Advanced Topics

Lecture 9

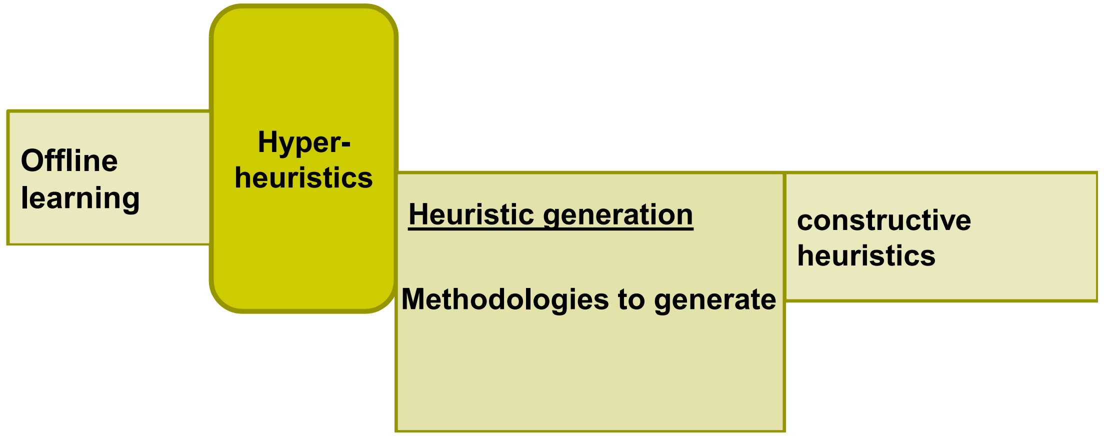
Ender Özcan

Computational Optimisation & Learning Lab

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Policy Matrix Evolution for Generation of Heuristics



S. Asta, E. Özcan, and A.J. Parkes, CHAMP: Creating Heuristics via Many Parameters for Online Bin Packing, Expert Systems With Applications, vol. 63, pp. 208-221, doi:10.1016/j.eswa.2016.07.005, 2016

Computational Optimisation & Learning Lab

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

1D Offline Bin Packing

Pack a **set** of items of sizes s_i for $i = 1, \ldots, n$

- Sizes are integer values and $s_i \in [1, C]$
- C is the fixed capacity of each bin

in such a way that

- Never exceed bin capacity
- Minimise number of bins used

Standard NP-hard problem

1D Online Bin Packing

Pack a **stream** of items of sizes s_i for $i = 1, \ldots$

- Sizes are integer values and $s_i \in [1, C]$
- C is the fixed capacity of each bin

upon their arrival (one item at a time)

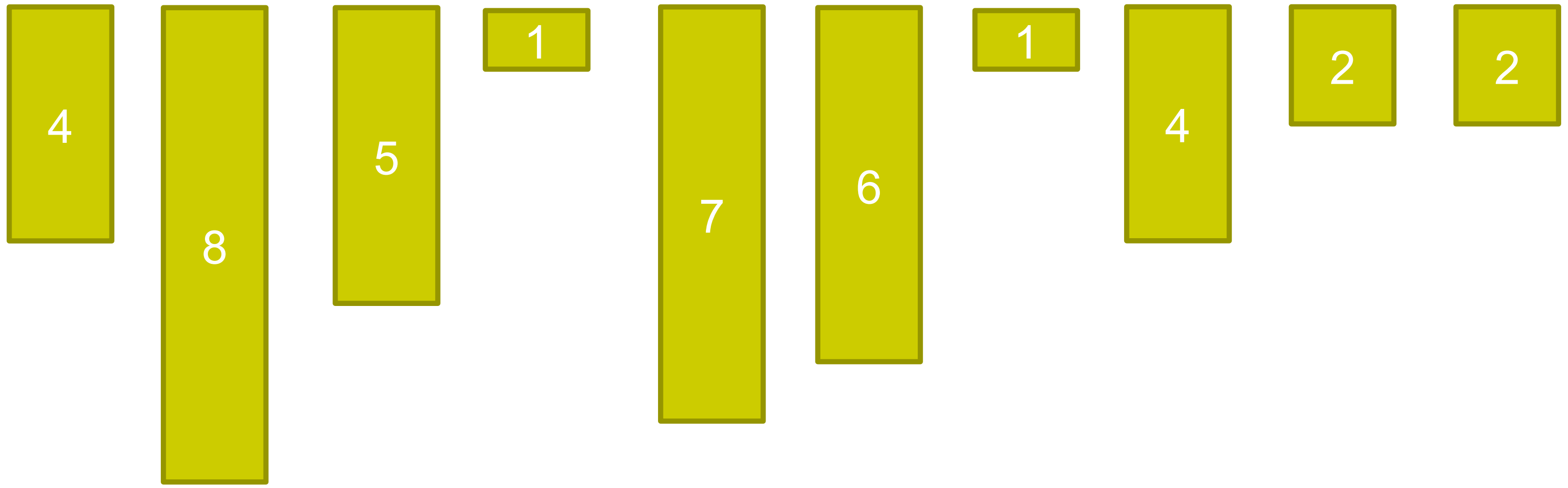
in such a way that

- Never exceed bin capacity
- Minimise number of bins used (Maximise the average bin-fullness) **at the end**

Standard Heuristics: First-fit and Best-fit

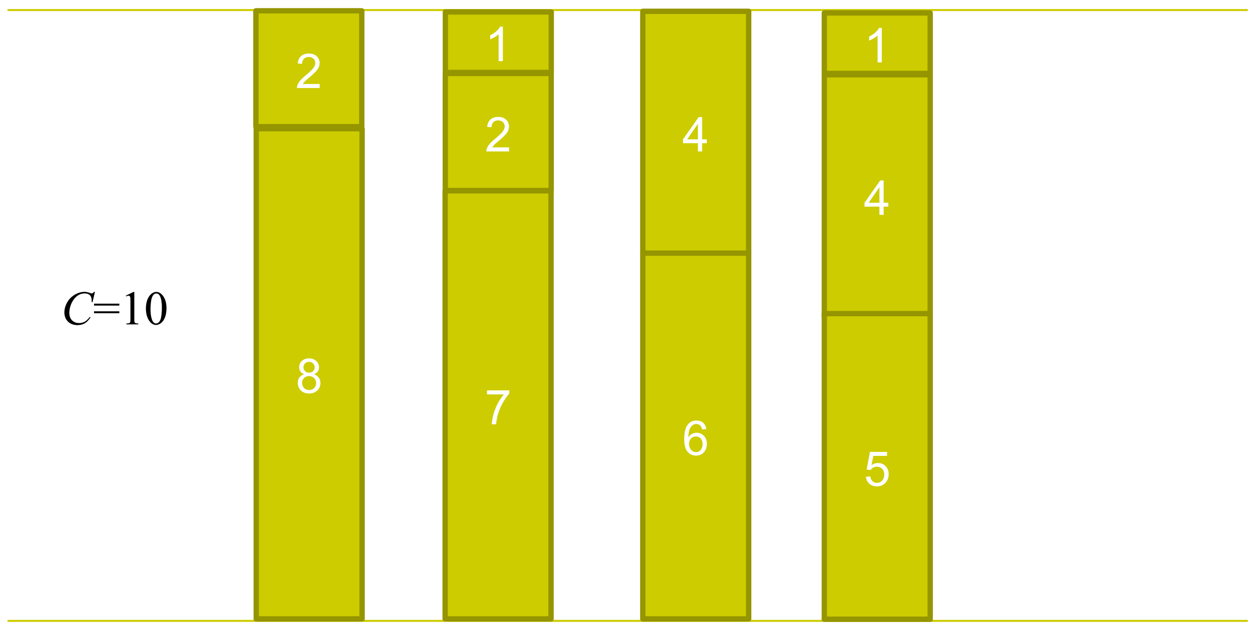
- **First Fit Heuristic (BP-FF).** Place the items in the order in which they arrive. Place the next item into the lowest numbered (left most) bin in which it fits. If it does not fit into any open bin, start a new bin.
- **Best Fit Heuristic (BP-BF).** Place the items in the order in which they arrive. Place the next item into that bin which will leave the least room left over after the item is placed in the bin. If it does not fit in any bin, start a new bin.

$S = \{4, 8, 5, 1, 7, 6, 1, 4, 2, 2, \ldots\}$ $C=10$



Standard Heuristics: First-fit and Best-fit

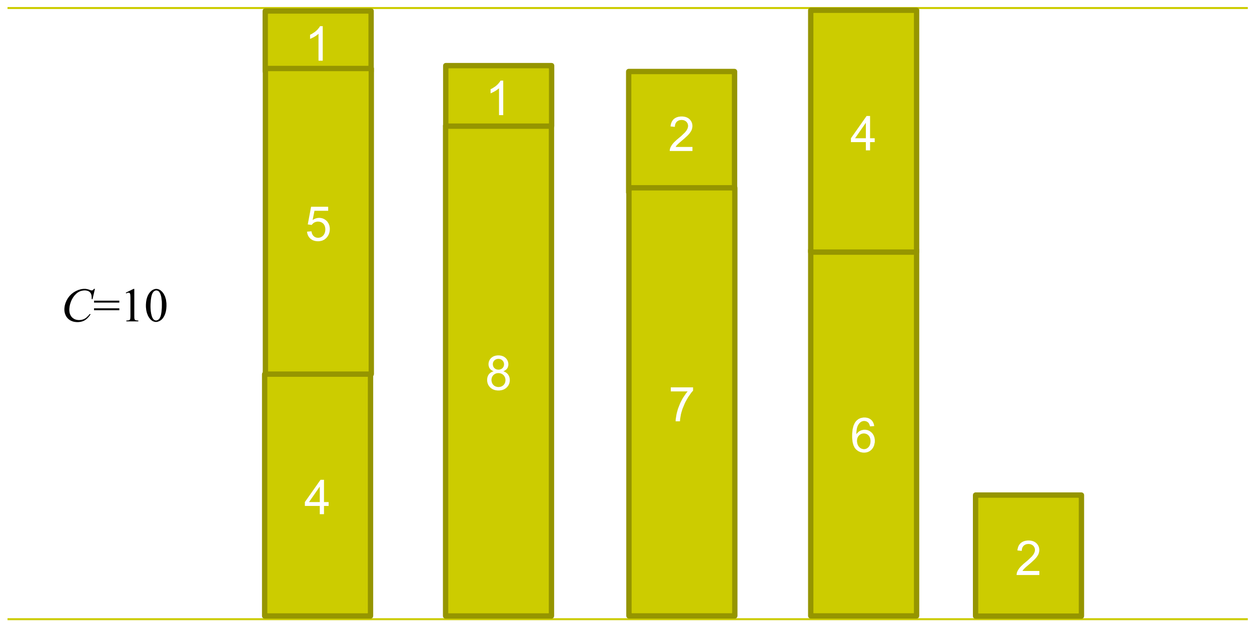
$S = \{8, 7, 6, 5, 4, 4, 2, 2, 1, 1\}$ C=10 **Offline Case**



4 bins (average bin-fullness: 100%)

Standard Heuristics: First-fit and Best-fit

$S = \{4, 8, 5, 1, 7, 6, 1, 4, 2, 2, \ldots\}$ C=10 **Online Case**



5 bins (average bin-fullness: 80%)

“Index Policies”

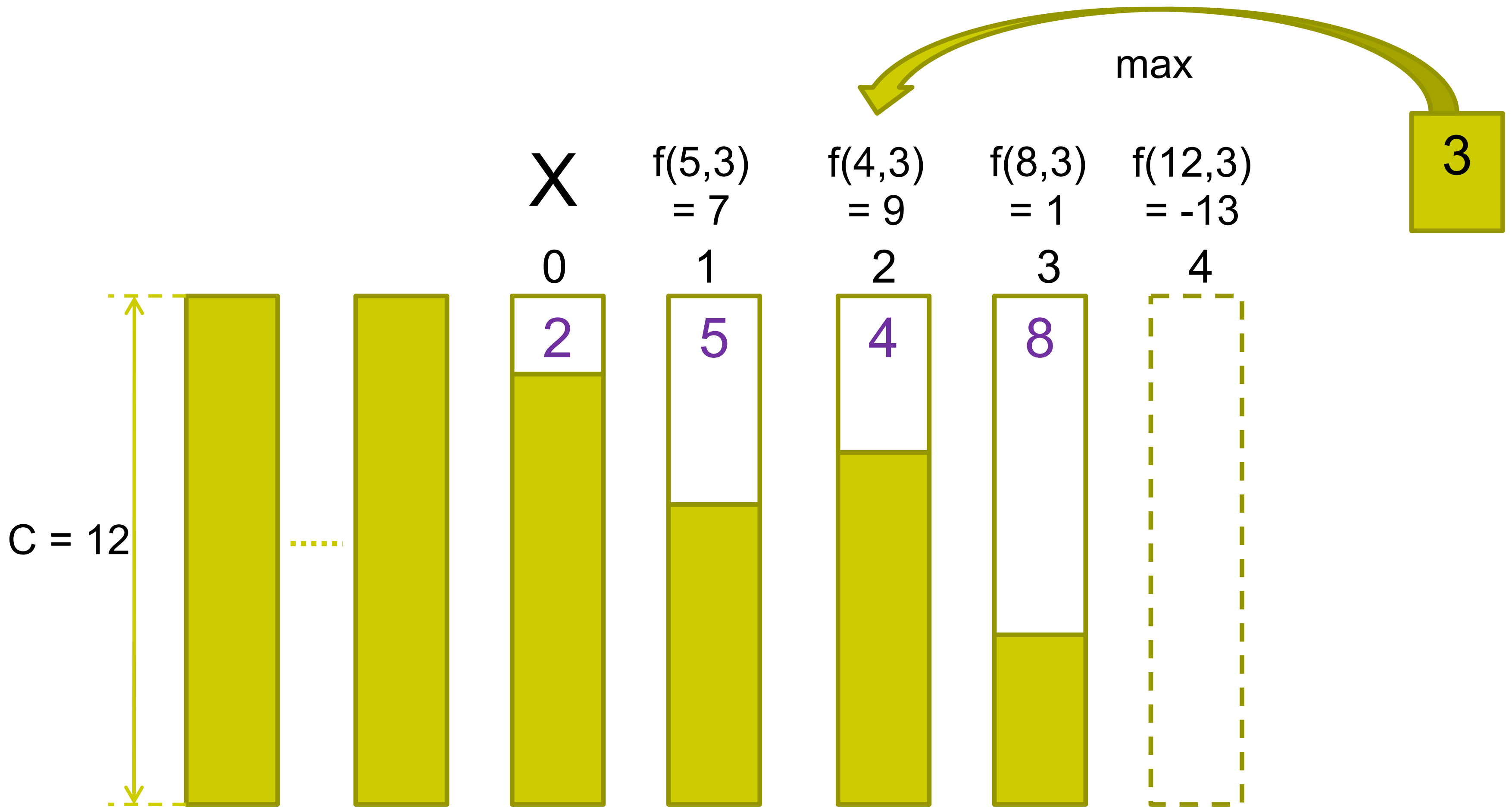
- “index policy”: each choice option is given a score, or “index value” independently of the other options
  - The option with the highest index value is taken
  - Also need a rule to break ties
- Although index policies are a special case, in some situations, they can be optimal, or at least very good

Index Policies for 1D Online Bin Packing

- Given
  - r : remaining capacity of bin (residual space)
  - s : item size
- score of bin is f(r,s) for some function f

- Given a new item of size then place into bin with largest value of f(r,s)
- We will break any ties using FF:
  - **place item in earliest bin with the best available score**

1D Bin Packing – Intermediate Step



1D Online Bin Packing

- A new empty bin is always available (**open**)

- A bin is **closed** if it can take no more items
  - e.g. if **residual space** is smaller than size of any item

- We need a *good* “policy”, i.e. a method to assign a new item upon its arrival to one of the *open* bins

Potential General Method for 1D Online Bin Packing

- On arrival of new item of size s_i
  - Inspect the current set of open bins
  - Simultaneously use the entire set of residual spaces in the open bins to pick where to place the new item

- This is difficult and expensive (in general)

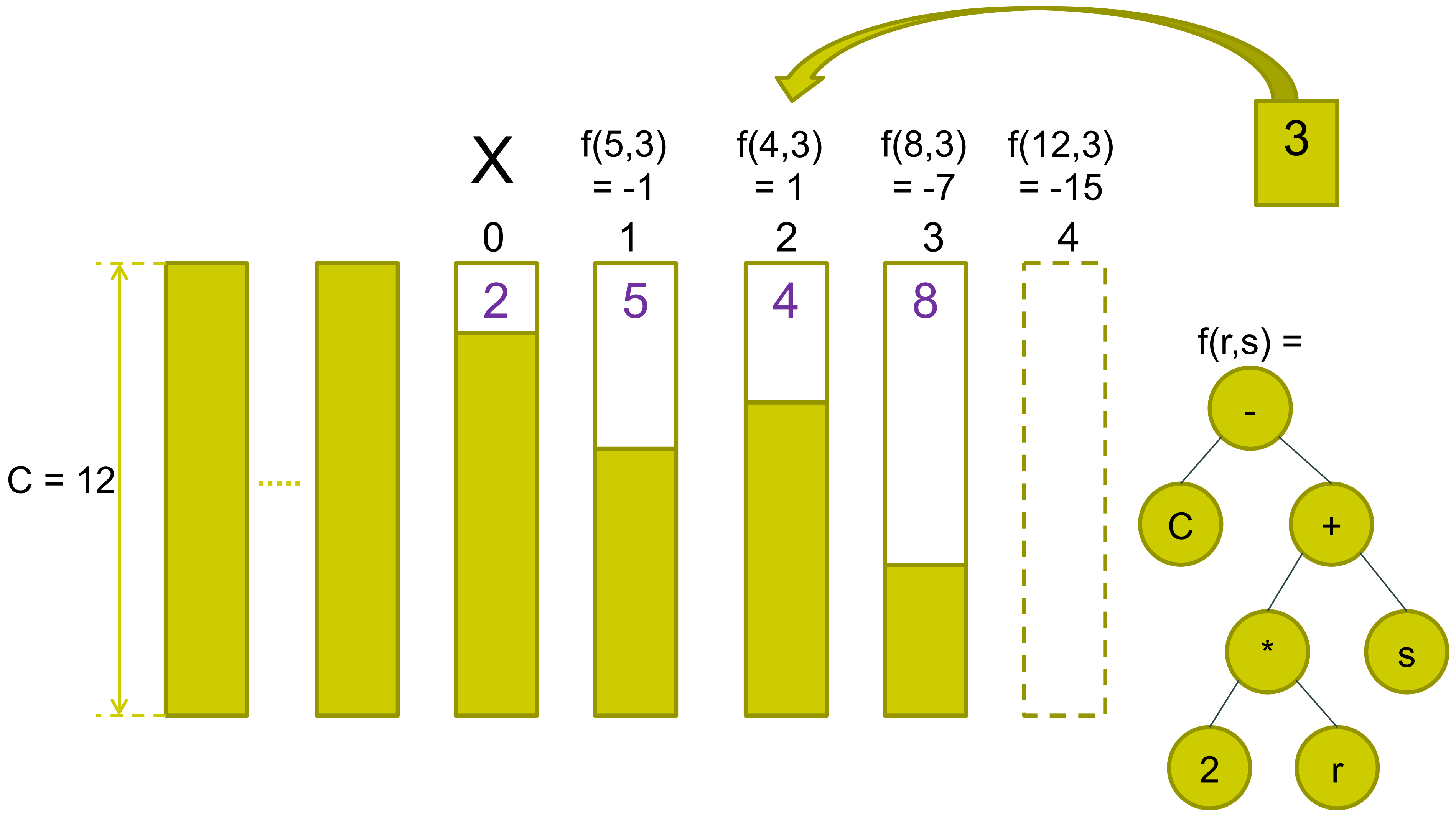
Generating Heuristics

- Within search methods, often have score functions, “index functions” to help make some choice
  - difficult to invent successful ones; want to automate this
- GP approach: evolve arithmetic score functions

E. K. Burke, M. R. Hyde, G. Kendall, J. R. Woodward: Automatic heuristic generation with genetic programming: evolving a jack-of-all-trades or a master of one. GECCO 2007: 1559-1565 [PDF]

- Use Genetic Programming to learn f(r,s)
- f(r,s) is represented as arithmetic function tree
- Automatically creates functions that at least match FF, BF

GP – 1D Online Bin Packing



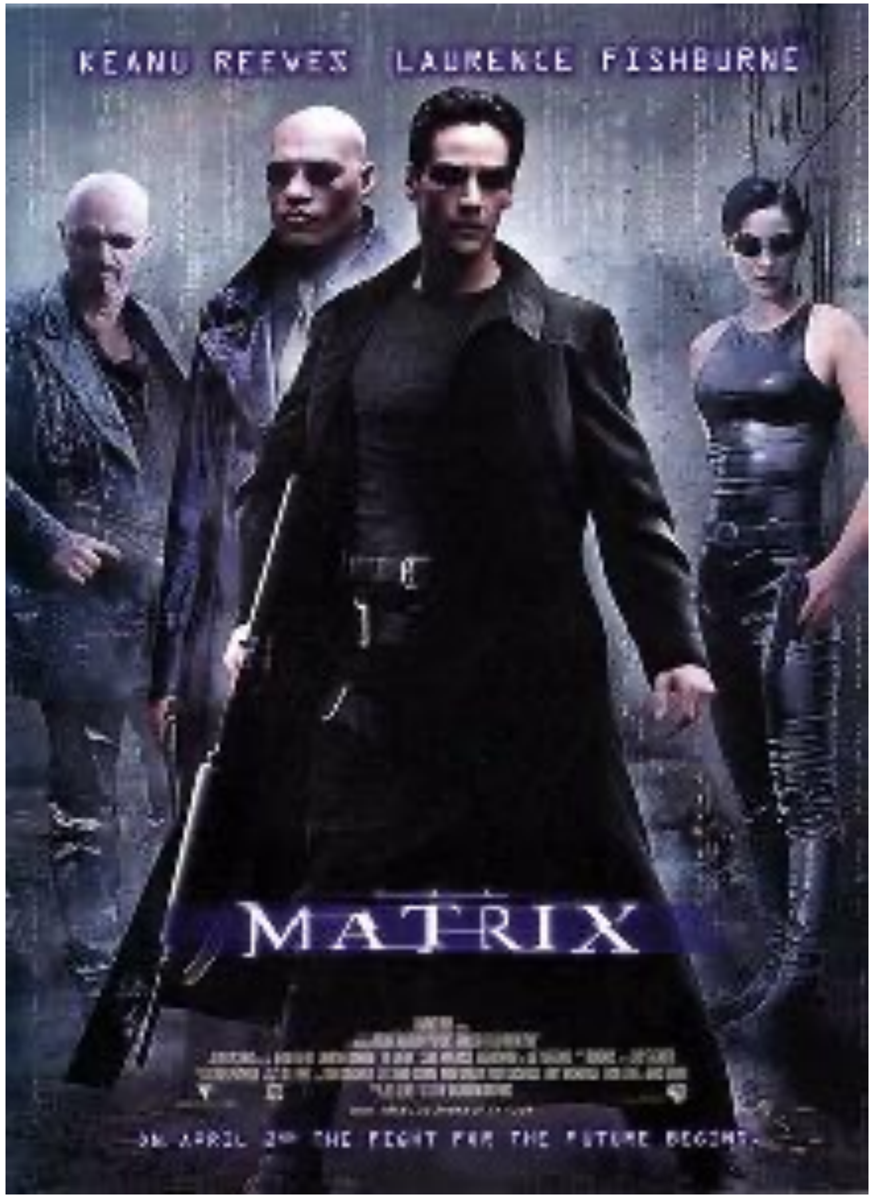
Generating Heuristics

Challenge:

- Space of functions, as used in GP,
  - is hard to understand
  - potentially biased because of the choice of representation
  - some perfectly good functions might have “bloated” representations

Can one do even better?

Is there another way to find policies?

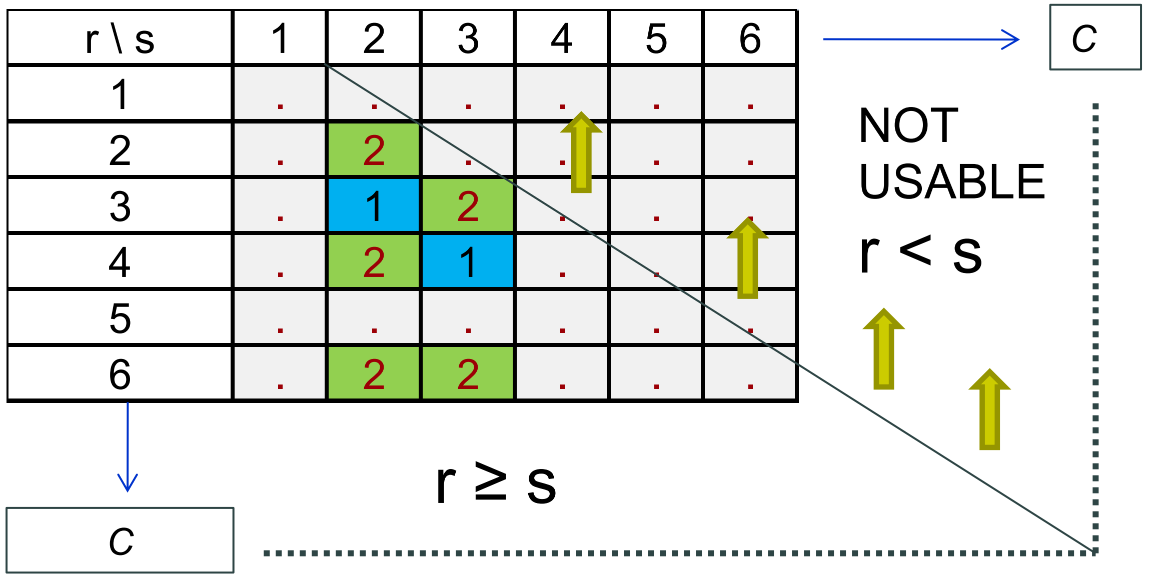


Source: http://en.wikipedia.org/wiki/The_Matrix

Matrix View of Policies/Heuristics

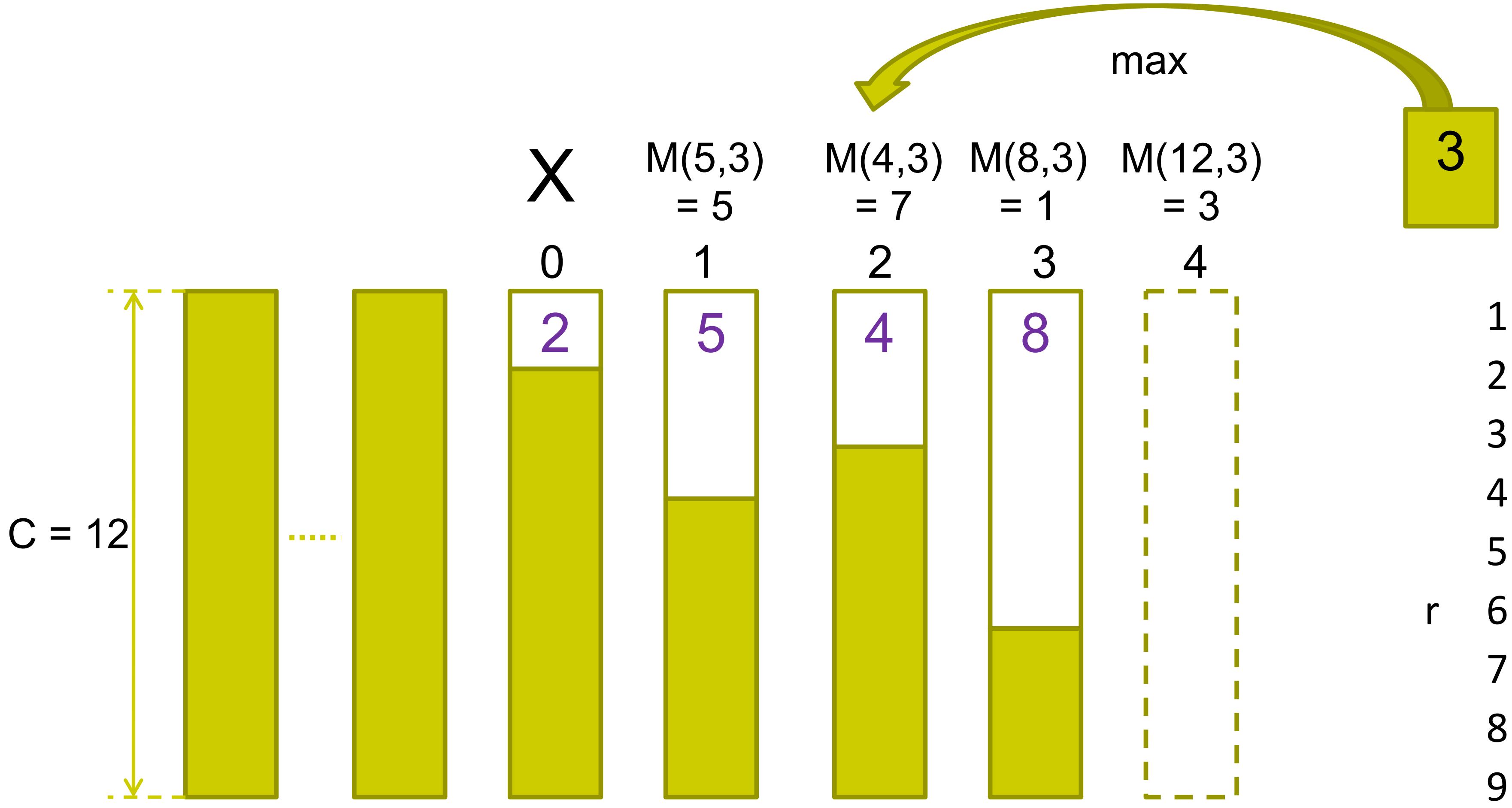
- Since all item sizes (s) and residual capacities (r) are integer, then f(r,s) is simply a large (*C*x*C*) matrix M(r,s) of **parameter values**

M(r,s)



r \ s	1	2	3	4	5	6
1	.	.	.	.	.	.
2	.	2	.	.	.	.
3	.	1	2	.	.	.
4	.	2	1	.	.	.
5	.	.	.	.	.	.
6	.	2	2	.	.	.

Policy Matrix – 1D Online Bin Packing



$$\begin{array}{c|ccc} r \backslash s & 2 & 3 & 4 \\ \hline 1 & * & * & * \\ 2 & * & * & * \\ 3 & * & * & * \\ 4 & * & 7 & * \\ 5 & * & 5 & * \\ 6 & * & * & * \\ 7 & * & * & * \\ 8 & * & 1 & * \\ 9 & * & * & * \\ 10 & * & * & * \\ 11 & * & * & * \\ 12 & * & 3 & * \end{array}$$

Uniform (random) Instances

We empirically studied matrix policies on Uniform Bin Packing problems

UBP(C, s_{min}, s_{max}, N) (problem generator)

- Bin capacity C
- N items are generated with integer sizes independently taken uniformly at random from the range [s_{min}, s_{max}]
  - N is usually taken to be large: 100k

UBP(6,2,3)

- (Bin capacity 6, items are size 2 or 3 only.)
- The only perfect packings are
  - 2+2+2
  - 3+3
- Hence the ‘obvious’ policy is …
- … “never mix even and odd sizes”

UBP(6,2,3)

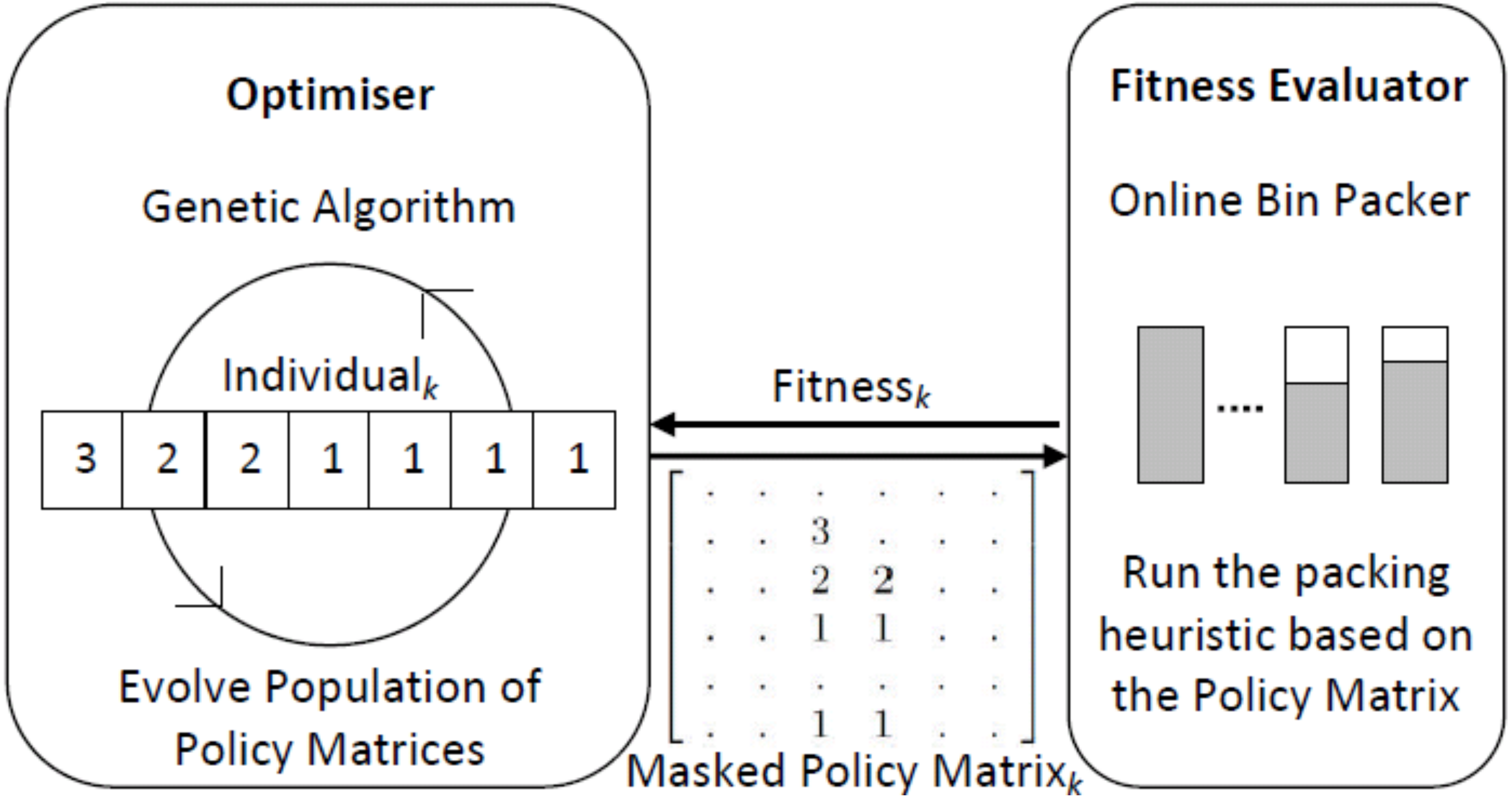
- … “never mix even and odd sizes”
- Index policy as a matrix:
  - rows: residual capacity of the bin
  - columns: item size
- Ties are broken using First-Fit (FF)
- Grey entries “.” are never usable

resid \ item	1	2	3	4	5	6
1	.	.	.	.	.	.
2	.	2	.	.	.	.
3	.	1	2	.	.	.
4	.	2	1	.	.	.
5	.	.	.	.	.	.
6	.	2	2	.	.	.

Creating Heuristics viA Many Parameters - CHAMP

- Basic idea:
  - Take values in matrix M(r,s) to be integers
  - Do (meta-)heuristic search to find good choices for M(r,s): Evaluation is by simulation
- Our Original Expectation:
  - the matrix will tweak the functions from GP and might slightly improve performance
- Potential expected disadvantages:
  - matrices can be much more verbose than functions
  - they fail to take into account of the good structure captured by functions

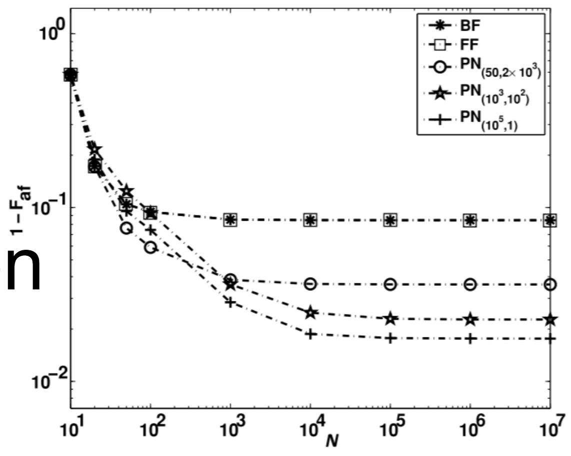
CHAMP-GA Architecture



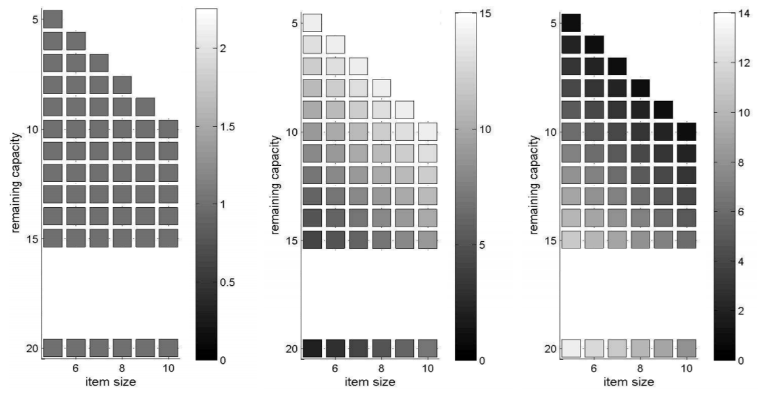
$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 3 & \cdot & \cdot & \cdot \\ \cdot & \cdot & 2 & 2 & \cdot & \cdot \\ \cdot & \cdot & 1 & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & 1 & \cdot & \cdot \end{bmatrix}$$

Implementation Details

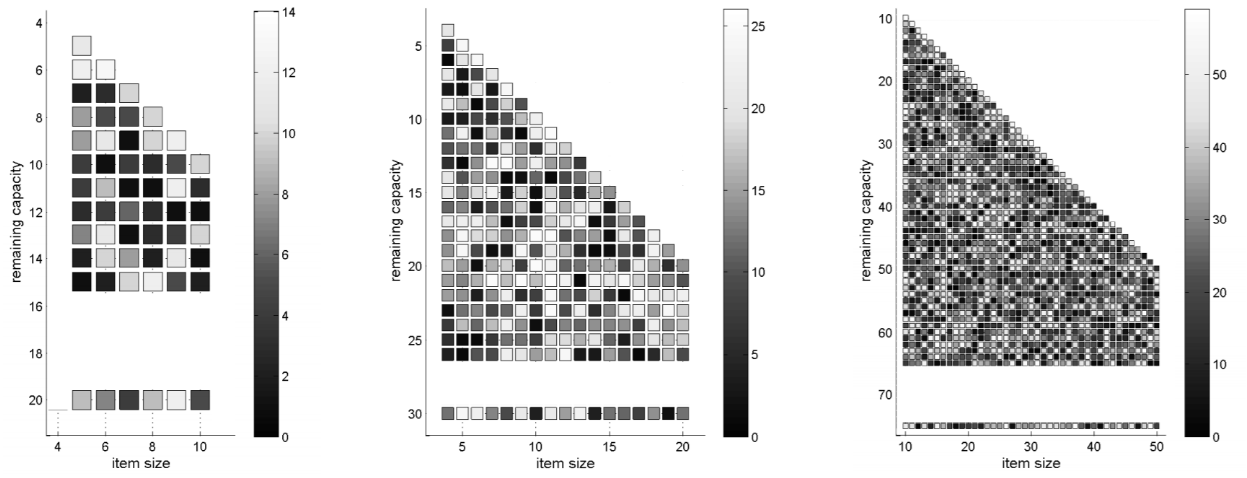
- Apply a standard GA for training
  - Trans-generational (with weak elitism, population size:C/2), tournament selection (tour size:2), Uniform Crossover, standard mutation (with probability 1/*L*), termination after 200 iterations, number of trials: 1.
- Only the active members of the matrix are stored as integer/binary values ($GA_{Original}$/GA_{Binary}) in the chromosome (GA_{FFinit}: $GA_{Original}$ where initial population contains -seeded with- an individual representing FF)
- Evaluation:
  - write matrix to a file
  - use matrix as input for a program that packs many items



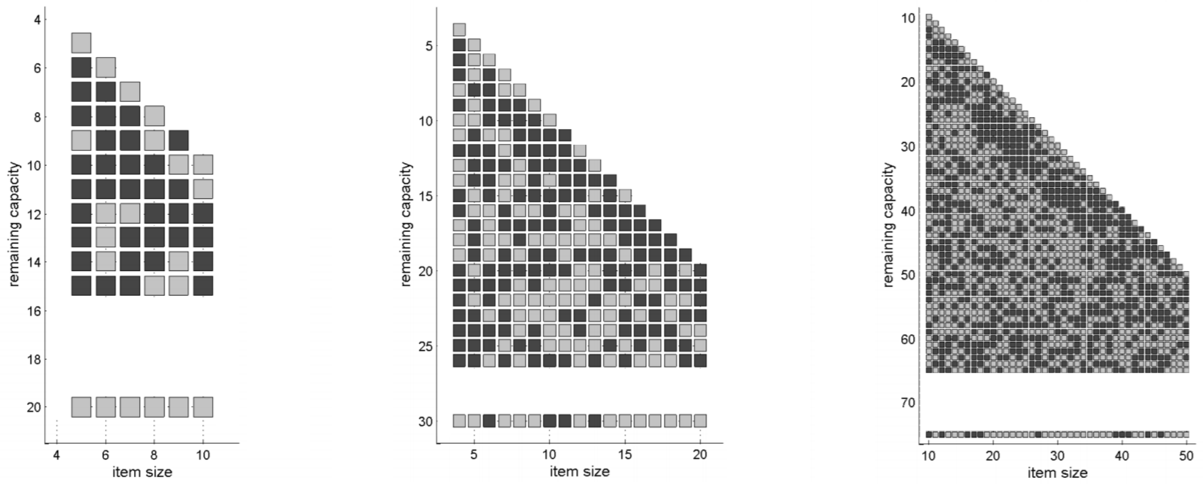
Examples of good evolved matrices


FF BF WF

integer



binary



(a) UBP$(20, 5, 10, 10^5)$ (b) UBP$(30, 4, 20, 10^5)$ (c) UBP$(75, 10, 50, 10^5)$

- Does not look like a smooth function
  - "Weird"
  - Seems to have spikes

Results – Best of runs for GA

Method	UBP$(6, 2, 3, 10^5)$	UBP$(15, 5, 10, 10^5)$	UBP$(20, 5, 10, 10^5)$	UBP$(30, 4, 20, 10^5)$	UBP$(30, 4, 25, 10^5)$	UBP$(40, 10, 20, 10^5)$	UBP$(60, 15, 25, 10^5)$	UBP$(75, 10, 50, 10^5)$	UBP$(80, 10, 50, 10^5)$	UBP$(150, 20, 100, 10^5)$
BF	92.30	99.62	91.55	96.84	98.38	90.23	92.55	96.08	96.39	95.82
FF	92.30	99.55	91.54	96.68	97.93	90.22	92.55	95.91	96.29	95.64
WF	91.70	86.58	90.54	88.61	84.10	88.66	90.80	87.94	89.25	87.73
Harmonic	-	74.24	90.04	73.82	74.21	89.10	85.18	71.59	72.96	71.97
$GA_{Original}$	**99.99**	**99.63**	98.18	99.41	98.39	**96.99**	**99.68**	98.22	**98.54**	**97.88**
GA_{FFinit}	**99.99**	99.61	98.15	99.10	99.25	96.05	98.28	98.43	97.87	96.92
GA_{Binary}	**99.99**	99.61	**98.42**	**99.58**	**99.55**	96.75	96.96	**98.45**	98.46	97.63

Conclusions

- Can use standard metaheuristics to create policies expressed in matrix representation
    - Policies exist that out-perform standard heuristics
    - Finding the policies is easier than expected
    - There are many different policies with similar performance
    - The policies are “weirder” than expected
        - The good policies could have “random” structures
        - Not necessarily easy to capture with an algebraic function of GP
    - The results can be “analysed” (inspected) to produce simple policies that out-perform standard ones
        - and that then scale to larger problems

References

- E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. Woodward (2009). Exploring hyper-heuristic methodologies with genetic programming. In C. Mumford and L. Jain (eds.), Computational Intelligence, Intelligent Systems Reference Library, pp. 177-201. Springer [PDF]
- E. Özcan, and A. J. Parkes, Policy Matrix Evolution for Generation of Heuristics, Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO '11), Natalio Krasnogor (Ed.). ACM, New York, NY, USA, pp. 2011-2018 (won the best paper award in the Self-* track), 2011 [original-PDF]. [PDF]
- A. J. Parkes, E. Özcan, M. Hyde, Matrix Analysis of Genetic Programming Mutation, EuroGP 2012, Lecture Notes in Computer Science 7244, pp. 158-169, 2012. [PDF]
- S. Asta, E. Özcan and A. J. Parkes, Dimension reduction in the search for online bin packing policies, Proceedings of the 2013 Genetic and Evolutionary Computation Conference Companion, pp. 65-66, 2013. [PDF]
- S. Asta, E. Özcan, and A.J. Parkes, CHAMP: Creating Heuristics via Many Parameters for Online Bin Packing, Expert Systems With Applications, vol. 63, pp. 208-221, doi:10.1016/j.eswa.2016.07.005, 2016 [original PDF]. [PDF]

Data Science (Machine Learning) Improved Hyper-heuristic Optimisation

Computational Optimisation & Learning Lab

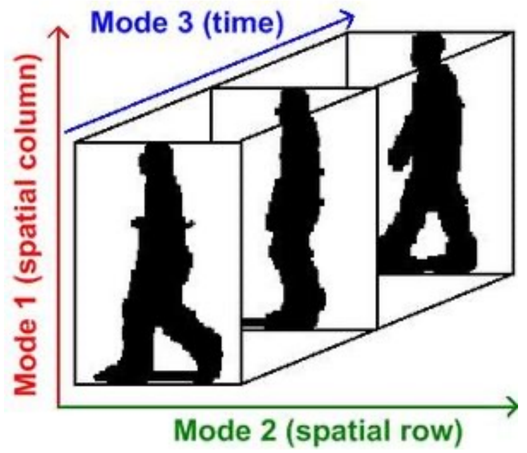
The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Single Objective Hyper^2-heuristic: A Data Science Improved Hyper-heuristic

- Many real-world data are multidimensional
  - Very high-dimensional (big) with a large amount of redundancy
- Multi-dimensional arrays representing such data describe a tensor

Many applications in signal processing, psychometrics, social network analysis, genomics and more



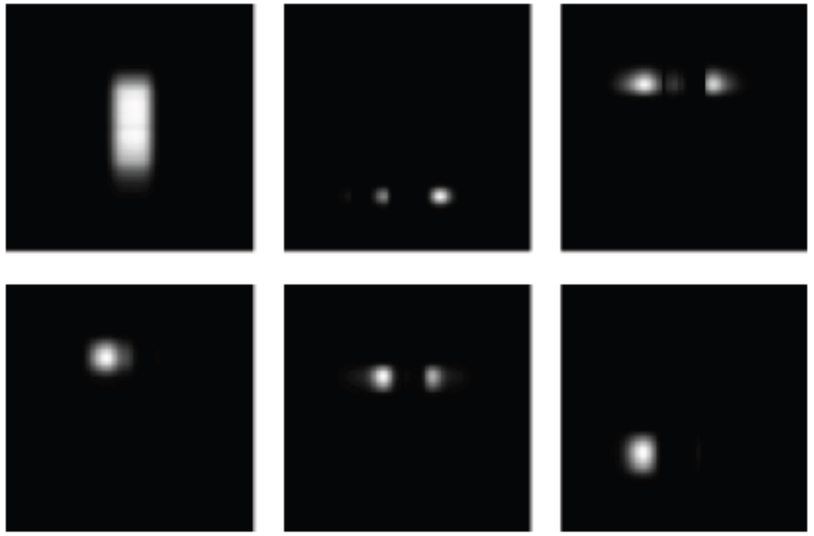
S. Asta and E. Özcan, A Tensor-based Selection Hyper-heuristic for Cross-domain Heuristic Search, Information Sciences, vol. 299, pp. 412-432, 2015

Tensor Factorisation

- There are different decomposition methods, we use Canonical Polyadic (CP) factorisation

$$\hat{\mathcal{T}} = \sum_{k=1}^{K} \lambda_k \mathbf{a}_k \circ \mathbf{b}_k \circ \mathbf{c}_k$$

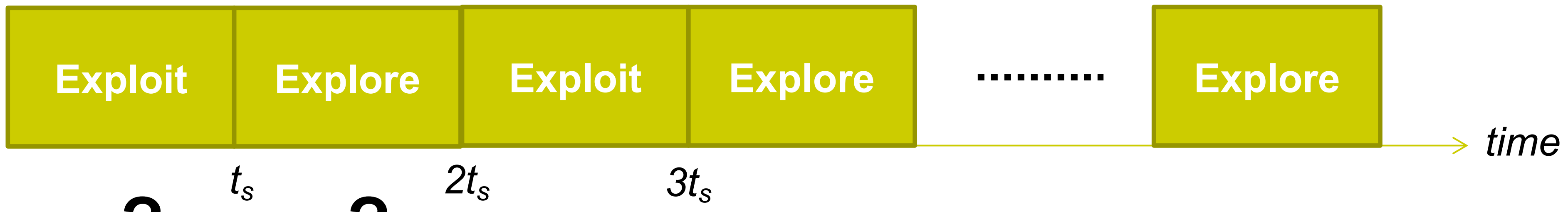
  - This gives a projection of 3D data onto 1D vectors
  - Helps to discover latent structures in data, quantifying the relationship between pairs of different components



SOURCE: B. Krausz, C. Bauckhage, Action recognition in videos using nonnegative tensor factorization., in: ICPR, IEEE, 2010, pp. 1763–1766.

Proposed Approach – Ideas

- The balance between exploration and exploitation is crucial (e.g. ILS)



? **?**

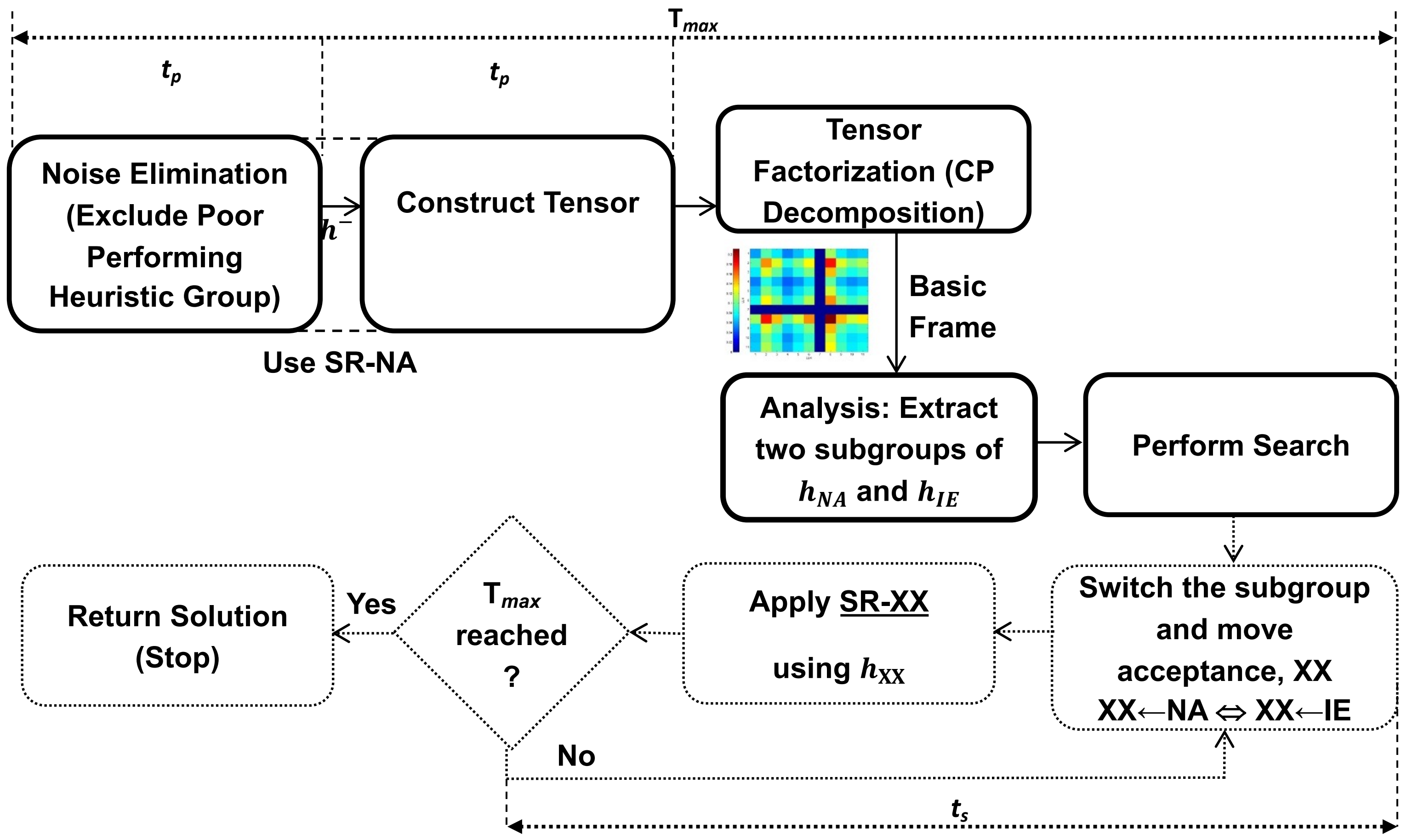
IE **NA**

***h*: set of low level heuristics (MU+RC+LS)**

$$h_{IE} \cup h_{NA} = h \quad (h_{IE} \cap h_{NA} = \emptyset)$$

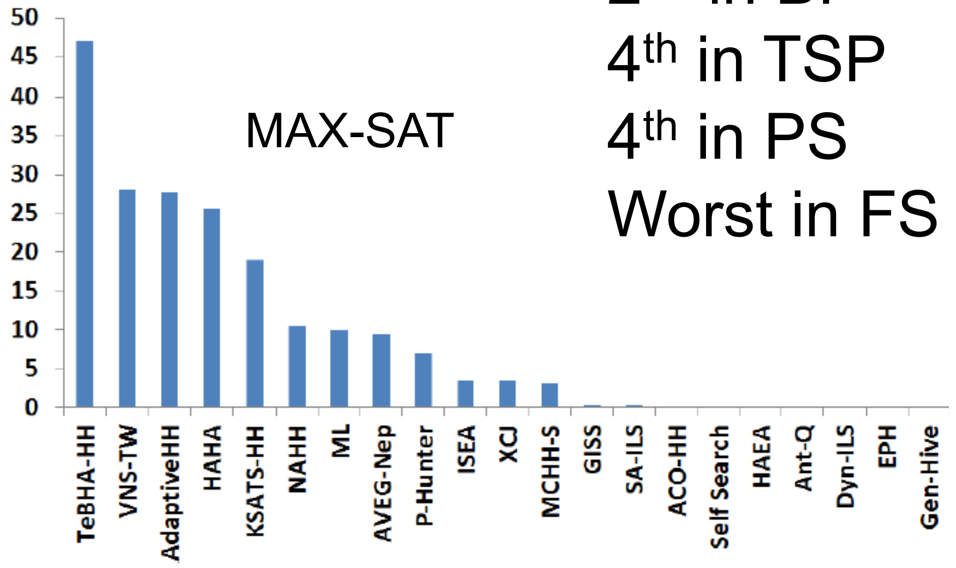
- Mix move acceptance methods
- Use machine learning to partition the low level heuristics associated with each method

Proposed Approach – TeBHA-HH

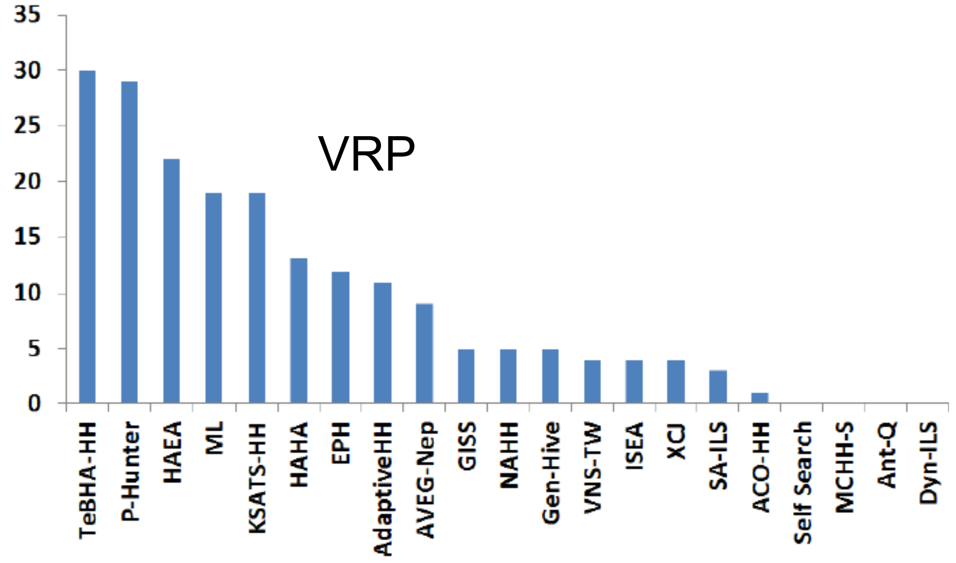


T_{max}
t_p
t_p
Noise Elimination (Exclude Poor Performing Heuristic Group)
h^-
Construct Tensor
Tensor Factorization (CP Decomposition)
Use SR-NA
Basic Frame
Analysis: Extract two subgroups of h_{NA} and h_{IE}
Perform Search
Switch the subgroup and move acceptance, XX XX←NA ⇔ XX←IE
Apply SR-XX using h_{XX}
T_{max} reached ?
Yes
No
Return Solution (Stop)
t_s

Results



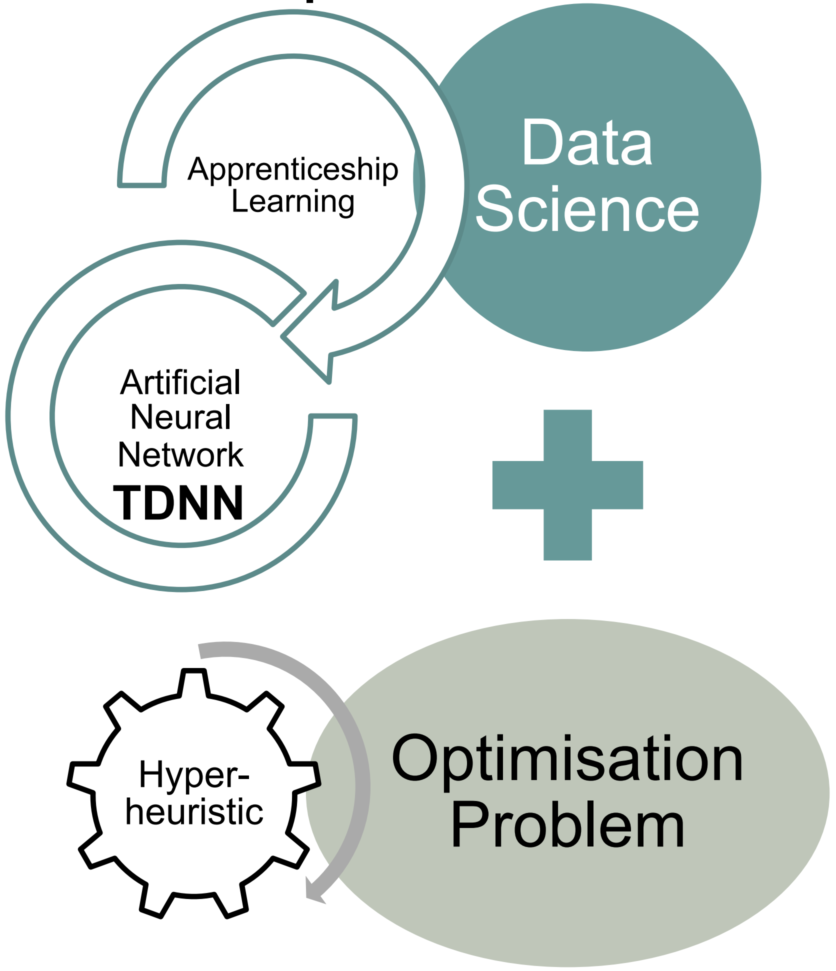
2nd in BP
4th in TSP
4th in PS
Worst in FS



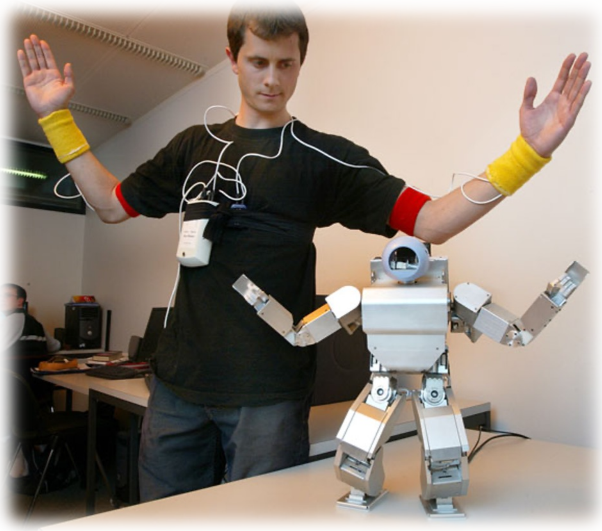
Rank	Name	Score
1	AdaptiveHH	162.83
2	**TeBHA-HH**	**148.85**
3	VNS-TW	118.83
4	ML	117.50
5	P-Hunter	84.75
6	EPH	83.25
7	NAHH	68.50
8	HAHA	65.58
9	ISEA	62.50
10	KSATS-HH	52

A Single Objective Hyper^2-heuristic – Apprenticeship Learning

Expert: MCF-AM

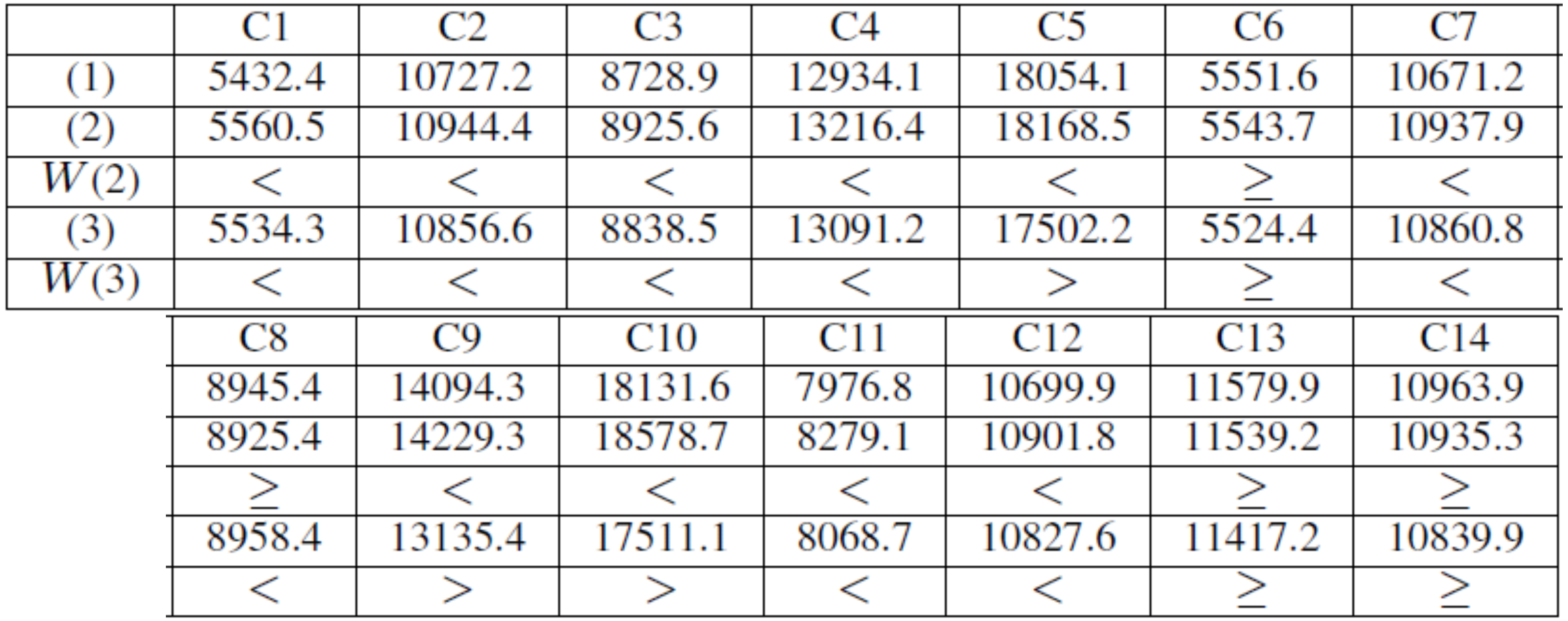


Open Vehicle Routing


http://www.calinon.ch/images/hoap2-xsens01.jpg

Automated generation of a new hyper-heuristic by observing an expert hyper-heuristic in operation

Results on Open VRP



	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14
(1)	5432.4	10727.2	8728.9	12934.1	18054.1	5551.6	10671.2	8945.4	14094.3	18131.6	7976.8	10699.9	11579.9	10963.9
(2)	5560.5	10944.4	8925.6	13216.4	18168.5	5543.7	10937.9	8925.4	14229.3	18578.7	8279.1	10901.8	11539.2	10935.3
$W(2)$	<	<	<	<	<	≥	<	≥	<	<	<	<	≥	≥
(3)	5534.3	10856.6	8838.5	13091.2	17502.2	5524.4	10860.8	8958.4	13135.4	17511.1	8068.7	10827.6	11417.2	10839.9
$W(3)$	<	<	<	<	>	≥	<	<	>	>	<	<	≥	≥

(1) vs (2)
<: 10
>: 0
≥: 4

(3) vs (2)
<: 8
>: 3
≥: 3

R. Tyasnurita, E. Özcan and R. John, Learning Heuristic Selection using a Time Delay Neural Network for Open Vehicle Routing, Proc. of the 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 1474-1481.

(1) TDNN-ALHH (objective values and distance)
(2) EXPERT (MCF-AM)
(3)TDNN-ALHH (objective values)

Hardware
Intel(R) Core(TM)i7 Windows 7 Enterprise
(3.40 GHz) with 16 GB RAM

Trial
30independent runs

Time Limit
600 nominal seconds

Summary I

- Hyper-heuristic research originated from a job shop scheduling application and has been rapidly growing since then.
- Generation hyper-heuristics are commonly used in the area
  - Train and test fashion
    - Does the selected subset of training instances is sufficiently representative of the test set?
    - Training is time-consuming (delta/incremental evaluation, surrogate functions)
  - The generated/evolved heuristics might not be easy to interpret, yet they can outperform human designed heuristics

Summary II

- There is empirical evidence that machine learning/analytics/ data science help to improve the hyper-heuristic search process
  - Problem features vs solution/state features
  - Offline versus online learning – Life long learning
- There is still a lack of benchmarks
- Automated design of search methodologies is extremely challenging
  - Addressed in almost complete absence of a mathematical and theoretical understanding

Q&A

Thank you.

Ender Özcan ender.ozcan@nottingham.ac.uk

University of Nottingham, School of Computer Science
Jubilee Campus, Wollaton Road, Nottingham
NG8 1BB, UK
http://www.cs.nott.ac.uk/~pszeo