The Genetic and Evolutionary Computation Conference

Automated Heuristic Design

Gabriela Ochoa, Matthew Hyde & Edmund Burke
Automated Scheduling, Optimisation and Planning (ASAP) Group,
School of Computer Science, The University of Nottingham

{gxo, mvh}@cs.nott.ac.uk

GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.

Agenda

- First Section: Introduction
  - General Introduction and Motivation
  - What is a Hyper-heuristic?
  - Classification of Hyper-heuristic Approaches
- Second Section: Heuristic Selection Methodologies
  - Case Study 1: Graph-based Hyper-heuristic
  - Case Study 2: *HyFlex* and Heuristic Selection
  - The Cross-domain Heuristic Search Challenge (CHeSC 2011)
  - Conclusion and Future Work
- Third Section: Heuristic Generation Methodologies
  - Introduction
    - Hyper-heuristic Definition
    - What's the Point?
  - Case Study 1: SAT
  - Case Study 2: Flow Shop
  - Case Study 3: Bin Packing
  - Conclusion

Introduction

- Search and optimization problems are everywhere, and search algorithms are getting increasingly powerful
- They are also getting increasingly complex
- Only autonomous self-managed systems that provide high-level abstractions can turn search algorithms into widely used methodologies
- Research goal: software systems able to automatically tune, configure, or even generate and design optimisation algorithms and search heuristics.

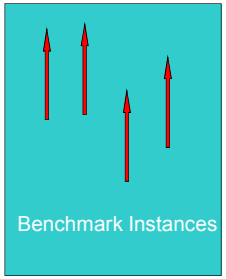
Introduction

- Several approaches to automated heuristic design
  - *Offline* approaches
    - Automated algorithm configuration
    - Meta-learning
    - Performance prediction
  - *Online* approaches
    - Adaptive memetic algorithms
    - Adaptive operator selection
    - Parameter control in evolutionary algorithms
    - Adaptive and self-adaptive search algorithms
    - Reactive search
    - Algorithm portfolios
  - Intelligent optimisation (offline and online)
  - Hyper-heuristics (offline and online)

Motivation

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

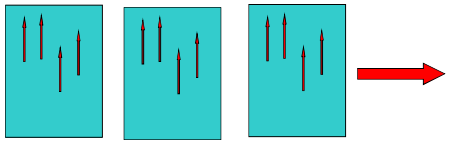


e.g. Exam Timetabling

Motivation

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

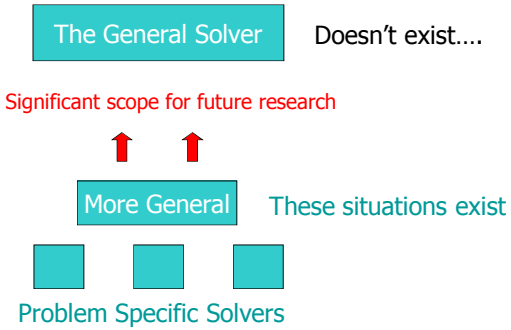
Motivation

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

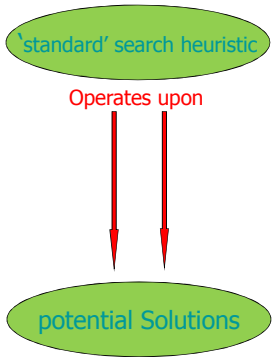
Research challenges

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

Motivation



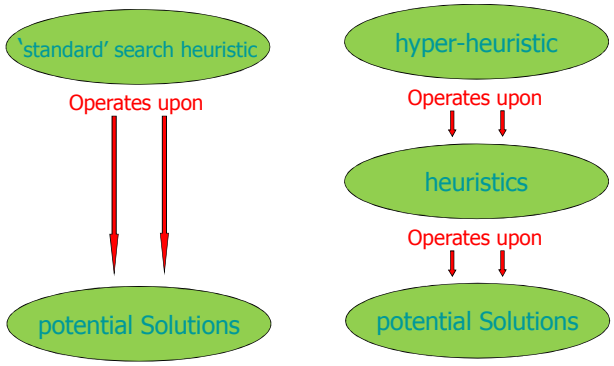
What is a Hyper-heuristic?



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, pp.449-468.

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)

Classification of hyper-heuristics

Search paradigms

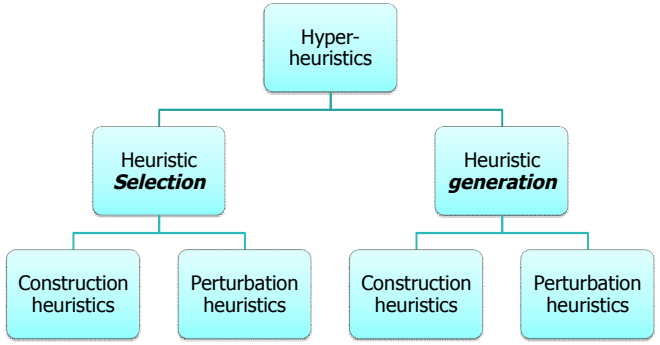
Perturbation

- 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)

Hyper-heuristics
- Online learning
- Offline learning
- No 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

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)

The Genetic and Evolutionary Computation Conference

Section2: Heuristic Selection Methodologies

Case Study 1: A constructive Hyper-heuristic

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

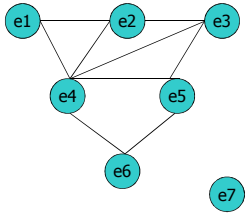
E.K.Burke, B.McCollum, A.Meisels, S.Petrovic & R.Qu. A Graph-Based Hyper Heuristic for Educational Timetabling Problems. EJOR, 176: 177-192, 2007.

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

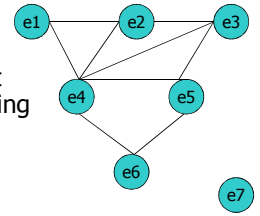
- How can we represent/model this problem?
  - There are 7 exams, e1 ~ e7
  - 5 students taking different exams
    - s1: e1, e2, e4
    - s2: e2, e3, e4
    - s3: e3, e4, e5
    - s4: e4, e5, e6
    - s5: e7
  - let's ignore rooms at the moment



Examination timetabling

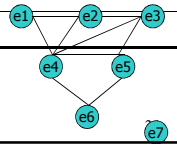
Can be modelled as graph colouring problems

- 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

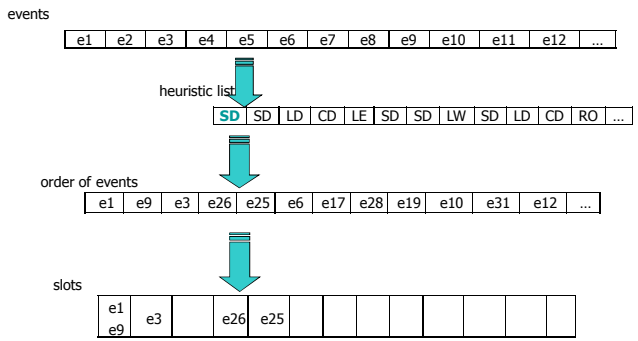


Graph-based hyper-heuristics

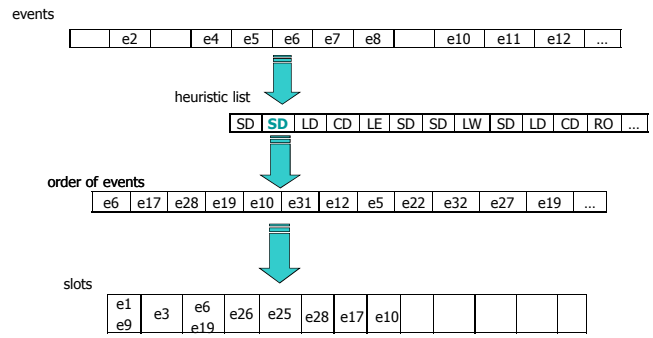
Graph Heuristics	Ordering strategies
Largest degree (LD)	Number of clashed events
Largest weighted degree (LW)	LD with number of common students
Saturation degree (SD)	Number of valid remaining time periods
Largest enrolment (LE)	Number of students
Colour degree (CD)	Number of clashed event that are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



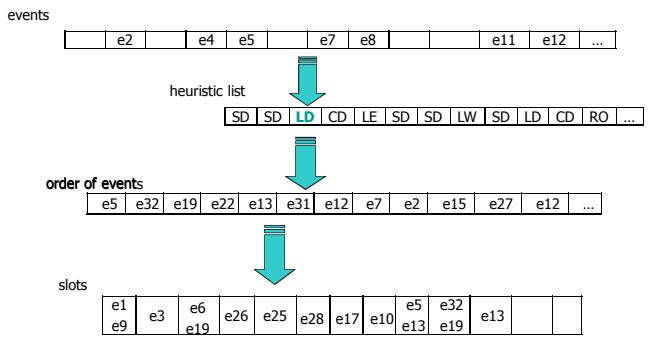
Graph-based hyper-heuristics



Graph-based hyper-heuristics



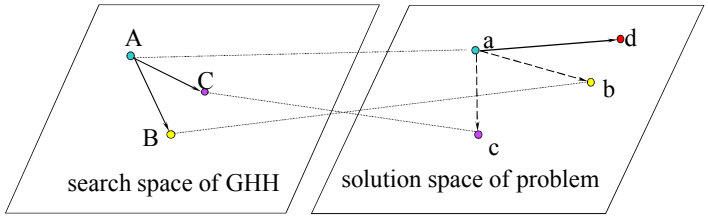
Graph-based hyper-heuristics



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

The Genetic and Evolutionary Computation Conference

Heuristic Selection Methodologies

Case Study 2: HyFlex and automated heuristic selection

Hyper-heuristics Research Challenge

Challenge

- Can we develop the ability to automatically work well on different problems?
- Raising the level of generality
- Develop search methodologies that are more generally applicable

However …

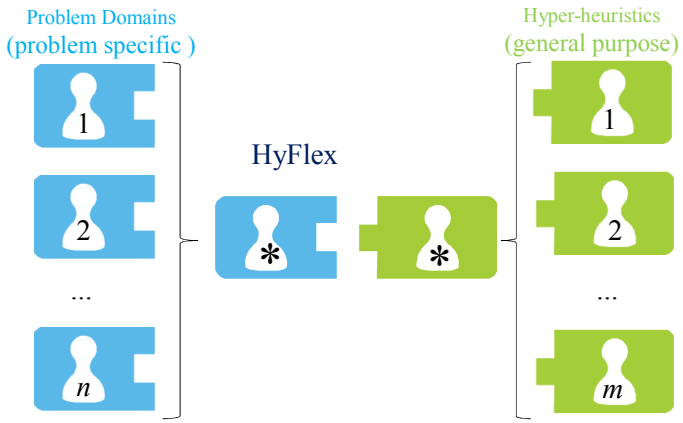
Current hyper-heuristic research

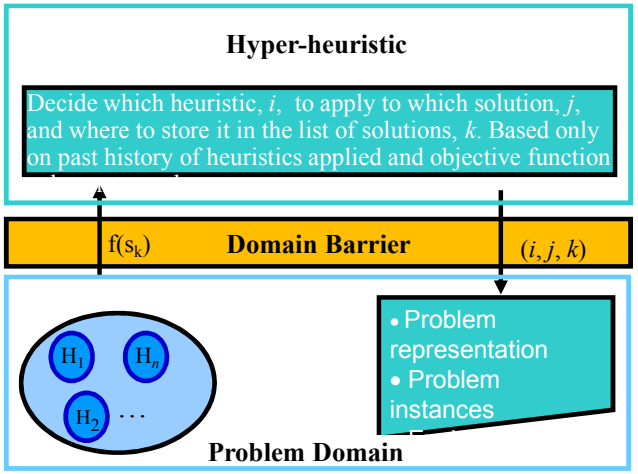
- Papers deal with very few problems: sometimes 2, rarely 3, … mostly only 1!
- **Question**: Can we produce a benchmark to test the generality of heuristic search algorithms?

HyFlex (Hyper-heuristics Flexible framework)

- A software framework (problem library) for designing and evaluating general-purpose search algorithms
- Provides the *problem-specific* components
- Efforts focused on designing high-level strategies

HyFlex: re-use and Interchange

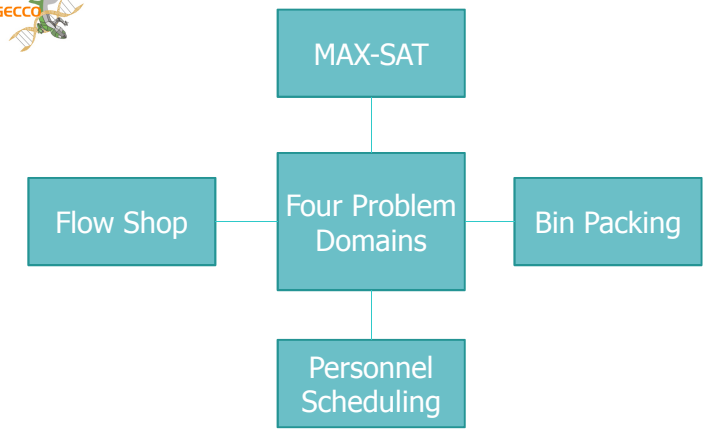




HH fremework:(Cowling P., Kendall G. and Soubeiga, 2000, 2001), (E. K. Burke et al., 2003)
Extension: J. Woodward, A. J. Parkes, G. Ochoa, A Mathematical Framework for Hyper-heuristics. PPSN Hyper-heuristics Workshop. 2008

Overview of the problem domain modules

1. A routine to initialise (randomised) solutions
2. A set of heuristics to modify solutions
   a. Mutational: makes a random modification
   b. Ruin-recreate: partially destroy a solution and rebuild it using a constructive procedure
   c. Local-search: iterative procedures searching on the neighbourhood of solutions
   d. Crossover: takes parent solutions and produce offspring solution
3. A set of interesting instances, that can be easily loaded (`LoadInstance(i)`)
4. A population or list of solutions

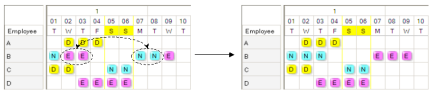


Personnel scheduling

Instances: Wide range of data sets (Industry, Academia, +10 countries)

	BCV-A.12.1	1294	12	5	31	[2, 7]
	BCV-A.12.2	1953	12	5	31	[2, 7]
	ORTEC01	**270**	16	4	31	[4]
	ORTEC02	290	16	4	31	[4]
	GPost	**5**	8	2	28	
	GPost-B	**3**	8	2	28	
	QMC-1	16	19	3	28	
	QMC-2	29	19	3	28	
	Ikegami-2Shift-DATA1	**0**	28	2	30	[9]
	Ikegami-3Shift-DATA1	6	25	3	30	[9]
	Ikegami-3Shift-DATA1.1	13	25	3	30	[9]
	Ikegami-3Shift-DATA1.2	12	25	3	30	[9]
	Millar-2Shift-DATA1	**0**	8	2	14	[9]
	Millar-2Shift-DATA1.1	**0**	8	2	14	[9]
	Valouxis-1	**20**	16	3	28	[13]

Low level heuristics: 12, different types. LS based on new, horizontal and vertical moves



Horizontal swap: move shifts in single employee's work pattern

HyFlex Hyper-heuristics

- Access to interesting problem domains and instance data
- Rich variety of low-level heuristics
- Example: Adaptive Iterated Local Search
  - On-line learning mechanisms for intelligently selecting the mutation operation in the perturbation phase
    - Choice function
    - Extreme value based adaptive operator selection
  - Good overall performance across the four test domains
  - Note: additional slides will be added at the presentation

CHeSC 2011 CROSS-DOMAIN HEURISTIC SEARCH CHALLENGE

asap automated scheduling optimisation & planning research

http://www.asap.cs.nott.ac.uk/chesc2011

The University of Nottingham

APTIASOLUTIONS

PATAT 2010

EVENTMAP Staff Roster Solutions

Queen's University Belfast

CARDIFF UNIVERSITY PRIFYSGOL CAERDYDD

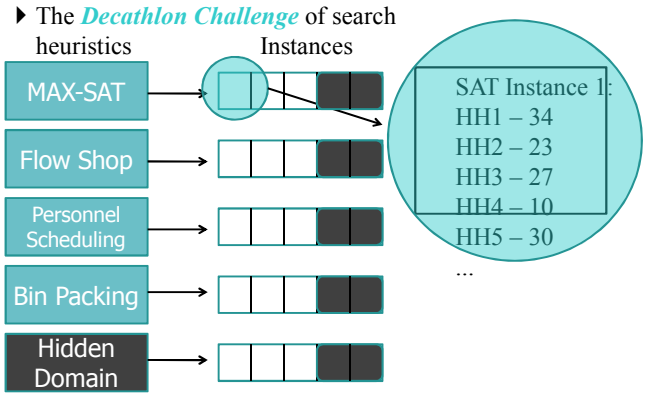
EPSRC Engineering and Physical Sciences Research Council

ÉCOLE POLYTECHNIQUE MONTRÉAL

The Cross-domain Heuristic Search Challenge (CHeSC 2011)

Description of the challenge

- The *Decathlon Challenge* of search heuristics



Conclusions of 1st Section

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

References: Hyper-heuristics

- E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. Woodward (2010). A Classification of Hyper-heuristics Approaches, *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, M. Gendreau and J-Y Potvin (Eds.), Springer, pp.449-468.
- E. K. Burke, B. McCollum, A. Meisels, S. Petrovic & R. Qu. A Graph-Based Hyper Heuristic for Educational Timetabling Problems. *European Journal of Operational Research*, 176: 177-192, 2007.
- E. K. Burke, M. Gendreau G. Ochoa, J. Walker, Adaptive Iterated Local Search for Cross-domain Optimisation. *Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2011)*, ACM
- D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers and Operations Research*, 34:2403– 2435, 2007.
- P. Ross, P. (2005) Hyper-heuristics, Chapter 17 in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Methodologies* (Eds. E.K.Burke and G.Kendall), Springer, 529–556.

References : Automated Heuristic Design

This a small sample of books, survey papers, and other journal papers

- R. Battiti, M. Brunato, F. Mascia (2008) *Reactive Search and Intelligent Optimization*, Operations Research/Computer Science Interfaces Series, Vol. 45, Springer.
- M. Birattari (2009). Tuning Metaheuristics: A machine learning perspective. Studies in Computational Intelligence,197. Springer, Berlin, Germany.
- A.E. Eiben, Z. Michalewicz, M. Schoenauer, and J.E. Smith (2007) Parameter Control in Evolutionary Algorithms, in (Lobo et al,2007), pp. 19-46.
- A. Fialho, L. Da Costa, M. Schoenauer and M. Analyzing Bandit-based Adaptive Operator Selection Mechanisms. In: Annals of Mathematics and Artificial Intelligence , Springer, 2010.
- F. Hutter, h. Hoos H, Leyton-Brown K, Stutzle T (2009) Paramils: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research (JAIR),* 36:267-306.
- F.G. Lobo, C.F. Lima, and Z. Michalewicz (eds.), (2007) *Parameter Setting in Evolutionary Algorithms*, Studies in Computational Intelligence, Springer.
- Y.S. Ong, M.H Lim, N. Zhu, K.W. Wong (2006) Classification of adaptive memetic algo rithms: a comparative study. *IEEE Transactions on Systems, Man, and Cybernetics*, Part B 36(1):141-152

The Genetic and Evolutionary Computation Conference

Section 3
Heuristic Generation Methodologies

Outline

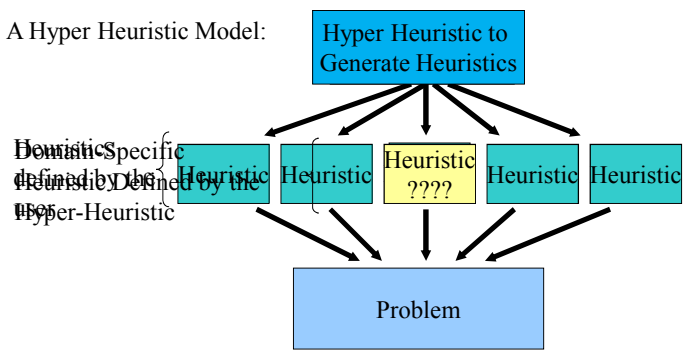
- Introduction to this section
  - Hyper-Heuristic Definition
  - What's the Point?
- Case Study 1: SAT
- Case Study 2: Flow Shop
- Case Study 3: Bin Packing
- Conclusion

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:



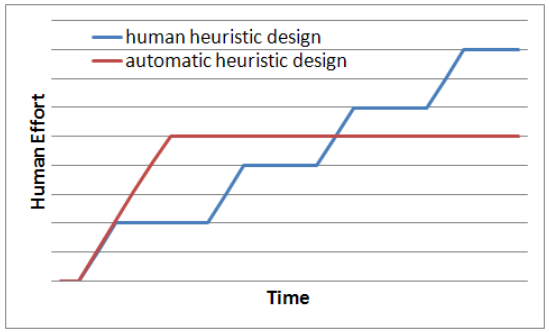
Heuristics defined by the Hyper-Heuristic

Domain-Specific Heuristic Defined by the user

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?



The Genetic and Evolutionary Computation Conference

Heuristic Generation Methodologies

Case Study 1

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

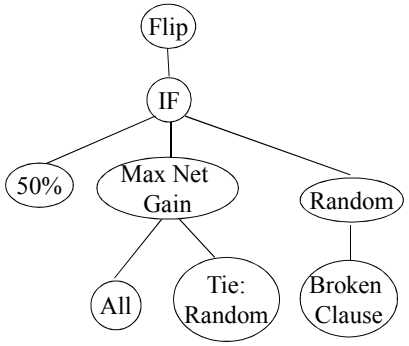
- Make a boolean expression true
- (¬A or B or C) AND (B or ¬C or E) AND (¬B or A or ¬D) AND (…) AND (…) …
- Hundreds/thousands of variables and clauses
- Local search heuristics iteratively choose a variable to flip.

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.



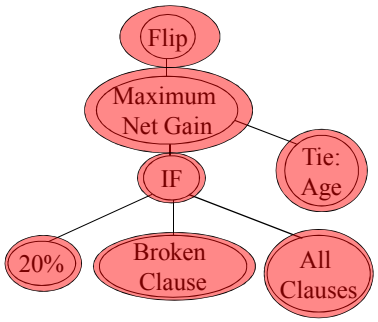
Evolving New SAT Heuristics

- 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 1
         | MAX_SCR 1 | MAX_SCR 1, op
         | IFV prob, v, v
         | MIN_SCR 1 | MIN_SCR 1, op
         | MAX_AGE 1 | MAX_AGE 1, op
l     →  ALL | ALL_USC
         | RAND_USC | USC
         | IFL prob, l, l
         | SCR_Z 1 | SCR_Z 1, 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

The Genetic and Evolutionary Computation Conference

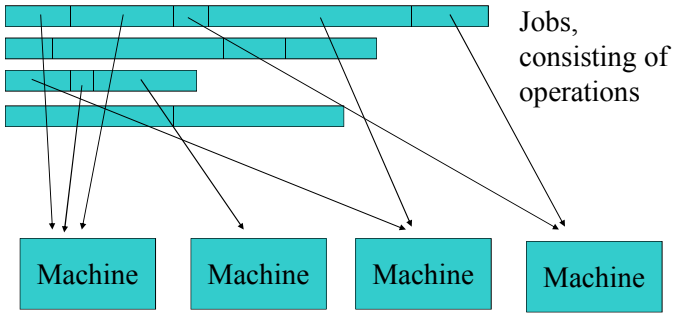
Heuristic Generation Methodologies

Case Study 2

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

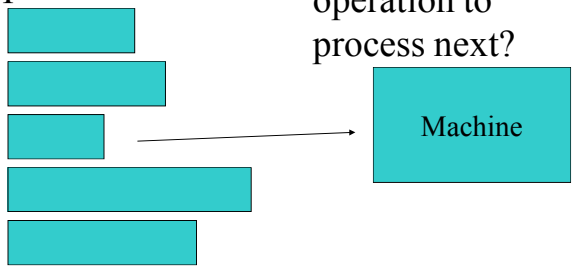
Job-Shop Scheduling



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'

The Genetic and Evolutionary Computation Conference

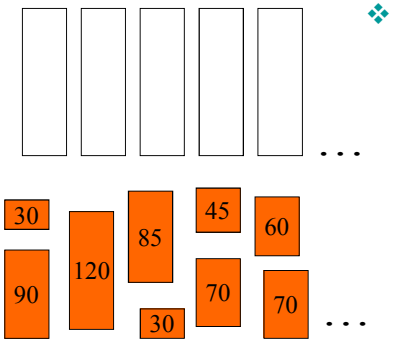
Heuristic Generation Methodologies

Case Study 3

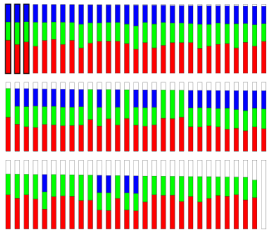
CASE STUDY 3

- One Dimensional Bin Packing
- Burke, Hyde, Kendall, and Woodward 2007
- Heuristics can be evolved that are specialised to different types of problems
- Extended to two dimensional packing heuristics in Burke, Hyde, Kendall, and Woodward 2010

The Bin Packing Problem

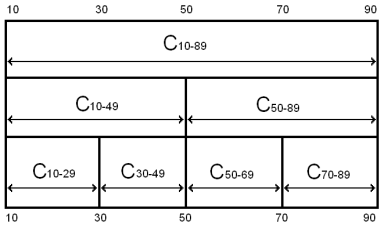


- Pack all the pieces into as few bins as possible



The Bin Packing Problem Set

- Online
- 7 problem classes
- Bin Capacity 150
- 120 items



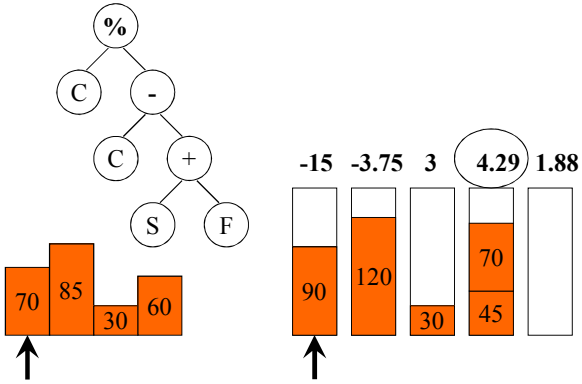
7 Training sets

7 Validation sets

GP Parameters Outline

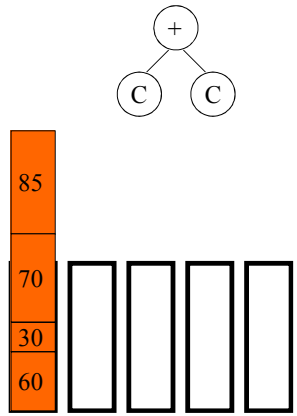
- 50 generations
- 90% crossover
- 10% reproduction
- Functions and terminals:
  - Bin Capacity — C
  - Bin Fullness — F
  - Piece Size — S
  - +, - , *, %, ≤
- 1000 population
- Fitness proportional selection

Evolving Bin Packing Heuristics

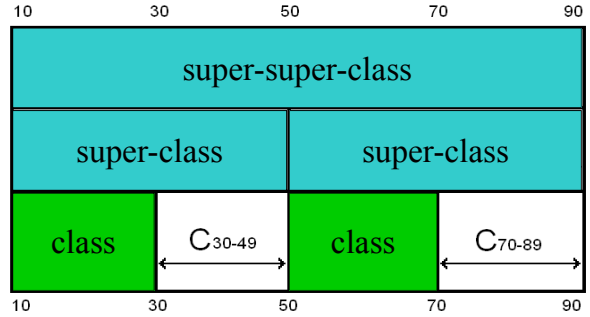


Illegal Heuristics

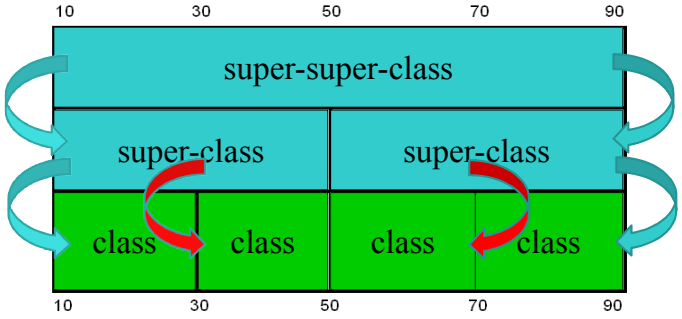
- Permitted
- High penalty
- The system evolves an understanding of the rules



Results - Specialisation of Heuristics



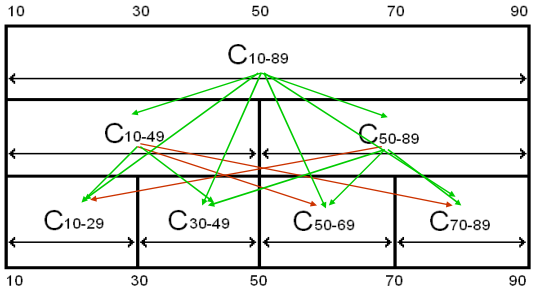
Results - Specialisation of Heuristics



Results - Robustness of Heuristics

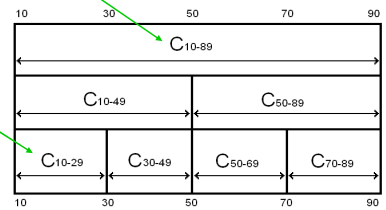
= all legal results

= some illegal results



Example of an evolved heuristic

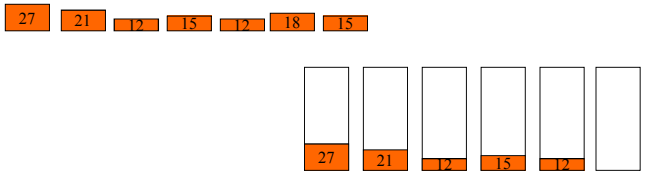
- Heuristic evolved on instances with the widest distribution
- Tested on instances with piece sizes between 10-29



- The heuristic performs very badly, by putting just one piece into each bin

Example of an evolved heuristic

- 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

References

- Burke E. K., Hyde M., and Kendall G., and Woodward J. 2010. "A Genetic Programming Hyper-Heuristic Approach for Evolving Two Dimensional Strip Packing Heuristics". IEEE Transactions on Evolutionary Computation 14(6). pp. 942--958
- Burke E. K., Hyde M., Kendall G., and Woodward J. 2007. "Automatic Heuristic Generation with Genetic Programming: Evolving a Jack-of-all-Trades or a Master of One", *Proceedings of the Genetic and Evolutionary Computation Conference.* London, UK. July 2007. pp. 1559--1565
- 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

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/