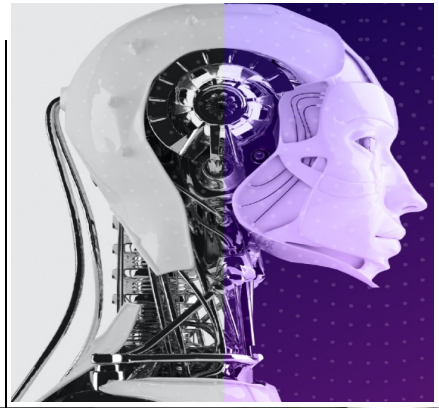
Hyper-heuristics I



Lecture 7

Ender Özcan

Computational Optimisation & Learning Lab

The University of Nottingham

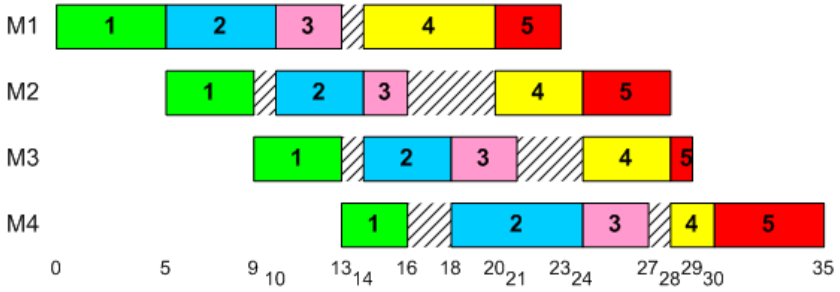
UNITED KINGDOM · CHINA · MALAYSIA

State-of-the-art in (Meta)Heuristic Optimisation

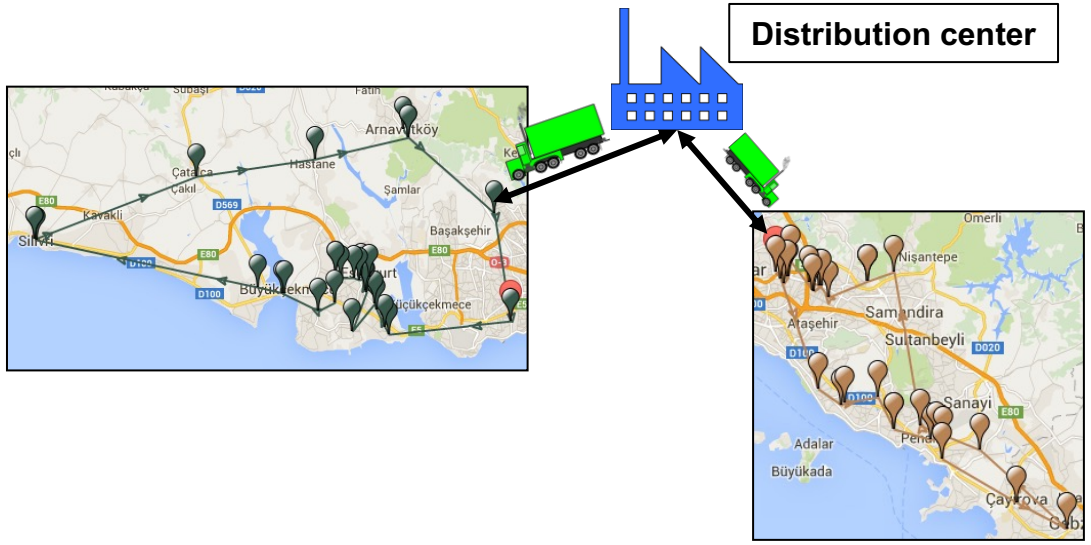
Trial and Error:

- Design and implement algorithmic components
- Configure the algorithm and tune the parameters: Test on selected instances (revisit the design options)
- Performance analyses (revisit the design options)

Flowshop Scheduling



Vehicle Routing



Nurse Rostering

John

03	04	05	06	07	08	09	10	11	12	13	14	15	16
M	T	W	T	F	S	S	M	T	W	T	F	S	S
E	E	N	N	D				D	N	N			

Gem

03	04	05	06	07	08	09	10	11	12	13	14	15	16
M	T	W	T	F	S	S	M	T	W	T	F	S	S
D	D			D	D	D	D	D	D				

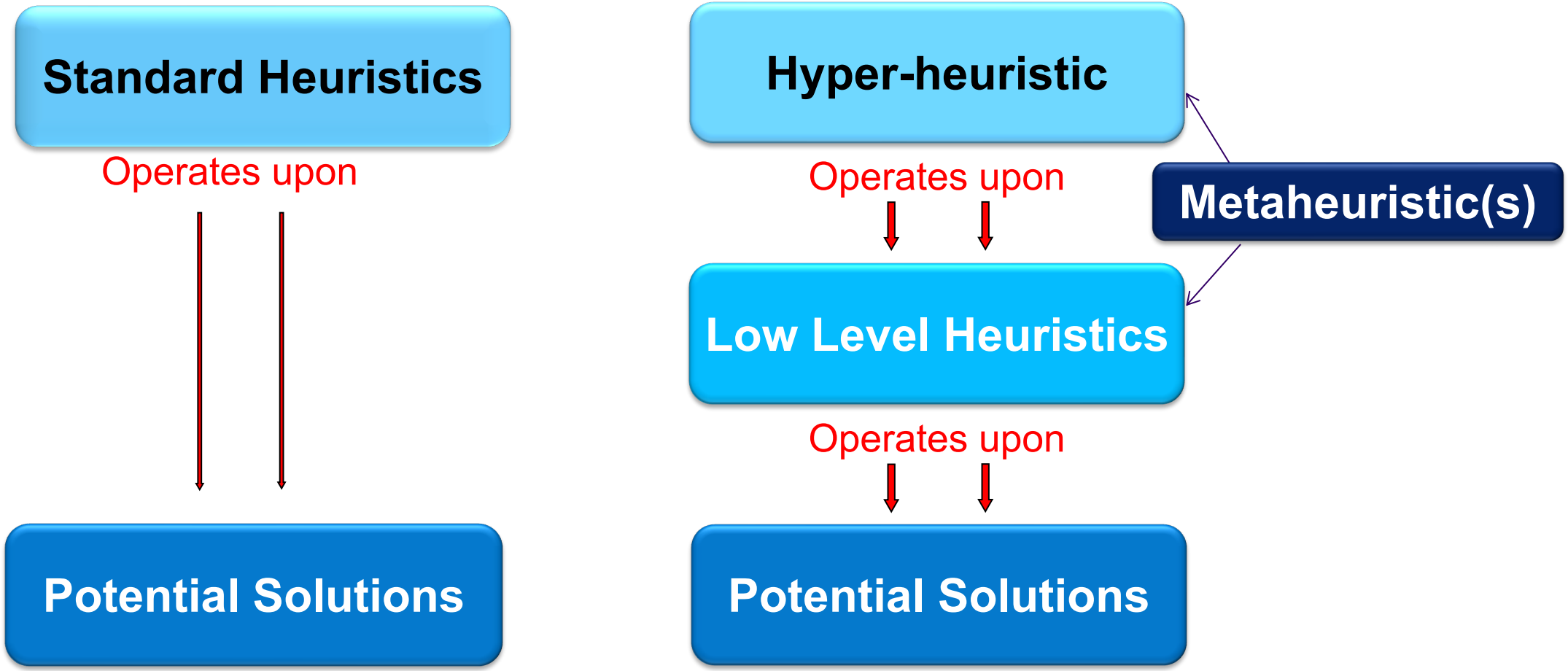
Hyper-heuristics

A hyper-heuristic is a search method or learning mechanism for selecting or generating heuristics to solve computationally difficult problems

- A class of methodologies for cross-domain search

- J. H. Drake, A. Kheiri, E. Özcan, and E. K. Burke, Recent Advances in Selection Hyper-heuristics, European Journal of Operational Research, DOI:10.1016/j.ejor.2019.07.073 (invited review) (Open Access - available online [PDF]).
- E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, Hyper-heuristics: A Survey of the State of the Art, Journal of the Operational Research Society, 64 (12) , pp. 1695-1724, 2013. [PDF]

Different Search Spaces



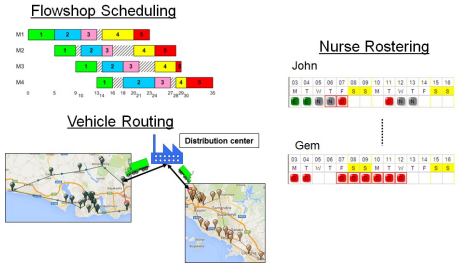
Motivation – Cross Domain Search

- Hyper-heuristic research is motivated by raising the level of generality of search methods . What are the limits?
- **Grand Challenge**

More General

A B C

Problem Specific Solvers



Significant scope for future research

The General Solver

Doesn't exist….

Hyper-heuristics: Origins

1961-63 · 1975 · 1990-95 · 1997 · 2001

Cowling P.I., Kendall G. and Soubeiga E., 2001. A Hyperheuristic Approach to Scheduling a Sales Summit, selected papers from PATAT 2000, Springer, LNCS 2079, 176-190. [PDF]

Fisher H. and Thompson G.L., 1963. Probabilistic Learning Combinations of Local Job-shop Scheduling Rules. Ch 15,:225-251, Prentice Hall, New Jersey.

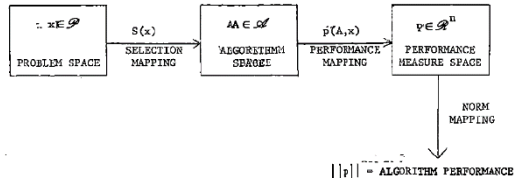
Crowston W.B., Glover F., Thompson G.L. and Trawick J.D. Probabilistic and Parameter Learning Combinations of Local Job Shop Scheduling Rules. ONR Research Memorandum, GSIA,CMU, Pittsburgh, (117), 1963

THE ALGORITHM SELECTION PROBLEM

John R. Rice
Computer Science Department
Purdue University
West Lafayette, Indiana 47907

July 1975

CSD-TR 152



Storer R. H., Wu S. D. , Vaccari R., 1992. New Search Spaces for Sequencing Problems with Application to Job Shop Scheduling, INFORMS, 38(10), 1495-1509.

Fang H.-L., Ross P. and Corne D., 1994. A Promising Hybrid GA/Heuristic Approach for Open-Shop Scheduling Problems., in'ECAI' , 590-594.

Denzinger J., Fuchs M. and Fuchs M., 1997. High performance ATP systems by combining several AI methods. In Proc. of the 15th IJCAI, 102-107.

A Selection Hyper-heuristic Framework for Cross-domain Search

Cowling P.I., Kendall G. and Soubeiga E., 2001. A Hyperheuristic Approach to Scheduling a Sales Summit, selected papers from PATAT 2000, Springer, LNCS 2079, 176-190. [PDF]

- No domain knowledge, other than that embedded in a range of simple knowledge-poor heuristics.
- Robust enough to effectively handle a wide range of problems and problem instances from a variety of domains.

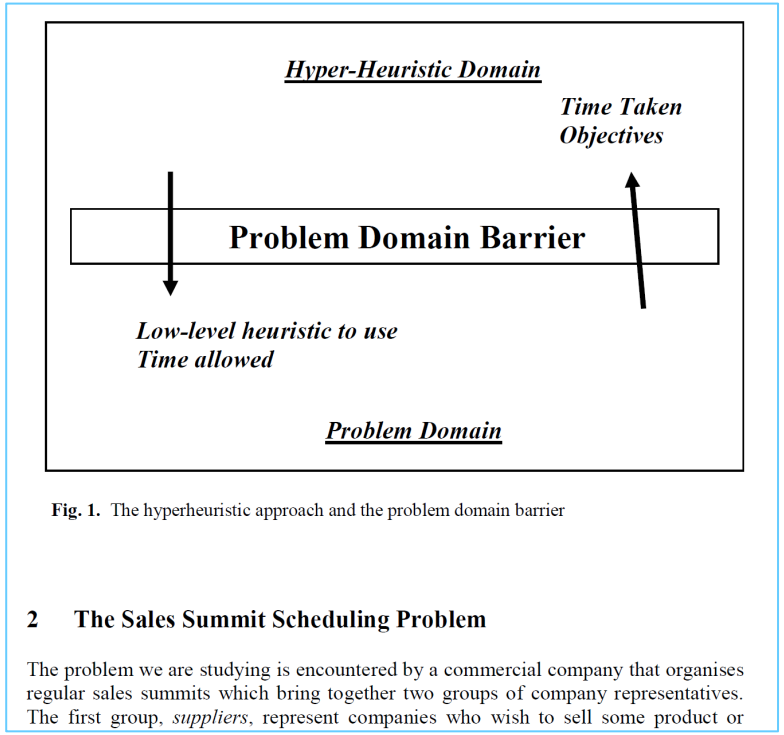


Fig. 1. The hyperheuristic approach and the problem domain barrier

2 The Sales Summit Scheduling Problem

The problem we are studying is encountered by a commercial company that organises regular sales summits which bring together two groups of company representatives. The first group, *suppliers*, represent companies who wish to sell some product or

Characteristics of Hyper-heuristics (initial framework)

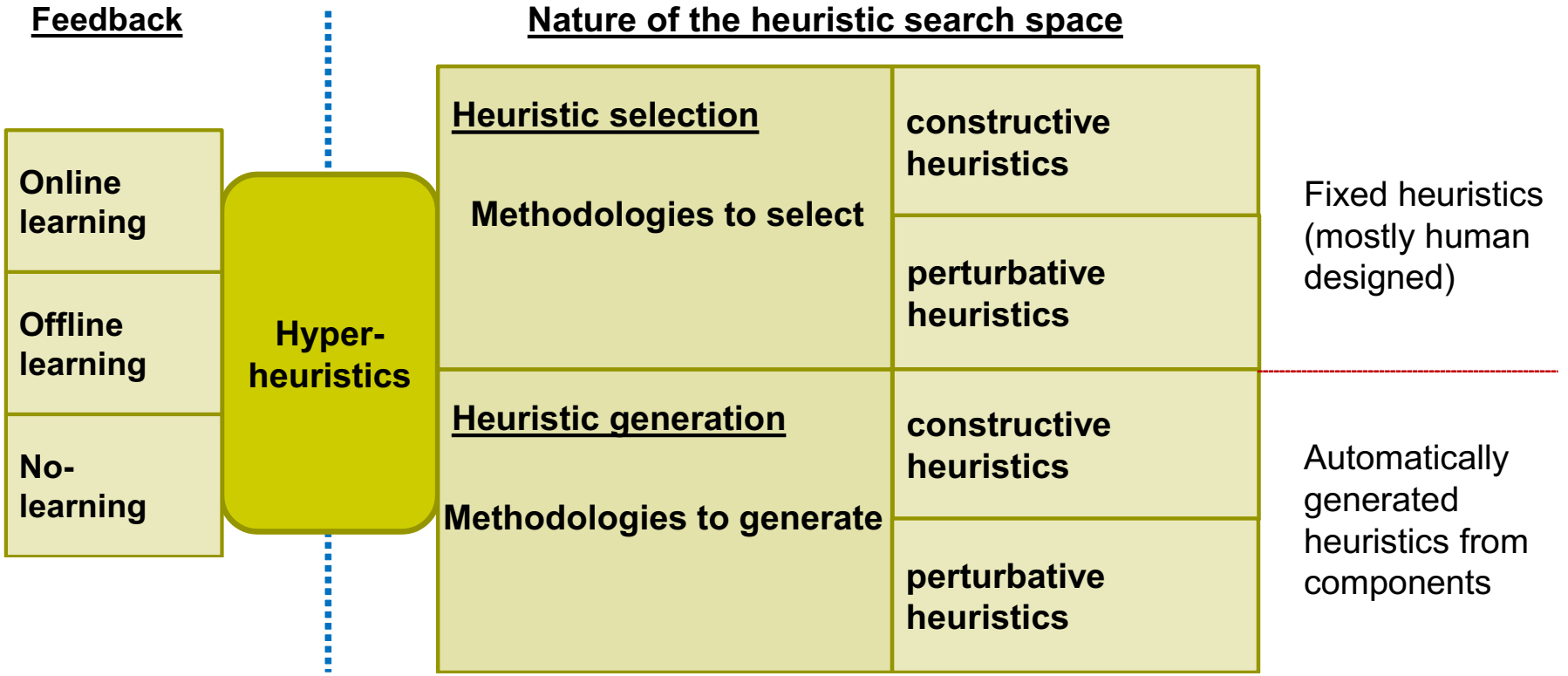
- Operate on a **search space of heuristics (neighbourhood operators)** rather than directly on a search space of solutions
- Aim is to take advantage of strengths and avoid weaknesses of each heuristic (operator)
- No problem specific knowledge is required during the search over the heuristics (operator) space
- Easy to implement, practical to deploy (easy, cheap, fast)
- Existing (or computer generated) heuristics (operators) can be used within hyper-heuristics

Related Areas

- Reactive search
- Algorithm selection/portfolios
- Adaptive operator selection
- Meta-learning
- Co-evolution/multimeme memetic algorithms/Memetic computing
- Variable Neighbourhood Search
- Cooperative (Distributed) Search
- Parameter control (e.g., in EAs)
- Algorithm configuration,…

Classification of Hyper-heuristics

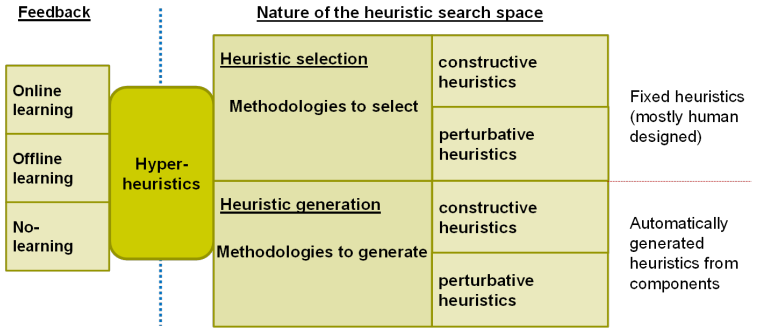
E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan and J. Woodward, A Classification of Hyper-heuristic Approaches: Revisited, In Gendreau, M, and Potvin, JY. (eds.), Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol. 272, pp. 453-477. Springer Cham, 2019. [PDF]

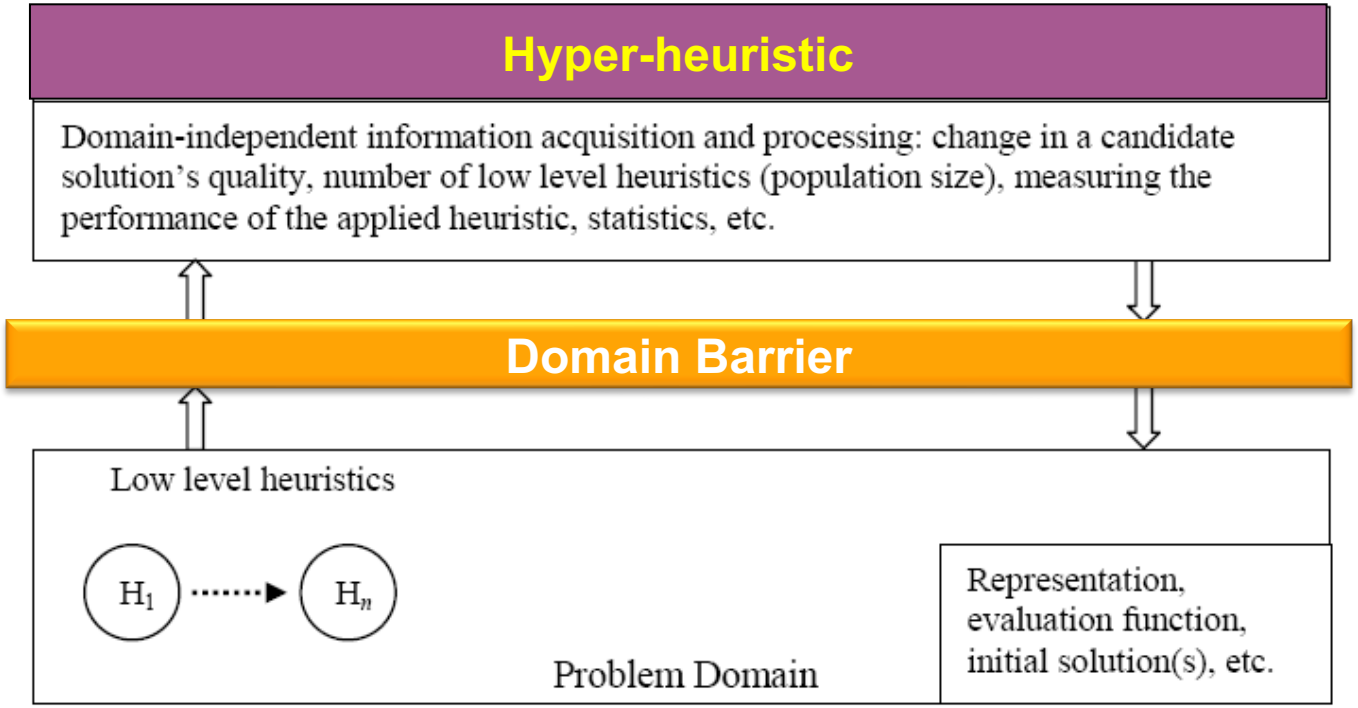


Software Tools for Heuristic Selection/Generation

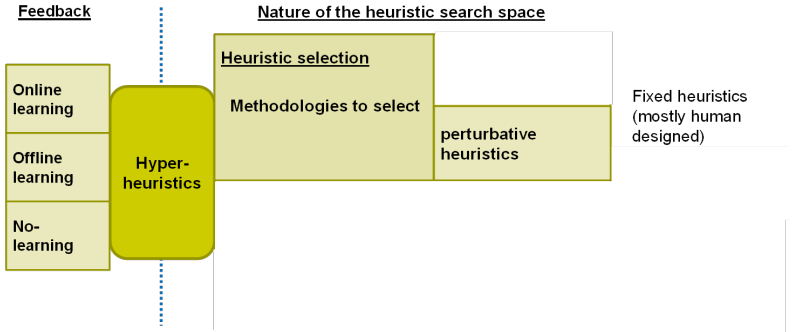
- SATzilla: algorithm portfolio oriented data-driven framework
- Simple Neighborhood-based Algorithm Portfolio in PYthon (snappy) (*old location:* **http://researcher.watson.ibm.com/researcher/files/us-samulowitz/snappy.zip**)
- Hyper-heuristics Flexible Interface (HyFlex): **http://www.asap.cs.nott.ac.uk/external/chesc2011/**
- ParHyFlex extends MPI: **https://tinyurl.com/zrfz2vw**
- Hyperion provides a white-box framework giving a metaheuristic/hyper-heuristic full access to the problem domain: **https://github.com/MitLware/mitl-hyperion**
- ECJ: **http://cs.gmu.edu/~eclab/projects/ecj/**

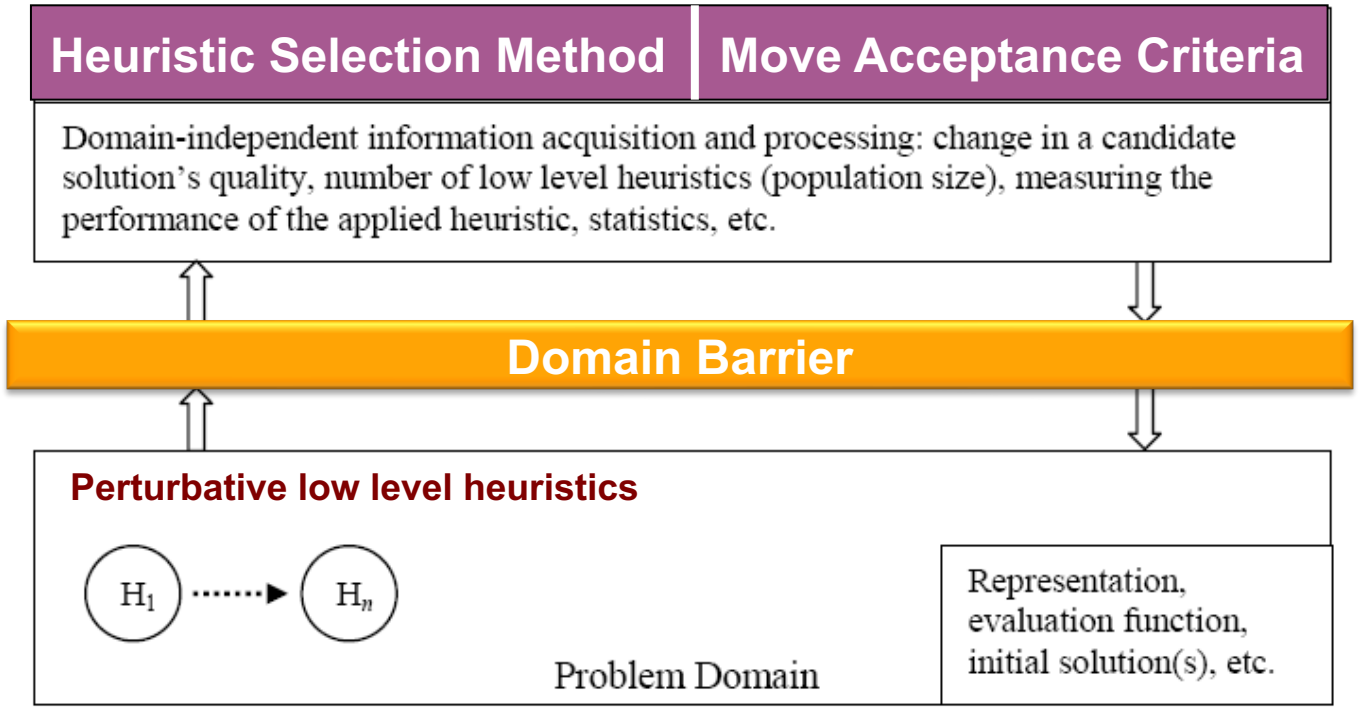
A Hyper-heuristic Framework





A Selection Hyper-heuristic Framework – Single Point Search



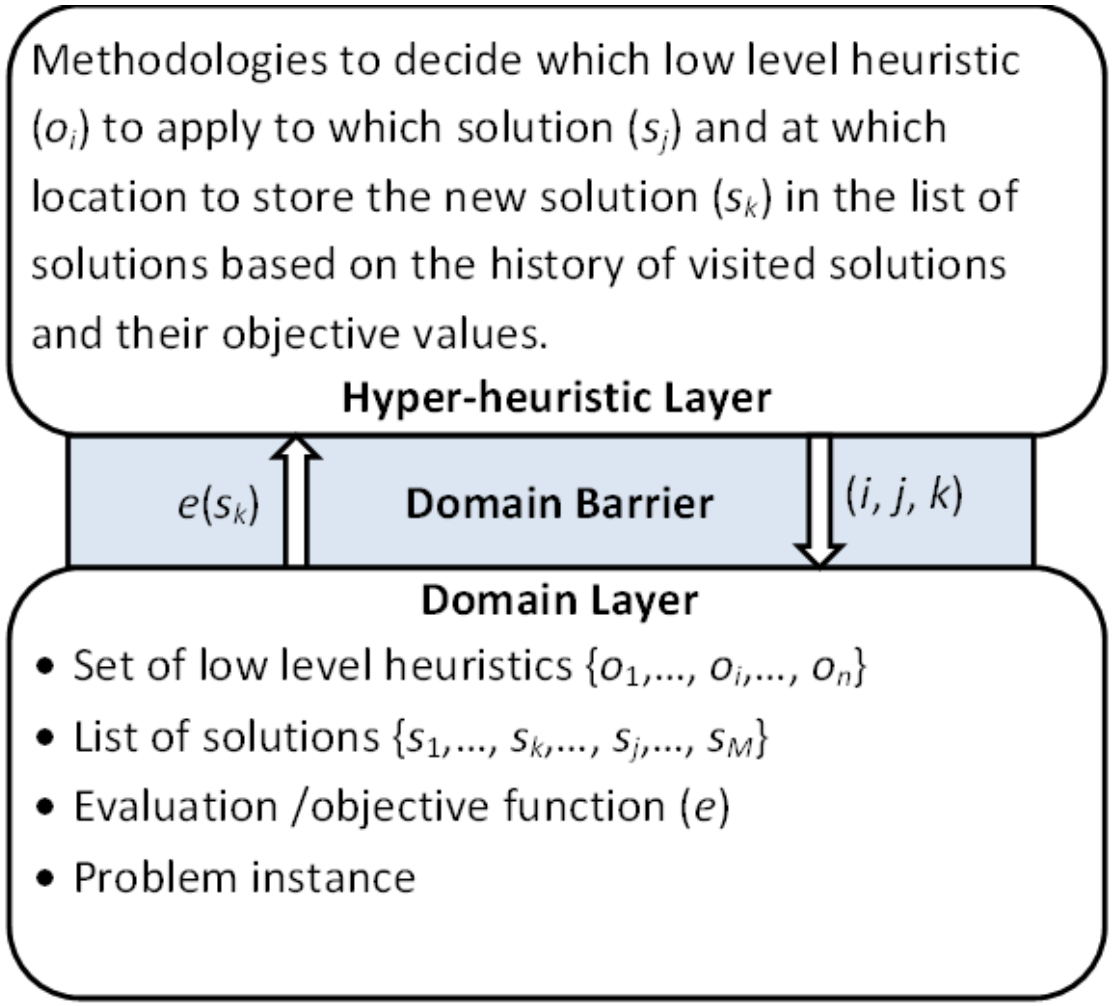


A Selection Hyper-heuristic Framework – Single Point Search

1. generate initial candidate solution *p*
2. while (termination criteria not satisfied){
3. select a heuristic (or subset of heuristics) *h* from {H_1, ..., H_n}
4. generate a new solution (or solutions) *s* by applying *h* to p
5. decide whether to accept *s* or not
6. if (*s* is accepted) then
7. *p*=*s* }
8. return *p*;

Hyper-heuristics Flexible Interface (HyFlex)

http://www.asap.cs.nott.ac.uk/external/chesc2011/ (web archive)

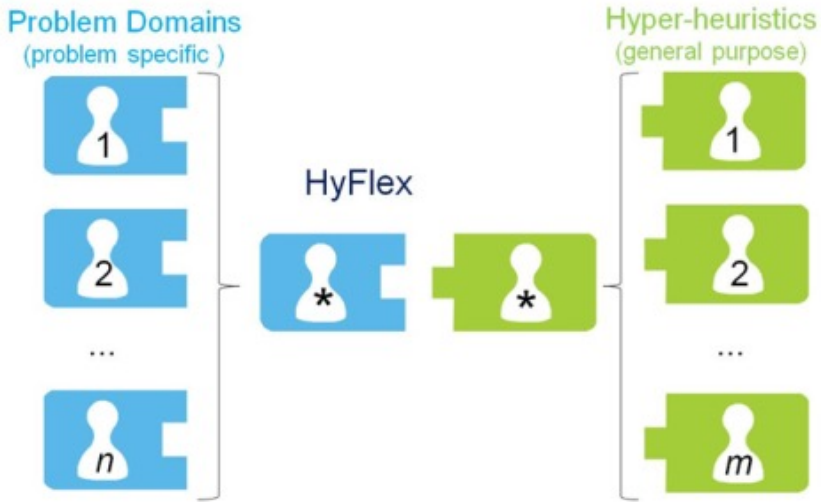


Ochoa G, Hyde M, Curtois T, Vazquez-Rodriguez JA, Walker J, Gendreau M, Kendall G, McCollum B, Parkes AJ, Petrovic S, Burke EK (2012) HyFlex: a benchmark framework for cross-domain heuristic search. In: Evolutionary Computation in Combinatorial Optimization, LNCS 7245, pp 136–147

Hyper-heuristics Flexible Interface (HyFlex)

http://www.asap.cs.nott.ac.uk/external/chesc2011/ (web archive)

- Defines behaviours of components and arranges the interaction between them



Separation between the problem-specific and the general-purpose parts, both of which are reusable and interchangeable through the HyFlex interface

HyFlex v1.0 Java Implementation

- Currently there are 6 problem domain implementations
- heuristic types: mutational, ruin-recreate, local search, crossover
- parameters: intensity, depth of search

MAX-SAT

Bin Packing

Flow Shop

Personnel Scheduling

TSP

VRP

Heuristic IDs	LLH0	LLH1	LLH2	LLH3	LLH4	LLH5	LLH6	LLH7
MAX-SAT	MU_0	MU_1	MU_2	MU_3	MU_4	$\mathbf{MU_5}$	$\mathbf{RC_0}$	HC_0
Bin Packing	MU_0	RC_0	$\mathbf{RC_1}$	MU_1	HC_0	$\mathbf{MU_2}$	$\mathbf{HC_1}$	$\mathbf{XO_0}$
PS	HC_0	HC_1	HC_2	HC_3	$\mathbf{HC_4}$	RC_0	RC_1	$\mathbf{RC_2}$
PFS	MU_0	MU_1	MU_2	MU_3	$\mathbf{MU_4}$	RC_0	$\mathbf{RC_1}$	HC_0
TSP	MU_0	MU_1	MU_2	MU_3	$\mathbf{MU_4}$	$\mathbf{RC_0}$	HC_0	HC_1
VRP	MU_0	MU_1	RC_0	$\mathbf{RC_1}$	HC_0	XO_0	$\mathbf{XO_1}$	$\mathbf{MU_2}$

Heuristic IDs	LLH8	LLH9	LLH10	LLH11	LLH12	LLH13	LLH14
MAX-SAT	$\mathbf{HC_1}$	XO_0	$\mathbf{XO_1}$				
PS	XO_0	XO_1	$\mathbf{XO_2}$	$\mathbf{MU_0}$			
PFS	HC_1	HC_2	$\mathbf{HC_3}$	XO_0	XO_1	XO_2	$\mathbf{XO_3}$
TSP	$\mathbf{HC_2}$	XO_0	XO_1	XO_2	$\mathbf{XO_3}$		
VRP	HC_1	$\mathbf{HC_2}$					

Example Domain: (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

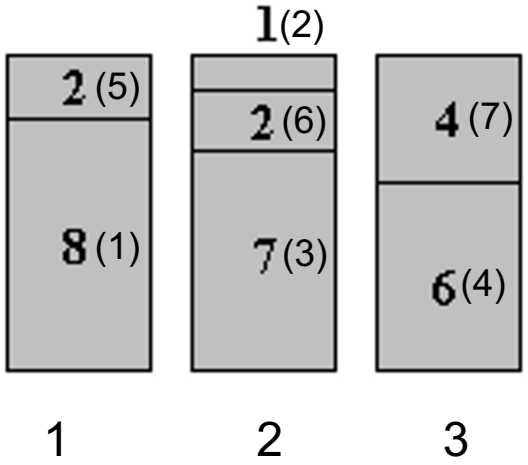
Example Problem Instance

- Given the set of (n=) 7 items

  i: 1 2 3 4 5 6 7

  S = {8, 1, 7, 6, 2, 2, 4}

  and bins of size (C=)10, pack the items into as few bins as possible



i: 1 2 3 4 5 6 7

<1, 2, 2, 3, 1, 2, 3>

Example Domain: Bin Packing
Low Level Mutational Heuristics

Bin Packing MU_0 RC_0 **RC_1** MU_1 HC_0 **MU_2** **HC_1** **XO_0**

- Parameter 'intensity of mutation'
  - [0.2: repeat 1 time] – [1.0: repeat 5 times]
- **Swap** (MU_0): Select two different pieces at random, and swap them if there is space.
- **Split a Bin** (MU_1): This heuristic selects a bin at random from those with more pieces than the average. It then splits this bin into two bins, each containing half of the pieces from the original bin.
- **Repack the Lowest Filled Bin** (MU_2): Remove all of the pieces from the lowest filled bin, and repack them into the other bins if possible, with the best-fit heuristic.

Example Domain: Bin Packing
Low Level Local Search Heuristics

Bin Packing	MU_0	RC_0	$\mathbf{RC_1}$	MU_1	HC_0	$\mathbf{MU_2}$	$\mathbf{HC_1}$	$\mathbf{XO_0}$

- Parameter 'depth of search'
  - 0.2: repeat 10 times – 1.0: repeat 20 times
- **First-improvement/IE – Swap** (HC_0) : Select two different pieces at random, and swap them if there is space, and if it will produce an improvement in fitness.
- **First-improvement/IE – Swap from Lowest Bin** (HC_1): Take the largest piece from the lowest filled bin, and exchange with a smaller piece from a randomly selected bin.

Example Domain: Bin Packing
Low Level Ruin&Recreate Heuristics

- Parameter ‘intensity of mutation’
  - 0.2: x=3, 0.4: x=6, 0.6: x=9, 0.8: x=12, 1.0: x=15
- **Destroy *x* Highest Bins** (RC_0): Remove all the pieces from the x highest filled bins
- **Destroy *x* Lowest Bins** (RC_1): Remove all the pieces from the x lowest filled bins

Crossover

- (XO_0): Exon Shuffling Crossover

Philipp Rohlfshagen and John Bullinaria. A genetic algorithm with exon shuffling crossover for hard bin packing problems. In Proceedings of the 9th annual conference on Genetic and evolutionary computation (GECCO’07), pages 1365–1371, London, U.K., 2007.

asap research
automated scheduling optimisation & planning

The University of Nottingham

CHeSC 2011 benchmark based on HyFlex v1.0

- MAX-SAT
- Bin Packing
- Flow Shop
- Personnel Scheduling
- TSP
- VRP

(TSP, VRP: Hidden)

- 10 public training instances
- **5 test instances (3 training + 2 hidden/all hidden)**

- Set problem instance
- Set time limit (10 min.)
- Perform 31 runs
- Report median

Ranking: Formula 1 scoring system

Rank	Score
1	10
2	8
3	6
4	5
5	4
6	3
7	2
8	1
rest	0

Organising Partners:

Queen's University Belfast

EPSRC Engineering and Physical Sciences Research Council

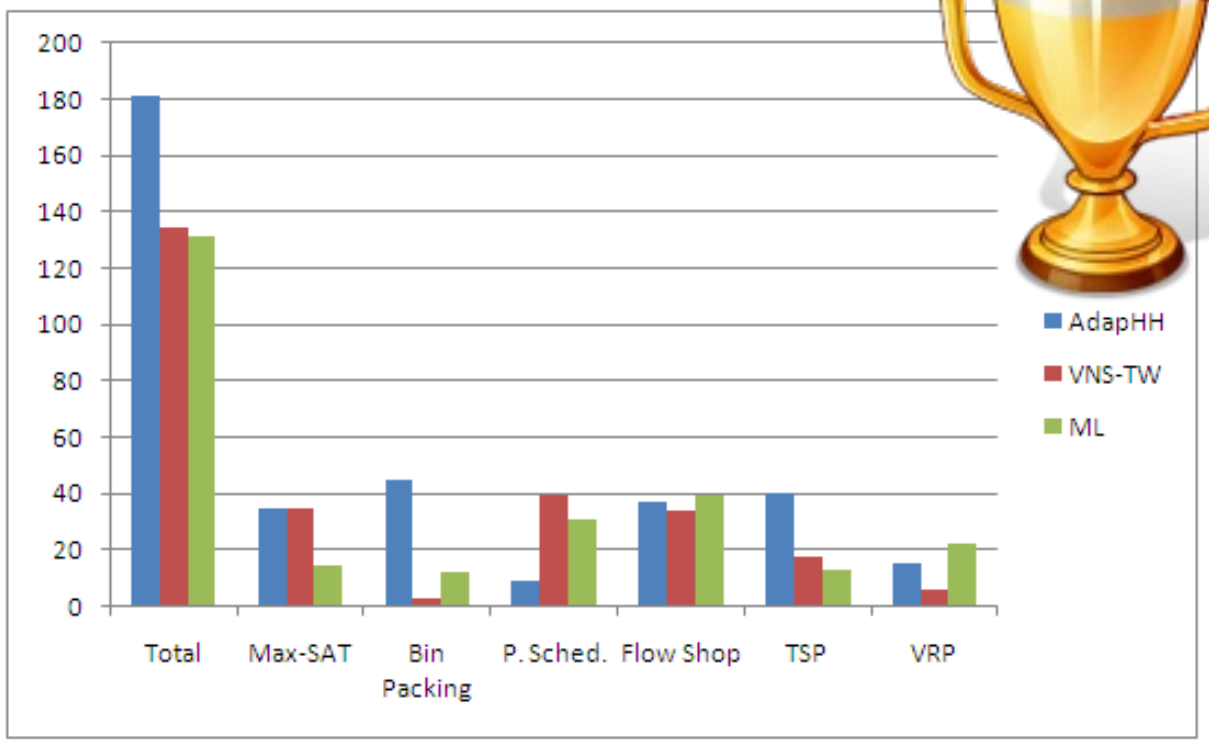
Sponsor:

ÉCOLE POLYTECHNIQUE MONTRÉAL

EVENTMAP

CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

And the winner is...



AdapHH – M. Mısır
K. Verbeeck
P. De Causmaecker
G. Vanden Berghe

AdapHH – Overview

$$p_i = w_1\Big[\Big(C_{p,best}(i)+1\Big)^2\Big(t_{remain}/t_{p,spent}(i)\Big)\Big]\times b + w_2\Big(f_{p,imp}(i)/t_{p,spent}(i)\Big) - w_3\Big(f_{p,wrs}(i)/t_{p,spent}(i)\Big) + w_4\Big(f_{imp}(i)/t_{spent}(i)\Big) - w_5\Big(f_{wrs}(i)/t_{spent}(i)\Big)$$

$$b = \begin{cases} 1, & \sum_{i=0}^{n} C_{p,best}(i) > 0 \\ 0, & otw. \end{cases}$$

$$avg = \Big\lfloor \Big(\sum_i^n QI_i\Big)/n \Big\rfloor$$

$$pl = ph_{duration}/t_{subset}$$

$$ph_{duration} = t_{total}/ph_{requested}$$

$$exc(i) = t_{perMove}(i)/t_{perMove}(fastest)$$

$$\sigma > 2.0 \ ; \ exc(i) > 2\varpi \ ; \ nb > 1$$

$$pr_i = \Big((C_{best}(i)+1)/t_{spent}\Big)^{(1+3tf^3)}$$

$$k = \begin{cases} ((l-1).k + iter_{elapsed})/l, & \text{if } cw = 0 \\ ((l-1).k + \sum_{i=0}^{cw} k.0.5^i.tf)/l, & \text{otherwise} \end{cases}$$

$$tf = (t_{exec} - t_{elapsed})/t_{exec}$$

$$cw = iter_{elapsed}/k$$

Relay hybridisation

Input: $list_{size} = 10; \gamma \in (0.02, 50); p, p' \in [0:1]$
1 $\gamma = (C_{best,s}+1)/(C_{best,r}+1)$
2 **if** $p \leq (C_{phase}/pl)^{\gamma}$ **then**
3 select LLH using a LA and apply to $S \rightarrow S'$
4 **if** $size(list_i) > 0$ *and* $p' <= 0.25$ **then**
5 select a LLH from $list_i$ and apply to $S' \rightarrow S''$
6 **else**
7 select a LLH and apply to $S' \rightarrow S''$
  end
end

$$l = l_{base} + (l_{initial} - l_{base} + 1)tf^3$$

AILLA move acceptance

Input: $i = 1, K \geq k \geq 0, l > 0$
for i=0 **to** l-1 **do** $best_{list}(i) = f(S_{initial})$
1 **if** $adapt\_iterations \geq K$ **then**
2 **if** $i < l-1$ **then**
3 $i++$
  end
end
4 **if** $f(S') < f(S)$ **then**
5 $S \leftarrow S'$
6 $w\_iterations = 0$
7 **if** $f(S') < f(S_b)$ **then**
8 $i = 1$
9 $S_b \leftarrow S'$
10 $w\_iterations = adapt\_iterations = 0$
11 $best_{list}.remove(last)$
12 $best_{list}.add(0, f(S_b))$
  end
13 **else if** $f(S') = f(S)$ **then**
14 $S \leftarrow S'$
15 **else**
16 $w\_iterations++$
17 $adapt\_iterations++$
18 **if** $w\_iterations \geq k$ *and* $f(S') \leq best_{list}(i)$ **then**
19 $S \leftarrow S'$ and $w\_iterations = 0$
  end
end

CHeSC Results

Rank	Hyper-heuristic	Score	Rank	Hyper-heuristic	Score
1	AdapHH	181.00	11	ACO-HH	39.00
2	VNS-TW	134.00	12	GenHive	36.50
3	ML	131.50	13	DynILS	27.00
4	PHUNTER	93.25	14	SA-ILS	24.25
5	EPH	89.75	15	XCJ	22.50
6	HAHA	75.75	16	AVEG-Nep	21.00
7	NAHH	75.00	17	GISS	16.75
8	ISEA	71.00	18	SelfSearch	7.00
9	KSATS-HH	66.50	19	MCHH-S	4.75
10	HAEA	53.50	20	Ant-Q	0.00

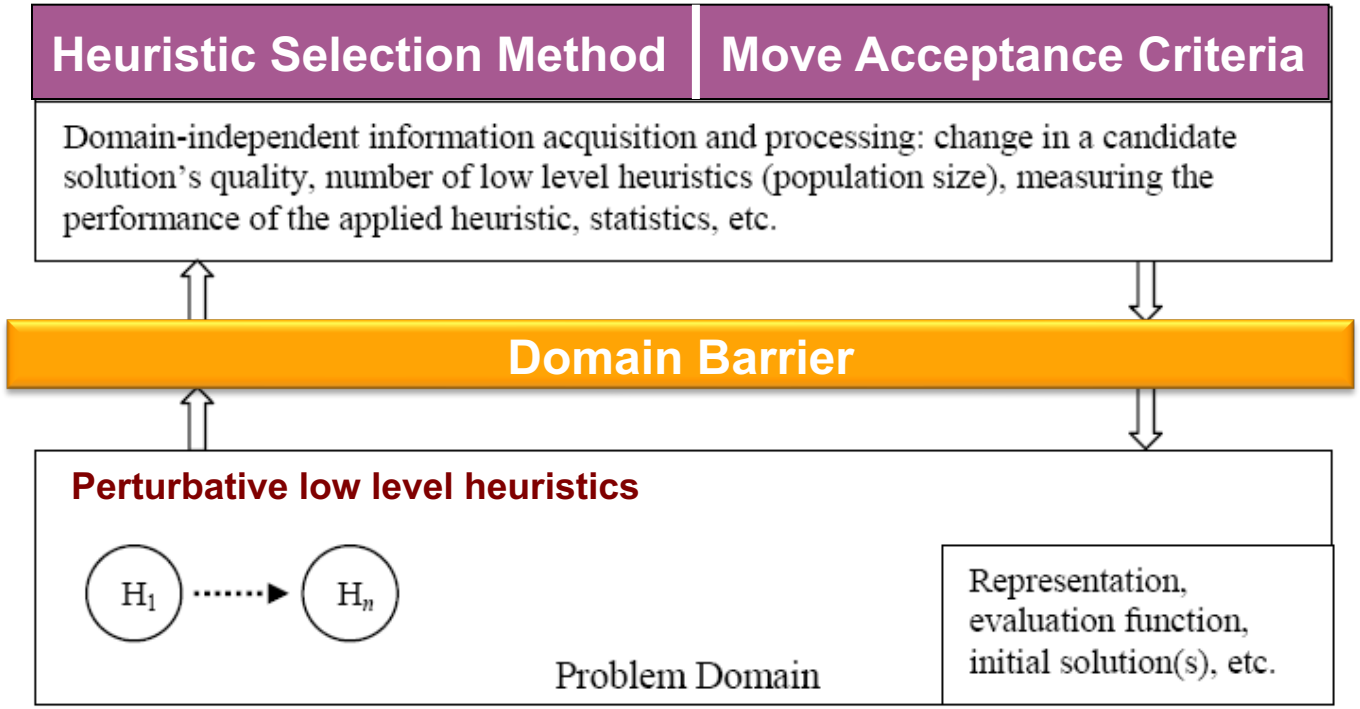
Extended HyFlex Problem Domains

- There are 3 new problem domains, each with 10 instances added to HyFlex benchmark

problem domain	abbrev.	init	ls	mut	rr	xo	total
Max-SAT	SAT	1	2	4	1	2	10
Bin Packing	BP	1	2	3	2	1	9
Permutation Flow Shop	PFS	1	4	5	2	3	15
Personnel Scheduling	PSP	1	4	1	3	3	12
Traveling Salesman	TSP	1	6	5	1	3	16
Vehicle Routing	VRP	1	4	4	2	2	13
0-1 Knap Sack	KP	1	6	5	2	3	17
Quadratic Assignment	QAP	1	2	2	3	2	10
Max-Cut	MAC	1	3	2	3	2	11

* S. Adriaensen, G. Ochoa, and A. Nowe, A benchmark set extension and comparative study for the hyex framework, In IEEE Congress on Evolutionary Computation, 2015, pp. 784-791.

A Selection Hyper-heuristic Framework – Single Point Search (revisited)



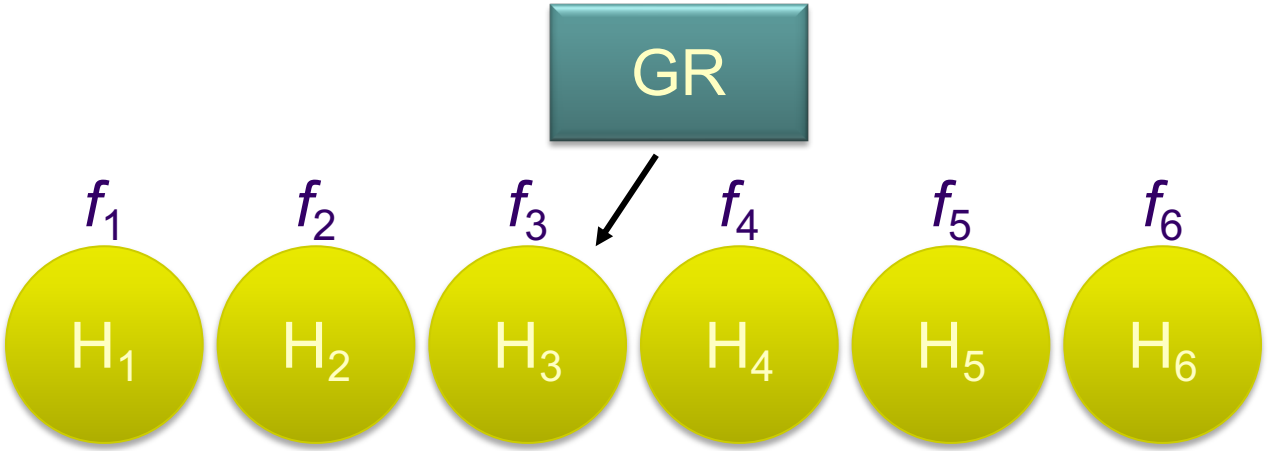
Heuristic (Operator) Selection

with no learning with learning

Component name	Reference(s)
Heuristic selection **with no learning**	
➡ Simple Random	Cowling et al (2000, 2002b)
➡ Random Permutation	Cowling et al (2000, 2002b)
Heuristic selection **with learning**	
Peckish	Cowling and Chakhlevitch (2003)
Greedy	Cowling et al (2000, 2002b); Cowling and Chakhlevitch (2003)
➡ Random Gradient	Cowling et al (2000, 2002b)
➡ Random Permutation Gradient	Cowling et al (2000, 2002b)
Choice Function	Cowling et al (2000, 2002b); Maashi et al (2015); Drake et al (2015)
Reinforcement Learning	Nareyek (2003); Pisinger and Ropke (2007)
Reinforcement Learning with Tabu Search	Burke et al (2003); Dowsland et al (2007)
Quality Index and Tabu based Learning Heuristic Selection	Mısır et al (2009, 2012)
Dominance-based Selection	Kheiri and Özcan (2011; 2015)
Probability-based Selection	Lehrbaum and Musliu (2012)
Adaptive pursuit	Walker et al (2012)

Heuristic Selection – Greedy (GR)

- Apply each low level heuristic to the candidate solution and choose the one that generates the best objective value



$f_3 < f_1, f_2, f_4, f_5, f_6$ at step n

Heuristic Selection – Reinforcement Learning (RL)

- A machine learning technique
- Inspired by related psychological theory
  - Reward and punishment
- Concerned with how an agent ought to take actions in an environment to maximize some notion of long-term reward
- Maintains a score for each heuristic
  - If an improving move then increase (e.g., +1), otherwise decrease (e.g., -1) the score of the heuristic

RL – Example Iteration

- There are 2 low level heuristics in a given domain implementation

utility score for each heuristic

	h_1	h_2
Step i	[3	5]

Choose the heuristic with max score: h_2
Apply to current solution: $h_2(s) \rightarrow s'$

Update scores for the next step:
If s' is s better than (or equal) s (improvement)
then give reward to h_2 ↑score (+1)
Otherwise, penalise (worsening) h_2 ↓score (-1)

Assuming h_2 creates a worsening solution,
its score is decreased by 1: (5-1)=4

After the update:

Step i +1	[3	**4**]

Heuristic Selection – Choice Function (CF)

- The choice function maintains a record of the performance of each heuristic. Three criteria are maintained:

  1) Its individual performance

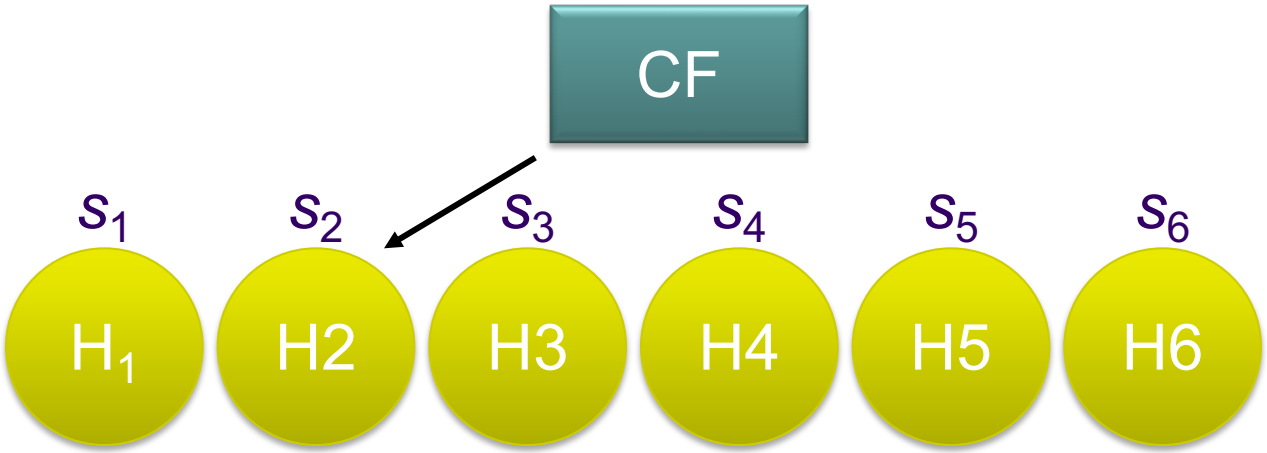
  2) how well it has performed with other heuristics

  3) the elapsed time since the heuristic has been called ($f_3(h_j)$)

$$f_1(h_j) = \sum_n \alpha^{n-1} \frac{I_n(h_j)}{T_n(h_j)} \qquad f_2(h_k, h_j) = \sum_n \beta^{n-1} \frac{I_n(h_k, h_j)}{T_n(h_k, h_j)}$$

$$F_n(h_j) = \alpha_n \, f_1(h_j) + \beta_n \, f_2(h_k, h_j) + \gamma_n \, f_3(h_j)$$

Heuristic Selection – Choice Function (CF)



$s_2 > s_1, s_3, s_4, s_5, s_6$ at step n

Simplified CF – Example Iteration

f_1 and f_2 are performance scores per iteration where α, β, γ are fixed (oversimplified for illustrative purposes)

- There are 2 low level heuristics in a given domain implementation.

Represents the heuristic invoked in previous step

Step i

	h_1	h_2	
	3	5	f_1
h_1	1	5	f_2
h_2	2	3	
	7	4	f_3

Assume that at **Step (i -1), h_2 was invoked** and α=1.0, β=0.5, γ=1.0 are fixed for all steps, so

$F_i(h_1)=f_1(h_1) + 0.5\, f_2(h_2, h_1) + f_3(h_1)= 3+0.5*2+7 = 11$

$F_i(h_2)=f_1(h_2) + 0.5\, f_2(h_2, h_2) + f_3(h_2)= 5+0.5*3+4 = 10.5$

Choose the heuristic with max F_i score: h_1

Apply to current solution: $h_1(s)\rightarrow s'$

Update scores for the next step:

If s' is s better than (or equal) s (improvement) then give reward to $f_1(h_1)$ & $f_2(h_2, h_1)$ ↑scores (+1)

Otherwise, penalise them both by ↓scores (-1)

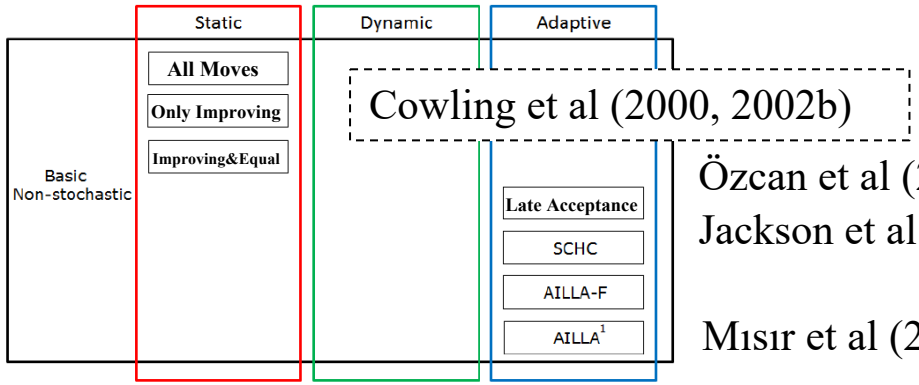
After the updates:

Step i +1

	h_1	h_2	
	4	5	f_1
h_1	1	5	f_2
h_2	**3**	3	
	0	4	f_3

Assuming $\boldsymbol{h_1}$ creates **an improving solution**, $f_1(h_1)$ and $f_2(h_2, h_1)$ scores are increased by 1: (3+1)=**4** and (2+1)=**3**, also since h_1 is used now, $f_3(h_1)$ is reset to **0**.

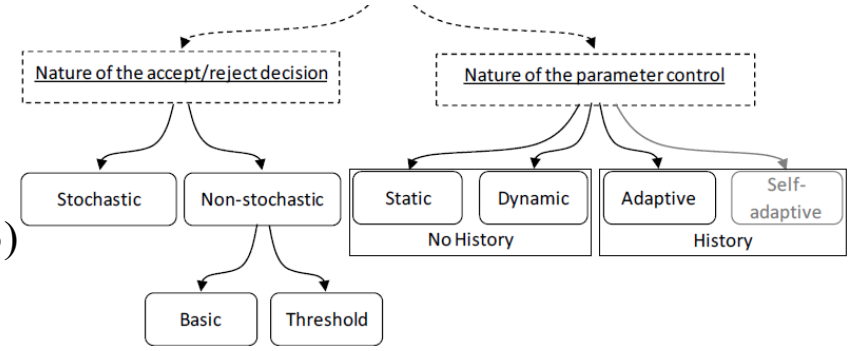
Move Acceptance

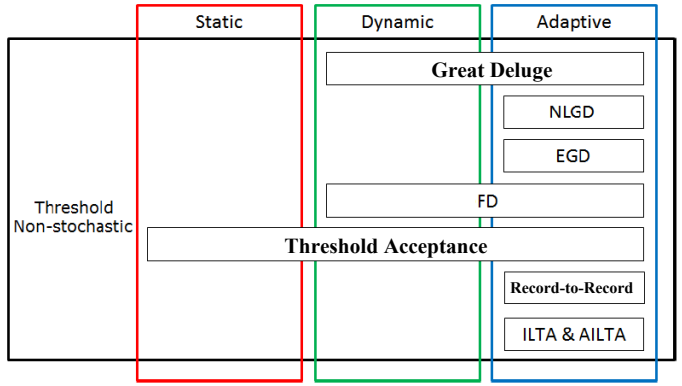


Cowling et al (2000, 2002b)

Özcan et al (2009);
Jackson et al. (2013)

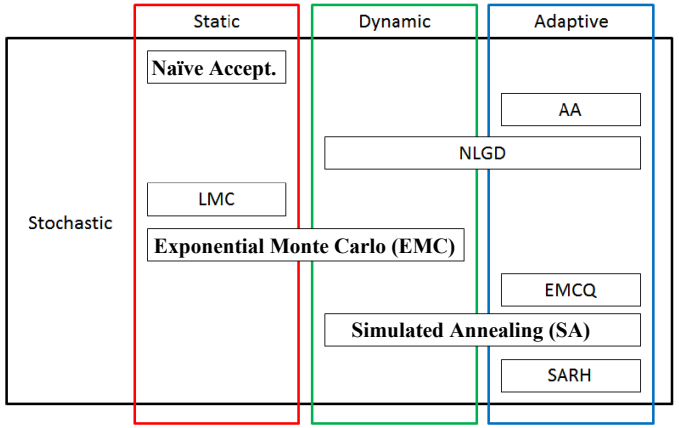
Mısır et al (2012)





Kendall and Mohamad (2004b) || Bilgin et al. (2006)

Kheiri and Özcan (2015)
Kendall and Mohamad (2004b)
Mısır et al (2009)



Burke et al (2010); Kheiri and Özcan (2012); Asta and Özcan (2015)

Ayob and Kendall (2003)

Bai and Kendall (2005); Bilgin et al (2006); Pisinger and Ropke (2007); Antunes et al (2009)

Dowsland et al (2007); Bai et al. (2007a)

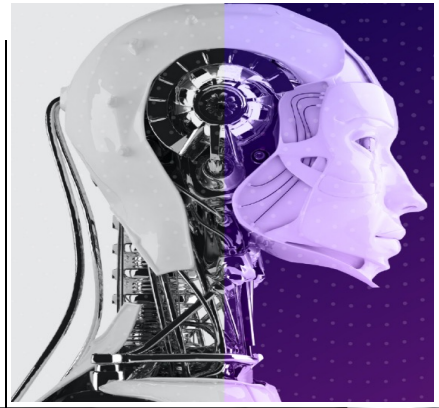
A Few Misconceptions about Hyper-heuristics – Personal View

- Hyper-heuristics do not require parameter tuning
- Hyper-heuristics are all tested under a fair setting (hyflex)
  - Time allocated (and instances used) for tuning
  - Reusable vs disposable heuristics
- Applying a hyper-heuristic to a new domain is *easy*
- Domain specific information should not be passed to the hyper-heuristics (objective value is *not* a domain specific information, *all others* are)

Focus of Recent Studies on Selection Hyper-heuristics

1) *Existing hyper-heuristics/metaheuristics for cross-domain search are tuned/configured for performance improvement*
2) Existing hyper-heuristics are applied to new problem domains
3) "New" hyper-heuristics are created and tested
   - Hyper^2-heuristics (based on metaheuristics and data science techniques) are emerging
     - *Multi-stage approaches (iterated)*
     - Generated hyper-heuristics
   - Multi-objective hyper-heuristics
4) Hybrid approaches are appearing (e.g., exact&inexact, contructive&perturbative, etc…)

An Iterated Multi-stage Selection Hyper-heuristic



A. Kheiri and E. Özcan, An Iterated Multi-stage Selection Hyper-heuristic, European Journal of Operational Research, (250)1:77–90, 2016

ASAP research group

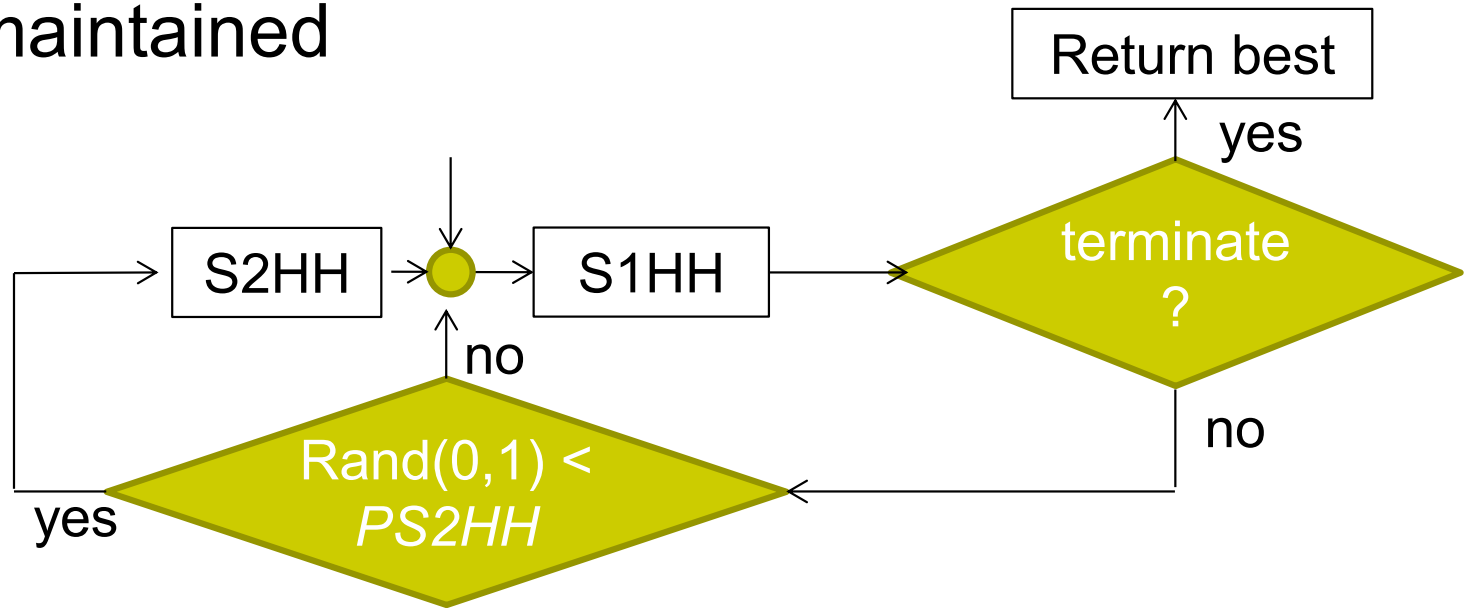
Computational Optimisation & Learning Lab

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

General Framework

- Crossover operators are ignored
- The parameter value of a low level heuristic is randomly reset if there is no improvement after its application, otherwise the same value is maintained



Stage 1 Hyper-heuristic

- Select a low level heuristic i with probability

$$score_i / \sum_{\forall k}(score_k)$$

- Apply the chosen heuristic
- Accept/reject based on an adaptive threshold acceptance method
- Stage 1 terminates if a duration of s_1 is exceeded without any improvement

An Adaptive Threshold Move Acceptance Method

- Moves are accepted using the criteria

if S_{new} $isBetterThan$ $S_{current}$ OR $f(S_{new})$ $isBetterThan$ $(1+\epsilon)f(S_{beststage})$ **then**

$S_{current} \leftarrow S_{new}$

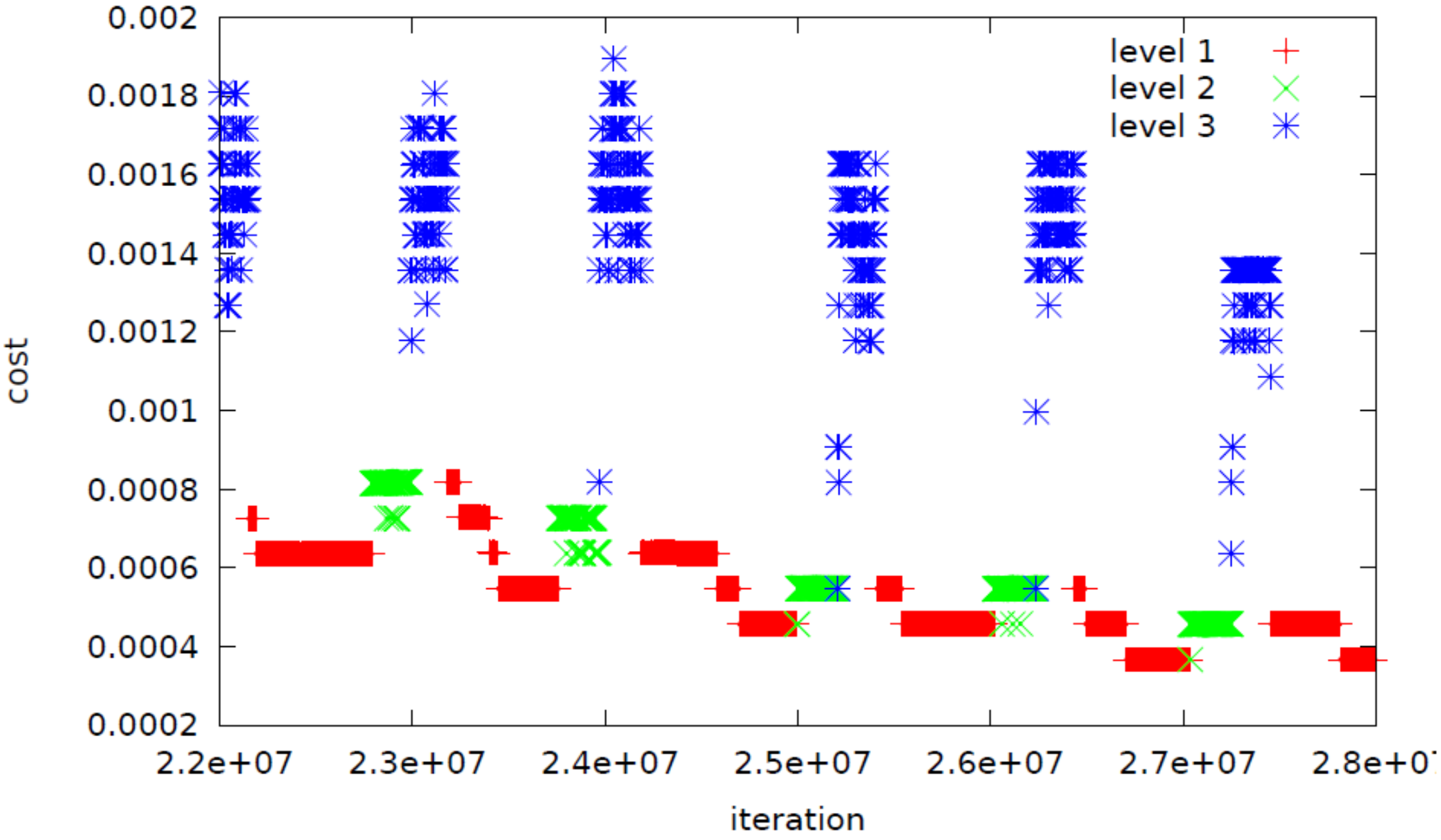
end

- where $\epsilon = \dfrac{\lfloor \log(f(S_{beststage})) \rfloor + c_i}{f(S_{beststage})}$, *C* is a circular list, and

  c_i is an integer value in $C = \{c_0, \ldots, c_i, \ldots, c_{(k-1)}\}$

  updated before/in Stage 2, fixed in Stage 1
- ϵ is updated if no improvement for a duration of *d*

Behaviour of Move Acceptance – Illustration



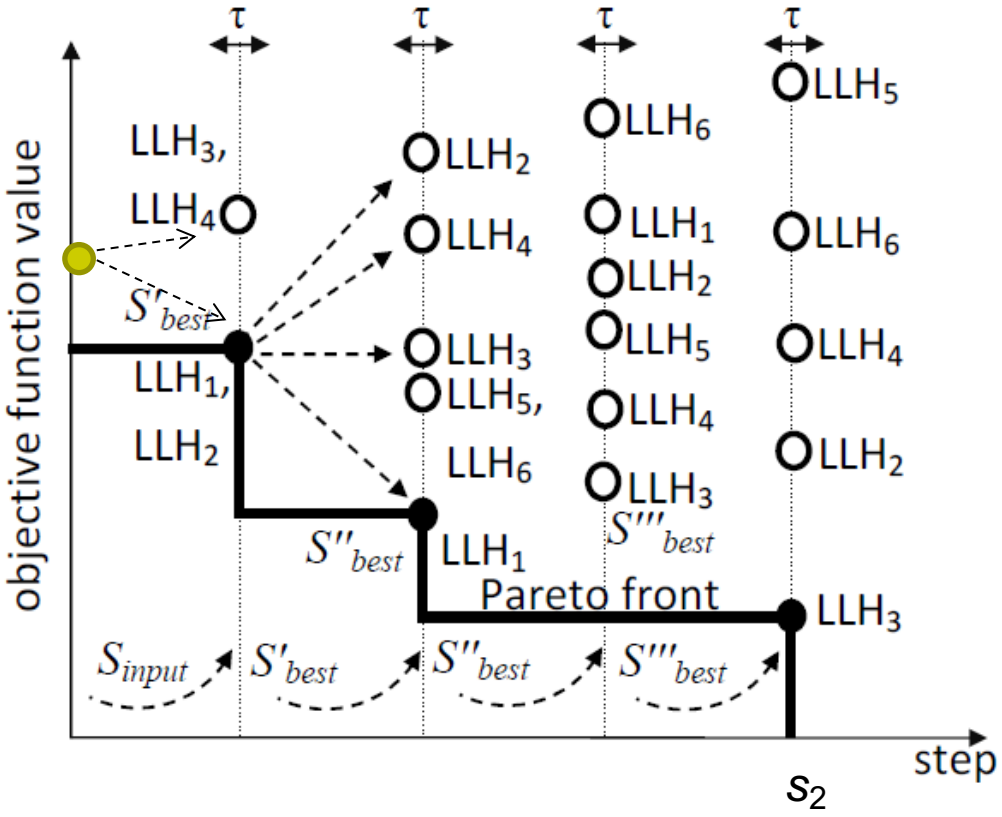
Stage 2 Hyper-heuristic

Given N LLHs, e.g., LLH_1, LLH_2
Pair up all and increase the number of LLHs to $N+N^2$

$LLH_3 \leftarrow LLH_1 + LLH_1$
$LLH_4 \leftarrow LLH_2 + LLH_2$
$LLH_5 \leftarrow LLH_1 + LLH_2$
$LLH_6 \leftarrow LLH_2 + LLH_1$

Reduce the Number of LLHs ($N+N^2 \rightarrow n$) + Assign Probabilities

LLH_1=2, LLH_2=1, LLH_3=1
50% 25% 25%

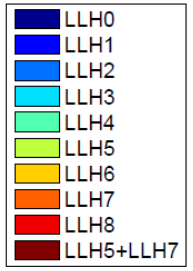


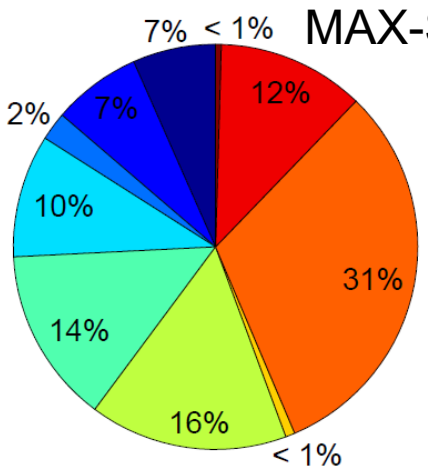
6 LLHs → **3 LLHs**

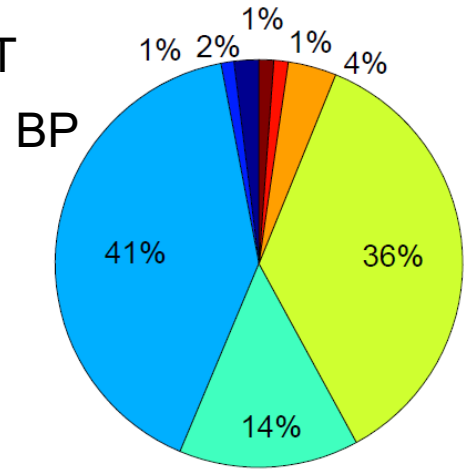
Parameter Tuning

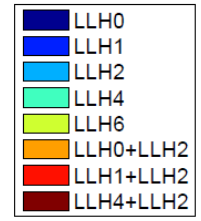
- The proposed approach introduces 6 system parameters to be set
  - $\tau = \{10, \mathbf{15}, 20, 30\}$ (in milliseconds)
  - $d = \{7, \mathbf{9}, 10, 12\}$ (in seconds)
  - $s_1 = \{10, 15, \mathbf{20}, 25\}$ (in seconds)
  - $s_2 = \{3, \mathbf{5}, 10, 15\}$ (in steps/iterations)
  - $P_{S2HH} = \{0.1, \mathbf{0.3}, 0.6, 0.9, 1.0\}$
  - $C = \{\{0\}, \{3\}, \{6\}, \{9\}, \{\mathbf{0, 3, 6, 9}\}\}$

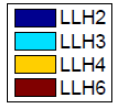
Relay hybridisation

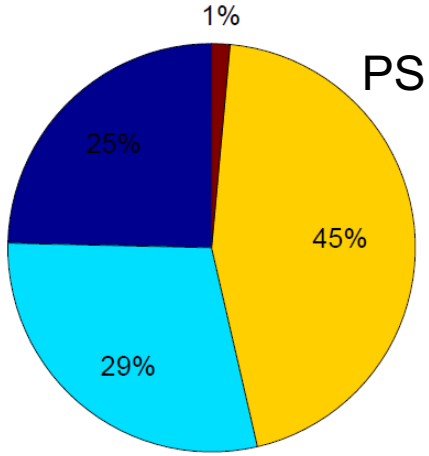


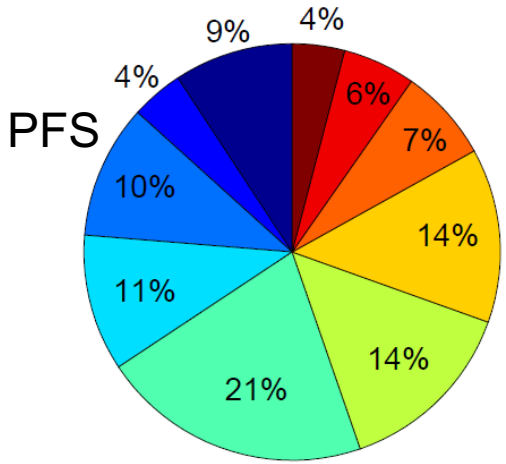


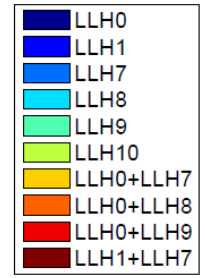


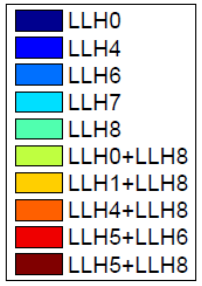


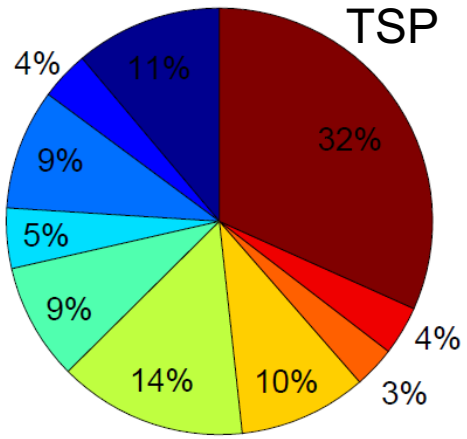


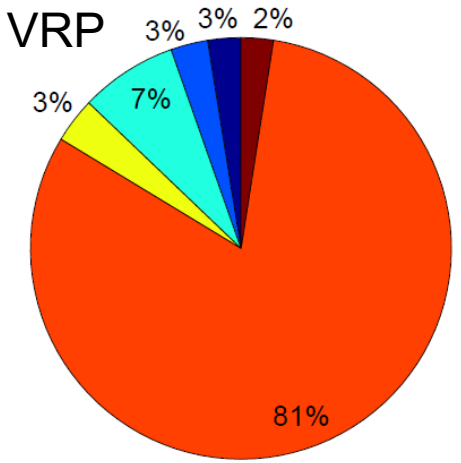


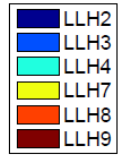




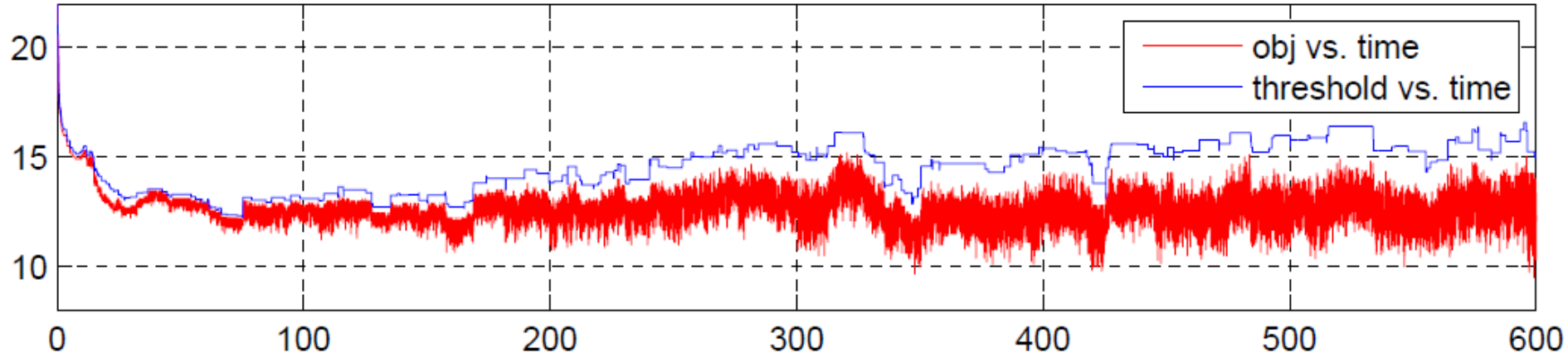




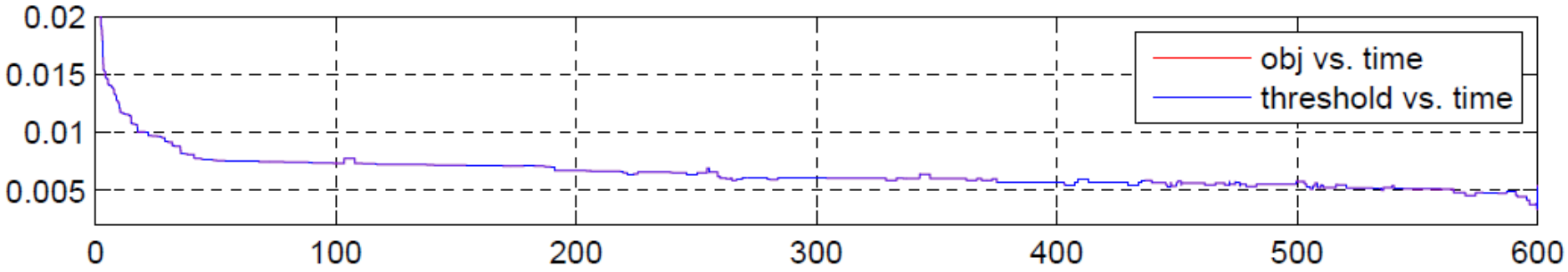




MAX-SAT



Bin Packing

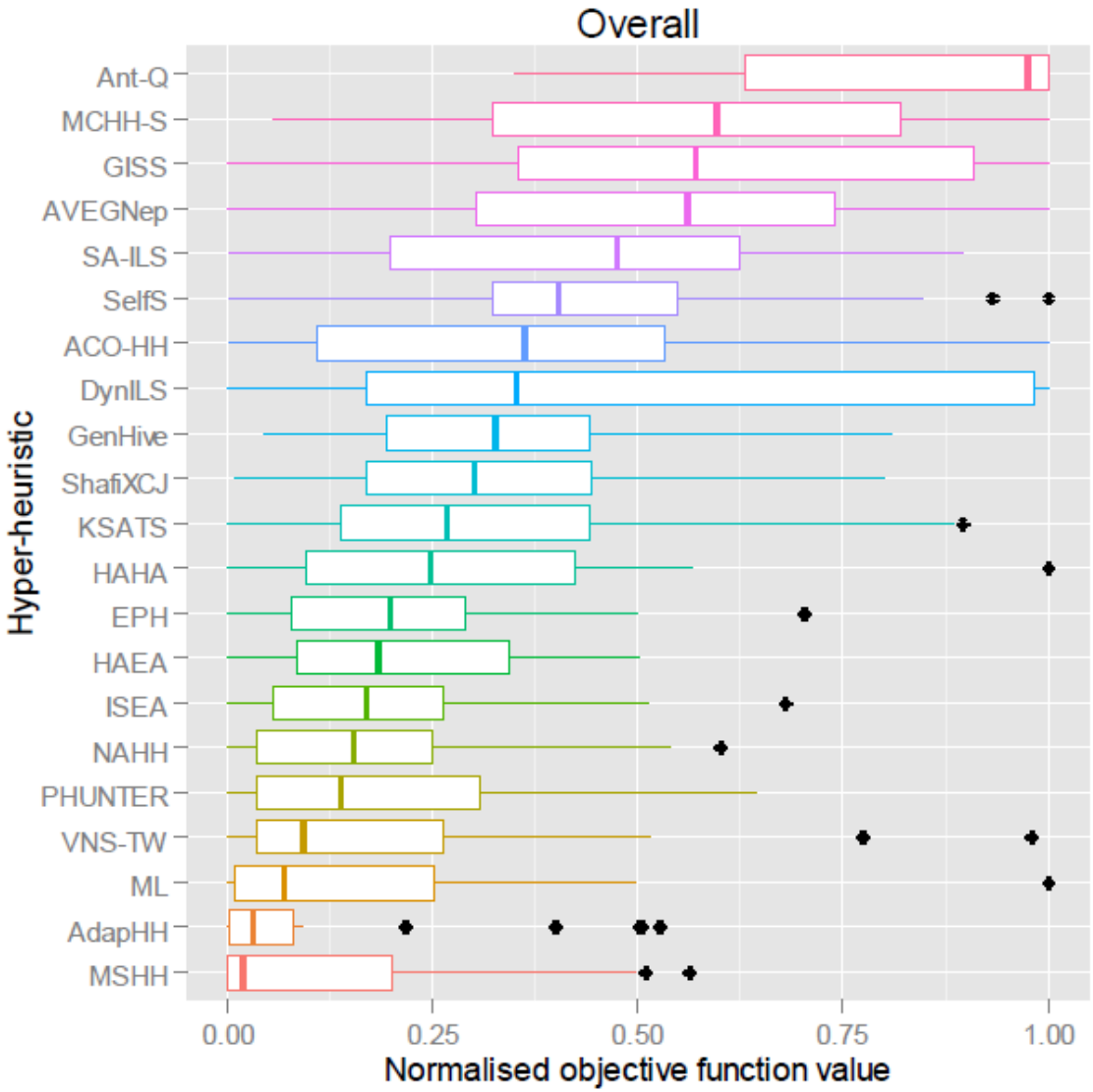

0.02
0.015
0.01
0.005
0
100
200
300
400
500
600
obj vs. time
threshold vs. time

Performance Comparison

		MSHH					S1HH				S2HH		
Domain	Instance	avg.	std.	median	min.	vs.	avg.	std.	min.	vs.	avg.	std.	min.
SAT	Inst1	**0.9**	0.7	1.0	**0.0**	>	6.4	4.5	1.0	>	15.0	4.6	3.0
	Inst2	**3.1**	3.9	2.0	**1.0**	>	21.3	13.3	3.0	>	44.9	9.8	18.0
	Inst3	**0.7**	0.5	1.0	**0.0**	>	7.1	7.7	**0.0**	>	26.3	14.0	1.0
	Inst4	**1.7**	1.0	1.0	**1.0**	>	5.7	4.3	**1.0**	>	20.0	4.6	12.0
	Inst5	**7.6**	0.9	7.0	**7.0**	>	10.4	1.5	**7.0**	>	15.4	1.7	13.0
BP	Inst1	0.0163	0.0014	0.0163	**0.0136**	<	**0.0159**	0.0010	0.0137	>	0.0198	0.0015	0.0160
	Inst2	**0.0037**	0.0015	0.0030	**0.0025**	>	0.0061	0.0015	0.0034	>	0.0104	0.0021	0.0077
	Inst3	**0.0050**	0.0015	0.0049	**0.0025**	≥	0.0054	0.0012	0.0027	>	0.0128	0.0011	0.0104
	Inst4	0.1084	0.0000	0.1084	**0.1083**	≤	**0.1084**	0.0000	0.1083	>	0.1084	0.0000	0.1084
	Inst5	**0.0050**	0.0019	0.0044	**0.0032**	≥	0.0055	0.0021	0.0032	>	0.0210	0.0015	0.0187
PS	Inst1	**25.5**	4.5	25.0	**16.0**	>	28.8	4.7	18.0	>	31.6	4.9	22.0
	Inst2	9668.9	217.8	9638.0	**9184.0**	≤	**9645.3**	159.6	9334.0	≤	9645.8	106.7	9391.0
	Inst3	**3283.7**	93.3	3270.0	**3132.0**	≥	3304.8	99.6	3134.0	≥	3309.9	110.2	3172.0
	Inst4	**1786.3**	172.1	1760.0	1545.0	≥	1801.0	142.3	1570.0	≥	1836.0	291.1	**1400.0**
	Inst5	**353.2**	21.2	350.0	**315.0**	>	724.4	657.3	320.0	>	810.7	621.5	360.0
PFS	Inst1	**6239.8**	14.9	6239.0	**6212.0**	>	6287.6	21.9	6249.0	>	6353.3	29.8	6301.0
	Inst2	26895.2	55.3	26889.0	**26775.0**	<	**26873.2**	30.7	26822.0	>	26976.9	54.7	26849.0
	Inst3	**6333.8**	19.0	6325.0	**6303.0**	>	6360.5	16.4	6323.0	>	6405.5	23.7	6369.0
	Inst4	**11363.8**	32.7	11359.0	**11320.0**	>	11429.9	43.8	11357.0	>	11529.3	35.9	11436.0
	Inst5	26711.9	47.0	26709.0	26630.0	≤	**26693.1**	40.7	**26608.0**	>	26779.1	49.8	26702.0
TSP	Inst1	**48208.1**	31.8	48194.9	**48194.9**	>	50032.0	571.1	49263.1	>	50326.5	606.6	49221.6
	Inst2	**2.09e^{+7}**	9.05e^{+4}	2.09e^{+7}	**2.07e^{+7}**	>	2.14e^{+7}	1.12e^{+5}	2.12e^{+7}	>	2.13e^{+7}	1.05e^{+5}	2.11e^{+7}
	Inst3	**6809.1**	7.1	6808.8	**6796.6**	>	7012.5	30.4	6964.6	>	7040.2	31.3	6988.6
	Inst4	**66840.2**	276.5	66843.6	**66236.8**	>	68908.4	382.4	68159.9	>	70241.9	704.6	68791.0
	Inst5	**53011.4**	469.7	52910.2	**52341.3**	>	54411.1	595.1	53686.0	>	55814.8	946.4	53992.4
VRP	Inst1	70998.4	3840.3	70506.5	**63948.2**	≤	**70223.0**	2960.2	64273.2	>	84103.9	7225.8	68958.3
	Inst2	**13421.8**	251.6	13359.6	**13303.9**	≥	13658.0	471.4	13319.6	>	13695.8	473.9	13320.0
	Inst3	148498.2	1625.8	148436.2	145466.5	≤	**148232.6**	1935.3	145426.5	≥	149553.2	2377.8	**145362.7**
	Inst4	21016.4	488.2	20671.4	**20650.8**	≤	**20991.3**	478.0	20653.5	>	21131.9	510.3	20657.5
	Inst5	**148813.7**	1272.5	149193.7	**146334.6**	≥	148999.1	1217.1	146844.9	>	150282.6	1616.3	146666.9

Performance Comparison to CHeSC 2011 competitors

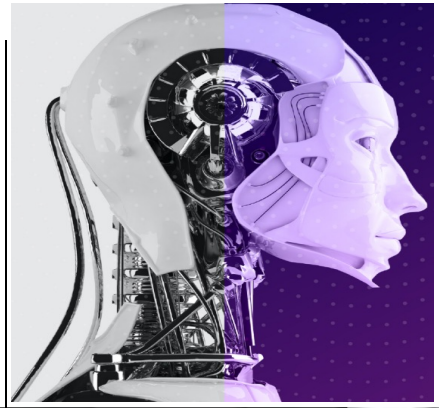
- Top with a CHeSC 2011 score of **163.60**



Conclusions

- The proposed use of two hyper-heuristics with effective components synergise well producing an easy-to-implement, easy-to-maintain, and successful hyper-heuristic for cross domain search
- The adaptive threshold move acceptance captures the trade-off between the extent of improvement that a heuristic can generate and the number of steps it takes to achieve that improvement

Parameter Tuning for Cross-domain Heuristic Search



D. B. Gumus, E. Özcan and J. Atkin, An Investigation of Tuning a Memetic Algorithm for Cross-domain Search, Proc. of the 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 135-142, 2016.

ASAP research group

Computational Optimisation & Learning Lab

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

A Cross-domain Parameter Tuning Study

- A previous study shows that a steady state memetic algorithm (SSMA) performs better than its trans-generational variant across 6 problem domains of HyFlex*
- This study
  - Investigated its performance on 9 problem domains (5 instances from each domain)
  - Performed parameter tuning using the Taguchi design of experiments (orthogonal array) method

*E. Ozcan, S. Asta and C. Altintas, Memetic Algorithms for Cross-domain Heuristic Search, The 13th Annual Workshop on Computational Intelligence (UKCI), 2013, pp. 175-182.

A Steady State Memetic Algorithm

Algorithm 1 : Pseudocode of Steady-state Memetic Algorithm

Create a population of *popsize* random individuals
Set the parameter values for *Intensity of Mutation* (IoM),
Depth of Search (DoS) and *tournament size* (toursize)
Apply a random local search method (hill climbing) to each individual
while (termination criterion is not satisfied)
  Parent1 ← Select-Parent(population, toursize)
  Parent2 ← Select-Parent(population, toursize)
  Child ← ApplyCrossover (Rand(1, MAX_CROSSOVER), Parent1, Parent2)
  Child ← ApplyMutation (Rand(1, MAX_MUTATION), IoM, Child)
  Child ← ApplyLocalSearch(Rand(1, MAX_LOCALSEARCH), DoS, Child)
  WorstOf(population) ← Child
end while

Experimental Design

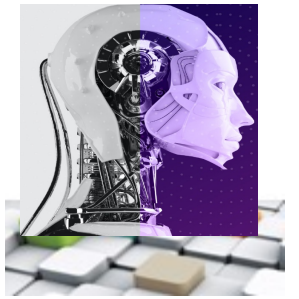
Parameter levels for memetic algorithm

Parameters	Value Options
Intensity of Mutation (IoM)	{0.2, 0.4, 0.6, 0.8, 1.0}
Depth of Search (DoS)	{0.2, 0.4, 0.6, 0.8, 1.0}
Population size (PopSize)	{5, 10, 20, 40, 80}
Tournament size (TourSize)	{2, 3, 4, 5}

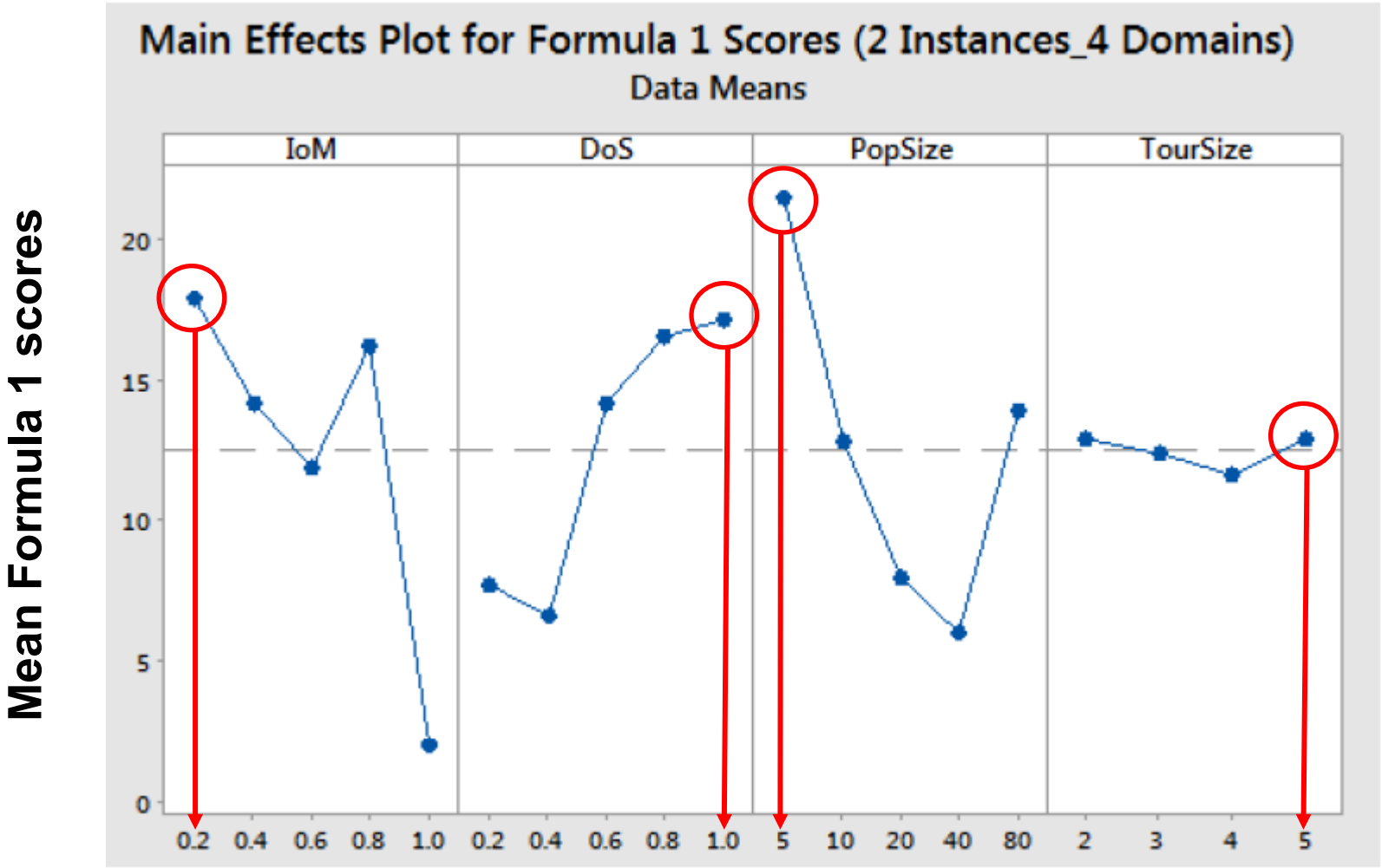
- 2 instances from each of the four public HyFlex problem domains are used for training
- The best configuration after tuning is tested on all instances

Taguchi orthogonal array

Experiment number	IoM	DoS	Pop size	Tour size
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	1	4	4	4
5	1	5	5	2
6	2	1	2	3
7	2	2	3	4
8	2	3	4	1
9	2	4	5	1
10	2	5	1	2
11	3	1	3	4
12	3	2	4	1
13	3	3	5	2
14	3	4	1	3
15	3	5	2	4
16	4	1	4	2
17	4	2	5	3
18	4	3	1	4
19	4	4	2	2
20	4	5	3	1
21	5	1	5	4
22	5	2	1	3
23	5	3	2	1
24	5	4	3	2
25	5	5	4	3

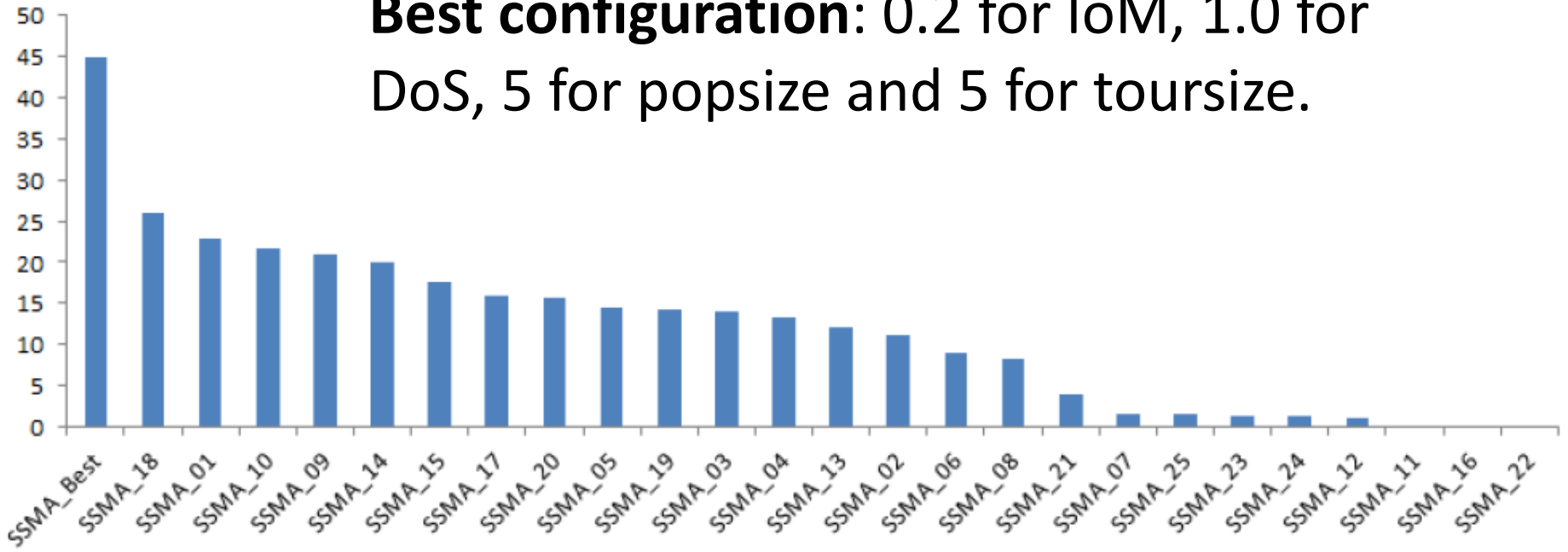


Main Effects Plot – Parameter tuning with 2 instances from 4 public domains

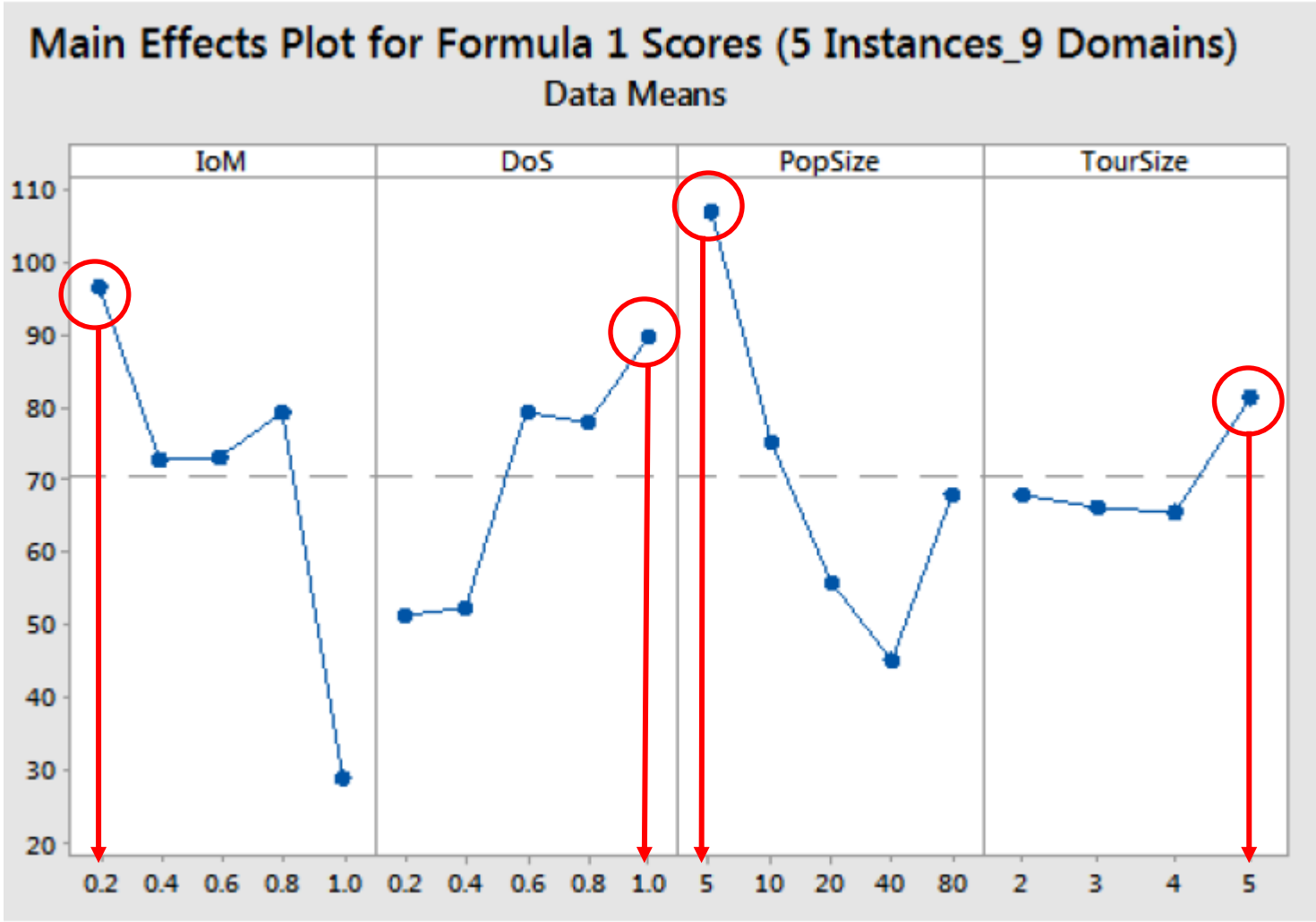


Confirmation Run on the Training Instances

Best configuration: 0.2 for IoM, 1.0 for DoS, 5 for popsize and 5 for toursize.



Main Effects Plot – Parameter tuning with 5 instances from 9 public domains



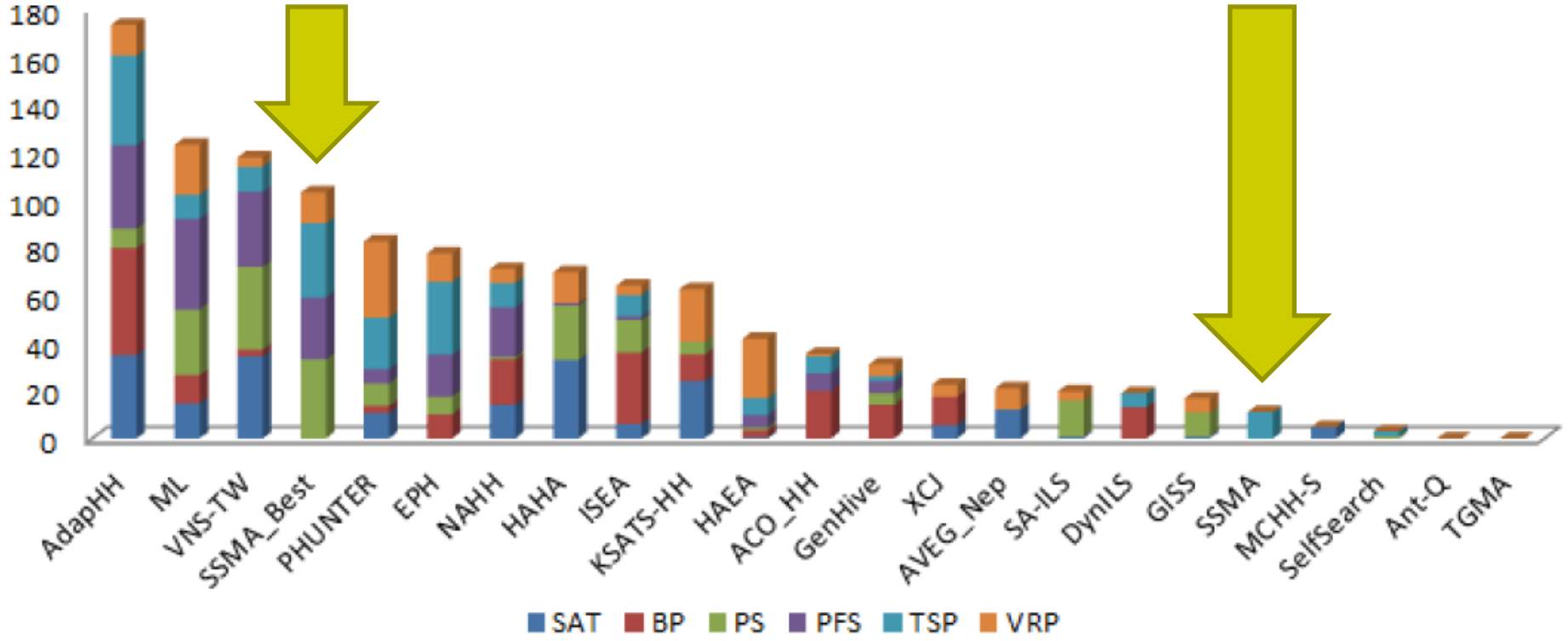
Influence of Parameter Tuning

		SSMA				SSMA-Best		
PD	ID	mean	median	best	vs.	mean	median	best
SAT	1	21.161	22	8	<	**9.484**	**9**	**3**
	2	52.484	53	37	<	**30.065**	**36**	**9**
	3	35	37	10	<	**15.387**	**10**	**3**
	4	27.742	27	26	<	**13.613**	**13**	**9**
	5	19.194	18	14	<	**12.742**	**13**	**9**
BP	1	0.083	0.082	0.074	<	**0.063**	**0.063**	**0.058**
	2	0.015	0.013	0.011	<	**0.011**	**0.012**	0.008
	3	0.022	0.022	0.018	>	0.034	0.034	0.029
	4	0.1115	0.1112	0.1105	<	**0.1102**	**0.11**	**0.1098**
	5	0.043	0.041	0.036	>	0.061	0.06	0.054
PS	1	51.129	50	37	<	**21.548**	**21**	**16**
	2	72015.613	70477	52056	<	**9705.129**	**9677**	**9477**
	3	13203.71	11859	5581	<	**3211.452**	**3219**	**3146**
	4	2942.419	2655	1820	<	**1596.871**	**1589**	**1344**
	5	435.645	431	385	<	**318.065**	**315**	**290**
PFS	1	6257.806	6258	6231	<	**6249.581**	**6251**	**6219**
	2	26884	26884	26813	<	**26812.452**	**26811**	**26754**
	3	6351.871	6363	6318	<	**6336.387**	**6333**	**6303**
	4	11441.806	11441	11410	<	**11376.613**	**11375**	**11333**
	5	26699.226	26703	26626	<	**26632.548**	**26640**	**26515**
TSP	1	48227.747	**48194.92**	**48194.92**	≤	**48221.583**	**48194.92**	**48194.92**
	2	21155458.17	21160875.64	20969185.56	<	**20912006.397**	**20885233.01**	**20789116.98**
	3	6825.552	6825.663	6800.708	<	**6811.145**	**6811.518**	**6799.111**
	4	68123.369	68059.971	67423.655	<	**67029.810**	**67043.255**	**66518.735**
	5	53810.138	53748.537	52685.992	<	**53503.576**	**53457.422**	**52247.568**
VRP	1	**71768.053**	71480.081	67820.589	≤	71946.305	**70776.497**	**65967.938**
	2	14324.522	14411.658	13358.611	<	**13826.295**	**13384.024**	**13328.791**
	3	176206.081	177131.584	167704.512	<	**147512.686**	**148001.434**	**143921.208**
	4	21647.018	21675.195	20678.096	<	**21275.493**	**21648.051**	**20654.219**
	5	152642.04	152829.839	149032.551	<	**147372.783**	**147228.056**	**145266.409**

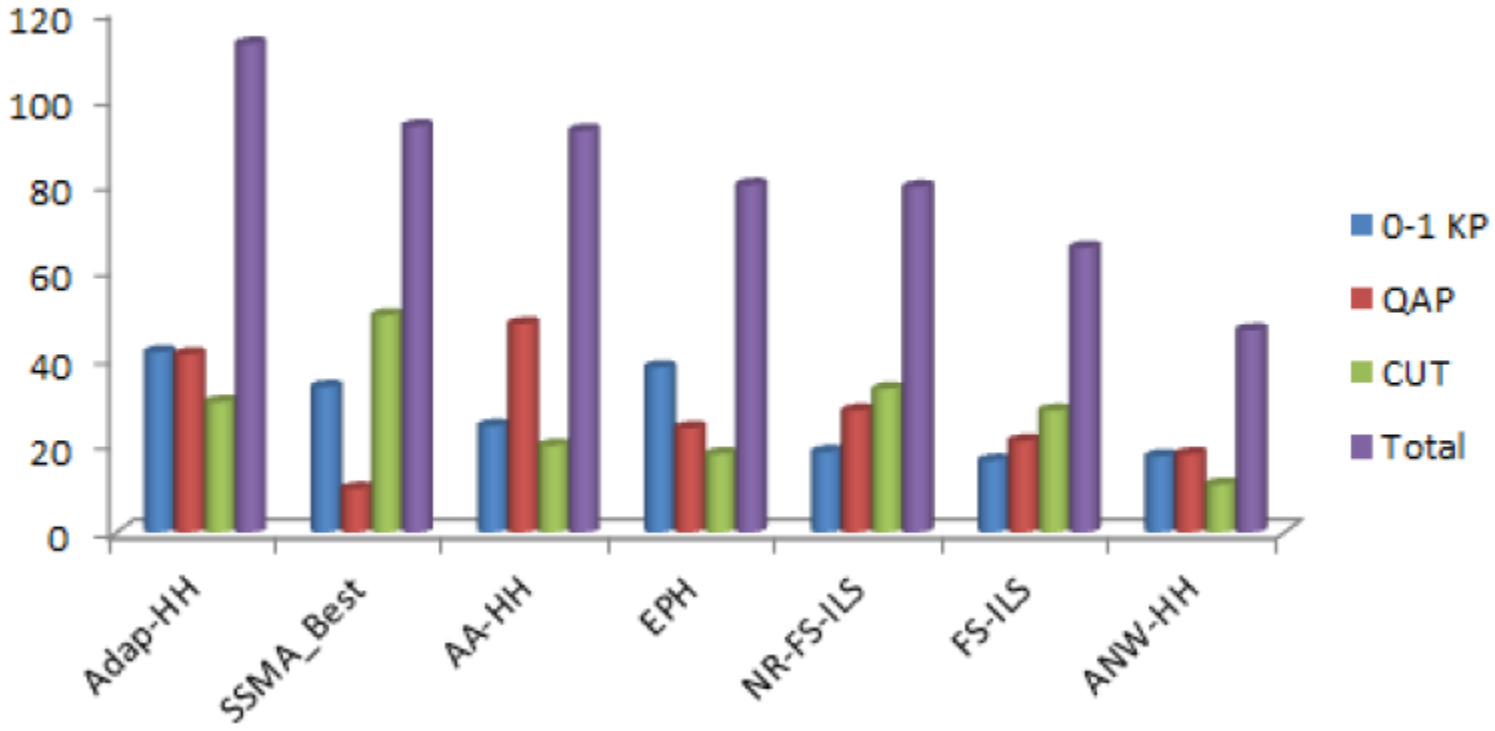
Influence of the Setting {toursize=popsize=5}

- SSMA-Best selects the best individual as both Parent 1 and 2 for crossover.
- After the application of a chosen crossover, the new offspring will be identical to the parents.
- For the HyFlex domains and instances, the crossover operators do not seem to have much of a positive influence on the overall performance of SSMA.
- Hence, SSMA under the best setting actually performs a single point based search rather than multipoint turning the overall algorithm into an approach similar to Iterated Local Search, where mutation is followed by local search at each step.

Performance of SSMA-Best wrt CHeSC 2011 Competitors



Performance of SSMA-Best on the Extended HyFlex Domains



Conclusions

- Parameter tuning has actual value in cross-domain search.
- The tuning experiments indicate the success of the SSMA-Best, which, significantly, outperforms the two memetic algorithm variants and even some of the competing search methods in CHeSC2011.
- The key observations from the empirical results are that crossover should not be used in this case and that single point based search should be preferred.

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://cs.nott.ac.uk/~pszeo