SAT Information size and its implications for industrial optimization

Maciej Drozdowski

Institute of Computing Science, Poznań University of Technology, Poland

This talk is about:

- information size of combinatorial problems
- exponential amount of information in SAT
- information acquisition in algorithms for combinatorial optimization
- fairer comparison of AI/ML methods with randomized metaheuristics

Related work

There is a connection between the combinatorial optimization **algorithm performance** and the **amount of information**:

- The *No Free Lunch Theorems* are proved using information - theoretic arguments[1]
- In graph coloring all greedy algorithms visiting nodes in a deterministic order and coloring nodes with the lowest feasible color, fail on some instance,
  – but *random sequential algorithm* visiting nodes in a random sequence cannot deterministically fail because it is connected to a source of unlimited amount of information[2] . . .

[1] D.H.Wolpert, W.G. Macready, No Free Lunch Theorems for Optimization, IEEE Trans. on Evolutionary Computation 1(1), April 1997.

[2] M.Kubale (ed.), Graph Colorings, American Mathematical Society, Providence, Rhode Island, 2004.

SAT

Definition

INPUT I: sums $k_j, j = 1, \ldots, m$, of binary variables, or their negations, chosen over a set of n binary variables $x_1, \ldots, x_n$.

REQUEST: find an assignment of 0/1 values to $x_1, \ldots, x_n$, i.e. vector $\overline{x}$, such that the conjunction $F(I, \overline{x}) = \prod_{j=1}^{m} k_j$ is 1.
If such a vector does not exist then signal $\emptyset$.

$|I|$ – instance I size, i.e., length of the string encoding I

"yes" instance – if for instance I : $\exists \overline{x} : F(I, \overline{x}) = 1$

"no" instance – otherwise

Search problem, e.g. SAT, as a string relation

Let:
Σ – an alphabet
e – some reasonable encoding scheme over Σ,
Σ^+ – a set of strings encoding instances of SAT using scheme e over alphabet Σ.

SAT-search is an example of a string relation:

Definition

Search problem Π is a string relation

$$R[\Pi, e] = \left\{ (a, b) : \begin{array}{l} a \in \Sigma^+ \text{ is the encoding of } I \text{ and} \\ b \in \Sigma^+ \text{ is the encoding of a solution} \\ \text{under coding scheme } e \end{array} \right.$$

Fixed code algorithm

Definition

Fixed code algorithm is an algorithm which is encoded in limited number of immutable bits.

Thus, a fixed code algorithm:

- does not change its code during the runtime,
- is not a source of randomness.

$|A|$ – the size of fixed code algorithm A in bits.

Truly random bit sequence

Definition

Truly random bit sequence (TRBS) is a sequence of bits, that has no shorter representation.

Thus:

- the only way to represent it is to store it in its whole entirety,
- an N-bit TRBS has information content N bits.

Information conservation

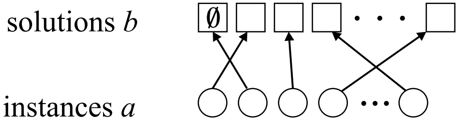
Postulate

Information is not created ex nihilo *by fixed code algorithms.*

Postulate

An algorithm to solve a problem must use at least the same amount of information as the amount of the information in the problem.

SAT as a string relation



SAT can be thought of as string relation $R[SAT, e]$:

- a mapping from strings a – instances, to strings b – solutions,
- set of arcs from instances a to solutions b,

How much information does such an object comprises?

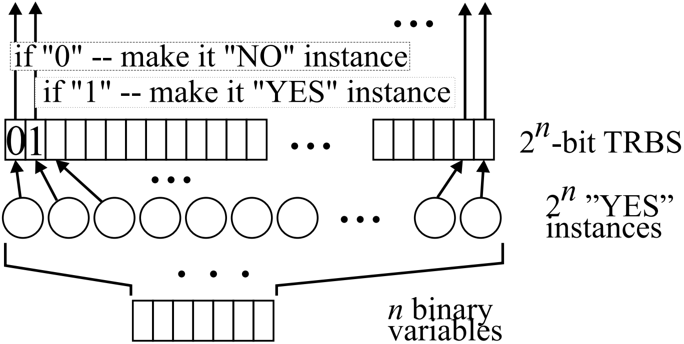
Exponential Information Content of SAT

Theorem

The amount of information in SAT grows at least *exponentially with instance size.*

Exponential Information Content of SAT

Proof: the idea



Exponential Information Content of SAT

Proof: 2^n of "Yes" instances in n variables

- n – number of variables,
- Let there be 4 clauses for $i = 1, \dots, n$:
  $k_{i1} = x_a + x_b + \widetilde{x_i}$, $k_{i2} = \overline{x_a} + x_b + \widetilde{x_i}$,
  $k_{i3} = x_a + \overline{x_b} + \widetilde{x_i}$, $k_{i4} = \overline{x_a} + \overline{x_b} + \widetilde{x_i}$.
- $\widetilde{x_i}$ – a variable x_i with or without negation
- No valuing of x_a, x_b alone makes the four clauses simultaneously equal 1.
- The four clauses may simultaneously become equal 1 only if $\widetilde{x_i} = 1$ (i.e. either $x_i = 1$ or $\overline{x_i} = 1$).
- Satisfying formula $F = k_{11} k_{12} k_{13} k_{14} \dots k_{n4}$ depends on valuing of variables $\widetilde{x_i}$ for $i = 1, \dots, n$.
- There are 2^n different ways of constructing formula F, leading to 2^n different "yes" instances with 2^n different solutions.

Exponential Information Content of SAT

Proof: Injecting 2^n-bit-long TRBS into SAT

- Consider a truly random bit sequence (TRBS) of length 2^n.
- Consider $j = 0, \ldots, 2^n - 1$ instances with clauses $k_{1j}, \ldots, k_{4j}$ and variables $\widetilde{x}_i$ as constructed above.
- If TRBS bit $j = 0, \ldots, 2^n - 1$ **is 1**, then the jth instance is constructed as "yes" instance by setting $\widetilde{x}_i$ in $k_{1j}, \ldots, k_{4j}$ consistently with number j binary encoding, for $i = 1, \ldots, n$.
- If TRBS bit $j = 0, \ldots, 2^n - 1$ **is 0**, then the jth instance is **spoiled** to a "no" instance by setting some $\widetilde{x}_i$ variable(s) in $k_{1j}, \ldots, k_{4j}$ inconsistently with number j binary encoding.

Exponential Information Content of SAT

Proof: amount of information

- cross-entropy of "Yes" instances

  $H(I, \text{"Yes"}) = -\sum_{I \in \mathcal{D}} p(I) \times \log q(I)$

  $\mathcal{D}$ – set of instances constructed in the above way,
  $p(I) = 1/|\mathcal{D}|$ – probability of encountering instance I,
  $q(I)$ – probability that I is a "Yes" instance;
- $q(I) = 1/2^n * 1/(2^{2^n})$

  $1/2^n$ – because a "Yes" instance can be spoiled in $2^n - 1$ ways
  2^{2^n} – because there are 2^{2^n} TRBSes of length 2^n.
- $H(I, \text{"Yes"}) = n + 2^n$.

Exponential Information Content of SAT

Proof: From the number of variables n to instance size $|I|$

- each number is limited from above by some constant K,
- $\Rightarrow$ the length of the encoding of the instance data is $|I| = 4n \times 3 \log K + \log K = 12n \log K + \log K$
- $n = (|I| - \log K)/(12 \log K)$
- $\Rightarrow$ cross-entropy of "Yes" instances: $H(I, \text{"Yes"}) = n + 2^n =$ $(|I| - \log K)/(12 \log K) + 2^{(|I| - \log K)/(12 \log K)}$ which is $\Omega(2^{d_1 |I|})$, where $d_1 = 1/(12 \log K) > 0$ is constant. $\square$

Algorithm information sources?

Knowing 1) the exponential lower bound on information size in SAT, and 2) by information conservation postulate, algorithms must acquire information to solve a problem like SAT.

Considering when an algorithm obtains information, types of algorithm are:

- **Off-line:** e.g. Machine Learning (AI), e.g. DRL, have large **static** amount of the information at the computation outset
- **On-line:** e.g. randomized metaheuristics increase the amount of information over the runtime by drawing random numbers
- **Hybrids:** e.g. hyper-heuristics, ML choosing branches and/or cuts in ILP, ML guiding MCTS

Machine Learning methods limitations

Observation

There always exists SAT instance I which size $\Omega(2^{d_1|I|})$ exceeds information size $|A|$ of any off-line information source fixed-code algorithm.

- $\Rightarrow$ ML are not a panacea, because:
- $\Rightarrow$ any pretrained ML method on a combinatorial optimization problem loses on solution quality with the increasing size of the problem,
- $\Rightarrow$ ML methods need re-training on new benchmarks with increasing instance sizes I.

Online information source algorithm

Proposition

Fixed code, online information source, algorithm with bounded bitrate v has less information than SAT.

Proof.

- assume the runtime is T (e.g. polynomially bounded),
- assuming limited random number acquisition speed v, the information acquired with the progress of time is $v \times T$,
- total amount of information in the instance I, the algorithm A, obtained in time T is $|I| + |A| + O(v \times T)$,
- which is less than $\Omega(2^{d_1|I|})$ bits of information in SAT. $\square$

How much information over time?

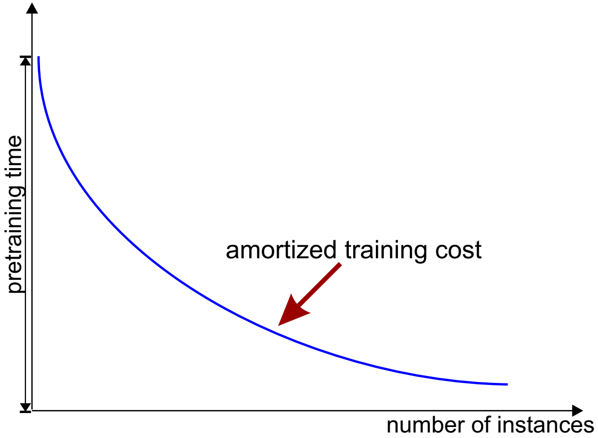
Comparing pretrained ML methods (static information gethered **offline**) with the randomized metaheuristics (information collected **online**) is biased because:

- while randomized metaheuristic collect information online,
- ML methods start with a lot of pre-computed information $|A|$,
  1) ML can amortize training cost,
  2) but the training cost is "magically" disappearing in the inference stage.

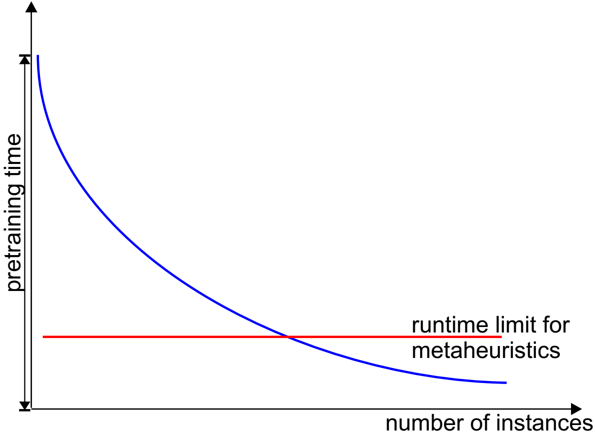
Unbounded training time $\Rightarrow$ unbounded algorithm information

- $\Rightarrow$ Is it fair to compare a metaheuristic with an ML method that has **unlimited training time?**
- $\Rightarrow$ How to compare pretrained ML methods with metaheuristics fairly?

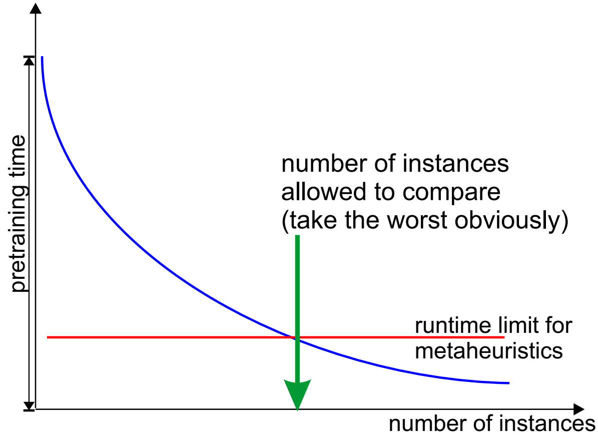
How to compare – a proposition



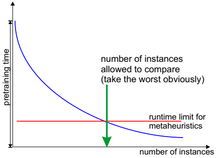
How to compare – a proposition



How to compare – a proposition



How to compare – a proposition



- the more training time given for the fixed training set
  $\Rightarrow$ the more time for a metaheuristic
- the less time for a metaheuristic
  $\Rightarrow$ the larger set of counterexamples for a ML methods
- the larger set of counterexamples for a ML methods
  $\Rightarrow$ the worse is the worst counterexample $\Rightarrow$
  $\Rightarrow$ the more training time to deal with counterexamples for a ML methods ...
- Only the worst-case performance really metters.

Existing research - example 1

S.Teck, T.San Pham, L.-M.Rousseau, P.Vansteenwegen, *Deep Reinforcement Learning for the Real-Time Inventory Rack Storage Assignment and Replenishment Problem*, EJOR 2025 in press.

Problems:

- decision agent developed by Deep Reinforcement Learning
- Training: 10k episodes $\times$ 10k decision points $=$ 4 days $\times$ 2 cores $==$ 192hours $\rightarrow$ problem: incomparable time units;
- Mathematical programming formulation given, Gurobi $\rightarrow$ problem: unknown Gurobi runtime;
- 5 competing (baseline) policies are greedy $\rightarrow$ problem: they do not acquire information online;

Existing research - example 1

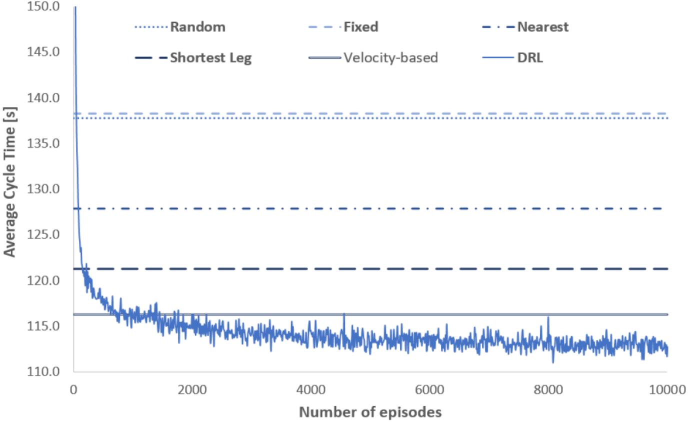


Figure 6: *Learning process of the decision-maker on a training instance with 600 storage locations.*

Existing research - example 1

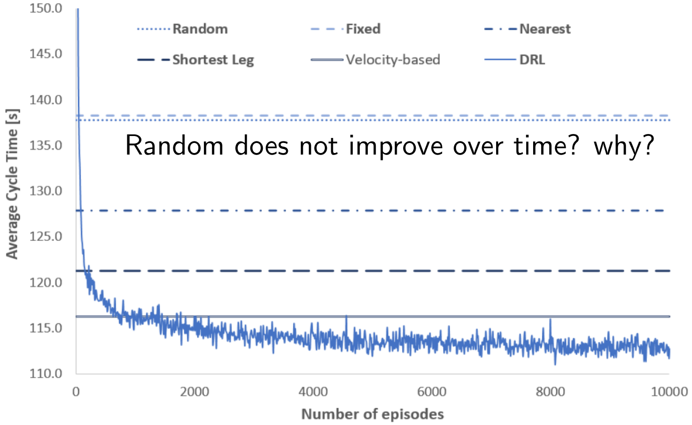


Figure 6: *Learning process of the decision-maker on a training instance with 600 storage locations.*

Existing research - example 2

W.Kool, H.van Hoof, M.Welling. *Attention, learn to solve routing problems!* International Conference on Learning Representations, 2019, arXiv:1803.08475.

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
	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)
TSP	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)

Existing research - example 2

W.Kool, H.van Hoof, M.Welling. *Attention, learn to solve routing problems!* International Conference on Learning Representations, 2019, arXiv:1803.08475.

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

	training [sec]:	33k			98k			165k		
		$n = 20$			$n = 50$			$n = 100$		
	Method	Obj.	Gap	Time	Obj.	Gap	Time	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)

Existing research - example 2

W.Kool, H.van Hoof, M.Welling. *Attention, learn to solve routing problems!* International Conference on Learning Representations, 2019, arXiv:1803.08475.

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

training [sec]: 33k (n = 20), 98k (n = 50), 165k (n = 100)

	Method	n = 20 Obj.	n = 20 Gap	n = 20 Time	n = 50 Obj.	n = 50 Gap	n = 50 Time	n = 100 Obj.	n = 100 Gap	n = 100 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)

Local search

Local search

Existing research - example 2

W.Kool, H.van Hoof, M.Welling. *Attention, learn to solve routing problems!* International Conference on Learning Representations, 2019, arXiv:1803.08475.

- at TSP size $n = 100$, training time 165k sec, Concorde time limit 3min $\Rightarrow$ test set size: 165k sec/180 sec$\approx$ 916.
- Their test set size: 10000 $\rightarrow$ problem: regression to the mean, no worst cases known.
- problem: local search methods stop by themselves.[3]
- problem: specific dataset – "*n* node locations are sampled uniformly at random in the unit square"
  $\rightarrow$ do these methods learn space-filling curves?

[3] discovered at the end of a chain of citations

Existing research - example 3

I.Echeverria, M.Murua, R.Santana, *Solving the flexible job-shop scheduling problem through an enhanced deep reinforcement learning approach*, 2024, arXiv:2310.15706

- Flexible job-shop, DRL, Markov Decision Process, Graph Neural Networks
- problem: no training time given, only "the maximum number of episodes . . . $\in [10000, 15000]$",
- they primarily compared to 6 greedy dispatching rules, but also implicitly OR-Tools CP solver.
- good: results on particular instances given:
- vdata benchmark – 6 instances out of 40 match the known upper bound (i.e. ML not a panacea)
- Behnke benchmark – 17 instances out of 45 better than OR-tools CP-SAT solver (ML inference time 61s, OR-Tools 1800s).

Finish

- **NP**-hard problems (like SAT) have exponential amount of information
- Algorithms obtain this information offline (ML) or online (randomized metheuristics)
- Comparing ML with randomized metaheuristics unfair if training cost ignored
- A fairer ML vs metaheuristic comparison proposed $\rightarrow$ amortized number of the worst cases
- Further work needed to compare metaheuristics and ML methods fairly and in unified way

Thank you for listening

details on SAT information size at `https://arxiv.org/abs/2401.00947`