Hyper-heuristics: Towards Automated Heuristic Design

Gabriela Ochoa, Matthew Hyde, Edmund K. Burke

Automated Scheduling, Optimisation and Planning (ASAP) Group, School of Computer Science, The University of Nottingham

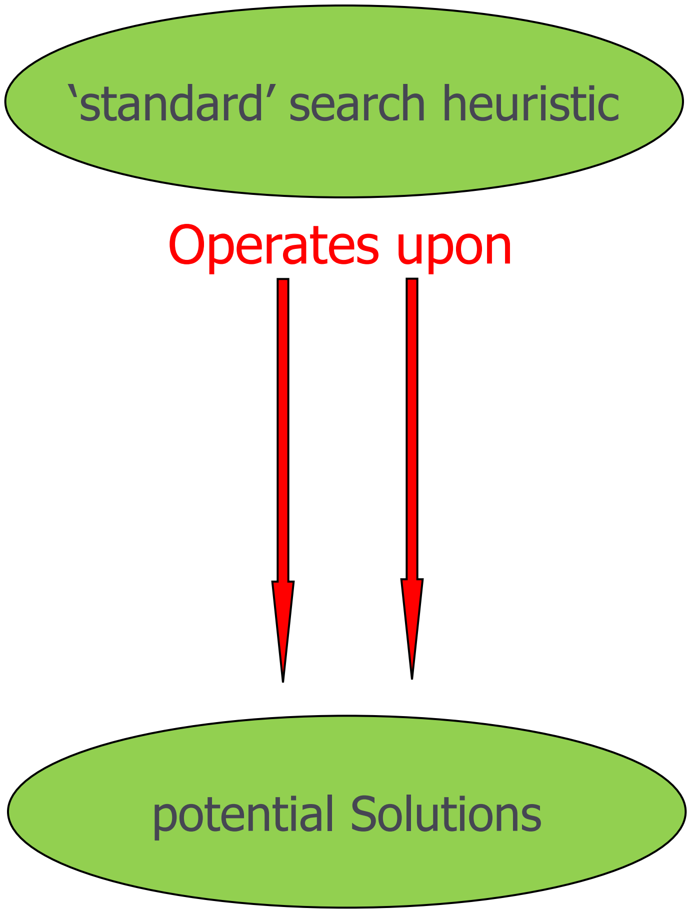
Content

- What is a hyper-heuristic?
- What motivates hyper-heuristic research?
- Origins and related areas
- Classification of hyper-heuristic approaches
  - Heuristic *selection* methodologies
  - Heuristic *generation* methodologies
- The 'Cross-domain Heuristic Search Challenge'

Content

- What is a hyper-heuristic?
- What motivates hyper-heuristic research?
- Origins and related areas
- Classification of hyper-heuristic approaches
  - Heuristic *selection* methodologies
  - Heuristic *generation* methodologies
- The 'Cross-domain Heuristic Search Challenge'

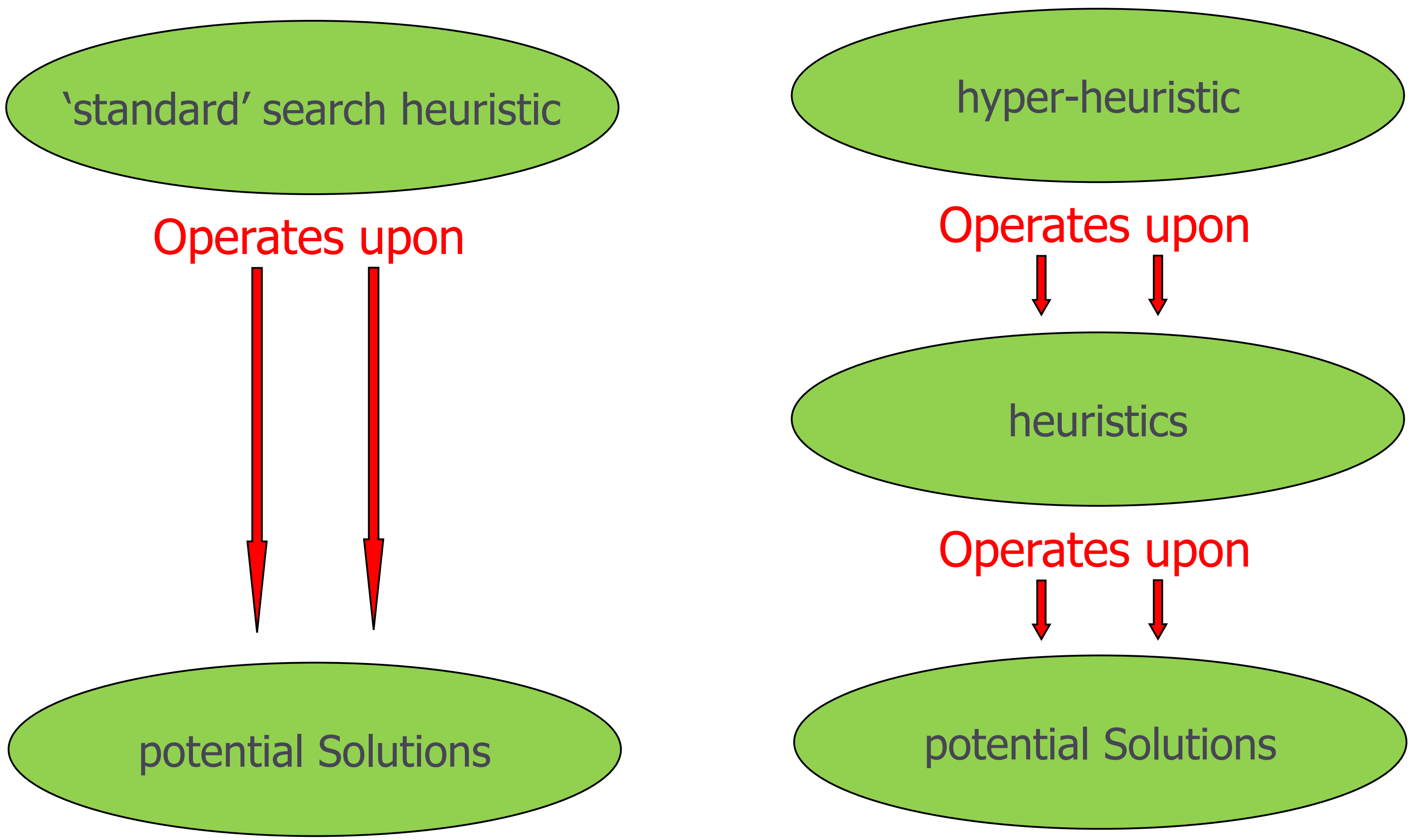
What is a hyper-heuristic?



What is a hyper-heuristic?

- Hyper-heuristics
  - "*Heuristics to choose heuristics*"

What is a hyper-heuristic?



What is a hyper-heuristic?

- All the term hyper-heuristic says is:
  - "Operate on a search space of heuristics"
- Most meta-heuristics operate directly on problems
- Hyper-heuristics operate on heuristics, which are then applied on the actual problems
- But … hyper-heuristics can be meta-heuristics
- Attempt to find the right method or heuristic in a particular situation

What is a hyper-heuristic?

- Recent research trend in hyper-heuristics
  - Automatically *generate* new heuristics suited to a given problem or class of problems
  - Combining, i.e. by GP, *components* or *building-blocks* of human designed heuristics
- New definition:

A hyper-heuristic is an automated methodology for selecting or generating heuristics to solve hard computational search problems

E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. Woodward (2009). A Classification of Hyper-heuristics Approaches, *Handbook of Metaheuristics,* International Series in Operations Research & Management Science, M. Gendreau and J-Y Potvin (Eds.), Springer (in press)

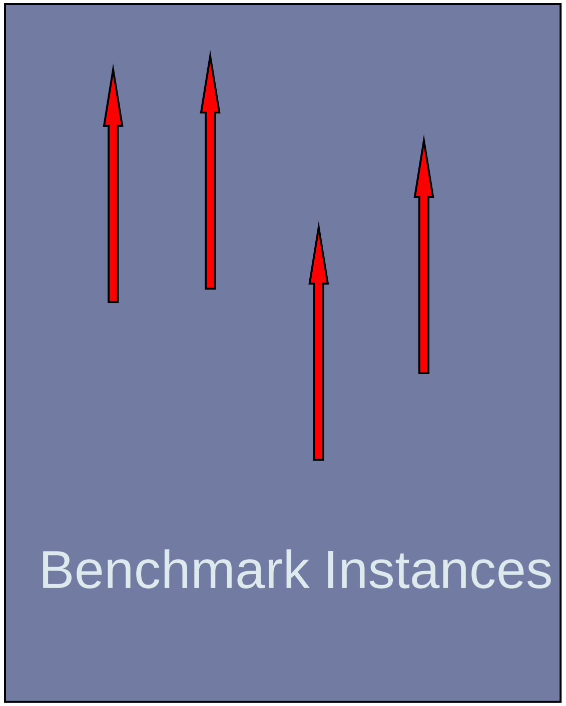
Content

- What is a hyper-heuristic?
- What motivates hyper-heuristic research?
- Origins and related areas
- Classification of hyper-heuristic approaches
  - Heuristic *selection* methodologies
  - Heuristic *generation* methodologies
- The 'Cross-domain Heuristic Search Challenge'
- References

What motivates hyper-heuristics?

The "Up the Wall" game

- We have a problem (e.g. exam timetabling) and a set of benchmark instances
- We develop new methodologies (ever more sophisticated)
- Apply methodologies to benchmarks
- Compare with other "players"
- The goal is to "get further up the wall" than the other players
- Consequence: Made to measure (handcrafted) *Rolls-Royce* systems

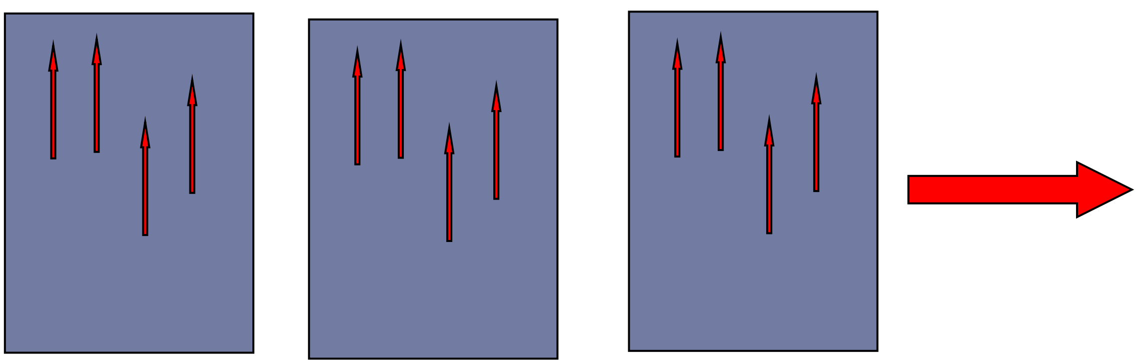


e.g. Exam Timetabling

What motivates hyper-heuristics?

The "Many Walls" game

- Can we develop the ability to automatically work well on different problems?
- Raising the level of generality
- Still want to get as high up the wall as possible … BUT…
- We want to be able to operate on as many different walls as possible
- Consequence: Off the peg, *Ford* model



One method that operates on several problems

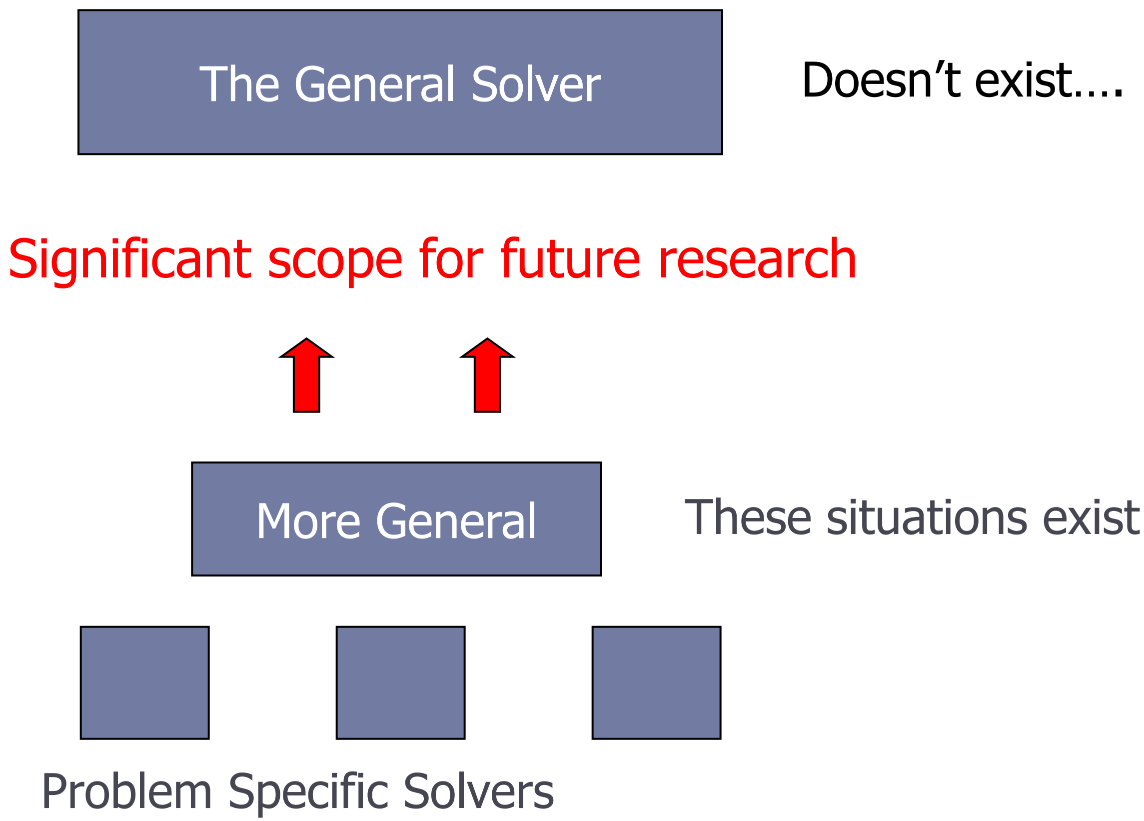
What motivates hyper-heuristics?

- Develop decision support systems that are *off the peg*
- Develop the ability to automatically work well on different problems

Research challenges

- Automate heuristic design
  - Now made by human experts
  - Not cheap!
- How general we could make hyper-heuristics
  - No free lunch theorem

What motivates hyper-heuristics?



Content

- What is a hyper-heuristic?
- What motivates hyper-heuristic research?
- Origins and related areas
- Classification of hyper-heuristic approaches
  - Heuristic *selection* methodologies
  - Heuristic *generation* methodologies
- The 'Cross-domain Heuristic Search Challenge'

Origins and early approaches

- Term *hyper-heuristics*
  - First used 1997 (Dezinger et. al): a protocol for combining several AI methods in automated theorem proving
  - Independently used in 2000 (Colwing et. al): 'heuristic to choose heuristics' in combinatorial optimisation
  - First journal paper (Burke et. al, 2003)
- The ideas can be traced back to the 60s and 70s
  - Automated heuristic sequencing (early 60s and 90s)
  - Automated planning systems (90s)
  - Automated parameter control in evolutionary algorithms (70s)
  - Automated learning of heuristic methods (90s)
  - Automated prioritising: "Squeaky Wheel" optimisation (1999)

Related areas

- Offline approaches
  - Automated algorithm configuration
  - Meta-learning
  - Genetic programming
- Online approaches
  - Adaptive memetic algorithms
  - Adaptive operator selection
  - Parameter control in evolutionary algorithms
  - Adaptive and self-adaptive search algorithms
  - Reactive search
  - Algorithm portfolios

Content

- What is a hyper-heuristic?
- What motivates hyper-heuristic research?
- Origins and related areas
- Classification of hyper-heuristic approaches
  - Heuristic *selection* methodologies
  - Heuristic *generation* methodologies
- The 'Cross-domain Heuristic Search Challenge'
- References

Classification of hyper-heuristics

Search paradigms

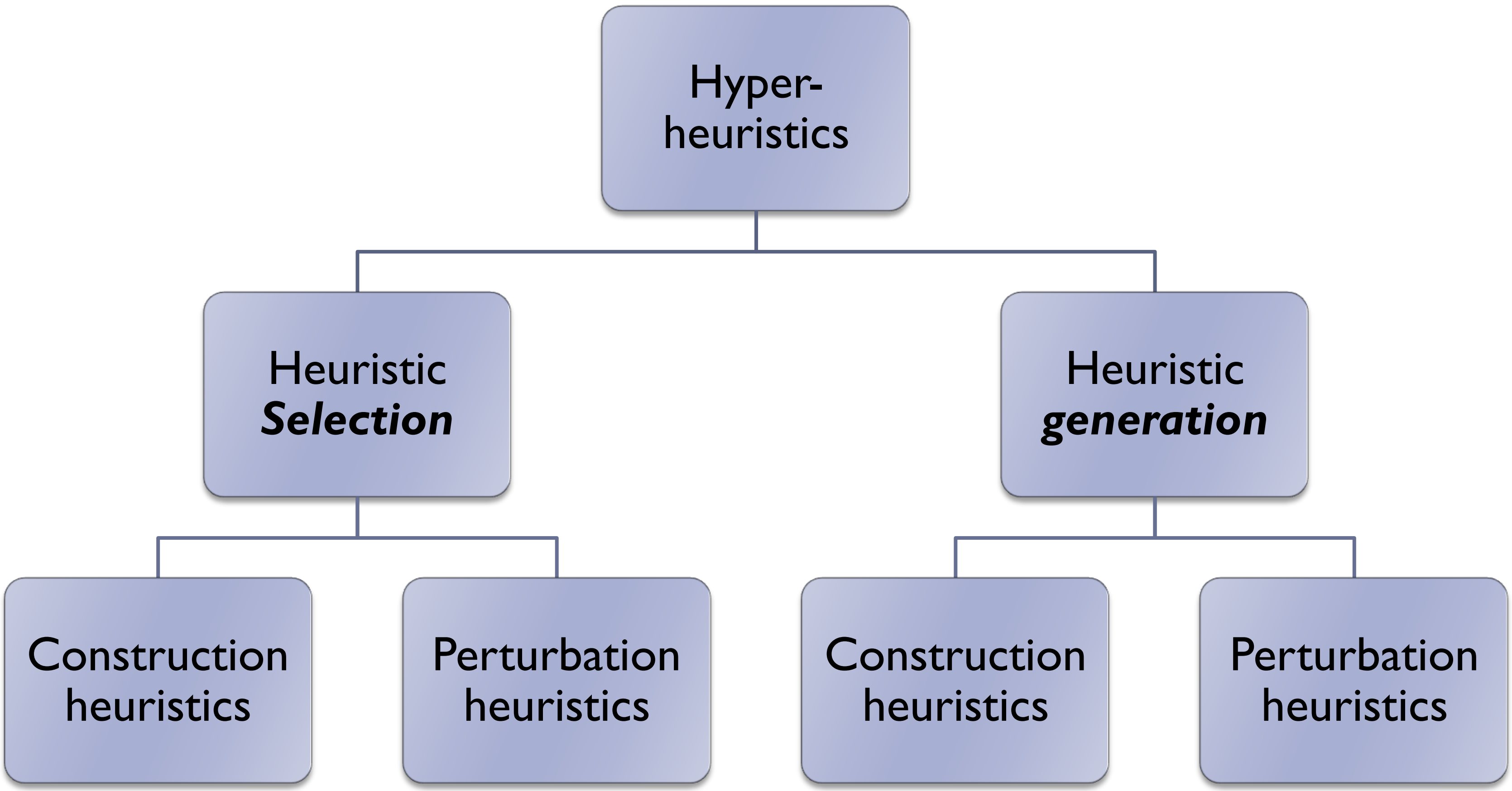
Perturbation

- Search space: complete candidate solutions
- Search step: modification of one or more solution components
- TSP: 2-opt exchanges

Construction

- Search space: partial candidate solutions
- Search step: extension with one or more solution components
- TSP: Next-neighbour

Classification of hyper-heuristics (nature of the search space)



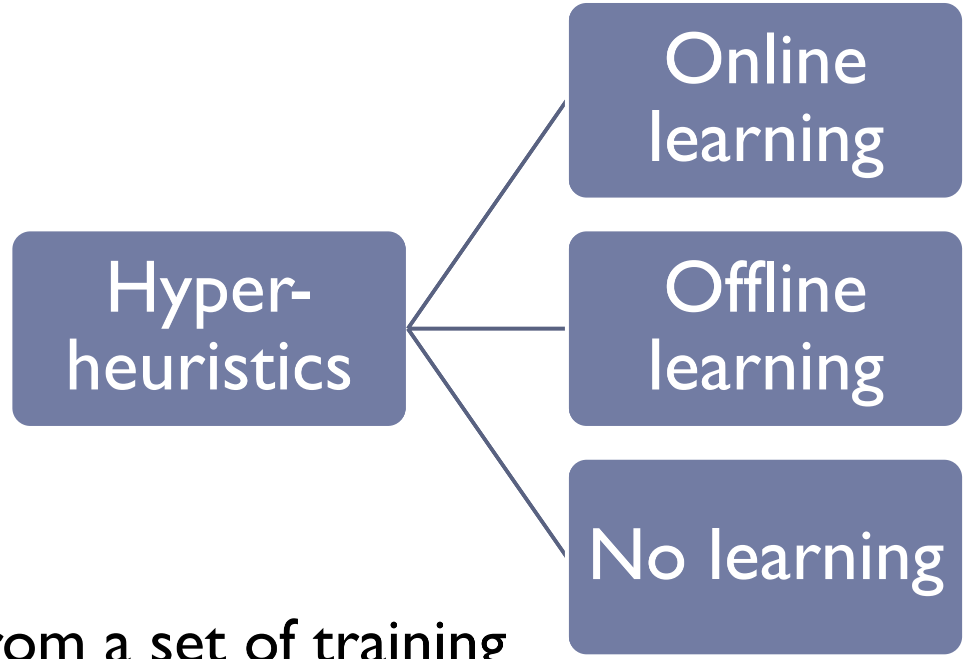
Fixed, human-designed low level heuristics

Heuristic *components*

Classification of hyper-heuristics (source of feedback during learning)

Online

- Learning while solving a single instance
- Adapt
- Examples: reinforcement learning, meta-heuristics



Offline

- Gather knowledge from a set of training instances
- Generalise
- Examples: classifier systems, case-based, GP

Content

- What is a hyper-heuristic?
- What motivates hyper-heuristic research?
- Origins and related areas
- Classification of hyper-heuristic approaches
  - Heuristic *selection* methodologies
  - Heuristic *generation* methodologies
- The 'Cross-domain Heuristic Search Challenge'

HHs based on construction heuristics vs. HHs based on perturbation heuristics

	Perturbation	Construction
Initial solution	Complete	Empty
Training phase	No (Online)	Yes (Offline) and No
Objective function	Yes	Other measures may be needed
Low-level heuristics	Operate in solution space	Operate in state space
Stopping condition	User-defined	(automatic) final state
Re-usability	Easy	Less (training required for each problem)

Case study 1 Constructive Hype-Heuristic : A Graph-Based Hyper-heuristic for Educational Timetabling Problems

Hyper-heuristics Tutorial

Graph-based hyper-heuristics

- A general framework (GHH) employing a set of low level constructive graph colouring heuristics
- Low level heuristics: sequential methods that order events by the difficulties of assigning them
  - 5 graph colouring heuristics
  - Random ordering strategy
- Applied to exam and course timetabling problem

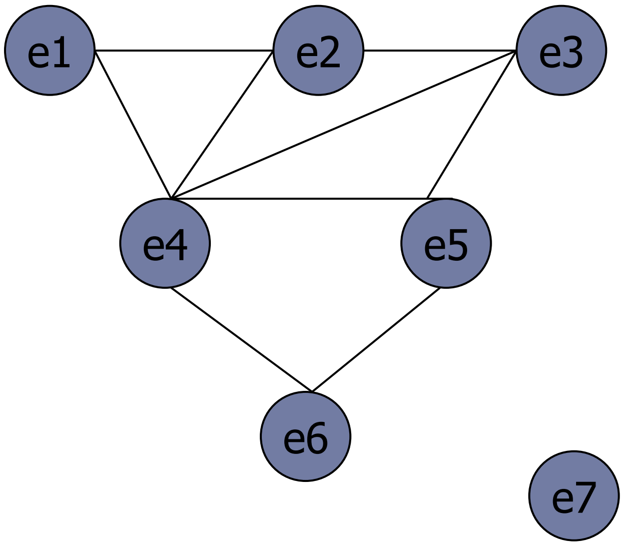
E.K.Burke, B.McCollum, A.Meisels, S.Petrovic & R.Qu. A Graph-Based Hyper Heuristic for Educational Timetabling Problems. EJOR, 176: 177-192, 2007.

Examination timetabling

- A number of exams *(e1, e2, e3, …)*, taken by different students *(s1, s2, s3, …)*, need to be scheduled to a limited time periods *(t1, t2, t3, …)* and certain rooms *(r1, r2, r3, …)*
- Hard Constraints
  - Exams taken by common students can't be assigned to the same time period
  - Room capacity can't be exceeded
- Soft Constraints
  - Separation between exams
  - Large exams scheduled early

Examination timetabling

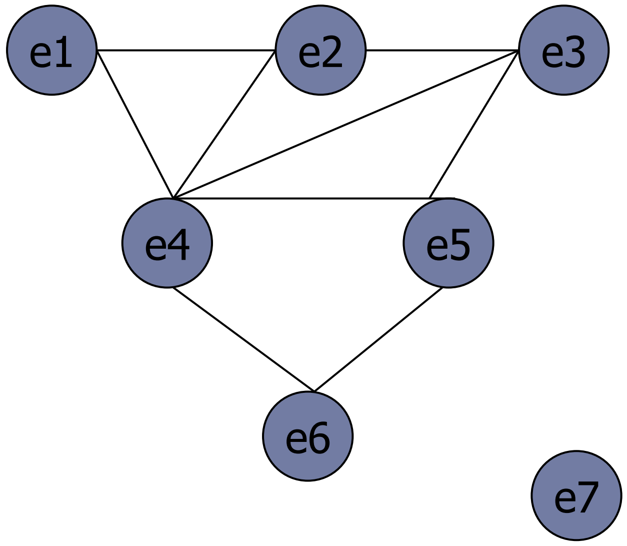
- How can we represent/model this problem?
  - There are 7 exams, e1 ~ e7
  - 5 students taking different exams
    - s1: e1, e2, e4
    - s2: e2, e3, e4
    - s3: e3, e4, e5
    - s4: e4, e5, e6
    - s5: e7
  - let's ignore rooms at the moment



Examination timetabling

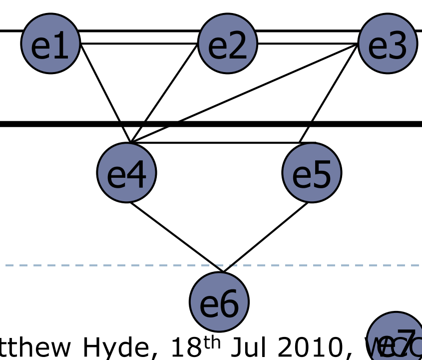
Can be modelled as graph colouring problems

- Nodes: exams
- Edges: adjacent exams (nodes) have common students
- Colours: time periods
- Objective: assign colours (time periods) to nodes (exams), adjacent nodes with different colour, minimising time periods used

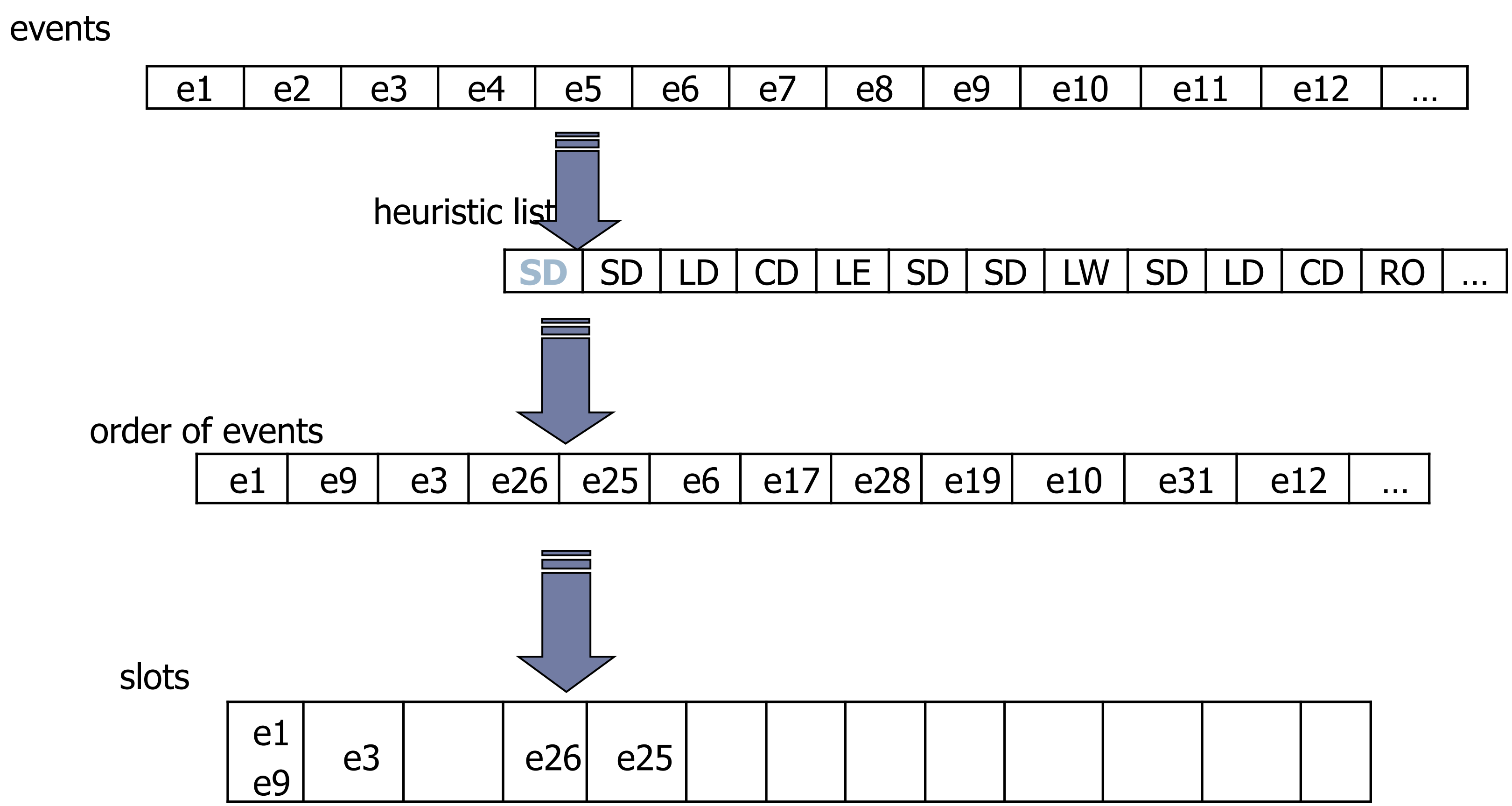


Graph-based hyper-heuristics

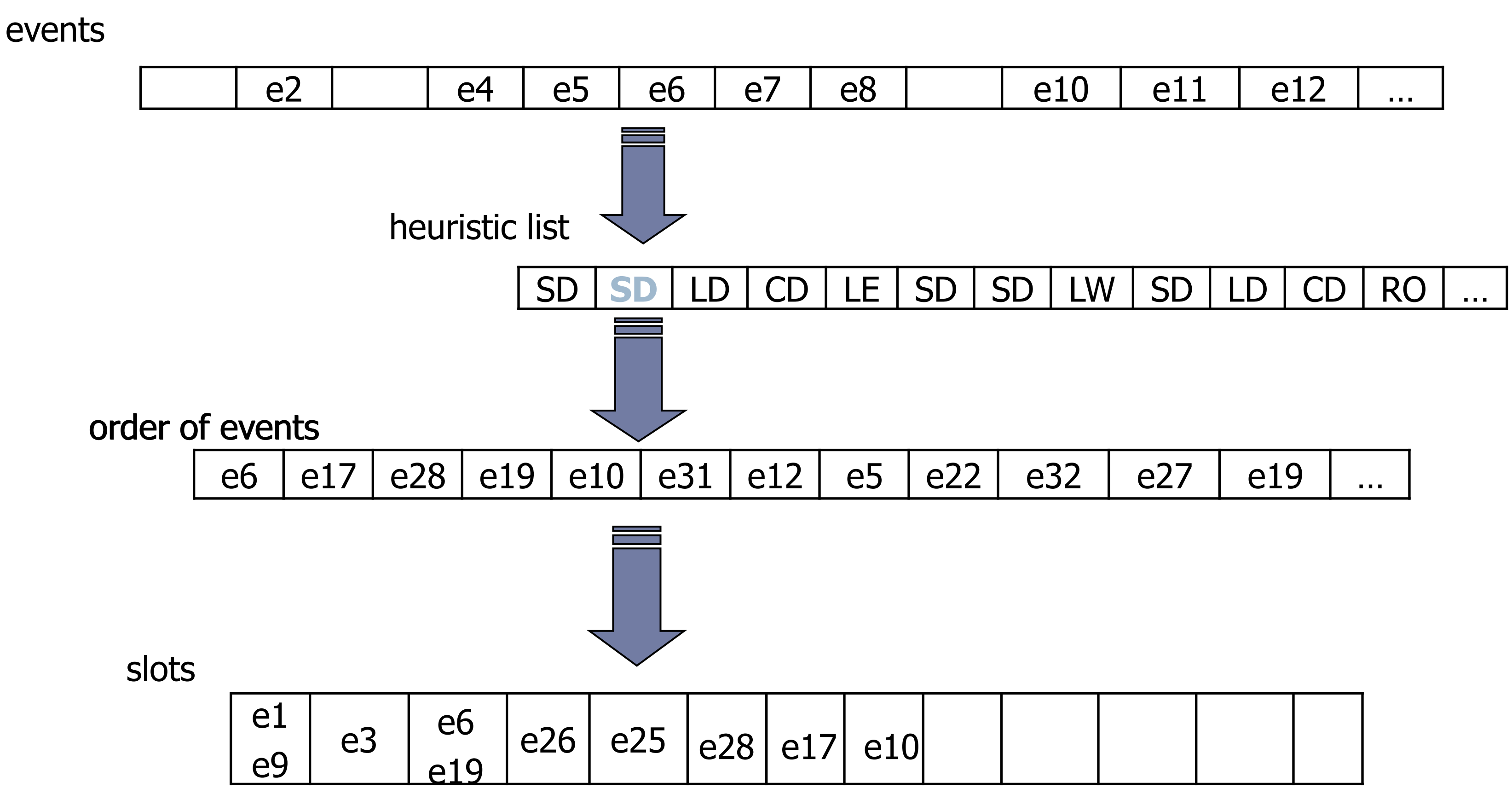
Graph Heuristics	Ordering strategies
Largest degree (LD)	Number of clashed events
Largest weighted degree (LW)	LD with number of common students
Saturation degree (SD)	Number of valid remaining time periods
Largest enrolment (LE)	Number of students
Colour degree (CD)	Number of clashed event that are scheduled
+	
Random ordering (RO)	Randomly



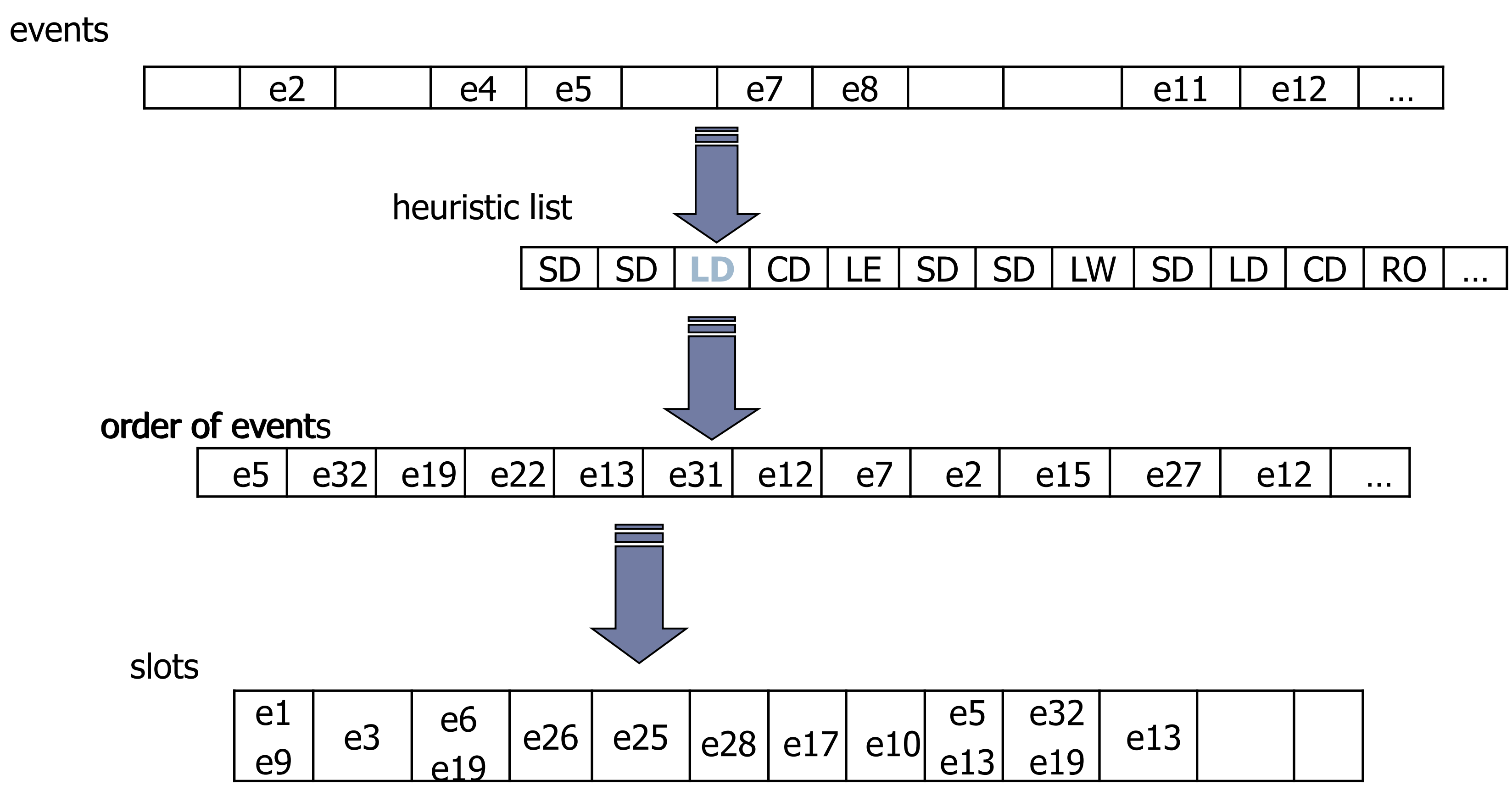
Graph-based hyper-heuristics



Graph-based hyper-heuristics



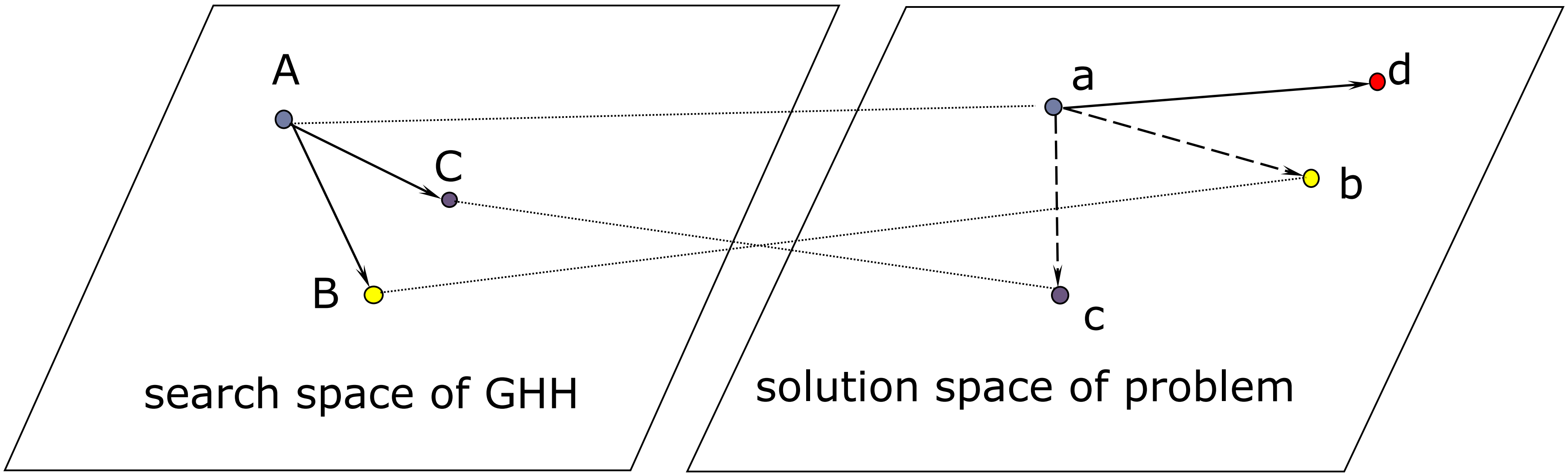
Graph-based hyper-heuristics



Graph-based hyper-heuristics

- Tabu Search at the high level
  - Neighbourhood operator: randomly change two heuristics in the heuristic list
  - Objective function: quality of solutions built by the corresponding heuristic list
  - Tabu list: visits to the same heuristic lists forbidden
- Other high-level search strategies tested
  - Steepest Descent
  - Variable neighbourhood search → best performing
  - Iterated Steepest Descent

Graph-based hyper-heuristics



Two search spaces

search space of heuristics: sequences of low level heuristics
solution space of problem: actual solutions

Application Domain	Reference(s)
Production Scheduling	(Fisher and Thompson, 1961, 1963) (Storer et al, 1992, 1995) (Dorndorf and Pesch, 1995) (Fang et al, 1993, 1994) (Norenkov and Goodman, 1997) (Hart and Ross, 1998; Hart et al, 1998) (Vázquez-Rodríguez et al, 2007a,b; Ochoa et al, 2009b) (Vázquez-Rodríguez and Petrovic, 2010) (Cano-Belmán and and J. Bautista, 2010)
Educational Timetabling	(Terashima-Marín et al, 1999) (Ahmadi et al, 2003; Asmuni et al, 2005) (Ross et al, 2004; Ross and Marín-Blázquez, 2005) (Burke et al, 2005a, 2006b) (Burke et al, 2007c; Qu and Burke, 2009; Ochoa et al, 2009a) (Pillay and Banzhaf, 2007; Pillay, 2008)
1D Packing	(Ross et al, 2002, 2003; Marín-Blázquez and Schulenburg, 2007) (Marín-Blázquez and Schulenburg, 2007)
2D Cutting and Packing	(Terashima-Marín et al, 2006, 2007, 2008a) (Garrido and Riff, 2007a,b)
Constraint Satisfaction	(Terashima-Marín et al, 2008b)
Vehicle Routing	(Garrido and Castro, 2009; Garrido and Riff, 2010)

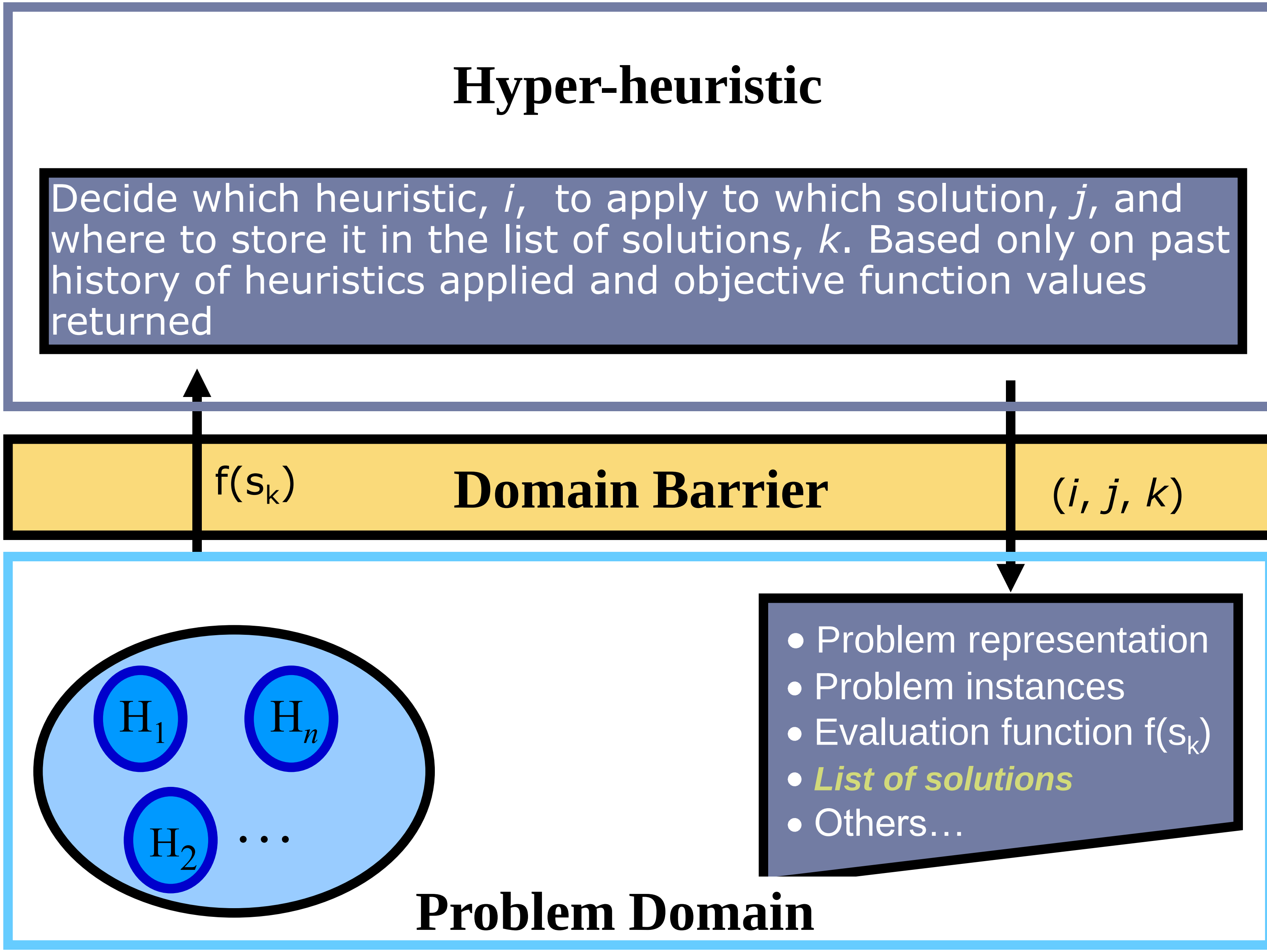
Application domains - hyper-heuristics based on construction heuristics

High-level strategy	Reference(s)
Hill-climbing	(Storer et al, 1992, 1995) (Gratch and Chien, 1996; Gratch et al, 1993) (Garrido and Castro, 2009)
Genetic Algorithms	(Dorndorf and Pesch, 1995) (Fang et al, 1993, 1994) (Norenkov and Goodman, 1997) (Hart and Ross, 1998; Hart et al, 1998) (Terashima-Marín et al, 1999) (Ahmadi et al, 2003) (Vázquez-Rodríguez et al, 2007a,b; Ochoa et al, 2009b) (Garrido and Riff, 2007a,b, 2010) (Pillay and Banzhaf, 2007; Pillay, 2008)
Meta-heuristics (TS, ILS, VNS)	(Ahmadi et al, 2003) (Burke et al, 2007c; Qu and Burke, 2009)
Fuzzy Systems	(Asmuni et al, 2005)
Scatter Search	(Cano-Belmán and and J. Bautista, 2010)
Case-based Reasoning (Offline)	(Burke et al, 2005a, 2006b)
Classifier Systems (Offline)	(Ross et al, 2002; Marín-Blázquez and Schulenburg, 2007) (Terashima-Marín et al, 2007)
Messy Genetic Algorithms (Offline)	(Ross et al, 2003, 2004; Ross and Marín-Blázquez, 2005) (Terashima-Marín et al, 2006, 2007, 2008a,b)

High-level strategies - hyper-heuristics based on construction heuristics

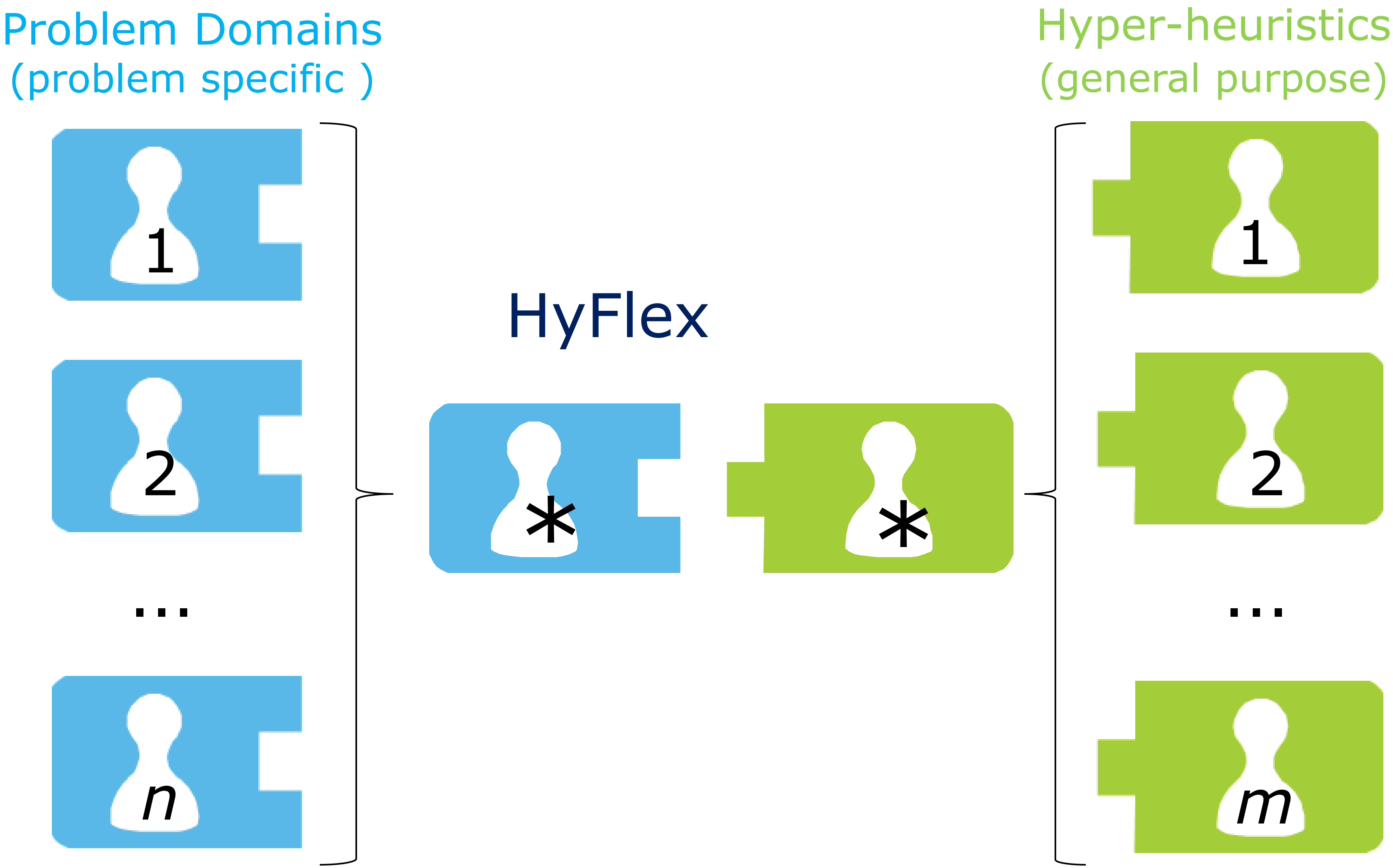
Case study 2 Pertrubation Hyper-heuristic: A Tabu-Search Hyperheuristic for Timetabling and Rostering

Hyper-heuristics Tutorial



HH fremework:(Cowling P., Kendall G. and Soubeiga, 2000, 2001), (E. K. Burke et al., 2003)

HyFlex: re-use and Interchange



Tabu search hyper-heuristics

- Heuristics selected according to learned ranks (using reinforcement learning)
- Dynamic tabu list of heuristics that are temporarily excluded from the selection pool
- Applied to
  - Nurse rostering
  - Course timetabling
- Produced good results/ comparable with state-of-the art
- More general than the tailor-made algorithms
- Later combined with SA acceptance

Burke, E.K., Kendall. G., Soubeiga. E. (2003) A Tabu-Search Hyperheuristic for Timetabling and Rostering, *Journal of Heuristics*,Vol 9

University course timetabling

- Schedule a number of courses, taken by a set of students and taught by lecturers, to a limited time period (usually on week basis) and rooms with certain features
- Related to exam timetabling problems, but with many differences on constraints
  - Courses scheduled consecutively
  - Courses can't be combined into one room
  - Preferred time periods

Tabu search hyper-heuristics

Course timetabling (Low-level heuristics)

- [N1]: Move a random event from its current timeslot to a random one
- [N2]: Same as [N1] but `1st improving hard constraints'
- [N3]: Same as [N1] but `1st improving soft and no worsening of hard constraints'
- [N4]: Swap the timeslots of two random events
- [N5]: Same as [N4] but `1st improving hard constraints'
- [N6]: Same as [N4] but `1st soft and no worsening of hard constraints'

Tabu search hyper-heuristics

Each heuristic k is assigned a rank r_k initialised to 0 and allowed to increase and decrease within interval $[r_{min}, r_{max}]$

Do:

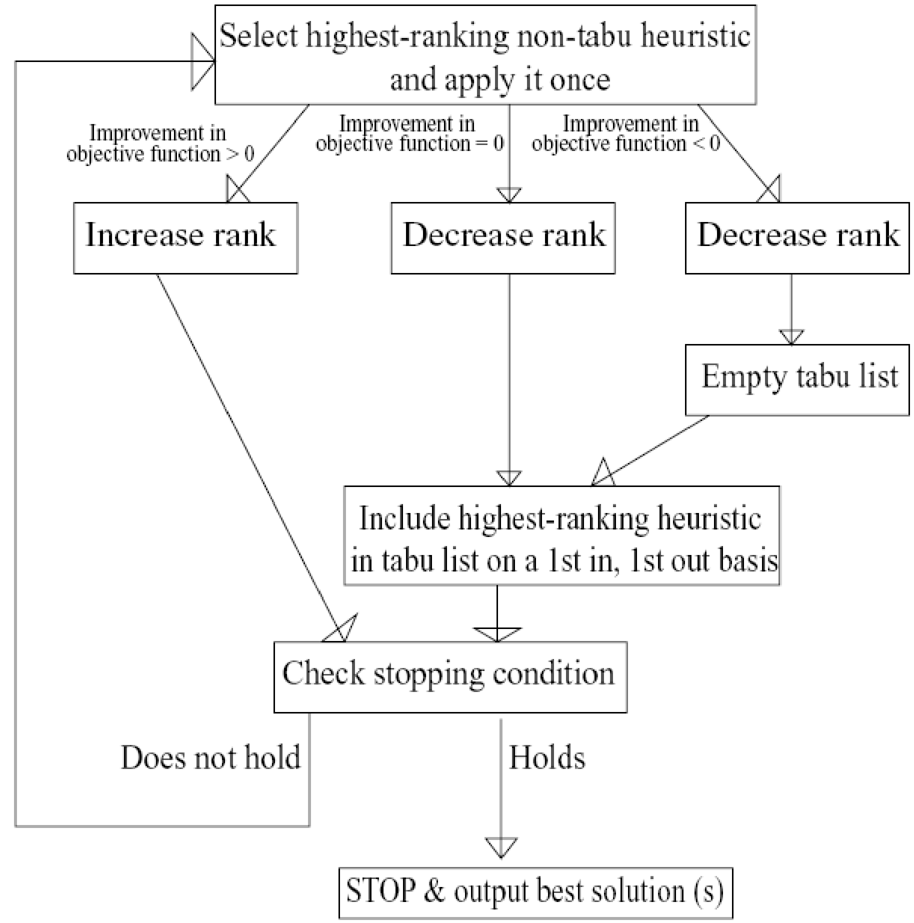
1- Select heuristic k with highest rank r_k and apply it once

2 - If $\Delta > 0$ then $r_k = r_k + \alpha$

- Otherwise $r_k = r_k - \alpha$,

Include heuristic k in TABULIST

Until Stop = true.



Application domain	Reference(s)
Channel assignment	Kendall and Mohamad (2004a,b)
Component placement	Ayob and Kendall (2003)
Personnel scheduling	Cowling et al (2000) Cowling and Chakhlevitch (2003) Han and Kendall (2003) Burke et al (2003b) Bai et al (2007a)
Packing	Dowsland et al (2007) Bai et al (2007a)
Planning	Nareyek (2003)
Production scheduling	Ouelhadj and Petrovic (2008, 2009, 2010)
Reactive power compensation	Antunes et al (2009)
Space allocation	Burke et al (2005c) Bai and Kendall (2005) Bai et al (2008)
Space-probe trajectory optimisation	Biazzini et al (2009)
Timetabling	Burke et al (2003b, 2005b) Bilgin et al (2006) Chen et al (2007) Bai et al (2007a) Ozcan et al (2009)
Vehicle routing problems	Pisinger and Ropke (2007) Meignan et al (2010)

Application domains - hyper-heuristics based on perturbation heuristics

Heuristic selection + Move Acceptance

Component name	Reference(s)
Heuristic selection with no learning	
Simple Random	Cowling et al (2000, 2002b)
Random Permutation	Cowling et al (2000, 2002b)
Greedy	Cowling et al (2000, 2002b); Cowling and Chakhlevitch (2003)
Peckish	Cowling and Chakhlevitch (2003)
Heuristic selection with learning	
Random Gradient	Cowling et al (2000, 2002b)
Random Permutation Gradient	Cowling et al (2000, 2002b)
Choice Function	Cowling et al (2000, 2002b)
Reinforcement Learning	Nareyek (2003); Pisinger and Ropke (2007); Bai et al (2007a)
Reinforcement Learning with Tabu Search	Burke et al (2003b); Dowsland et al (2007)
Deterministic move acceptance	
All Moves	Cowling et al (2000, 2002b)
Only Improvements	Cowling et al (2000, 2002b)
Improving and Equal	Cowling et al (2000, 2002b)
Non-deterministic move acceptance	
Monte Carlo	Ayob and Kendall (2003)
Great Deluge	Kendall and Mohamad (2004a); Bilgin et al (2006)
Record to Record Travel	Kendall and Mohamad (2004b)
Tabu Search	Chakhlevitch and Cowling (2005)
Simulated Annealing	Bai and Kendall (2005); Bilgin et al (2006); Pisinger and Ropke (2007); Antunes et al (2009)
Simulated Annealing with Reheating	Dowsland et al (2007); Bai et al (2007a)

Summary of 1st part

A hyper-heuristic is an automated methodology for selecting or generating heuristics to solve hard computational search problems

- Main feature: search in a space of heuristics
- Term used for '*heuristics to choose heuristics*' in 2000
- Ideas can be traced back to the 60s and 70s
- Two main type of approaches
  - Heuristic selection
  - Heuristic generation
- Ideas from online and offline machine learning are relevant, as are ideas of meta-level search

Future work

- Generalisation: By far the biggest challenge is to develop methodologies that work well across several domains
- Foundational studies: Thus far, little progress has been made to enhance our understanding of hyper-heuristic approaches
- Distributed, agent-based and cooperative approaches: Since different low-level heuristics have different strengths and weakness, cooperation can allow synergies between them
- Multi-criteria, multi-objective and dynamic problems: So far, hyper-heuristics have been mainly applied to single objective and static problems

Heuristic Generation Methodologies

Hyper-heuristics Tutorial

Outline

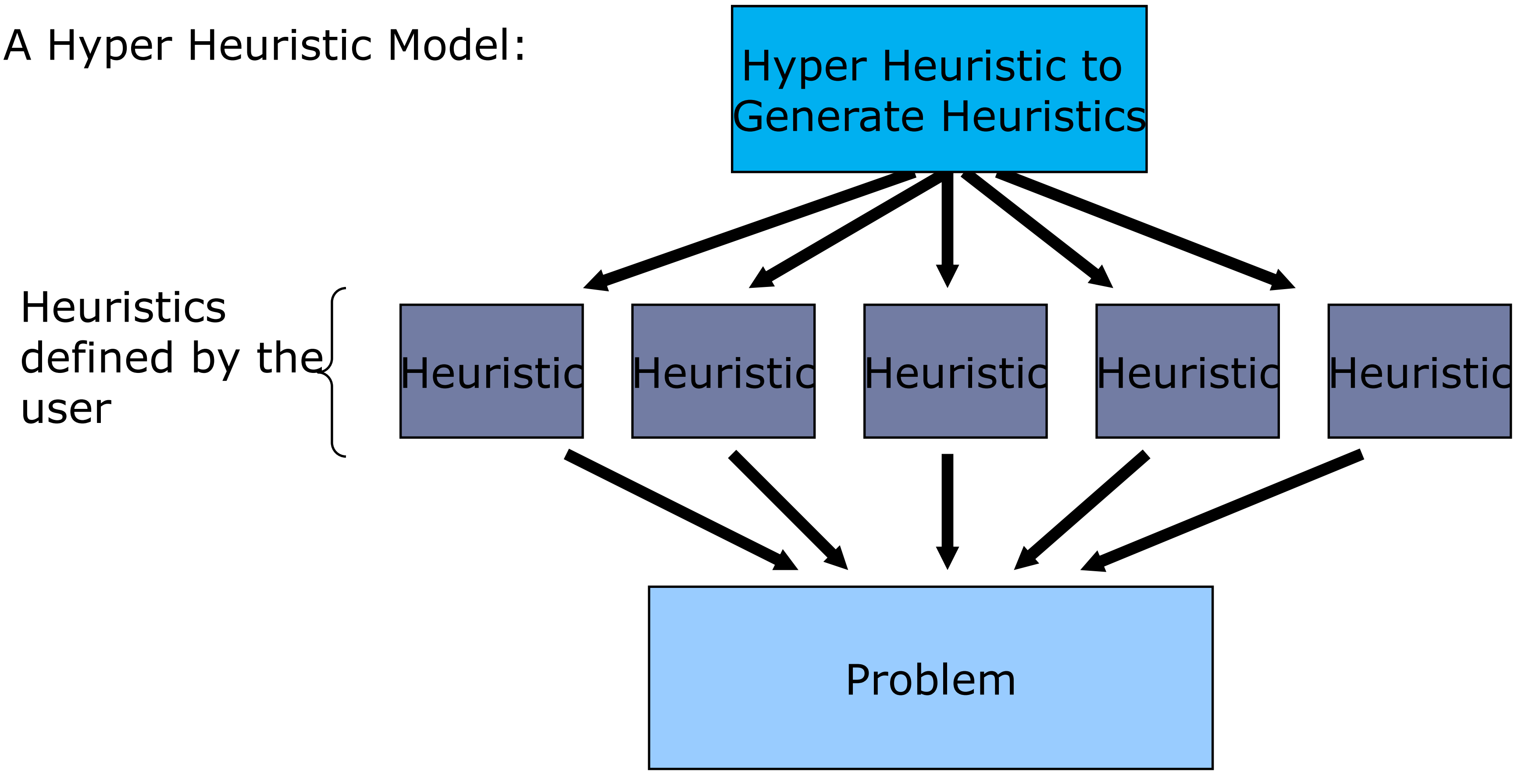
- Introduction to this section
  - Hyper-Heuristic Definition
  - What's the Point?
- Case Study 1: SAT
- Case Study 2: Flow Shop
- Case Study 3: Bin Packing
- Conclusion

Hyper-Heuristic Definition

"A hyper-heuristic is an automated methodology for selecting or **generating** heuristics to solve hard computational search problems"

Two Types of Hyper-Heuristic?

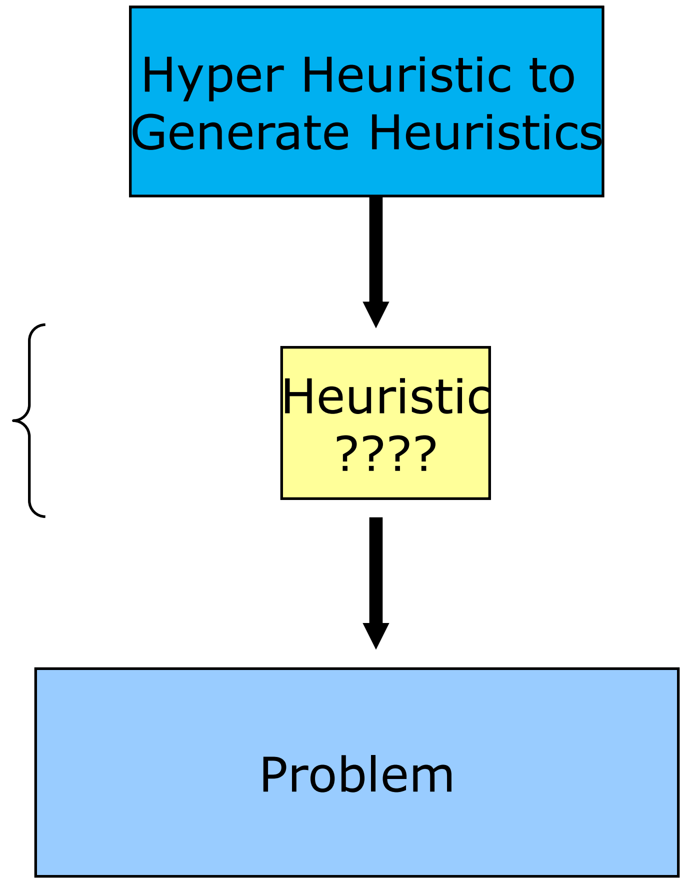
A Hyper Heuristic Model:



Two Types of Hyper-Heuristic?

A Hyper Heuristic Model:

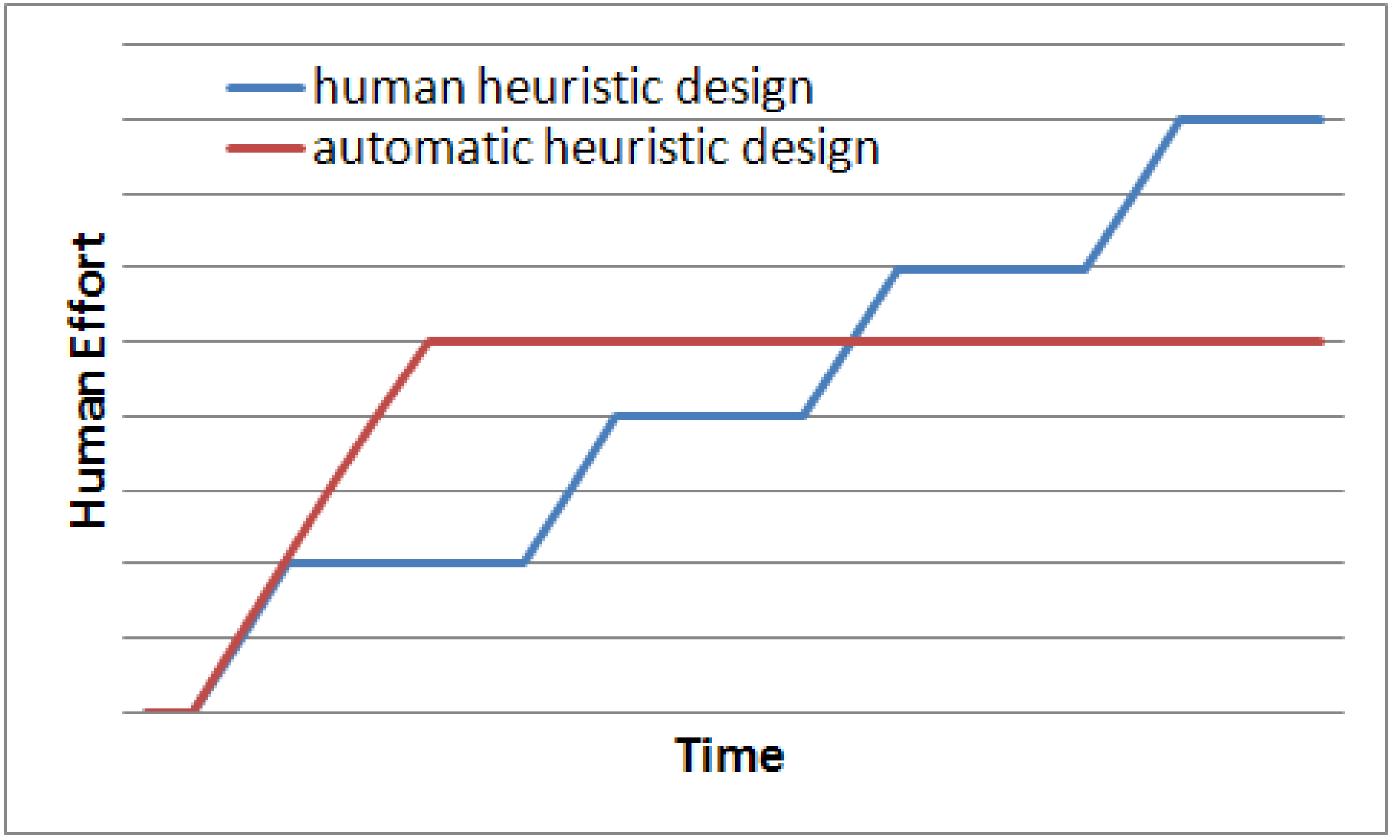
Domain-Specific Heuristic Defined by the Hyper-Heuristic



What's the Point?

- We spend a lot of time testing, and fine tuning, solution methods.
- They are usually specialised to a particular problem instance set, with certain characteristics.
- Automating this creative process can potentially save time and/or effort.
- Humans still have a creative role in heuristic generation, but the idea is that more of the process is automated.

What's the Point?



Heuristic Generation Methodologies
Case Study 1

Hyper-heuristics Tutorial

CASE STUDY 1

- Evolving Heuristics for SAT
- Bader-el-Din and Poli, 2007
- Based on Fukunaga, 2004, 2008
- SAT local search heuristics can be evolved from a set of components, obtained by analysing existing heuristics from the literature

Evolving Heuristics for SAT

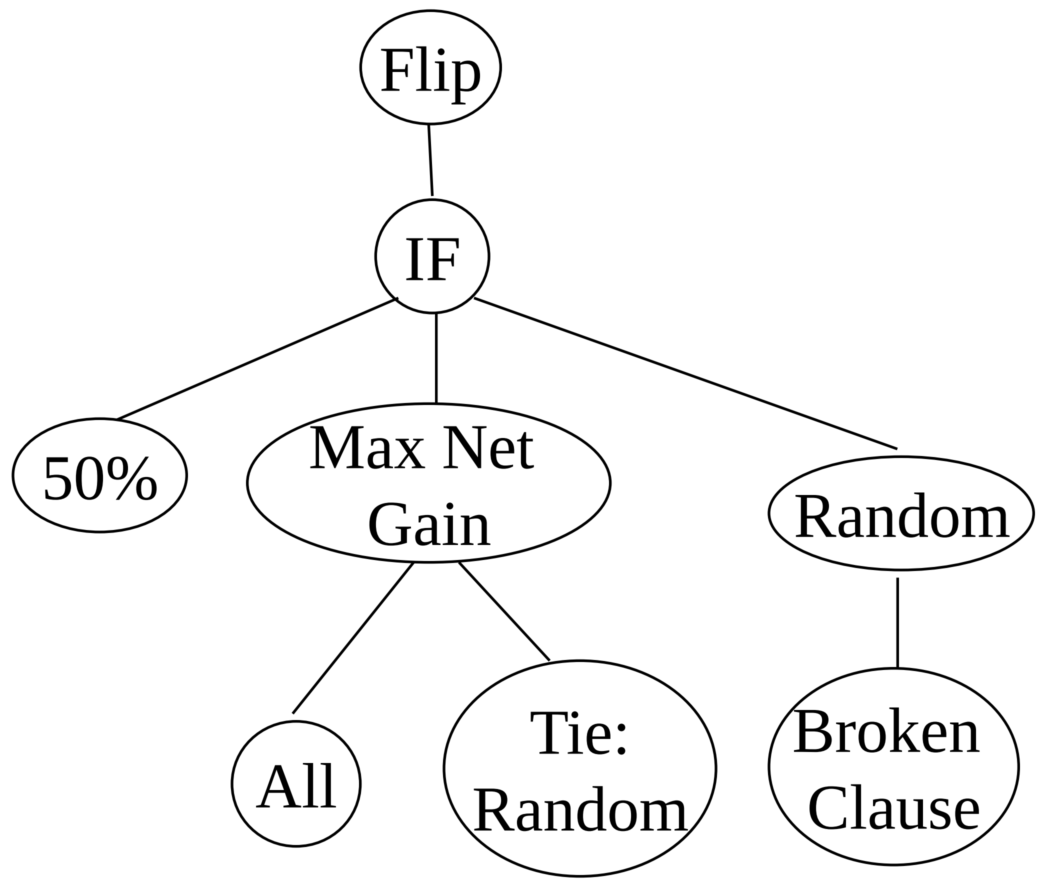
- Make a boolean expression true
- (¬A or B or C) AND (B or ¬C or E) AND (¬B or A or ¬D) AND (…) AND (…) …
- Hundreds/thousands of variables and clauses
- Local search heuristics iteratively choose a variable to flip.

Existing Heuristics for SAT

- GSAT
  - Flip variable which removes the most broken clauses (highest 'net gain')
- HSAT
  - Same as GSAT, but break ties by choosing the variable that has remained 'unflipped' for the longest
- HARMONY
  - Pick random broken clause BC. Select the variable V in BC with highest net gain, unless V has been flipped most recently in BC. If so, select V with probability p. Otherwise, flip variable with 2nd highest net gain

Existing Heuristics for SAT

- GWSAT
  - With probability 0.5, apply GSAT
  - Otherwise flip a random variable in a random broken clause.



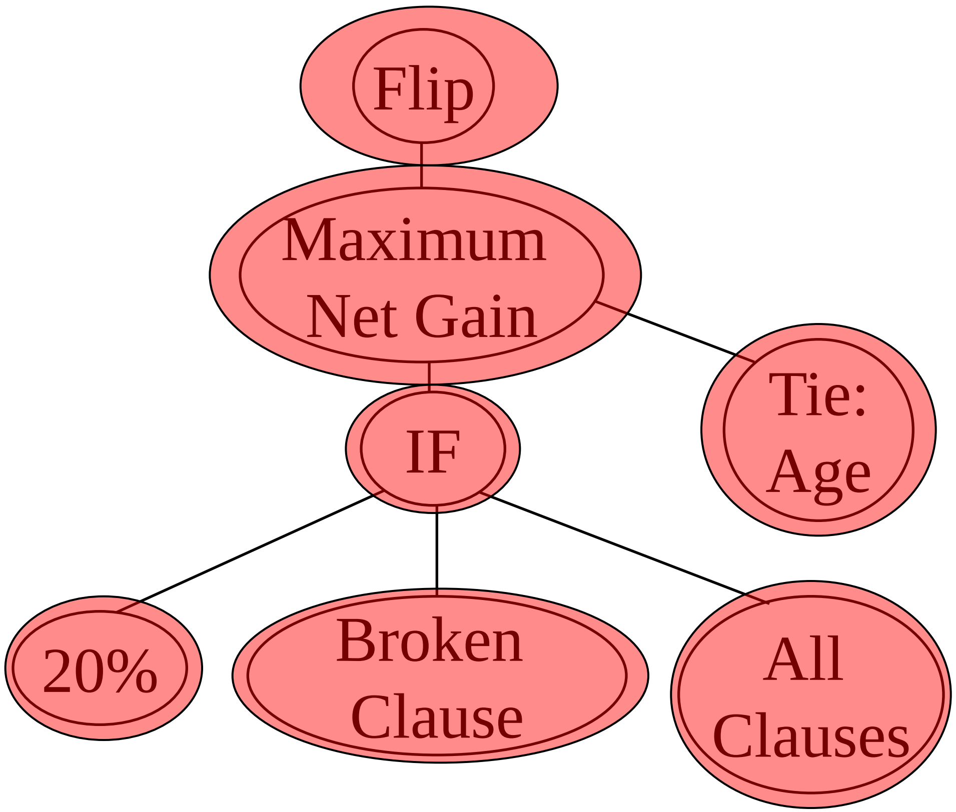
Evolving New SAT Heuristics

- They define a grammar, which can represent many heuristics from the literature, and new heuristics

```
start →  FLIP v
v     →  RANDOM l
        | MAX_SCR l | MAX_SCR l, op
        | IFV prob, v, v
        | MIN_SCR l | MIN_SCR l, op
        | MAX_AGE l | MAX_AGE l, op
l     →  ALL | ALL_USC
        | RAND_USC | USC
        | IFL prob, l, l
        | SCR_Z l | SCR_Z l, op
op    →  TIE_RAND | TIE_AGE
        | TIE_SCR | NOT_ZERO_AGE
prob  →  20 | 40 | 50 |
         70 | 80 | 90
```

Taken from: Bader-El-Din and Poli, "Generating SAT local-search heuristics using a GP hyper-heuristic framework", Proceedings of the 8th International Conference on Artificial Evolution. 2007. pp 37-49

Evolving New SAT Heuristics




```
start → FLIP v
v     → RANDOM l
        | MAX_SCR l | MAX_SCR l, op
        | IFV prob, v, v
        | MIN_SCR l | MIN_SCR l, op
        | MAX_AGE l | MAX_AGE l, op
l     → ALL | ALL_USC
        | RAND_USC | USC
        | IFL prob, l, l
        | SCR_Z l | SCR_Z l, op
op    → TIE_RAND | TIE_AGE
        | TIE_SCR | NOT_ZERO_AGE
prob  → 20 | 40 | 50 |
        70 | 80 | 90
```

Lessons – Case Study 1

- Existing local search heuristics were broken down into components
- These heuristics return a variable to flip, not a value or 'score'
- Local search heuristics evolved here, rather than constructive heuristics

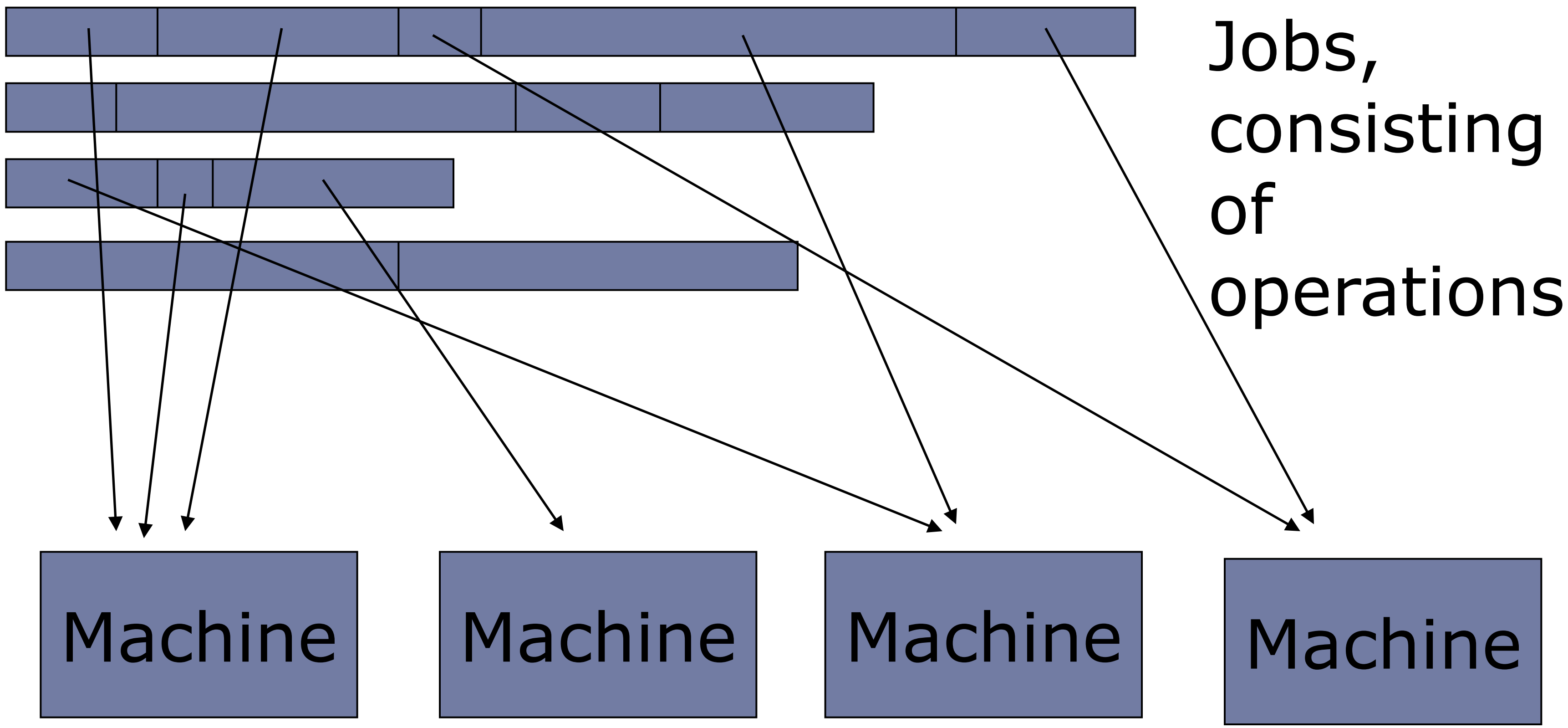
Heuristic Generation Methodologies Case Study 2

Hyper-heuristics Tutorial

CASE STUDY 2

- Multi-Objective Scheduling
- Tay and Ho, 2008
- In a multi-objective flexible job shop problem, composite dispatching rules can be evolved which dominate human created rules from the literature

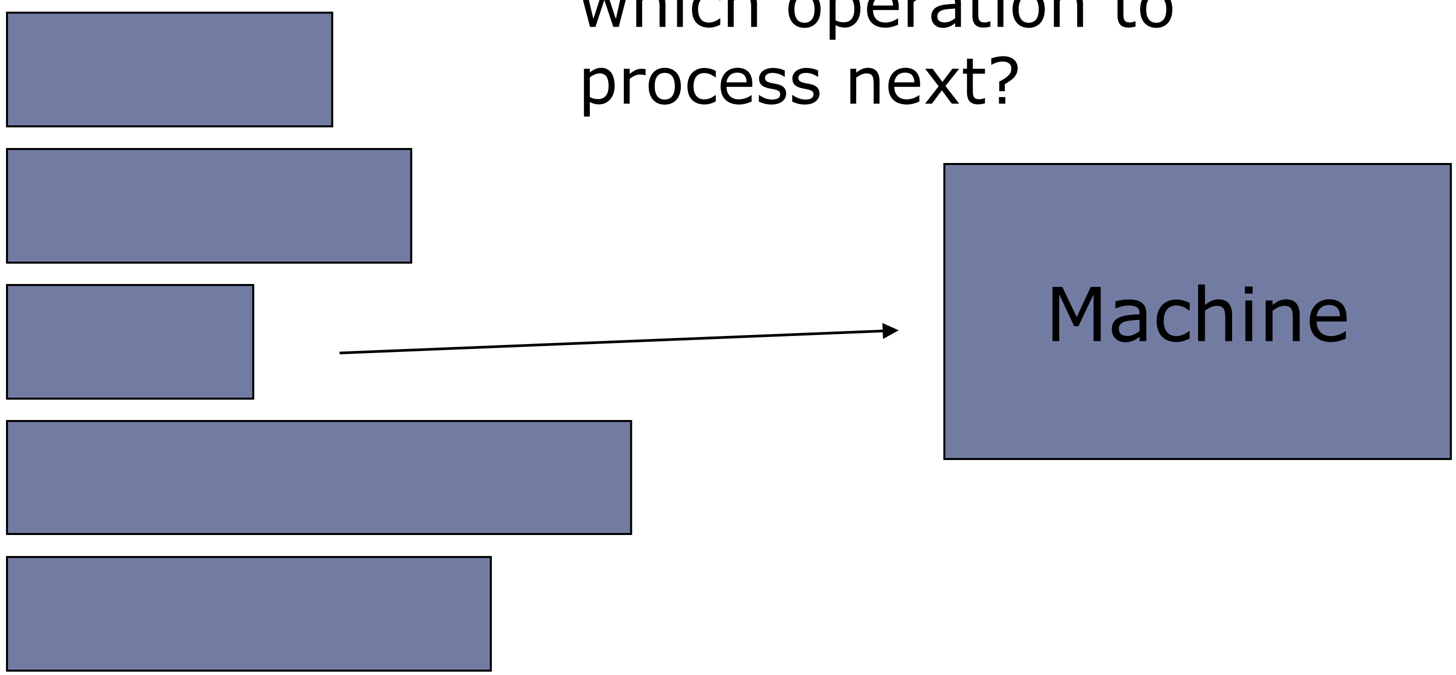
Job-Shop Scheduling



Job-Shop Scheduling

Queue of operations

How should we decide which operation to process next?



Dispatching Rules

- Existing dispatching rules from the literature can be written as formulas, containing:
- Release Date
- Due Date
- Operation Processing Time
- Job Processing Time Remaining
- Current Time
- Number of Operations in Job
- Total Job Processing Time
- + - * /

Evolved Dispatching Rules

- RD + 2PT + 2TPT + nOPS
- Higher priority to:
  - Smaller processing time
  - Jobs with less operations

- RD + DD + TPT + PT – 2(RD / nOPS)
- Higher priority to:
  - Smaller processing time
  - Jobs with **more** operations

Lessons – Case Study 2

- They found that some elements are useful, which are ignored in the literature
- So can discover **counter-intuitive heuristics**
- They **fix some of the algorithm**, and evolve one decision making component.
- Operations are assigned to machines with a fixed algorithm. The order of operations at each machine is decided by the evolved heuristic.

Sufficient Components

- Due date, processing time, current time
- Slack = due date – processing time – current time
- 'Slack' can be added as a single component
- Eliminates the need for slack to be evolved
- But, slight variations of slack cannot be evolved
- 'Expressivity' versus 'Design Effort'

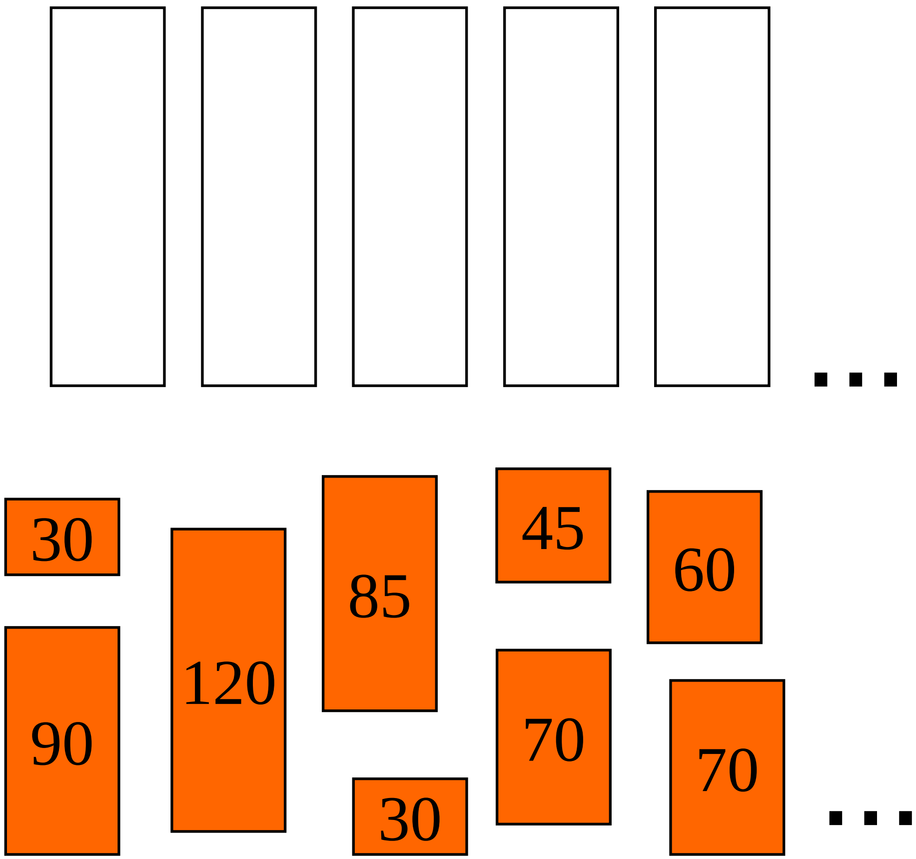
Heuristic Generation Methodologies
Case Study 3

Hyper-heuristics Tutorial

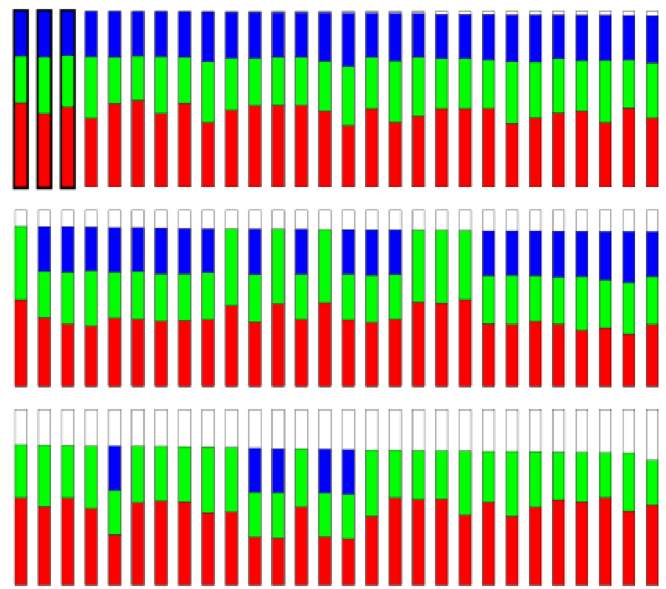
CASE STUDY 3

- One Dimensional Bin Packing
- Burke, Hyde and Kendall, 2007
- Heuristics can be evolved that are specialised to different types of problems

The Bin Packing Problem



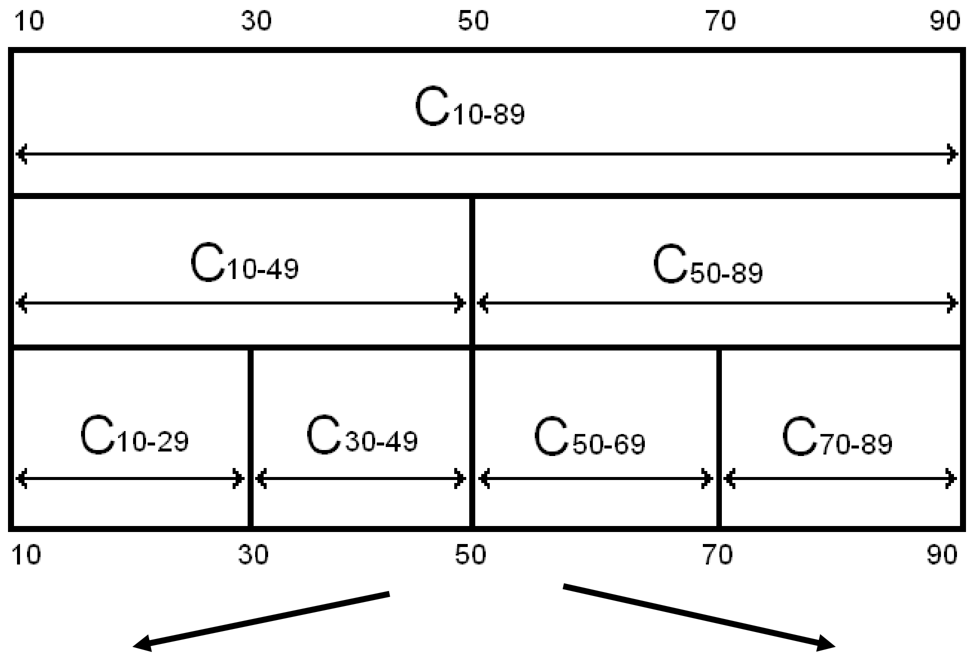
- Pack all the pieces into as few bins as possible



The Bin Packing Problem Set

- Online
- 7 problem classes

- Bin Capacity 150
- 120 items



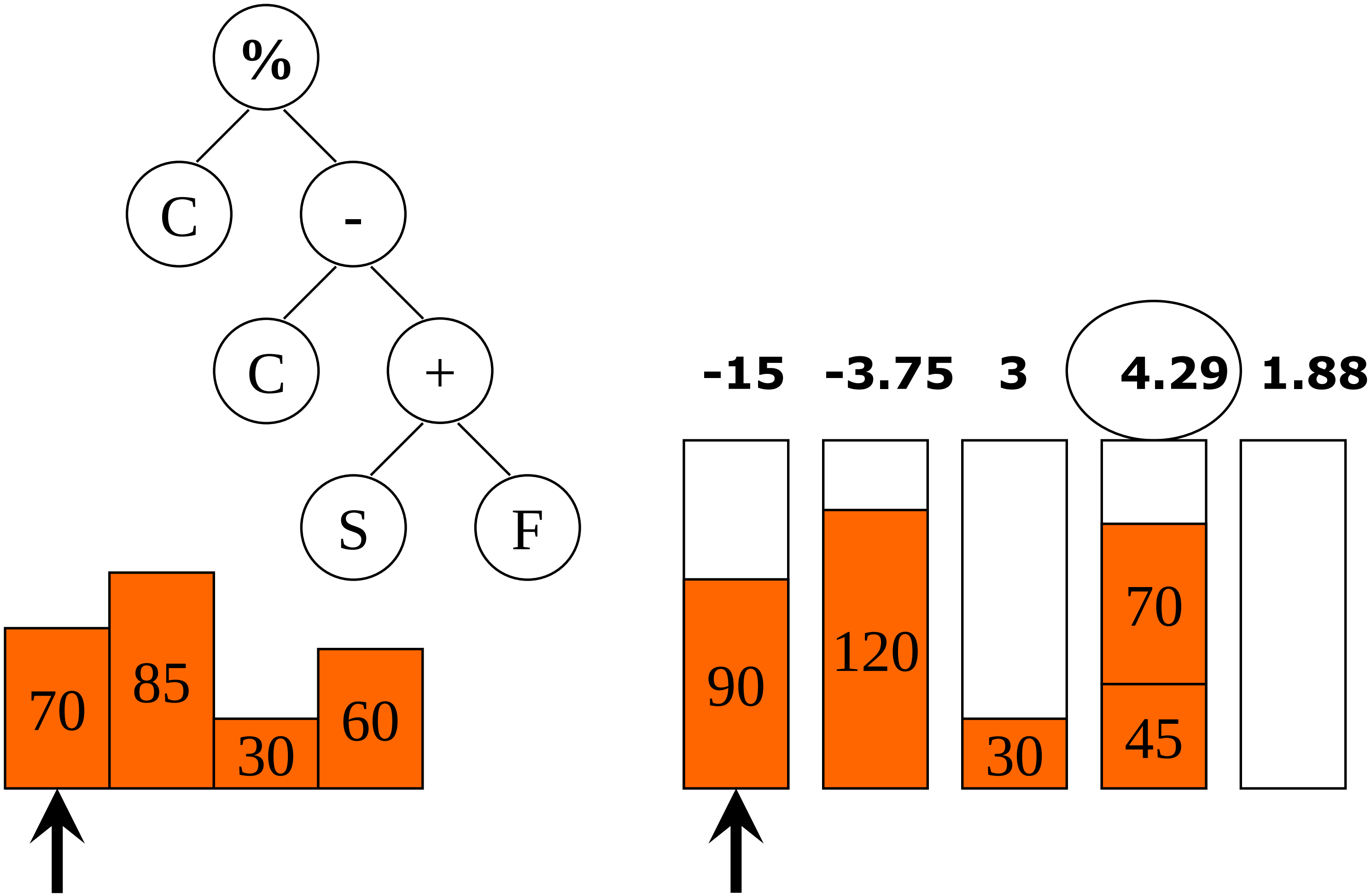
7 Training sets

7 Validation sets

GP Parameters Outline

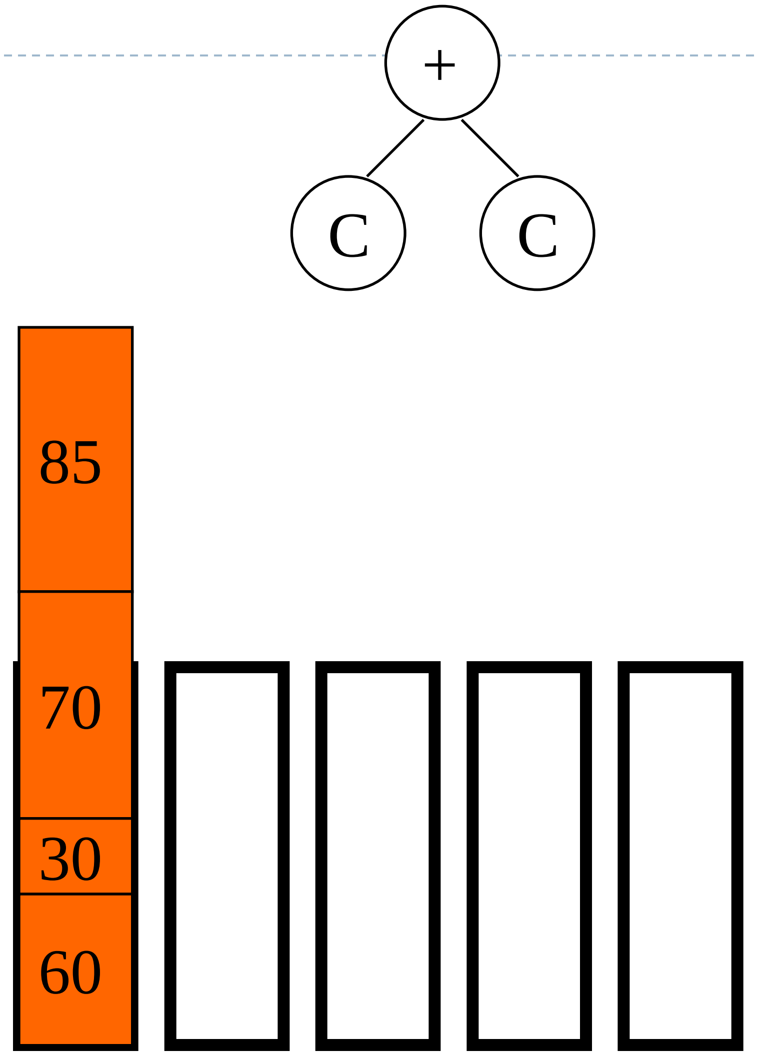
- 50 generations
- 90% crossover
- 10% reproduction
- Functions and terminals:
  - Bin Capacity C
  - Bin Fullness F
  - Piece Size S
  - +, - , *, %, ≤
- 1000 population
- Fitness proportional selection

Evolving Bin Packing Heuristics

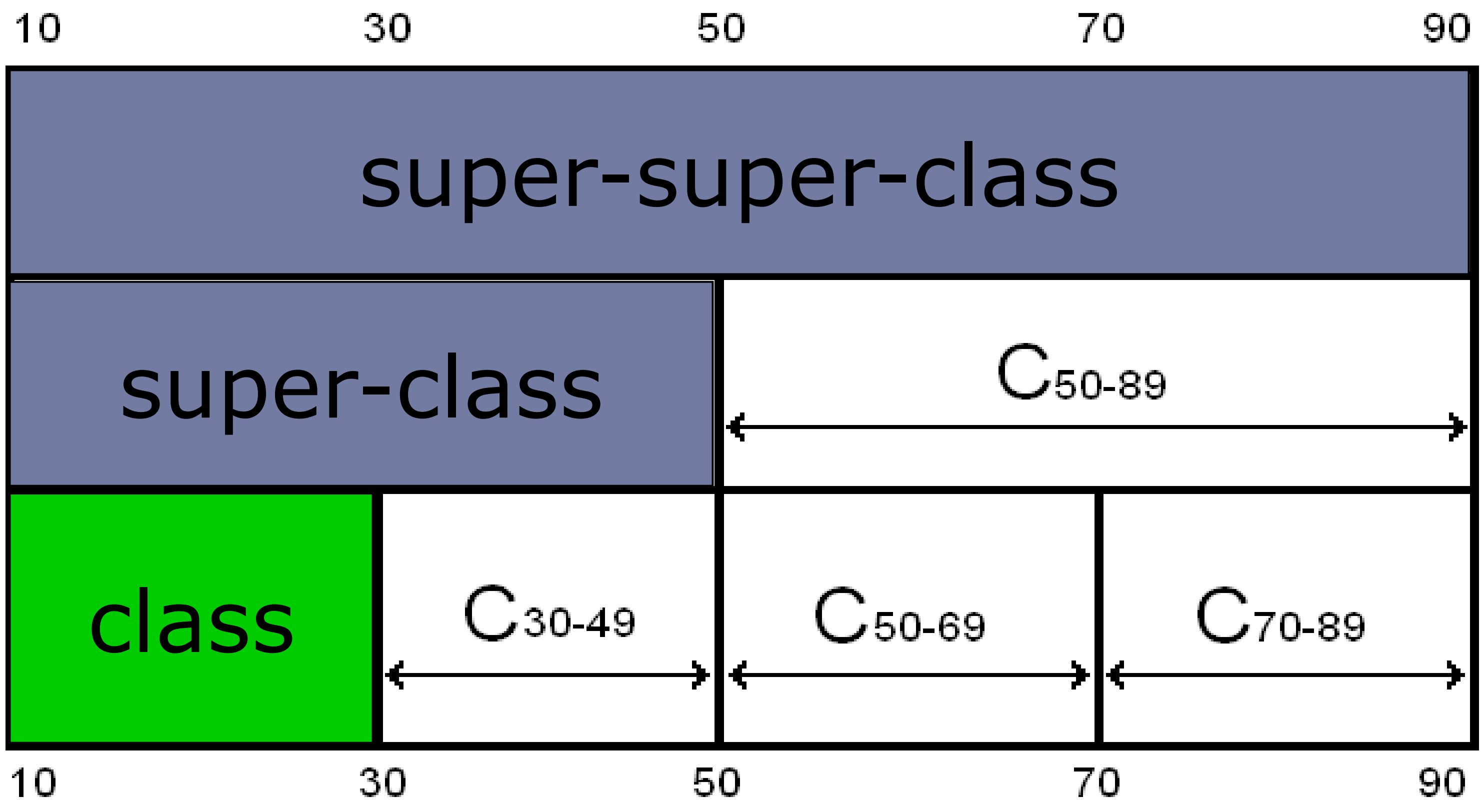


Illegal Heuristics

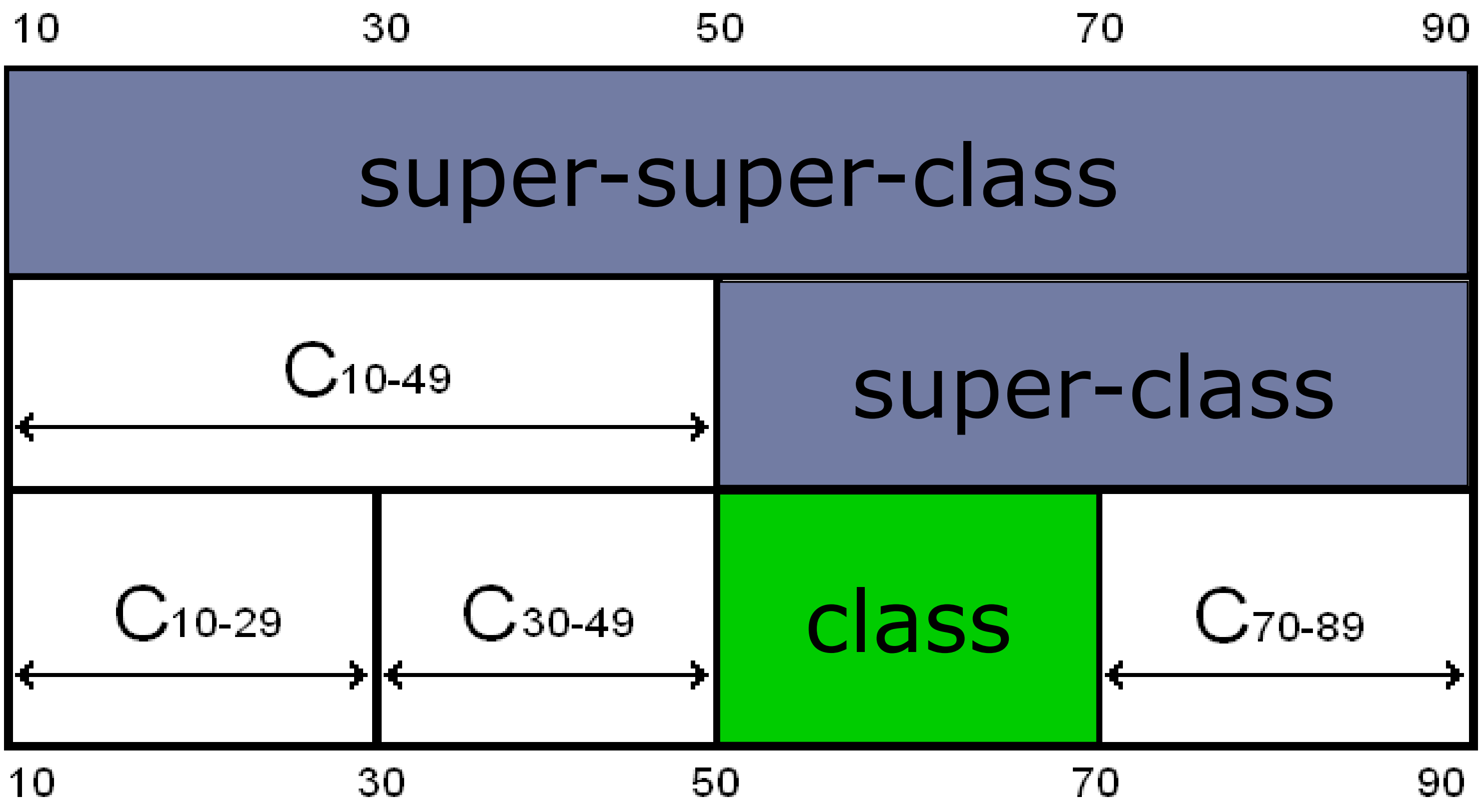
- Permitted
- High penalty
- The system evolves an understanding of the rules



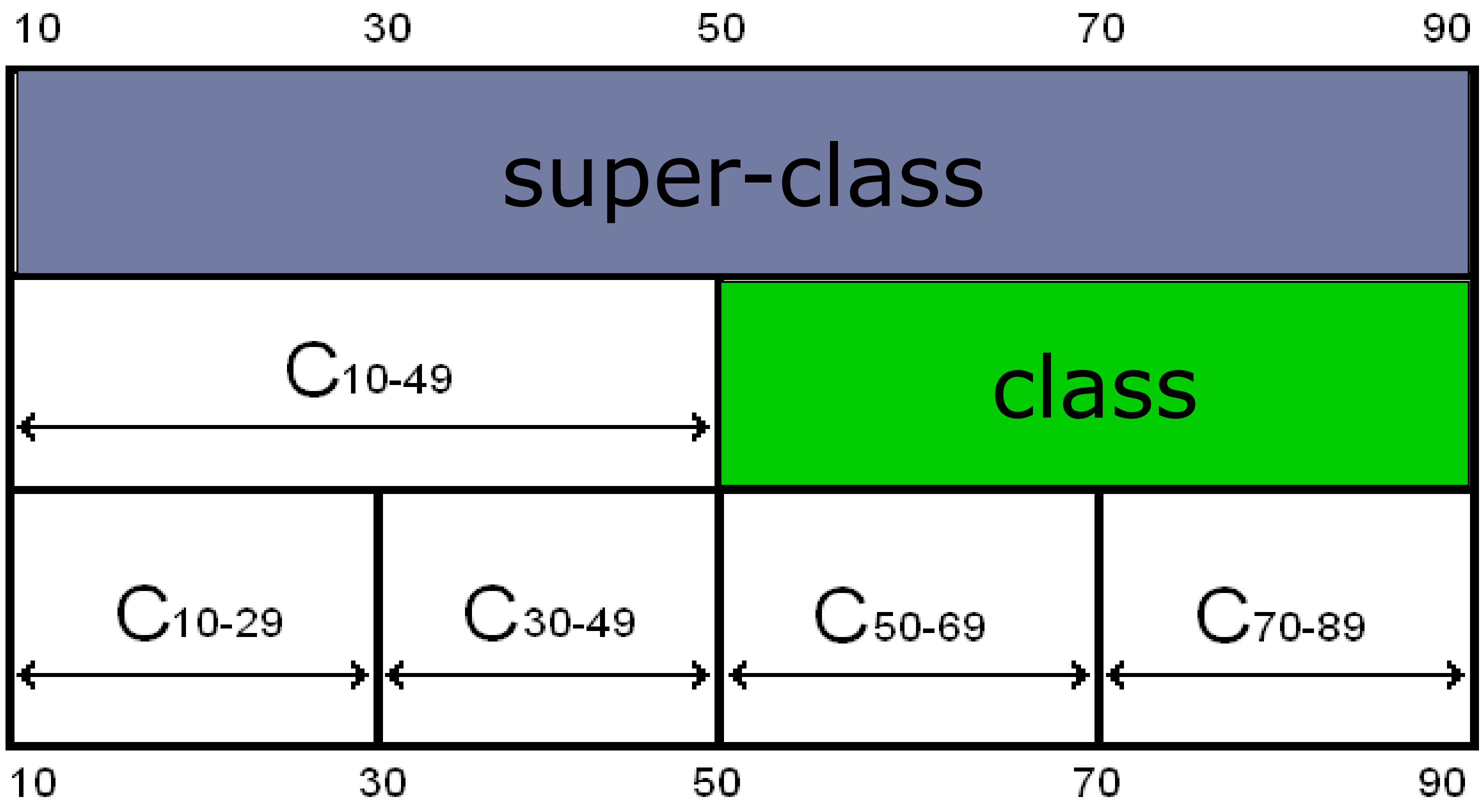
Results - Specialisation of Heuristics



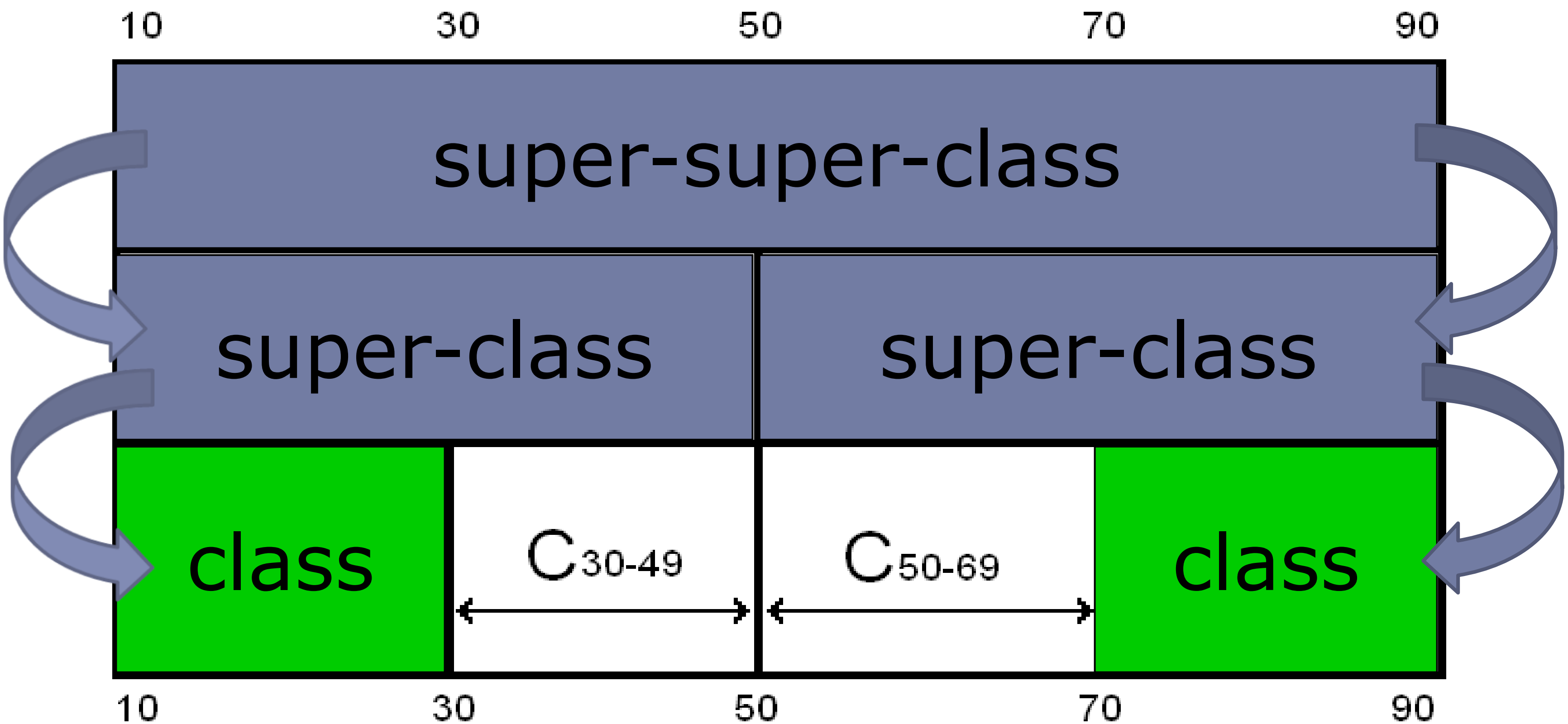
Results - Specialisation of Heuristics



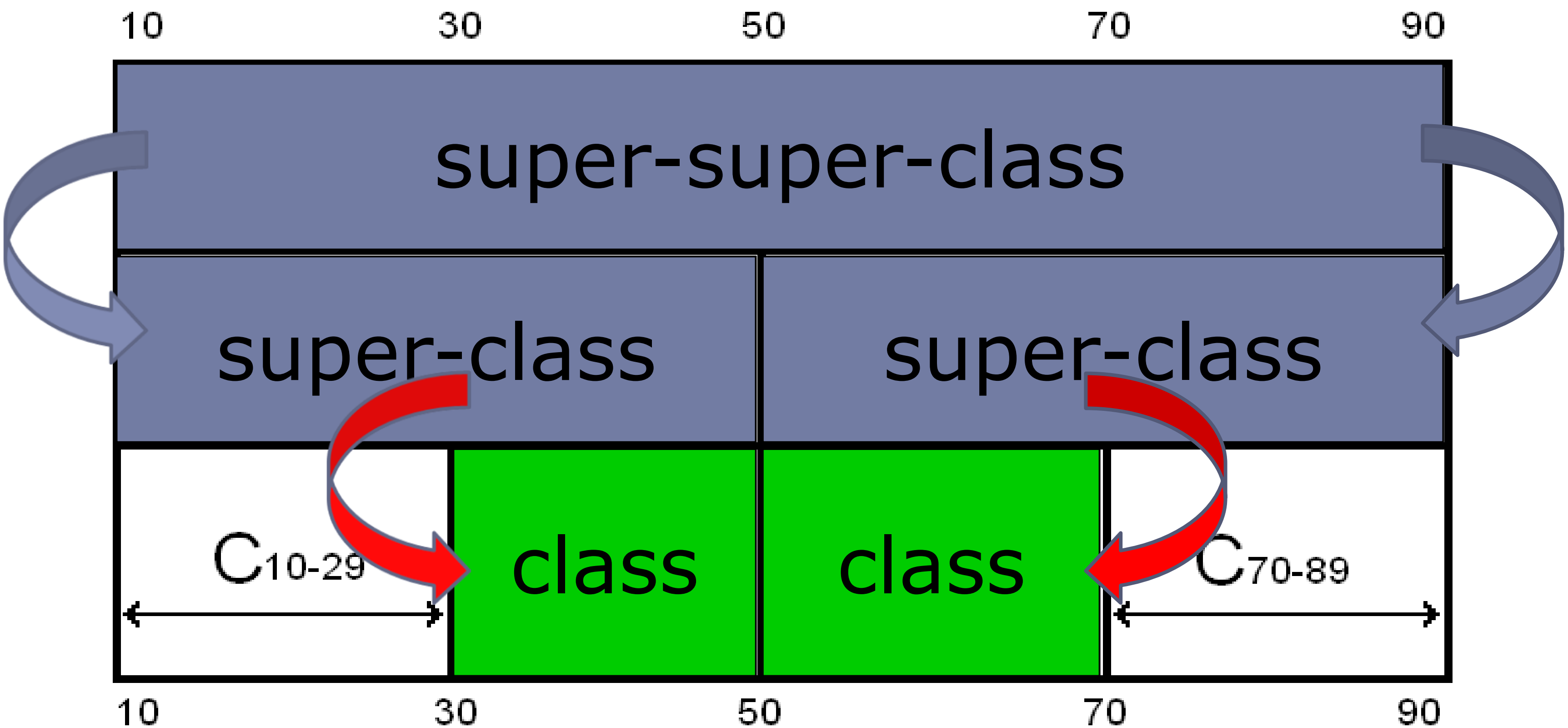
Results - Specialisation of Heuristics



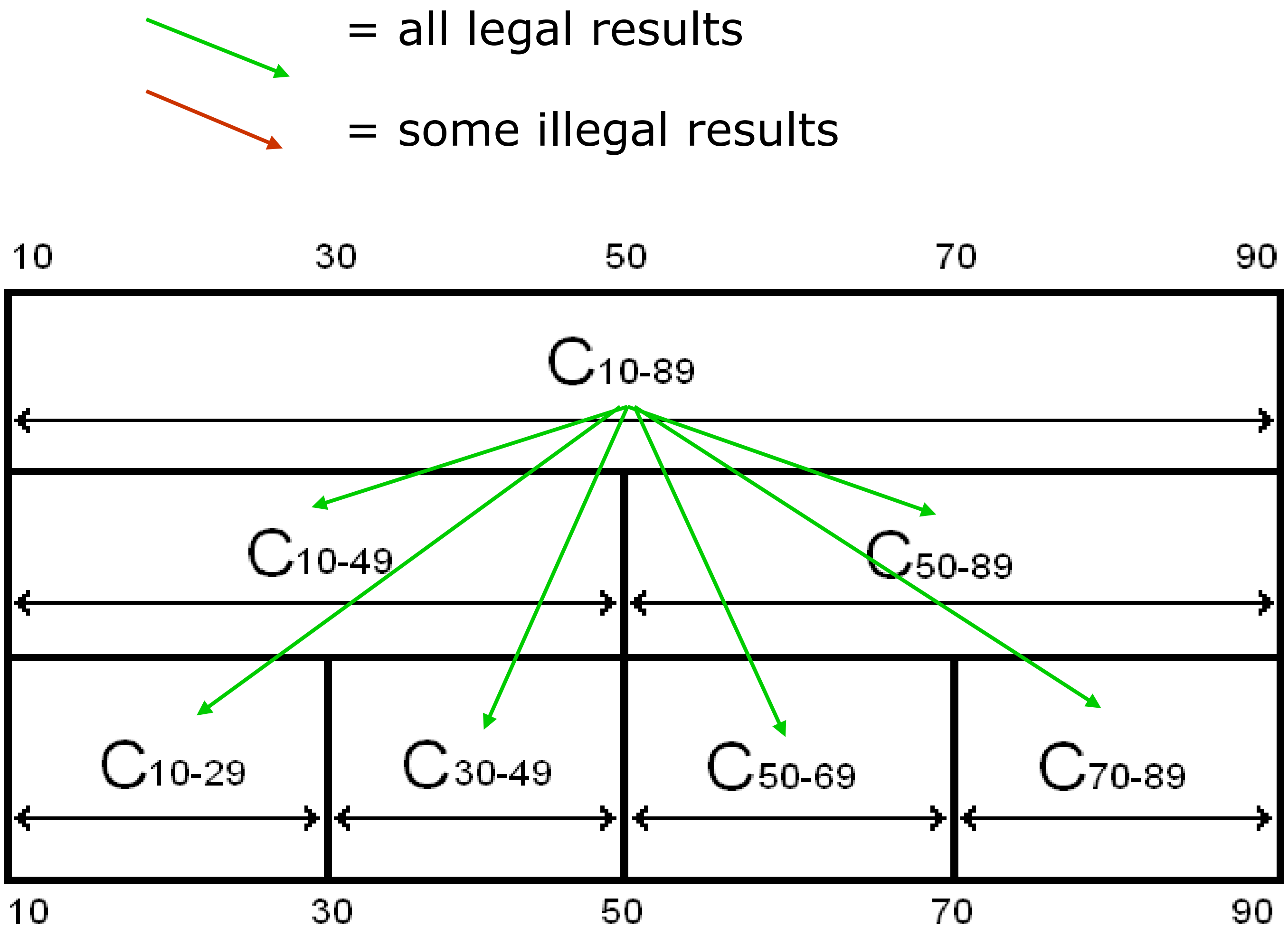
Results - Specialisation of Heuristics



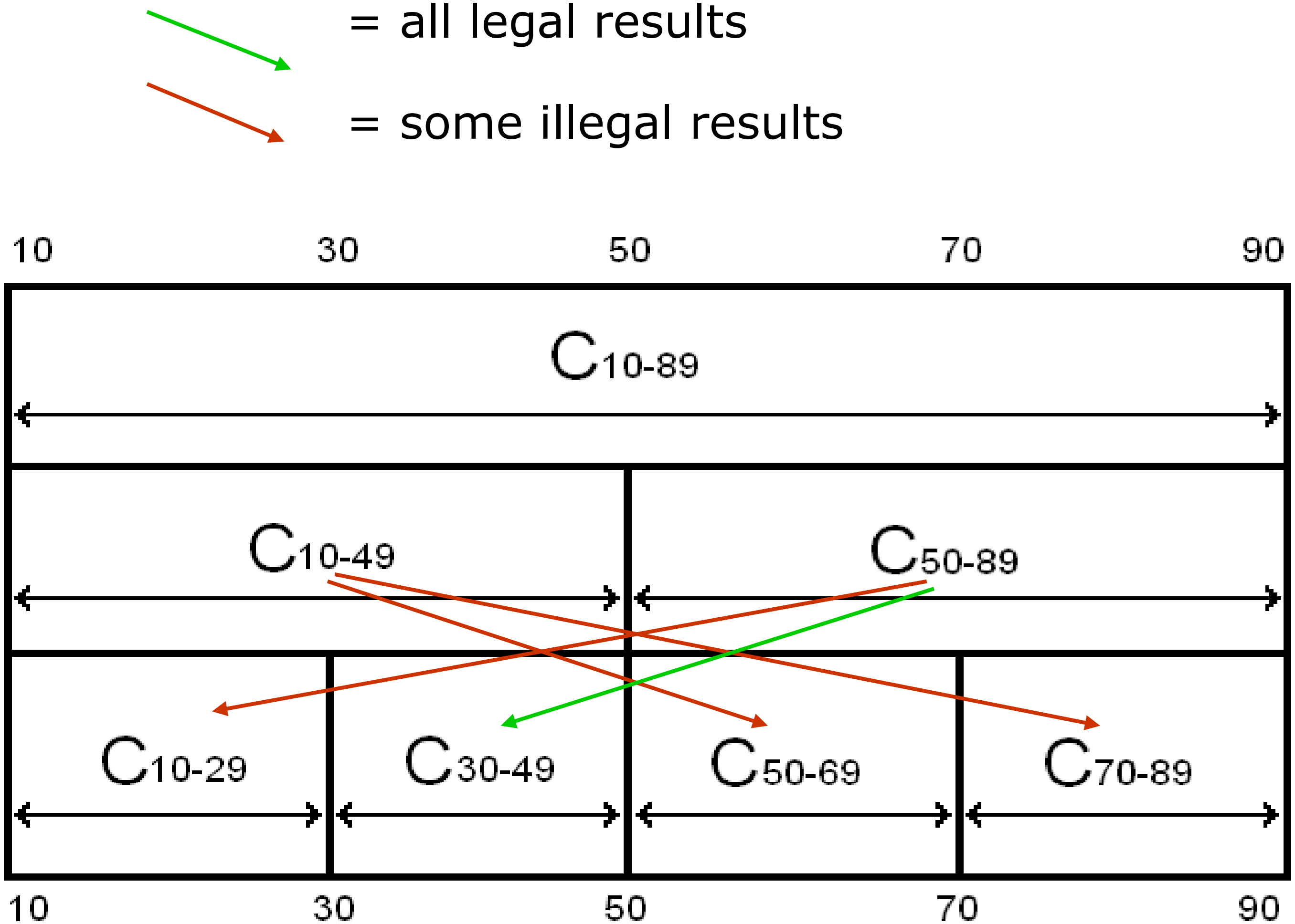
Results - Specialisation of Heuristics



Results - Robustness of Heuristics



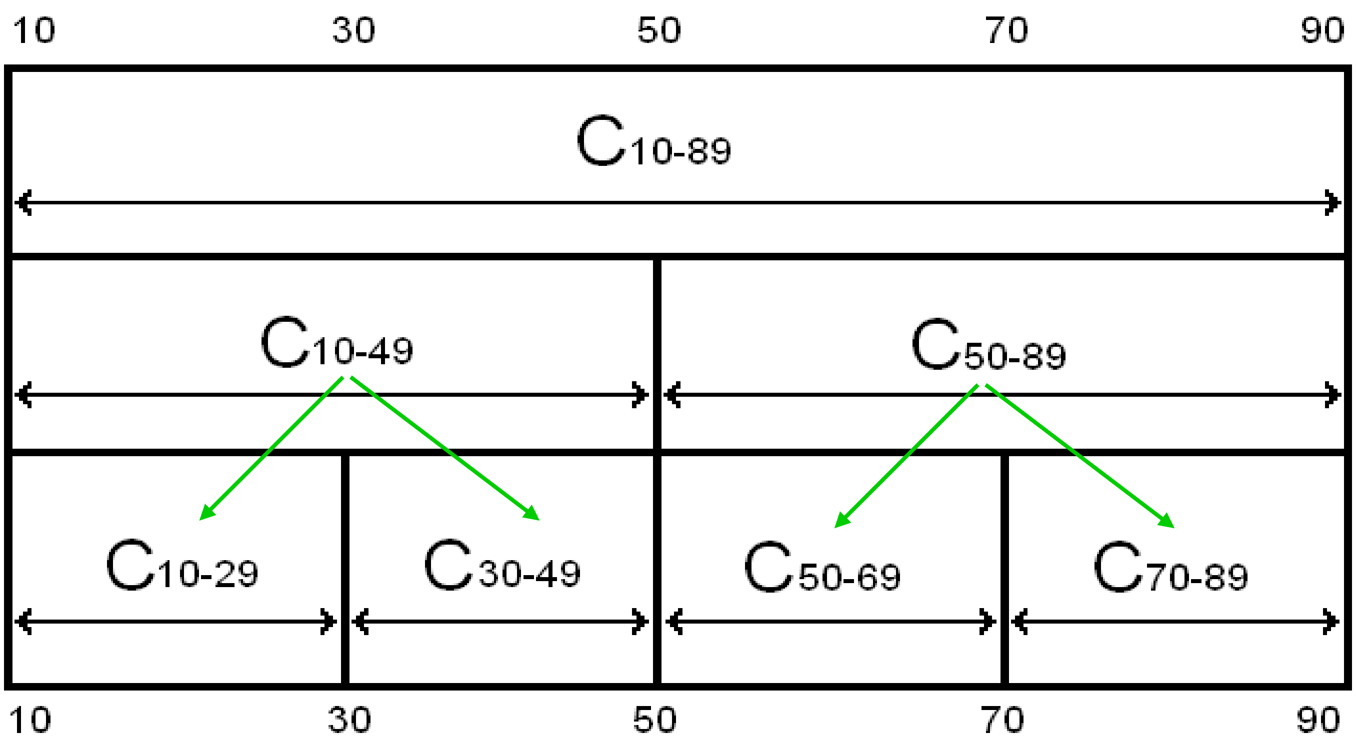
Results - Robustness of Heuristics



Results - Robustness of Heuristics

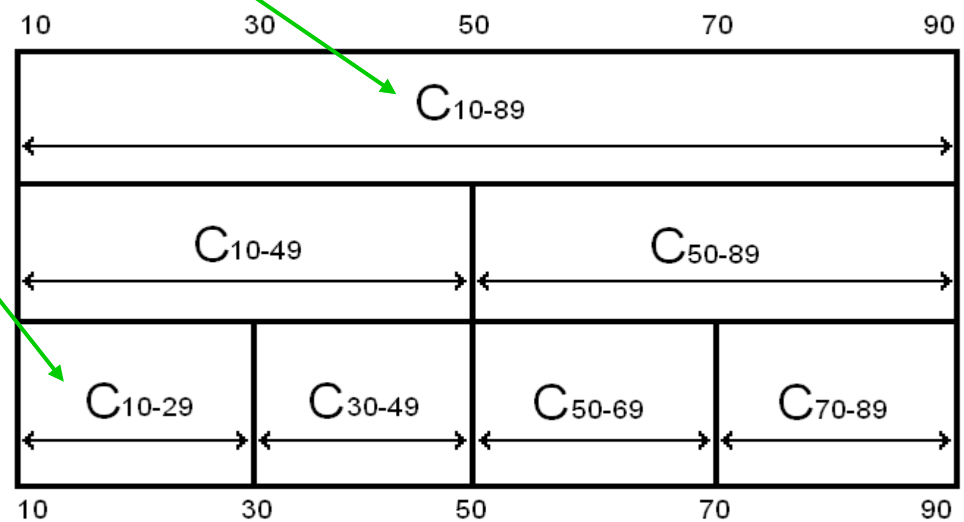
= all legal results

= some illegal results



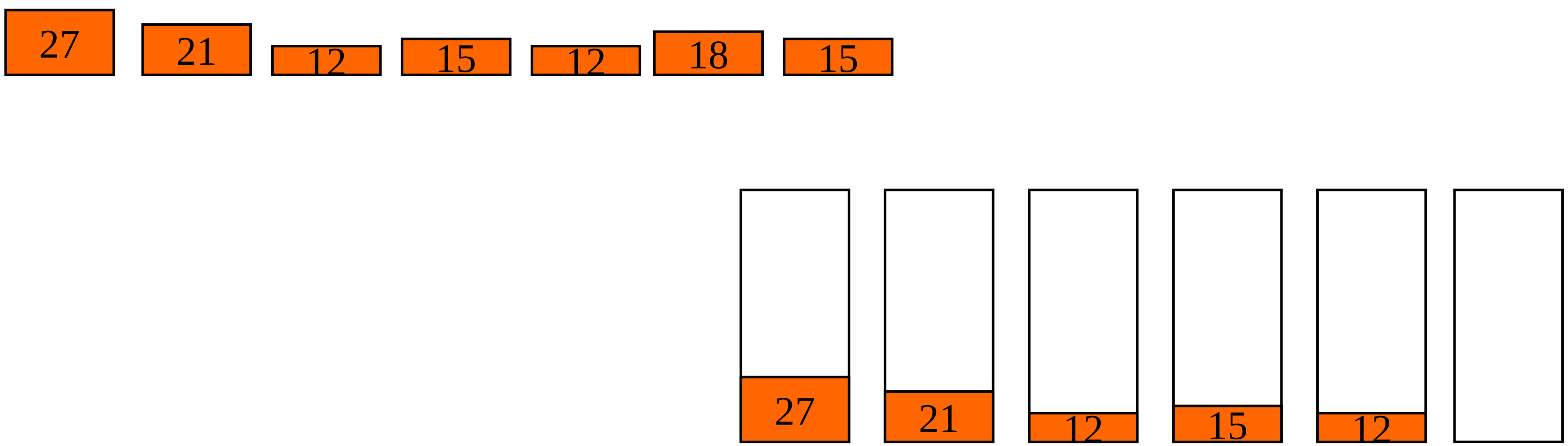
Example

- Heuristic evolved on instances with the widest distribution
- Tested on instances with piece sizes between 10-29
- The heuristic performs very badly, by putting one piece in each bin



Example

- The heuristic always scores the empty bin as the best



$$\frac{2S+F}{S+F} + \frac{C}{((\frac{F}{C}) \leq (2C-F)) + (C-S-F)}$$

Lessons – Case Study 3

- Heuristics can be **specialised** to specific types of sub problem
- Heuristics may not work at all on new instances if they contain different distributions of pieces
- The **training set must be carefully chosen** to ensure it represents every type of problem that the heuristic must solve in the future

Conclusion

- Presented three case studies which highlight different research issues
- Humans will (always?) still have a role in heuristic generation
- The hyper-heuristic automates the process of combining elements that have been **chosen by humans**
- Our role moves from designing heuristics to **designing the search space** in which the best heuristic is likely to exist

Generating Heuristics References

Refs

- Bader-El-Din, M. B. and R. Poli. 2007. Generating SAT local-search heuristics using a GP hyper-heuristic framework. *LNCS 4926. Proceedings of the 8th International Conference on Artificial Evolution* p37-49
- Joc Cing Tay and Nhu Binh Ho. 2008. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers and Industrial Engineering* 54(3) p453-473
- Alex S. Fukunaga. 2008. Automated discovery of local search heuristics for satisfiability testing. *Evolutionary Computation* 16(1) p31-61
- Geiger, C., Uzsoy, R., Aytug, H. Rapid Modeling and Discovery of Priority Dispatching Rules: An Autonomous Learning Approach. *Journal of Scheduling* 9(1) p7-34

Hyper-Heuristic Competition

- Cross-Domain Heuristic Search Challenge
- 4 problem domains
  - SAT
  - Bin Packing
  - Flow Shop
  - Personnel Scheduling
- Heuristics are provided for each
- **You do not know which domain you are solving**
- **You provide an algorithm which coordinates the use of the heuristics**

References

- Hyper-heuristic bibliography online
- http://www.cs.nott.ac.uk/~gxo/hhbibliography.html
- The Cross-domain Heuristic Search Challenge (CHeSC)
- http://www.asap.cs.nott.ac.uk/chesc2011/