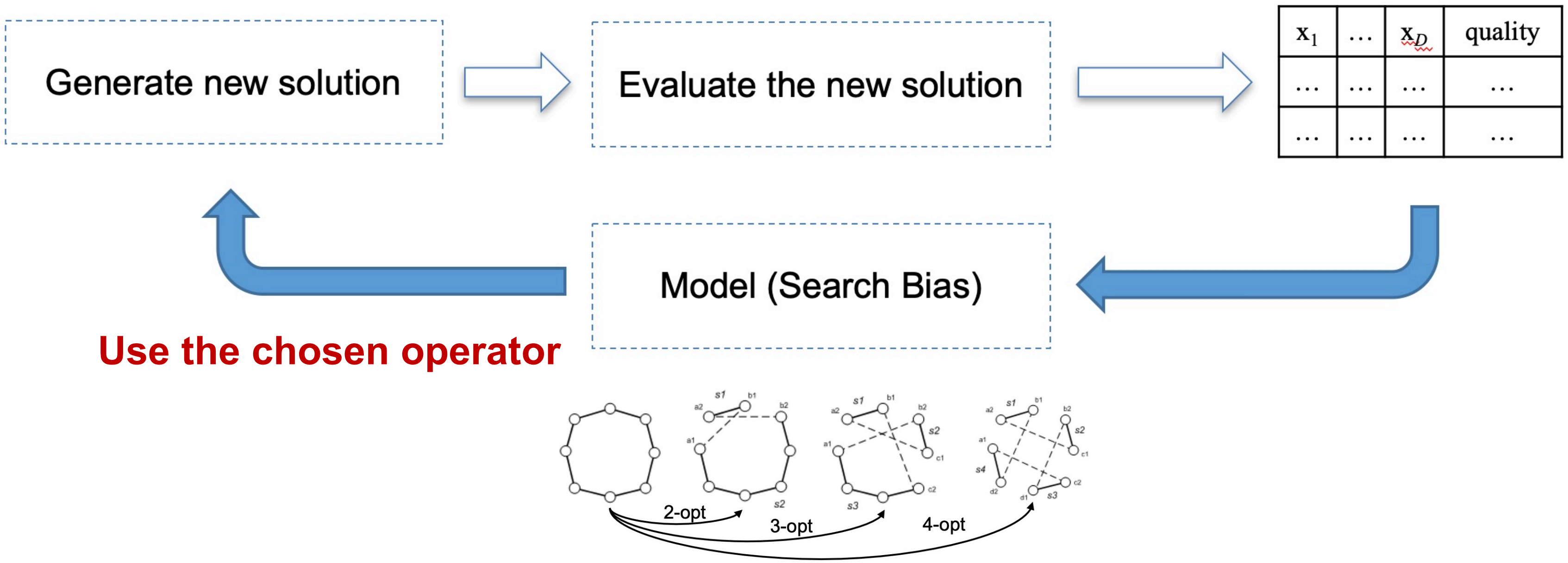
- Bayesian Optimization [1][48]
- Evolution Strategies [2]
- Estimation of Distribution Algorithms [49]



Learn for What?

To choose the operator (e.g., a probability distribution) for generating new solutions.

- Self-adaptation [3]
- Hyper-heuristic [4]
- ...



Learn for What?

To accelerate the evaluation of new solutions.

- Surrogate-assisted Evolutionary Optimization [5][47]
- Response Surface Model [6]
- ...

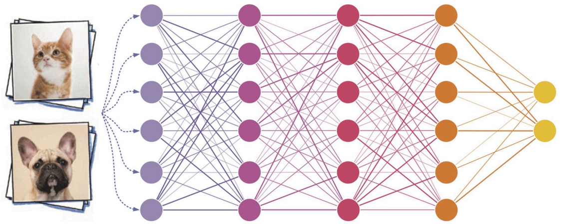
Generate new solution

Evaluate the new solution

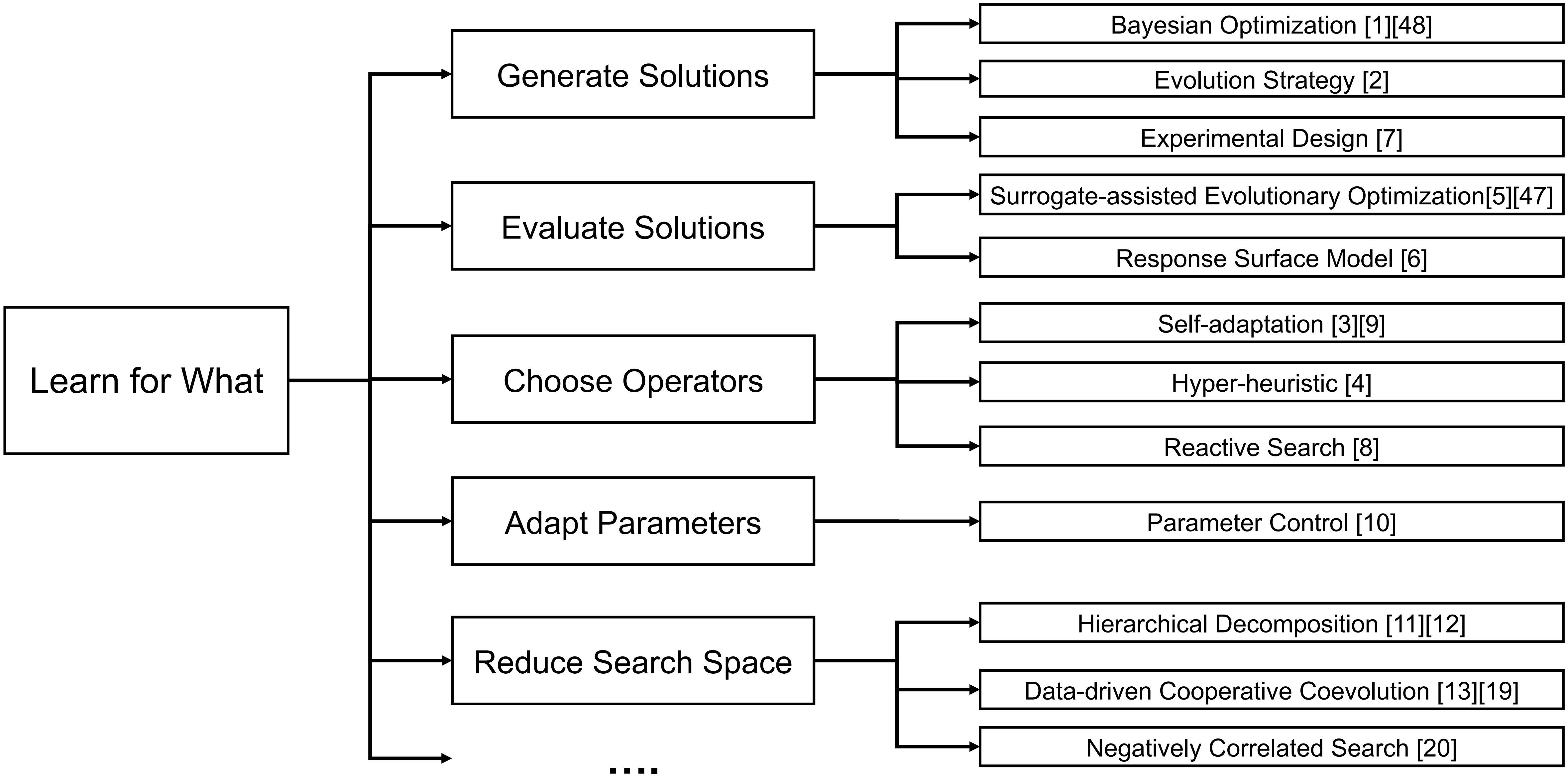
x_1	…	x_D	quality
…	…	…	…
…	…	…	…

Use the surrogate model for evaluation

Model (Search Bias)



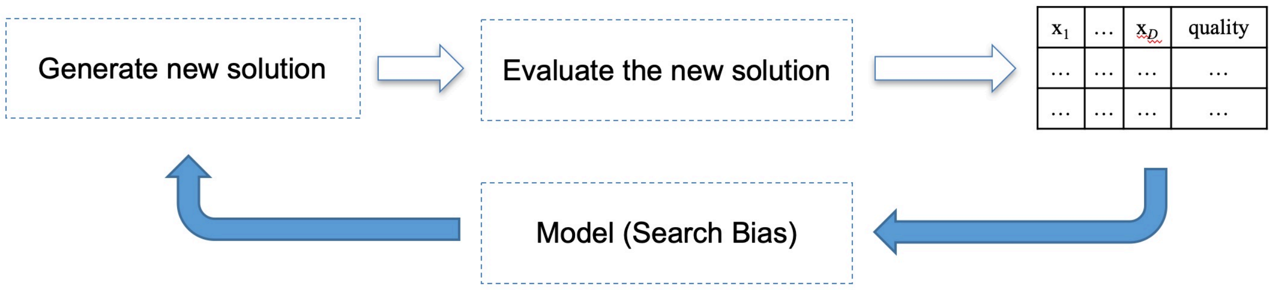
Learn for What?

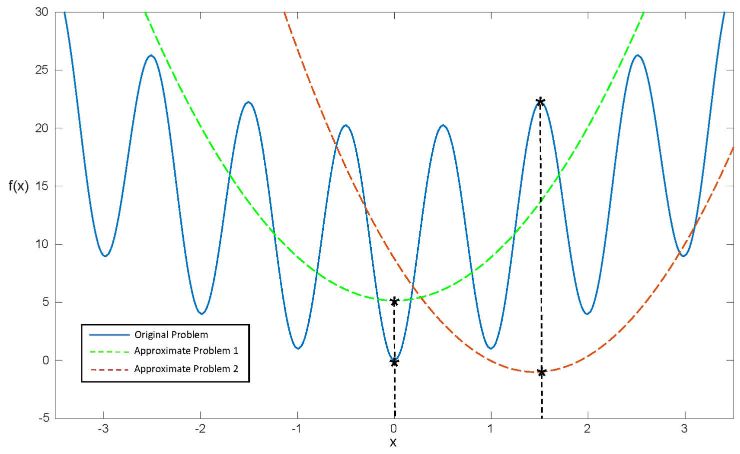


Dilemma of L2O

Shared framework induces Shared Challenge

- Learning is done on-the-fly (i.e., in the loop of an optimization algorithm)
- **Can one hard optimization problem be better solved by introducing another hard optimization problem (learning) into an optimization algorithm?**
  - No conclusive answer, although indirect evidence exists for some case studies [17] [18].





How can we obtain the green curve?

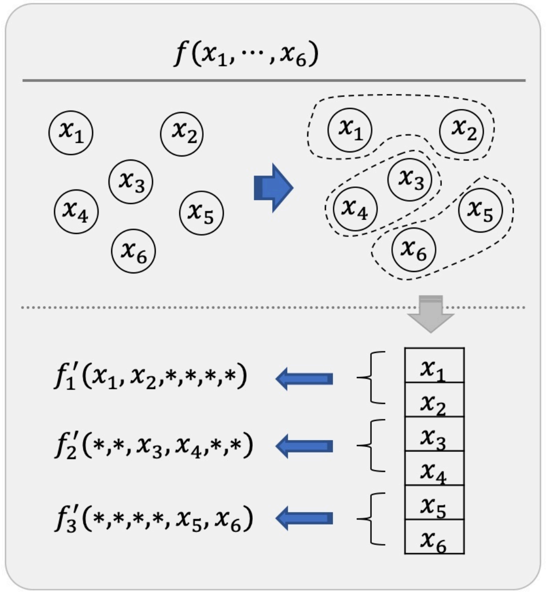
Dilemma of L2O

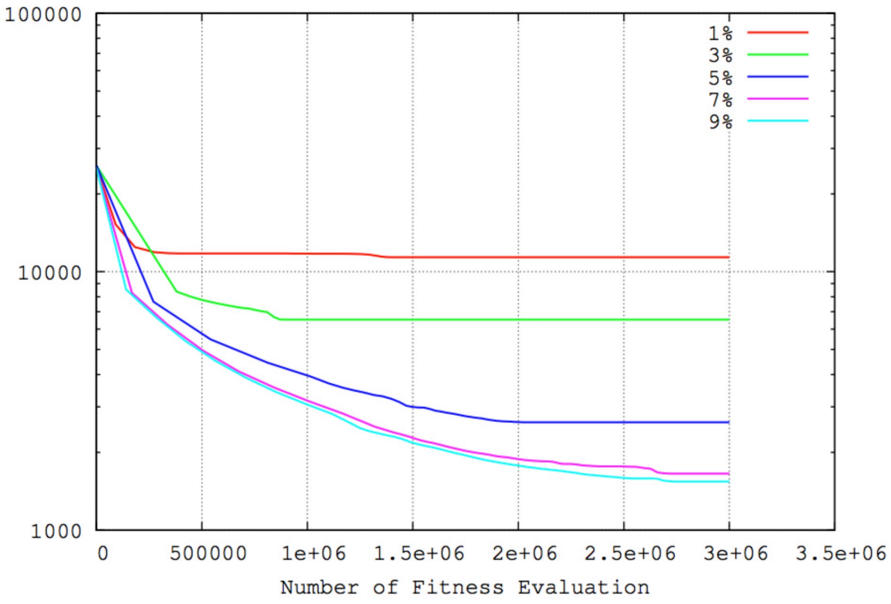
SUSTech Southern University of Science and Technology NICAL

An illustrative example: Learning interactions between decision variables

As the learning “accuracy” increases

- Marginal benefit of learning decreases, while learning cost increases
- What is the best trade-off?





Outline

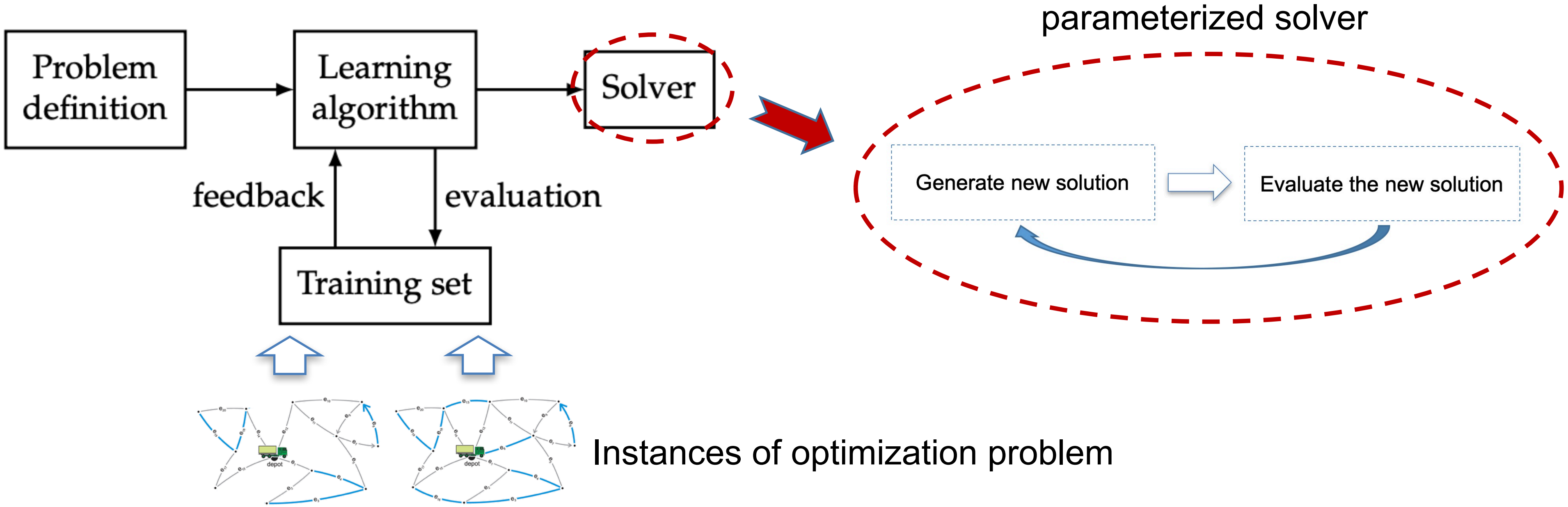
- Introduction
- L2O On-The-Fly
- L2O for Algorithm Design
- Summary

L2O for Algorithm Design – General Ideas

Learning-in-the-loop could be viewed as the counterpart of online learning in L2O context.

The rise of offline L2O : L2O for algorithm design

- Train an optimization algorithm (solver) with data.
- **A data-driven paradigm for algorithm/solver design.**

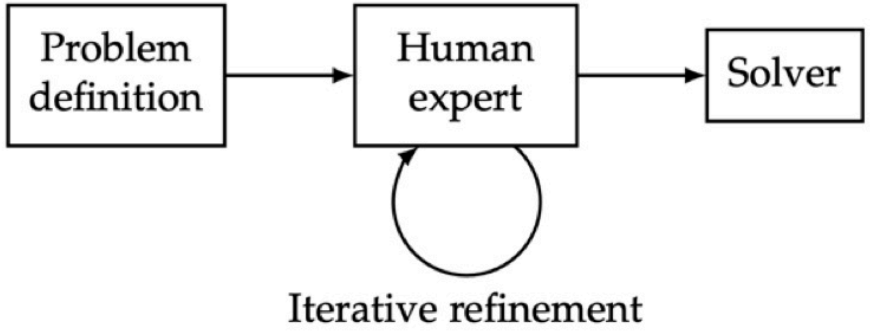


Instances of optimization problem

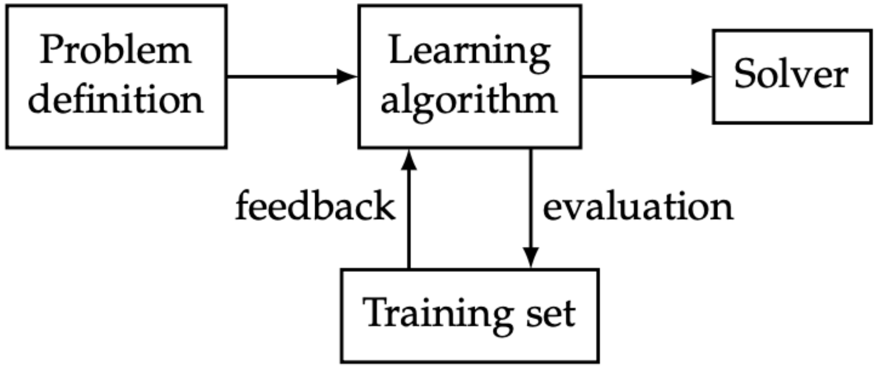
L2O for Algorithm Design – General Ideas

Why offline?

- Offline learning is, in general, more tractable and reliable.
- Less restricted by the time budget allowed for solving an optimization problem.
- Relief human labors + possibly better algorithm



Traditional Human-centric design process



Automated algorithm design process

L2O for Algorithm Design – General Ideas

Like all machine learning tasks, the high-level learning problem is defined by:

- Solver representation: *A* (a parameterized optimization algorithm, e.g., a heuristic algorithm)
- Training set : $I = \{s_i\}$,
- Some performance indicator: m (determined by the application/user, e.g., solution quality, runtime)
- **Learning Objective**: $\underset{A}{argmax}\, m(A, I)$

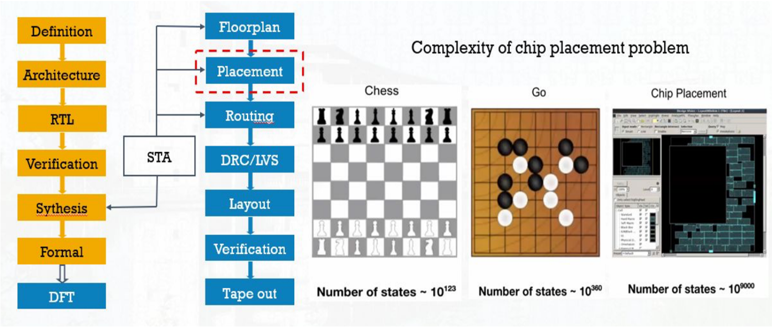
The high-level optimization problems are still tough since the objective function could be

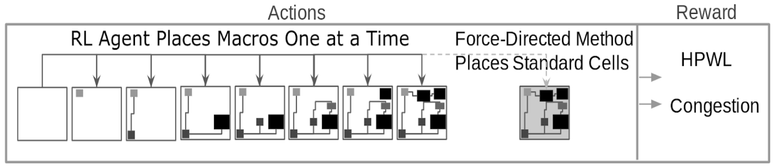
- Not differentiable
- No clear mathematical formulations
- Noisy (particularly for NP-Hard problems)

L2O for Algorithm Design – Recent Advances

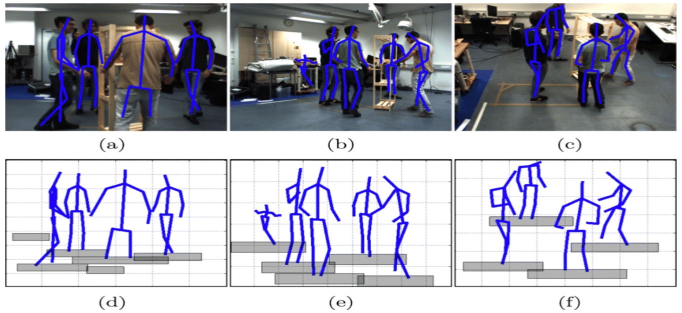
SUSTech Southern University of Science and Technology NICAL

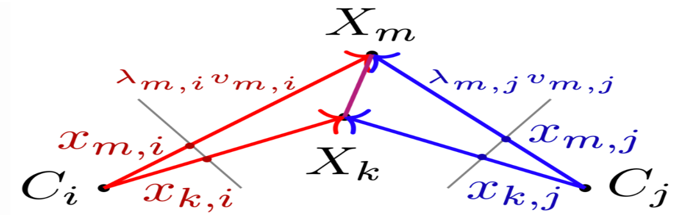
Chip Placement [14] (Nature'2021)



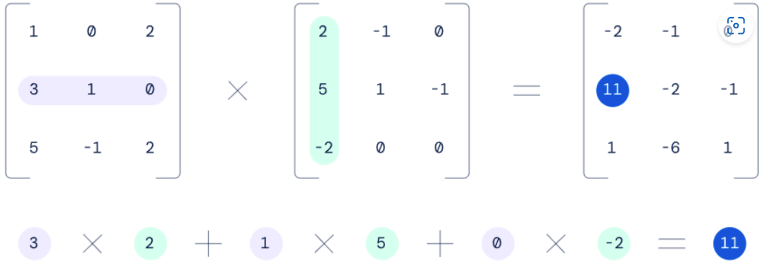


Multi-view Pose Estimation [16] (CVPR'2022 best paper)

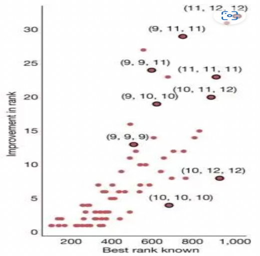




Matrix Multiplication [15] (Nature'2022)



Size (n, m, p)	Best method known	Best rank known	AlphaTensor rank Modular	AlphaTensor rank Standard
(2, 2, 2)	(Strassen, 1969)[2]	7	7	7
(3, 3, 3)	(Laderman, 1976)[15]	23	23	23
(4, 4, 4)	(Strassen, 1969)[2] (2, 2, 2) ⊗ (2, 2, 2)	49	47	49
(5, 5, 5)	(3, 5, 5) + (2, 5, 5)	98	96	98
(2, 2, 3)	(2, 2, 2) + (2, 2, 1)	11	11	11
(2, 2, 4)	(2, 2, 2) + (2, 2, 2)	14	14	14
(2, 2, 5)	(2, 2, 2) + (2, 2, 3)	18	18	18
(2, 3, 3)	(Hopcroft and Kerr, 1971)[16]	15	15	15
(2, 3, 4)	(Hopcroft and Kerr, 1971)[16]	20	20	20
(2, 3, 5)	(Hopcroft and Kerr, 1971)[16]	25	25	25
(2, 4, 4)	(Hopcroft and Kerr, 1971)[16]	26	26	26
(2, 4, 5)	(Hopcroft and Kerr, 1971)[16]	33	33	33
(2, 5, 5)	(Hopcroft and Kerr, 1971)[16]	40	40	40
(3, 3, 4)	(Smirnov, 2013)[18]	29	29	29
(3, 3, 5)	(Smirnov, 2013)[18]	36	36	36
(3, 4, 4)	(Smirnov, 2013)[18]	38	38	38
(3, 4, 5)	(Smirnov, 2013)[18]	48	47	47
(3, 5, 5)	(Sedoglavic and Smirnov, 2021)[19]	58	58	58
(4, 4, 5)	(4, 4, 2) + (4, 4, 3)	64	63	63
(4, 5, 5)	(2, 5, 5) ⊗ (2, 1, 1)	80	76	76

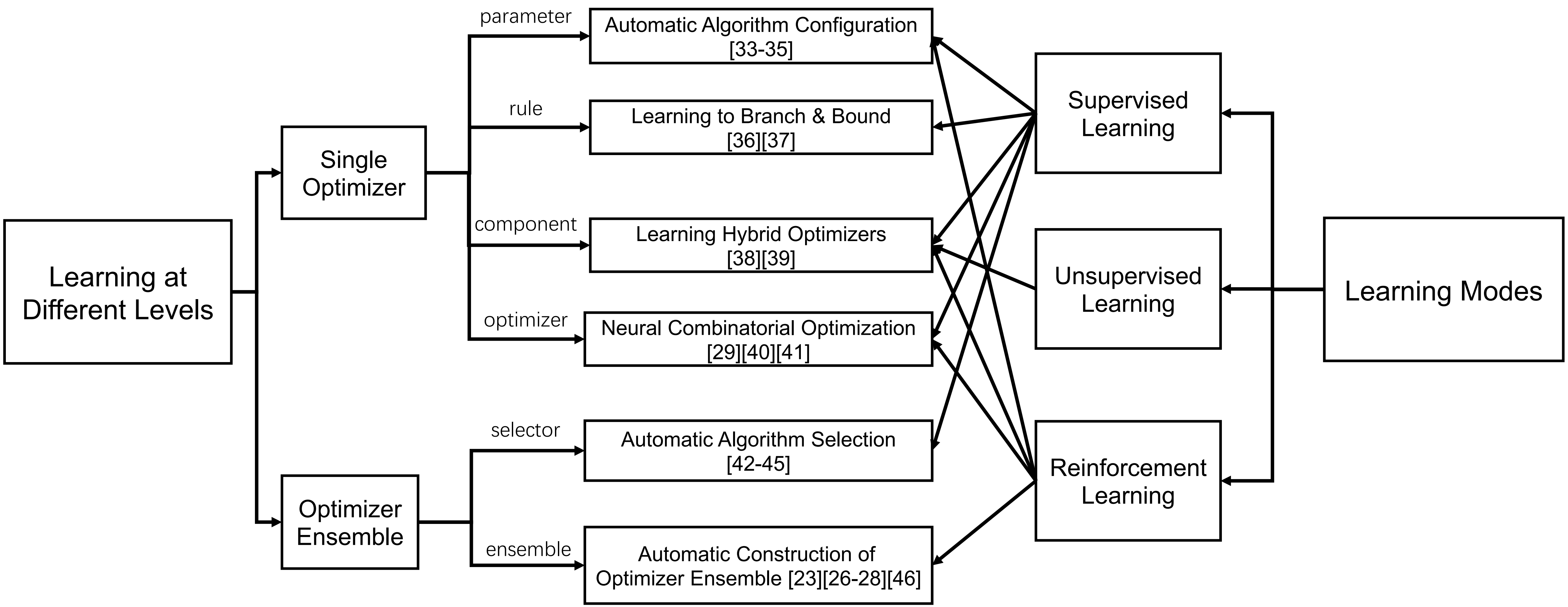


Outputs layouts in 6 hours, comparable to those designed by human experts taking several weeks

Achieves at least 10+ times speedup compared to the best known results

Multiplies two 4×4 matrices in only 47 multiplications, less than 49 multiplications required by Strawson's algorithm

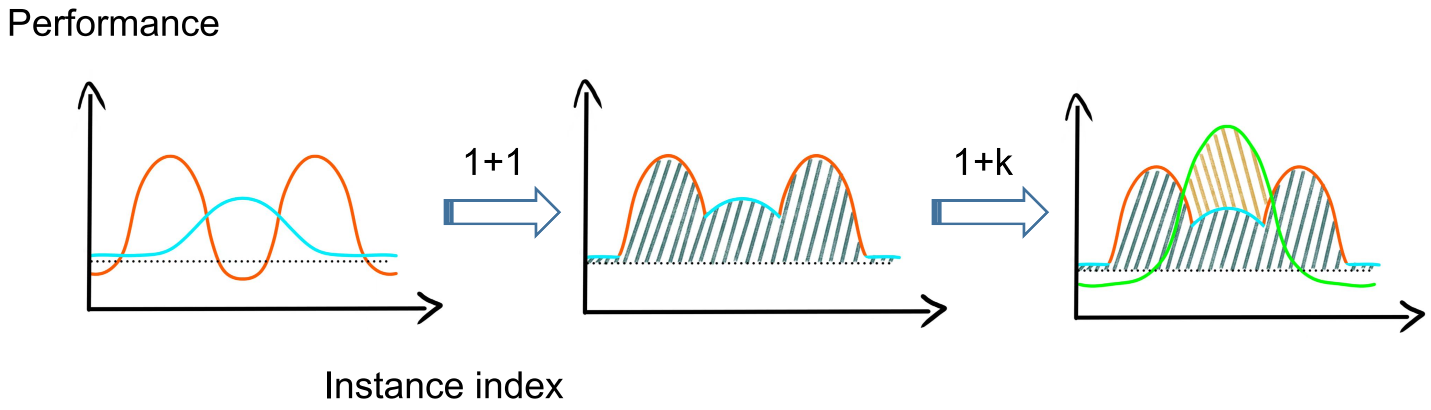
L2O for Algorithm Design – Recent Advances



Learning Optimizer Ensembles

Parallel Algorithm Portfolios (PAP) [21][22]

- Run multiple algorithms in parallel, introduce minimum implementation overhead.
- Inherently generalize better than a single algorithm (No-Free-Lunch).
- A framework adopted by many industrial software system.



Learning Optimizer Ensembles

PAP : $P = \{A_1, A_2 \dots A_k\}$, **Training set :** $I = \{s_i\}$

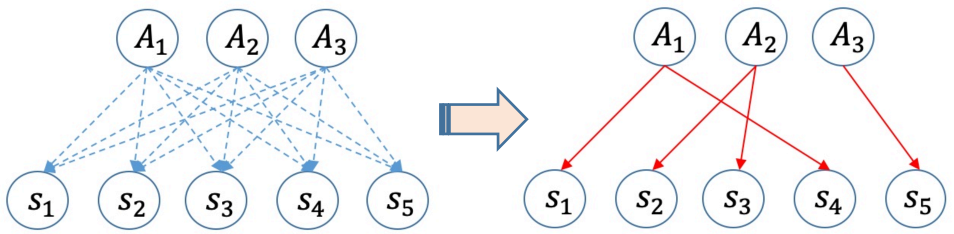
Performance indicator of a PAP: $m(P, s_i) = \min\{m(A_1, s_i), m(A_2, s_i), \dots, m(A_k, s_i)\}$

Performance indicator of an optimization algorithm

Objective Function : $\underset{P}{\operatorname{argmin}}\, m(P, I)$ not separable with respect to algorithms

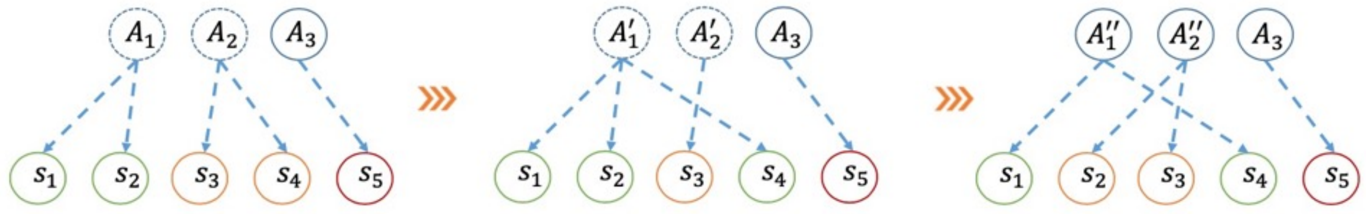
Learning Optimizer Ensembles

If the training set could be "clustered" in advance, *k* algorithms could be learned separately.



But how to cluster training instances without problem-specific feature engineering?

- **A problem-independent approach [23]:** using algorithm behavior data as instance feature



S. Liu, K. Tang and X. Yao, "Automatic Construction of Parallel Portfolios via Explicit Instance Grouping," *AAAI* 2019

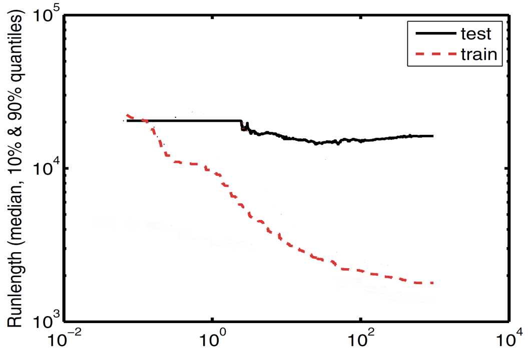
Learning Optimizer Ensembles

What if we don't have enough training instances?

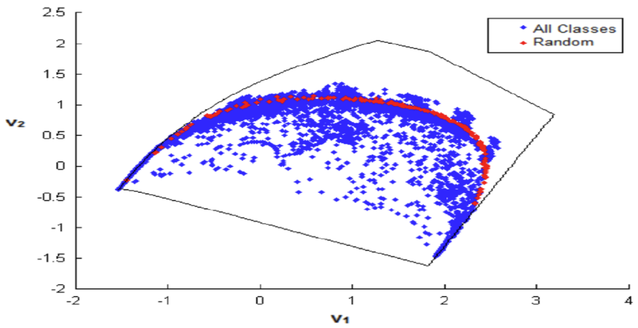
- Suppose a solver for combinatorial optimization problem is to be built.

Problem	No. of Instance	
TSP	143	http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html
CVRP	319	http://vrp.atd-lab.inf.puc-rio.br/index.php/en/
VRPSPDTW	85	https://github.com/senshineL/VRPenstein

Indeed, benchmark set is small [27]



Overfitting may also occurs [24]

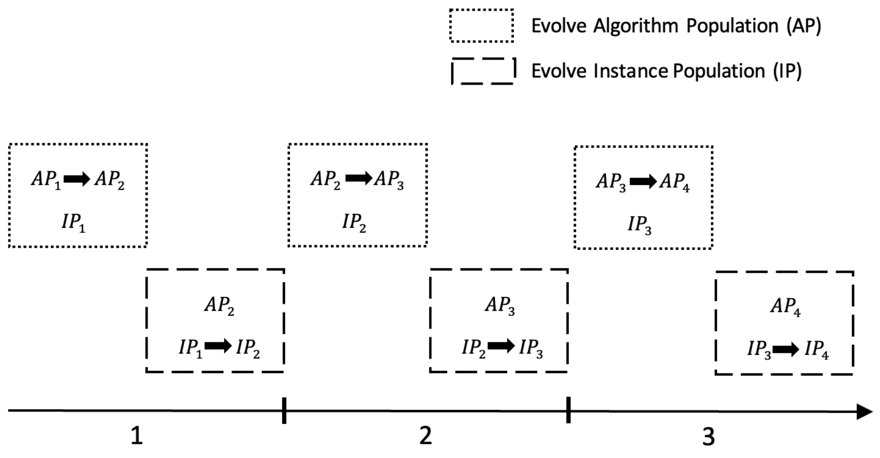


Randomly generated instances are biased [25]

Learning Optimizer Ensembles

A Co-Evolutionary Approach [26]: Alternatively improve the solver and generate pseudo instances to solve a *minimax* problem.

$$\min_{P} \max_{s \in I^*} m(A, s) \ :$$



In essence, iteratively maximize a generalization bound of the PAP.

$$J(A') - J(A) \leq \sum_{s \in I \cup I'} [m(A', s) - m(A, s)]$$

K. Tang, S. Liu, P. Yang and X. Yao, "Few-shots Parallel Algorithm Portfolio Construction via Co-evolution," *IEEE Transactions on Evolutionary Computation*, 2021

Performance of The Learned PAP

No more than 7 days * 40core Intel Xeon machines with 128 GB RAM (2.20 GHz, 30 MB Cache)

SAT_PAP[23]	TSP_PAP[28]	VRPSPDTW_PAP[26]
Competitive to SOTA	Significantly outperform SOTA	New best solution

	Agile Track		Parallel Track	
	#TOs	PAR-10(s)	#TOs	PAR-10(s)
SAT_PAP	**181**	**119**	**35**	**1164**
Priss6	225	146	-	-
PfolioUZK	-	-	**36**	**1185**
Plingeling-bbc	452	276	**33**	**1090**

Priss6：SAT'16 Competition (Agile Track) Winner

PfolioUZK：SAT'12 Competition (Parallel Track) Winner

Plingeling-bbc：SAT'16 Competition (Parallel Track) Winner

	#TOs	PAR-10(s)	ADR(‱)
TSP_PAP	**7**	**2369.38**	**0.02**
LKH	50	14242.85	0.56
EAX	16	4928.88	0.24
LKH-TUNED	43	12319.73	0.51
EAX-TUNED	14	4364.25	0.16
EAX_LKH	9	2921.51	0.12

LKH, EAX：**Best** TSP solvers so far

LKH/EAX-TUNED：LKH & EAX with **further finetuning**

EAX_LKH：**PAP** consisted with LKH & EAX

问题类型	样例数量	好于BKS的数量
Rdp	23	19
Cdp	17	9
RCdp	16	16
总计	56	44

BKS：Best Known Solutions by September 2020

Solver \ Cost	Rdp103	Rdp206	Cdp108	RCdp102	RCdp105
VRPSPDTW_PAP	**2594.64**	**1206.14**	**1932.49**	**2770.28**	**2946.36**
Co-GA	2616.16	1261.32	1951.24	2897.05	2981.26
p-SA	2626.77	1259.94	2063.73	2822.76	2981.54
ALNS-PR	2597.01	1213.68	1932.88	2783.62	2948.96

Outline

- Introduction
- L2O On-The-Fly
- L2O for Algorithm Design
- Summary

Summary

- **L2O is a general idea/methodology with long history and fruitful results.**

- **L2O could be viewed from many other facets, each leading to an instantiation of the idea.**
  - Learning mode viewpoint: Online/Offline mode
  - Model representation viewpoint: Heuristic rules, parameters of algorithms, neural networks...
  - Learning technique viewpoint: Supervised/Unsupervised/Reinforcement/Evolutionary Learning...

- **Recently, the “offline” mode of L2O has made quite a few impressive progresses in application domains, and may even lead to a paradigm shift for algorithm design.**

Road Ahead

- **What is the key challenge for L2O, in comparison to Learn to Classify/Predict/Rank?**
  - SOTA NCO solver is not as competitive as traditional solvers [31]

	TSP-50	TSP-100	TSP-500	TSP-1000
Best NCO Solver (POMO [29])	0.0006%	0.1278%	1.7621%	out of memory
Heuristic Solver (EAX [30])	0%	0%	0.0140%	0.0182%

- **Where to get the data?**
  - Deeper integration between optimization and **simulation**.
- **More rigorous theoretical foundation?**
  - Bridging theory of learning and optimization (e. g. PAC + Robust Optimization)
  - How much data do we need to learn an algorithm?
    - Some efforts are emerging [32]

References

1. D. R. Jones, M. Schonlau and W. J. Welch, “Efficient Global Optimization of Expensive Blackbox Functions.” Journal of Global Optimization, 1998.
2. N. Hansen and A. Ostermeier, “Completely Derandomized Self-adaptation in Evolution Strategies.” Evolutionary Computation, 2001.
3. A. C. Wilson et al., “The Marginal Value of Adaptive Gradient Methods in Machine Learning.” NeurIPS-2017.
4. E. K. Burke et al., “Hyper-Heuristics: A Survey of the State of the Art.” Journal of the Operational Research Society, 2013.
5. Y. Jin, M. Olhofer and B. Sendhoff, “A Framework for Evolutionary Optimization with Approximate Fitness Functions.” IEEE Transactions on Evolutionary Computation, 2002
6. R. H. Myers et al., “Response Surface Methodology: A Retrospective and Literature Survey.” Journal of Quality Technology, 2004.
7. M. I. Rodrigues and A. F. Iemma. Experimental design and process optimization. CRC Press, 2014.
8. R. Battiti, M. Brunato and A. Mariello, “Reactive Search Optimization: Learning while Optimizing." Handbook of Metaheuristics. Springer, 2019.
9. X. Lu, K. Tang, S. Menzel and X. Yao, “Dynamic Optimization in Fast-Changing Environments via Offline Evolutionary Search,” IEEE Transactions on Evolutionary, 2022.
10. G. Karafotias, M. Hoogendoorn and A. E. Eiben, “Parameter Control in Evolutionary Algorithms: Trends and challenges." IEEE Transactions on Evolutionary Computation, 2014.
11. K. Tang, J. Wang X. Li and X. Yao, “A Scalable Approach to Capacitated Arc Routing Problems Based on Hierarchical Decomposition.” IEEE Transactions on Cybernetics, 2017.
12. P. Yang, K. Tang and X. Yao, “Turning High-dimensional Optimization into Computationally Expensive Optimization,” IEEE Transactions on Evolutionary Computation, 2018.
13. W. Chen, T. Weise, Z. Yang and K. Tang, “Large-Scale Global Optimization using Cooperative Coevolution with Variable Interaction Learning,” PPSN-2010

References

14. A. Mirhoseini et al., “A Graph Placement Methodology for Fast Chip Design.” Nature 2021.
15. A. Fawzi et al., “Discovering Faster Matrix Multiplication Algorithms with Reinforcement Learning.” Nature 2022.
16. P .Hruby, T. Duff, A. Leykin and T. Pajdla, “Learning to Solve Hard Minimal Problems.” CVPR-2022.
17. C. Qian, K. Tang and Z. H. Zhou, “Selection Hyper-heuristics Can Provably be Helpful in Evolutionary Multi-objective Optimization.” PPSN-2016.
18. Z. A. Zhang, C. Bian and C. Qian, “Running Time Analysis of the (1+1)-EA using Surrogate Models on OneMax and LeadingOnes.” PPSN-2022.
19. L. Li and K. Tang, “History-Based Topological Speciation for Multimodal Optimization,” IEEE Transactions on Evolutionary Computation, 2015.
20. K. Tang, P. Yang and X. Yao, “Negatively Correlated Search,” IEEE Journal on Selected Areas in Communications, 2016.
21. F. Peng, K. Tang, G. Chen and X. Yao, “Population-based Algorithm Portfolios for Numerical Optimization,” IEEE Transactions on Evolutionary Computation, 2010.
22. K. Tang, F. Peng, G. Chen and X. Yao, “Population-based Algorithm Portfolios with Automated Constituent Algorithms Selection,” Information Sciences, 2014.
23. S. Liu, K. Tang and X. Yao, “Automatic Construction of Parallel Portfolios via Explicit Instance Grouping,” AAAI-2019.
24. F. Hutter, H. H. Hoos and Thomas Stützle, “Automatic Algorithm Configuration based on Local Search." AAAI-2007.
25. K. S. Miles and S. Bowly, “Generating New Test Instances by Evolving in Instance Space.” Computers & Operations Research, 2015.
26. K. Tang, S. Liu, P. Yang and X. Yao, “Few-shots Parallel Algorithm Portfolio Construction via Co-evolution,” IEEE Transactions on Evolutionary Computation, 2021.
27. S. Liu, K. Tang and X. Yao, “Generative Adversarial Construction of Parallel Portfolios,” IEEE Transactions on Cybernetics, 2022.

References

28. 刘晟材, 杨鹏, 唐珂. 近似最优并行算法组智能汇聚构造. 中国科学: 技术科学 2021.
29. Y. D. Kwon et al., “Pomo: Policy Optimization with Multiple Optima for Reinforcement Learning.” NeurIPS-2020.
30. Y. Nagata and S. Kobayashi, “A Powerful Genetic Algorithm Using Edge Assembly Crossover for the Traveling Salesman Problem.” INFORMS Journal on Computing, 2013.
31. S. Liu, Y. Zhang, K. Tang and X. Yao, “How Good Is Neural Combinatorial Optimization?” arXiv preprint arXiv:2209.10913, 2022.
32. M. F. Balcan et al., “How Much Data is Sufficient to Learn High-performing Algorithms? Generalization Guarantees for Data-driven Algorithm Design." STOC-2021.
33. S. Liu, K. Tang, Y. Lei and X. Yao, “On Performance Estimation in Automatic Algorithm Configuration,” AAAI-2020.
34. F. Hutter, H. H. Hoos and K. L. Brown, “Sequential Model-based Optimization for General Algorithm Configuration.” LION-2011.
35. M. Lindauer et al., “SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization.” Journal of Machine Learning Research, 2022.
36. Y. Bengio, A. Lodi, and A. Prouvost. "Machine learning for combinatorial optimization: a methodological tour d’horizon." European Journal of Operational Research, 2021.
37. A. M. Alvarez, Q. Louveaux and L. Wehenkel, “A Machine Learning-Based Approximation of Strong Branching.” INFORMS Journal on Computing, 2017.
38. X. Liang et al., “NeuroLKH: Combining Deep Learning Model with Lin-Kernighan-Helsgaun Heuristic for Solving the Traveling Salesman Problem." NeurIPS-2021.
39. J. Zheng, K. He, J. Zhou, Y. Jin and C.-M. Li, “Combining Reinforcement Learning with Lin-Kernighan-Helsgaun Algorithm for the Traveling Salesman Problem.” AAAI-2021
40. O. Vinyals, M. Fortunato, and N. Jaitly, “Pointer Networks,” NeurIPS-2015.
41. I. Bello, H. Pham, Q. V. Le, M. Norouzi and S. Bengio, “Neural Combinatorial Optimization with Reinforcement Learning,” ICLR-2017.

References

42. L. Xu, F. Hutter, H. H. Hoos and K. L. Brown, “SATzilla: Portfolio-based Algorithm Selection for SAT.” Journal of Artificial Intelligence Research, 2007.
43. K. L. Brown, E. Nudelman and Y. Shoham, “Empirical Hardness Models: Methodology and A Case Study on Combinatorial Auctions.” Journal of the ACM, 2009.
44. L. Kotthoff, “Algorithm Selection for Combinatorial Search Problems: A Survey.” AI Magazine, 2014.
45. F. Hutter, L. Xu, H. H. Hoos and K. L. Brown, “Algorithm Runtime Prediction: Methods & Evaluation.” Artificial Intelligence, 2014.
46. M. Lindauer, H. H. Hoos, K. L. Brown and T. Schaub, “Automatic Construction of Parallel Portfolios via Algorithm Configuration.” Artificial Intelligence, 2017.
47. D. Gorissen, T. Dhaene and F. D. Turck, “Evolutionary Model Type Selection for Global Surrogate Modeling.” Journal of Machine Learning Research, 2009.
48. J. Snoek, H. Larochelle and R. P. Adams, “Practical Bayesian Optimization of Machine Learning Algorithms.” NeurIPS-2012.
49. P. Larrañaga and J. A. Lozano, “Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation.” Springer Science & Business Media, 2001.

Collaborators

Dr. Shengcai Liu

Dr. Peng Yang

Dr. Juan Wang

Dr. Xiaofen Lu

Dr. Fei Peng

Dr. Zhenyu Yang

Prof. Xin Yao

Prof. Guoliang Chen

Thanks!

Comments/Questions are most welcome!