NTNU

Norwegian University of Science and Technology

Introduction to Hyper-heuristics

General purpose solvers

Xavier Sánchez Díaz

November 11, 2021

Outline

Optimisation problems are common

Motivation

Here's a couple of combinatorial and parametric optimisation problems for you:

1. Pack a selection of items inside a container with limited capacity, such that the total value of the items is maximal.
2. Assign guests to seats in order to avoid conflicting scenarios at a gala dinner.
3. Visit all REMA 1000s and Bunnpris stores in the city such that the distance travelled is minimal.
4. Fine tune hyperparameters of your machine learning model.

Optimisation problems are common

Motivation

Here's a couple of combinatorial and parametric optimisation problems for you:

1. Pack a selection of items inside a container with limited capacity, such that the total **value** of the items is **maximal**.
2. **Assign** guests to seats in order to avoid **conflicting** scenarios at a gala dinner.
3. **Visit** all REMA 1000s and Bunnpris stores in the city such that the **distance** travelled is **minimal**.
4. **Fine tune** hyperparameters of your machine learning model.

Optimisation problems are hard

Motivation

- **Knapsack Problems**: NP-hard with complexity of $\mathcal{O}(cn)$ via dynamic programming. Intractable when $c \gg n$.
- **Constraint Satisfaction Problems (CSP)**: NP-complete with upper-bound of $\mathcal{O}(d^n)$ for a maximum domain size d and n variables.
- **Vehicle Routing**: NP-complete with upper-bound of $\mathcal{O}(n!)$ for n visited nodes.
- **Parametric optimisation**: Can be transformed into a CSP if considering discrete domains so $\mathcal{O}(d^n)$

Let's try to solve the seating problem!

Heuristics

Three available tables (one with 5 seats and two with 6 seats each). Here are the hard requirements:

- The *Pharaoh's family* (Dad, Mom and two kids) can't be near the Priest. The family kids are intrigued by the Robot so they want to sit on the same table.
- Demis, Vangelis and Mikis (*The Greek*) need to be on the same table.
- Wilson and Akerfelt want to be seated with *The Greek*, but don't want to be in the same table than Parsons.
- Parsons and Gilmour prefer to be with one another.
- Dickinson, Harris, Dio and Summers—the *metalheads*— are easy going. No restrictions, just beer.
- The Priest doesn't want to be near the *metalheads* nor the Robot.

Intelligent search

Heuristics

Congratulations! You just used your brain to make an intelligent decision:

Intelligent search

Heuristics

Congratulations! You just used your brain to make an intelligent decision:

- Assign first the **most-conflicting** individual

Intelligent search

Heuristics

Congratulations! You just used your brain to make an intelligent decision:

- Assign first the **most-conflicting** individual
- Proceed with the *easy* constraints

Intelligent search

Heuristics

Congratulations! You just used your brain to make an intelligent decision:

- Assign first the **most-conflicting** individual
- Proceed with the *easy* constraints
- Assign later the **least-conflicting** people

Intelligent search

Heuristics

Congratulations! You just used your brain to make an intelligent decision:

- Assign first the **most-conflicting** individual
- Proceed with the *easy* constraints
- Assign later the **least-conflicting** people

This is a heuristic!

There's no *better* approach

No Free Lunch

However the same 'intelligent decision' can't be applied to *all* problem instances. Each instance is different and some methods are *better* than others depending on many factors specific to each problem. This is called the No Free Lunch theorem.

Can we characterise problems to decide **which method to use**?

Deciding what to do is hard

The Algorithm Selection Problem

Since no better approach exists, then I should use the best method available depending on the problem. This is NP-Complete.

Do I get better results if I **mix heuristics**? Is there a **common pattern** to all my examples so that I can solve them?

Is there really something to learn from my data?

The Computational Learning Theory

The problem of finding a function which accurately characterises a pattern for a certain kind of problem is NP-hard (as is determining if the next block in a shaft is a Diamond Ore in Minecraft). There's no way to know for sure.

But we can select, and mix, and generate, and modify, and see if we come up with a *generalisation*.

General solvers

Hyper-heuristics

This is the idea behind a hyper-heuristic—a general solver which operates over the **heuristic space** instead of the **solution space**.

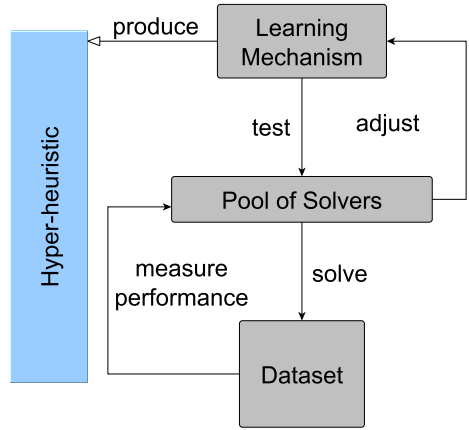
Similar to **algorithm portfolios** and **ensembles**, the hyper-heuristic operates on heuristics and **selects** or **generates** algorithms from existing solvers according to an **objective function**.

The Hyper-heuristic model

Hyper-heuristics

The Learning mechanism could be a **Machine Learning model** or a **metaheuristic**:

- Evolutionary and stochastic algorithms
- Swarm intelligence & Simulated Annealing
- Reinforced & Q-learning
- Neural Networks

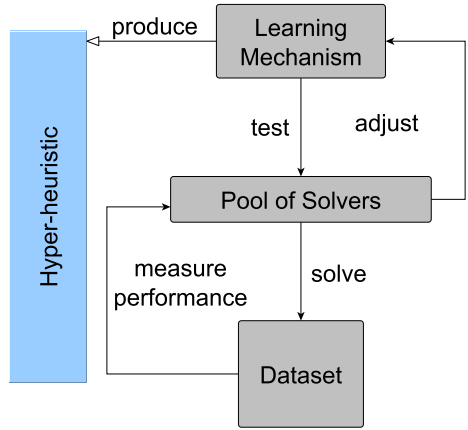


The Hyper-heuristic model

Hyper-heuristics

Both the Pool of available Solvers and the Dataset depend on your **problem domain**:

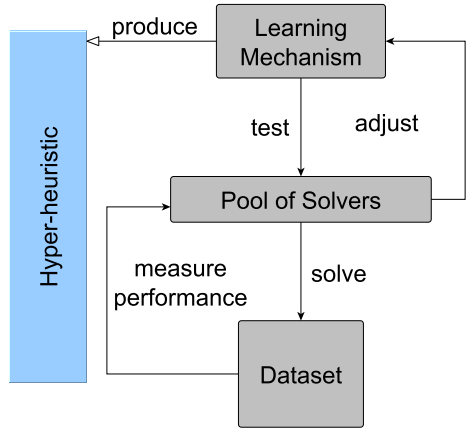
- Knapsack & Bin Packing
- Constraint Satisfaction
- Jobshop Scheduling & Vehicle Routing



The Hyper-heuristic model

Hyper-heuristics

The Performance Measure is usually an **objective function**, per instance or overall (per dataset)

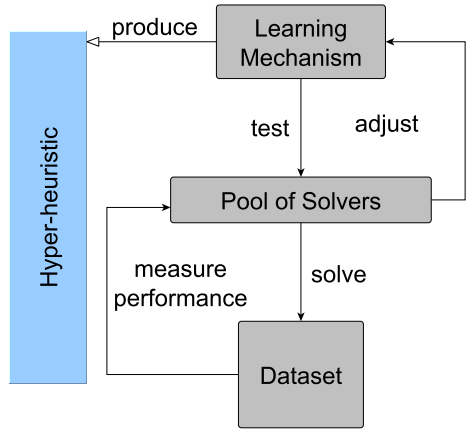


The Hyper-heuristic model

Hyper-heuristics

The output hyper-heuristic could be anything from

- A function (mapping heuristics to instances)
- A vector (or sequence of heuristics)
- A probability distribution (of heuristics)
- A trained model (or pattern description)



It is now a search problem

Hyper-heuristics

Using a **hyper-heuristic** the problem is *solved* by searching for a valid, feasible or a good approximation of the optimal solution.

Therefore, the difficulty now lies on **optimising the search**, which is another hard problem.

NTNU

It is now a search problem

Hyper-heuristics

Using a **hyper-heuristic** the problem is *solved* by searching for a valid, feasible or a good approximation of the optimal solution.

Therefore, the difficulty now lies on **optimising the search**, which is another hard problem.

We do our best.

NTNU

Thank you!

Slides available at

https://saxarona.github.io/project/hhs-intro/

References and Further Reading I

Fawaz Alanazi and Per Kristian Lehre.
Runtime analysis of selection hyper-heuristics with classical learning mechanisms.
In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2515–2523, July 2014.

Fawaz Alanazi and Per Kristian Lehre.
Limits to learning in reinforcement learning hyper-heuristics.
In Francisco Chicano, Bin Hu, and Pablo García-Sánchez, editors, *Evolutionary Computation in Combinatorial Optimization: 16th European Conference, EvoCOP 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings*, pages 170–185. Springer International Publishing, Cham, 2016.

Ruibin Bai, Edmund K. Burke, Michel Gendreau, Graham Kendall, and Barry McCollum.
Memory length in hyper-heuristics: An empirical study.
In *Computational Intelligence in Scheduling, 2007. SCIS '07. IEEE Symposium on*, pages 173–178, April 2007.

Edmund K. Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu.
Hyper-heuristics: a survey of the start of the art.
Journal of the Operational Research Society, 64(12):1695–1724, December 2013.

Edmund K. Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R. Woodward.
A classification of hyper-heuristic approaches.
In *Handbook of Metaheuristics*, pages 449–468, Boston, MA, 2010. Springer US.

Edmund K. Burke, Matthew R. Hyde, Graham Kendall, and John Woodward.
Automating the packing heuristic design process with genetic programming.
Evol. Comput., 20(1):63–89, March 2012.

Edmund K. Burke, Graham Kendall, Jim Newall, Emma Hart, Peter Ross, and Sonia Schulenburg.
Hyper-heuristics: An emerging direction in modern search technology.
Handbook of Metaheuristics, pages 457–474, 2003.

References and Further Reading II

James M. Crawford and Larry D. Auton.
Experimental results on the crossover point in satisfiability problems.
In *AAI-93 Proceedings*, pages 21–27, 1993.

John H. Drake, Matthew Hyde, Khaled Ibrahim, and Ender Özcan.
A genetic programming hyper-heuristic for the multidimensional knapsack problem.
In *11th IEEE International Conference on Cybernetic Intelligent Systems*, Limerick, Ireland, August 2012.

Carla P. Gomes and Bart Selman.
Algorithm portfolio design: Theory vs. practice.
In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, UAI'97, pages 190–197, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

Emma Hart and Kevin Sim.
On the life-long learning capabilities of a NELLI*: A hyper-heuristic optimisation system.
In *Parallel Problem Solving from Nature – PPSN XIII*, volume 8672 of *Lecture Notes in Computer Science*, pages 282–291, September 2014.

Emma Hart and Kevin Sim.
A hyper-heuristic ensemble method for static job-shop scheduling.
Evol. Comput., 24(4):609–635, December 2016.

Emma Hart and Kevin Sim.
On constructing ensembles for combinatorial optimisation.
In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, pages 5–6, New York, NY, USA, 2017. ACM.

References and Further Reading III

Juraj Hromkovič.

Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics.
Springer-Verlag New York, Inc., New York, NY, USA, 2001.

Matthew Hyde.

A Genetic Programming Hyper-Heuristic Approach to Automated Packing.
PhD thesis, University of Nottingham, March 2010.

Per Kristian Lehre and Ender Özcan.

A runtime analysis of simple hyper-heuristics: To mix or not to mix operators.
In *Proceedings of the Twelfth Workshop on Foundations of Genetic Algorithms XII*, FOGA XII '13, pages 97–104, New York, NY, USA, 2013. ACM.

Nuno Lourenço, Francisco B. Pereira, and Ernesto Costa.

The importance of the learning conditions in hyper-heuristics.
In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 1525–1532, New York, NY, USA, 2013. ACM.

Mustafa MısıR, Katja Verbeeck, P. Patrick De-Causmaecker, and Greet Vanden-Berghe.

An investigation on the generality level of selection hyper-heuristics under different empirical conditions.
Applied Soft Computing, 13(7):3335–3353, July 2013.

Eoin O'Mahony, Emmanuel Hebrard, Alan Holland, Conor Nugent, and Barry O'Sullivan.

Using case-based reasoning in an algorithm portfolio for constraint solving.
In *Proceedings of the Irish Artificial Intelligence and Cognitive Sciences Conference*, 2008.

José Carlos Ortiz-Bayliss.

Exploring Hyper-heuristic Approaches for Solving Constraint Satisfaction Problems.
PhD thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey, Monterrey, NL, México, December 2011.

References and Further Reading IV

Ender Özcan, Burak Bilgin, and Emin Erkan Korkmaz.
A comprehensive analysis of hyper-heuristics.
Intell. Data Anal., 12(1):3–23, January 2008.

Smiljana V. Petrovic and Susan L. Epstein.
Tailoring a mixture of search heuristics.
Constraint Programming Letters, 4:15–38, 2008.

Riccardo Poli and Mario Graff.
There is a free lunch for hyper-heuristics, genetic programming and computer scientists.
In *Genetic Programming: 12th European Conference, EuroGP 2009 Tübingen, Germany, April 15-17, 2009 Proceedings*, pages 195–207, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

John R. Rice.
The algorithm selection problem.
Advances in Computers, 15:65–118, 1976.

Tzur Sayag, Shai Fine, and Yishay Mansour.
Combining multiple heuristics.
In *STACS 2006: 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006. Proceedings*, pages 242–253, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

Kevin Sim and Emma Hart.
An improved immune inspired hyper-heuristic for combinatorial optimisation problems.
In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 121–128, New York, NY, USA, 2014. ACM.

Kevin Sim, Emma Hart, and Ben Paechter.
A lifelong learning hyper-heuristic method for bin packing.
Evol. Comput., 23(1):37–67, March 2015.

References and Further Reading V


Kate A. Smith-Miles.
Cross-disciplinary perspectives on meta-learning for algorithm selection.
ACM Comput. Surv., 41(1):6:1–6:25, January 2009.


Kate A. Smith-Miles, Davaatseren Baatar, Brendan Wreford, and Rhyd Lewis.
Towards objective measures of algorithm performance across instance space.
Computers and Operations Research, 45:12–24, May 2014.


Jerry Swan, John Woodward, Ender Özcan, Graham Kendall, and Edmund K. Burke.
Searching the hyper-heuristic design space.
Cognitive Computation, 6(1):66–73, 2013.


Xavier Sánchez-Díaz, José Carlos Ortiz-Bayliss, Ivan Amaya, Jorge M. Cruz-Duarte, Santiago Enrique Conant-Pablos, and Hugo Terashima-Marín.
A feature-independent hyper-heuristic approach for solving the knapsack problem.
Applied Sciences, 11(21), 2021.


Xavier F. C. Sánchez Díaz.
Analysis of a feature-independent hyper-heuristic model for constraint satisfaction and binary knapsack problems.
PhD thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey, 2017.


Richard J. Wallace.
Analysis of heuristic synergies.
In *Recent Advances in Constraints: Joint ERCIM/CoLogNET International Workshop on Constraint Solving and Constraint Logic Programming, CSCLP 2005, Uppsala, Sweden, June 20-22, 2005, Revised Selected and Invited Papers*, pages 73–87, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

References and Further Reading VI


Eric W. Weisstein.
Halting problem.
Internet. http://mathworld.wolfram.com/HaltingProblem.html.
From Mathworld, a Wolfram Web Resource.


Eric W. Weisstein.
Np-complete problem.
Internet. http://mathworld.wolfram.com/NP-CompleteProblem.html.
From Mathworld, a Wolfram Web Resource.


Eric W. Weisstein.
Np-hard problem.
Internet. http://mathworld.wolfram.com/NP-HardProblem.html.
From Mathworld, a Wolfram Web Resource.


Eric W. Weisstein.
Np-problem.
Internet. http://mathworld.wolfram.com/NP-Problem.html.
From Mathworld, a Wolfram Web Resource.


David H. Wolpert and William G. Macready.
No free lunch theorems for optimization.
IEEE Transactions on Evolutionary Computation, 1(1):67–82, April 1997.