University of Sheffield

Evaluating the Performance of Hyper-Heuristics

Ruitao Feng

Supervisor: Pietro Oliveto

A report submitted in partial fulfilment of the requirements
for the degree of MSc. Computer Science with Speech and Language Processing

in the

Department of Computer Science

May 15, 2019

Declaration

Name: Ruitao Feng

Signature: Ruitao Feng

Date: May 15, 2019

Abstract

Many successful applications of randomised search heuristics (such as evolutionary algorithms) to real-world optimisation problems have been reported. Despite these successes, it is still difficult to decide which particular search heuristic is a good choice for the problem at hand, and what parameter settings should be used.

The high level idea behind the field of hyper-heuristics is to overcome this difficulty by evolving the search heuristic for the problem rather than choosing one in advance. The overall goal is to automate the design and the tuning of the algorithm for the optimisation problem, and hence achieve a more generally applicable system.

Many hyper-heuristics decide which search operators to use according to how well they perform during the optimisation process. Hence, learning which operators are most effective for the problem at hand is considered crucial for the success of hyper-heuristics.

The aim of this project is to analyse the performance of widely used hyper-heuristics, for which several successes have been reported, against some recently proposed new hyper-heuristics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Every corner in engineering exists combinatorial optimisation problem, scheduling, logistics, supply chain designing and so on. It also has applications in several field, including auction theory, machine learning and artifical intelligence. The goal behinds those application is to find the optima from a finite set of objects. However, many combinatorial optimisation problem are NP-hard and cannot be solved by exhaustive search. That is, the problem cannot be solved by a polynomial algorithm without giving polynomial algorithms. Although it has not been proven that no polynomial-time algorithm could solve NP-hard problems, they are consider unsolvable in practice. In this case, the approaches to get the best result that considered close enough with the global optima in polynomial time were investigated, heuristics, in other words.

Although heuristics algorithms have achieved many successful application in solving real-world computational search problem, there are still some difficulties to apply them to a new encountered problem:

1. Select which particular search heuristic should be used.
2. What parameter setting should be applied.

As a result, hyper-heuristics found a place to overcome those difficulties. Seen as a high-level methodology, the hyper-heuristics automatically provides an adequate combination of the provided components to solve the given problem filed efficiently.

1.1 Aims and Objectives

The objective of this project is to analyse the performance of widely used hyper-heuristics. we will present a hyper-heuristics framework and several benchmark functions as problem field to evaluate their performance.

1.2 Overview of the Report

The rest of the paper is organized as follows. As the readers may not be experts in combinatorial optimization and hyper-heuristics, we will start with some basic notions. In next section we will formally present a related technology survey, the problem in combinatorial optimization, heuristics and hyper-heuristics. In Section 3, we will analyze the project and presents a simple hyper-heuristics framework and its architecture. Finally, we will present a summarization of this project

Chapter 2

Literature Survey

2.1 Combinatorial optimization

Combinatorial optimization is a topic that consists of finding an optimal object from a finite set of objects [1], optimization problems in which the feasible solutions are discrete and can be expressed using concepts from combinatorics (such as sets, subsets, combinations or permutations) and/or graph theory (such as vertices, edges, cliques, paths, cycles or cuts) [2]. It is a subset of mathematical optimization and a multidisciplinary research area which integrates three major scientific domains: mathematics, theoretical computer science and management. As a result, the concepts of combinatorial optimization could therefore be explained from three aspect [3]:

- On the complexity of combinatorial optimization problems, presenting basics about worst-case and randomized complexity;
- Classical solution methods, presenting the two most-known methods for solving hard combinatorial optimization problems, that are Branch-and-Bound and Dynamic Programming;
- Elements from mathematical programming, presenting fundamentals from mathematical

The application scenarios can be found everywhere in engineering. Unfortunately, many of those problems are proven to be NP-optimization problem which is considered impossible to be solved within polynomial time in practical. Some simple and classic problems are shown as follows:

- Job-shop Scheduling (JSP): given n jobs $J_1, J_2, \cdots, J_n$ of varying processing times, we need to be scheduled on m machines with varying processing power, while trying to minimize the makespan.
- Knapsack problem: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.
- The Traveling Salesman Problem: given the (x, y) positions of N different cities, find the shortest possible path that visits each city exactly once.

2.2 Heuristics & Meta-Heuristics

The exact algorithms, which guarantee to find the global optima, must explicitly examine every combination in the search space unless they could guarantee no need to be examined. As the computational complexity theory had shown that many NP problems were likely to be intractable by exact algorithms. The search space would be extremely large at each iteration in the case of NP optimization problems. That makes exact algorithms impossible to found any solution within polynomial time. However, we often don't need an exact global optima of such problems in practice. The near-optimal solutions are also acceptable if they could be provided within reasonable time. Hence the heuristic algorithm could be defined from 2 perspectives:

- An algorithm that could provide a feasible solution of each instance of a given problem field within reasonable time. The derivation between the feasible solution and global optima cannot be predicted generally.
- A technology that search for the best solution within the acceptable cost, but it can not guarantee to get the optimal solution, or even worse in most cases, can not explain the degree of approximate degree between the result and the optimal solution.

Although heuristics were believed to be the only hope to solve the optimization problem, the acceptance of heuristic algorithms were slow before it and achieved many success in solving the real-world computational search problem. A quote by Fred Glover by late seventies shown the common attitude to find a efficient heuristic:

"Algorithms are conceived in analytic purity in the high citadels of aca- demic research, heuristics are midwifed by expediency in the dark corners of the practitioner's lair. . . and are accorded lower status. [4]"

The early research on heuristics were always built on the intuitive and experiences. Simon and Newell [5] proposed a theory that the design of heuristics algorithms should focus on intuitive, insight and learning. In particular, a new type of heuristics called meta-heuristics attracted considerable research effort. In contrast to heuristics, the meta-heuristics build on simple problem-specific (local) search algorithms and aim at overcoming local optimality through some general-purpose mechanism. Examples include Evolutionary Algorithms (EAs), Ant Colony Optimisation (ACO) and Particle Swarm Optimization, which are all based on principles observed in nature but applied to optimisation. A detailed classfication of meta-heuristics are shown at 2.1.

2.3 Hyper-Heuristics

Both heuristic and meta-heuristic methodology requires problem-specific domain knowledge to design and determine the algorithm that could be applied to the new problems, or even a new instance of similar problems. It is still difficult to decide which of a set of heuristics to apply and what parameter setting would be a good choice to use. The theoretical understanding of which heuristic algorithm work efficient on particular problem and why also does not provide many guidance. Therefore, designing a efficient heuristic algorithm for given problem field is difficult and expensive. This fact proposed a new requirement, that is to

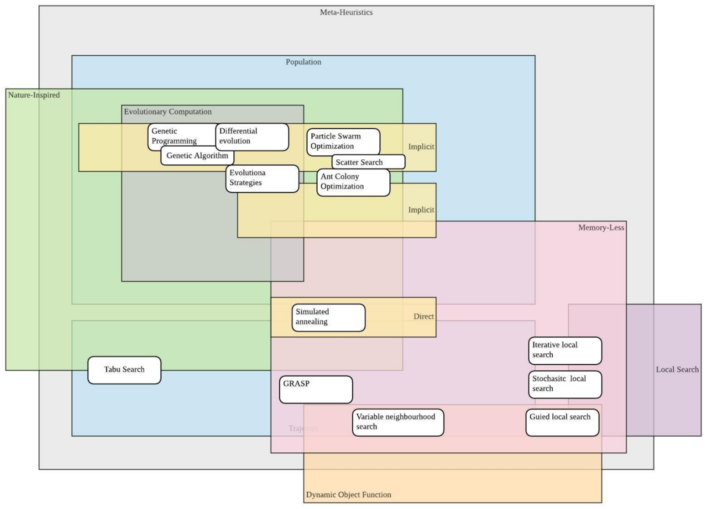


Figure 2.1: *Classification of meta-heuristics*

automate the design and the tuning of heuristic method to solve hard computational search problem [6, 7, 8]. It differentiate from the heuristic algorithms by operating on the search space of heuristics rather than searching directly for the solution to solve specific problem fields or their instance as 2.1.

	Heuristics	Meta-Heuristics	Hyper-Heuristics
Search Space	solutions to the underlying problem	solutions to a problem instance	heuristic algorithms
domain knowledge	Required	Required	Few or not required
	Tabu search	Random based hyper-heuristics	
Typical algorithms	Hill climbing	Genetic algorithms	Greedy based hyper-heuristics
	Greedy Algorithm	Simulated annealing	Hybrid hyper-heuristics

Table 2.1: *Difference between Heuristics, Meta-Heuristics and Hyper-Heuristics*

The term hyper-heuristics is first used by [9] in a peer reviewed conference paper in 2001. The professor Cowling further developed the idea and applied it to scheduling problem in [10, 11, 12]. The hyper-heuristics were considered to be a high-level approach that, select and apply a low-level heuristic from a fixed set of low-level heuristics at each decision point to solve a given problem. However, the idea behind the hyper-heuristics are not new and could be tracked back to the early 60s in many research field such as Computer Science, Operational Research and Artificial Intelligence. They all investigated and developed approaches to solve different problem and could be classified into four types:

- Automated heuristic sequencing
- Automated planning systems
- Automated parameter control in evolutionary algorithms
- Automated parameter control in evolutionary algorithms

And with all the success achieved by applying hyper-heuristics, more and more research interest are attracted to this area. As a result, several tutorial and introduction of hyper-heuristics are published over the last few years, including [13, 14, 15, 16, 17, 18]. In particular, a classification of hyper-heuristics are proposed in [17] that provide framework of investigating hyper-heuristic approaches. Basically, the classification considered two dimensions to classify a hyper-heuristic algorithm according to [17]:

- the nature of search space of the heuristic algorithm.
- the different source of feedback information.

From the perspective of the first dimension, we have (i) selection hyper-heuristic: iteratively choose from a set of heuristics to form a new heuristic algorithm. (ii) generation hyper-heuristic: Generate new heuristic algorithm from the component of existing one. The framework to construct the new solution can be further classified into two types: (i) constructive heuristic: consider a partial solution and improve the performance by adding new components. (ii) perturbation heuristic: consider a complete solution and improve the performance by modifying some of its components.

In recent years, people start to focusing on enchancing the theoretical understanding of hyper-heuristic approaches. Several publication icnluding [19, 20, 21, 22, 23] are released on run time analysis and other foundational studies. Besides, the application area of hyper-heuristics are also investigated such as [24] to use hyper-heuristics to solve real-world optimization problems.

Chapter 3

Analysis

The project aims to evaluate the performance of hyper-heuristics and provide theoretical runtime analysis. Thus, the implementation of our project is divided into 2 part: (i) Heuristics Framework; (ii) and Programming;. We will introduce all the module respectively along with the notation and possible ethical, professional and legal issues in this section.

3.1 Notation

In this paper, we use uppercase letters to denote vectors and lowercase to represent its element. The empty set is written with a empty set symbol $\emptyset$ while other set is denoted by listing its elements inside curly braces: $X = \{v_1, v_1, v_3\}$. Besides, we defined the set of integers $[n] \coloneqq \{1, 2, \cdots, n\}$ and $[0 \cdots n] \coloneqq \{0\} \cup [n]$. Given a vector $v \in R$ and an integer $i \in [n]$, then v_i denotes the i-th element of v. We use $\sim$ to denote "distributed as" e.g. if X is distributed as D we write: $X \sim D$.

3.2 Heuristics Framework

In our project, we will focus on selection hyper-heuristic and use 1 as the framework:

Algorithm 1 Simple Selection Heuristics Framework

```
1: procedure HYPER-HEURISTIC
2:     $x \sim H$                                        ▷ H denotes the search space of solutions
3:     while Termination criteria not satisfied do
4:         $\mathbf{Var} \sim D_p(NO_1, NO_2, \cdots, NO_n)$        ▷ Select a neighbourhood operator
5:         $x' \sim \mathbf{Var}(x)$
6:         $\mathbf{Acc} \sim D_p(AO_1, AO_2, \cdots, AO_n)$        ▷ Select a Acceptance operator
7:         if $\mathbf{ACC}(x, x')$ then $x \leftarrow x'$
8:     end while
9:     return $b$
```

The framework contains the three components: (i) the search space of solutions; (ii) neighbourhood operators; (iii) and acceptance operators. Each component has multiple method and will be introduced in the following subsections.

3.2.1 Bench-marking Functions

Since the neighbourhood operators (also known as variation operators) aims to pick another heuristic rule independently from the set of heuristics and applied on current solution of given problem in each iteration. Therefore, the neighbourhood operators and the set of low-level heuristics, depends on the choice of bench-marking functions. There are several bench-marking functions in the runtime analysis of hyper-heuristics:

1. **ONEMAX** function:

   The **ONEMAX** function is a simple bench-marking function that is also perfect to start with. The function counts the number of bits that valued as '1' in a given bitstring:

   $$\mathbf{ONEMAX} \coloneqq \sum_{i=0}^{n} X_i$$

   Therefore, the seach space of solution could be

   $$\mathbf{UNIF}\{0,1\}^n$$

   And the neighbourhood operators could be like: (i) **FLIPOneBIT** fucntion: randomly flip 1 bit; (ii) and **FLIPTWOBIT** function: randomly flip two bits in the bitstring.

2. **GAPPATH** function:

   The GAPPATH function, as a simple but powerful bench-marking funtion, is a variant of SPI function (short path with increasing values on the path):

   $$\mathbf{GAPPATH}(x) = \begin{cases} \mathbf{ZM}(x) & \mathbf{if}\,\mathbf{RIDGE}(x) \equiv 1(mod3) \\ \mathbf{ZM}(x) + 2n\mathbf{RIDGE}(x) & \mathbf{Otherwise} \end{cases}$$

   Where $\mathbf{ZM}(x) = \sum_{i=1}^{n}(1 - x_i)$ and

   $$\mathbf{RIDGE}(x) = \begin{cases} i & \mathbf{if}\, x = 1^i 0^{n-i} \mathbf{for}\, i \in [0 \cdots n] \\ 0\mathbf{RIDGE}(x) & \mathbf{Otherwise} \end{cases}$$

   The search space of solution and neighbourhood operators is exactly the same with **ONEMAX** function since they both operate on the bitstring.

   The **GAPPATH** function is powerful since it contains gaps where the function values are inferior than the rest of the path. As a result, the algorithm that only use one particular neighbourhood operator will have a infinite expected runtime. Only algorithms that can alternate the operators could found a feasible solution.

.

3.2.2 Acceptance operator

The acceptance operators decided whether to accept the current solution or not. There are two common-used deterministic acceptance operators: (i) **All Moves** (AM), (ii) and **Only Improvement** (OI).. As their name implies, the former operator accept all the solutions regardless of their fitness while the latter one only accept the moves that improved from the previous solution. Other non-deterministic acceptance operators will be further investigated if there is enough time.

3.3 Programming

In order to verify our runtime analysis and other performance criteria, we designed a program to run the hyper-heuristic framework with different parameters. The directory structure of the program would look like 3.1:

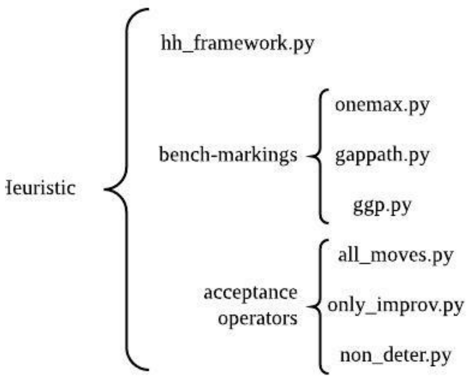


Figure 3.1: *Program Directory Structure*

3.4 Ethical, Professional and Legal Issues associated with project

It is in everyone's interests to promote research ethics, and support the integrity and reputation of research. We checked the objective and production carefully with the code of ethics (also with a exhaustive search) and find no obvious ethical, professional and legal issues associated with our project in all respects.

Chapter 4

Planning

4.1 Schedule

We will use the Scrum as our project management framework and have divied the development cycle into 4 sprints:

- The design sprint: Read the related papers and online courses to gain the fundamental knowledge of hyper-heuristics.
- The development sprint: Complete the hyper-heuristic framemwork and implement a program that could test the runtime of hyper-heuristic with different parameters.
- The validation sprint: We verify our framemwork with our developed program and draw figures to demonstrate our evalution.
- The optimization sprint: After we finished the implementation of our project, investigations on improvements and extensions will be launched especially on add more selection operators and acceptance operators.

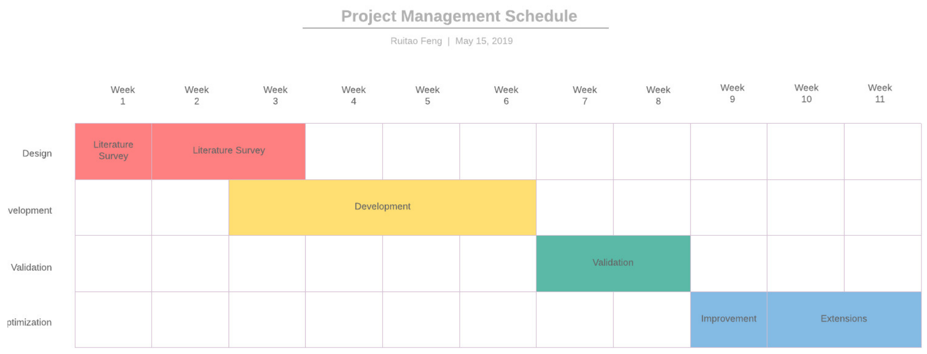
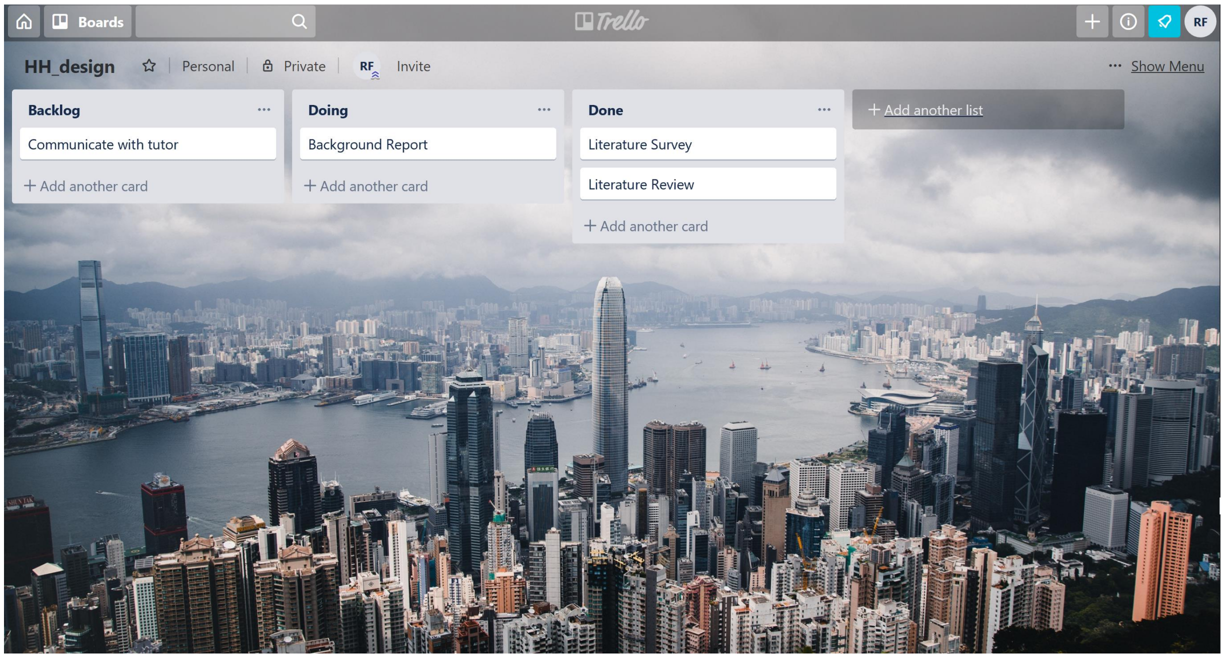


Figure 4.1: *Project Management Schedule*

4.2 Risk Management

The Trello board 4.2 is used for each sprint to organize our work. We will designed every task before the start of each sprint and change the status of each task if finished. The progress of each sprint could be easily visualized so we could flexibly make adjustment or ask for help. Besides, communication with supervisor is included in every sprint so we should make the progress under control with supervisor's experiences. some task in the schedule is optional to our requirement such as the improvements and extensions. We would abort those task if there is not enough time.

Figure 4.2: Trello Taskboard



Backlog	Doing	Done
Communicate with tutor	Background Report	Literature Survey
		Literature Review

Chapter 5

Conclusions

The high-level idea of hyper-heuristic approaches is to found solution on the search space of heuristics rather than directly on the search space of problem fields or problem instances. Hyper-heuristics offers a general framework to design algorithms that ideally can select and generate heuristics adapted to a particular problem instance so we could avoid the difficulty of design appropriate algorithms to every specific conditions. This potential will benefit all field that encountered the optimisation problem.

Our project provided performance evaluation on widely used hyper-heuristics and a set of toolkit to test new hyper-heuristic approach. The research help us understand hyper-heuristics in theoretical–a small step but a necessary one to build more efficient hpyer-heuristic algorithms. Future work would consider to add non-deterministic acceptance operators, more complex bench-marking functions and more sophisticated problem scenarios.

Bibliography

[1] Alexander Schrijver. A first course in combinatorial optimization. *Choice Reviews Online*, 42(03):42–1619–42–1619, 2013.

[2] Adam N. Letchford. What is the meaning of combinatorial optimization?

[3] Vangelis Th. Paschos. *Applications of combinatorial optimization.*

[4] Fred Glover. Heuristics for integer programming using surrogate constraints. *Decision sciences*, 8(1):156–166, 1977.

[5] Herbert A Simon and Allen Newell. Heuristic problem solving: The next advance in operations research. *Operations research*, 6(1):1–10, 1958.

[6] Sam Allen, Edmund K Burke, Matthew Hyde, and Graham Kendall. Evolving reusable 3d packing heuristics with genetic programming. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 931–938. ACM, 2009.

[7] Belarmino Adenso-Diaz and Manuel Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *Operations research*, 54(1):99–114, 2006.

[8] Samad Ahmadi, Rossano Barone, Peter Cheng, Peter Cowling, and Barry McCollum. Perturbation based variable neighbourhood search in heuristic space for examination timetabling problem. *Proceedings of multidisciplinary international scheduling: theory and applications (MISTA 2003), Nottingham*, pages 155–171, 2003.

[9] Kendall G Cowling P and Soubeiga E. A hyperheuristic approach to scheduling a sales summit, 2000.

[10] Peter Cowling, Graham Kendall, and Eric Soubeiga. A parameter-free hyperheuristic for scheduling a sales summit. In *Proceedings of the 4th Metaheuristic International Conference, MIC*, volume 2001, pages 127–131. Citeseer, 2001.

[11] Peter Cowling, Graham Kendall, and Limin Han. An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1185–1190. IEEE, 2002.

[12] Peter Cowling, Graham Kendall, and Eric Soubeiga. Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation. In *Workshops on Applications of Evolutionary Computation*, pages 1–10. Springer, 2002.

[13] Edmund Burke, Graham Kendall, Jim Newall, Emma Hart, Peter Ross, and Sonia Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. In *Handbook of metaheuristics*, pages 457–474. Springer, 2003.

[14] Peter Ross. Hyper-heuristics. In *Search methodologies*, pages 529–556. Springer, 2005.

[15] Konstantin Chakhlevitch and Peter Cowling. Hyperheuristics: recent developments. In *Adaptive and multilevel metaheuristics*, pages 3–29. Springer, 2008.

[16] Edmund K Burke, Mathew R Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan, and John R Woodward. Exploring hyper-heuristic methodologies with genetic programming. In *Computational intelligence*, pages 177–201. Springer, 2009.

[17] Edmund K Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan, and J Woodward. Handbook of metaheuristics, volume 146 of international series in operations research & management science, chapter a classification of hyper-heuristic approaches, 2010.

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

[19] 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*, pages 97–104. ACM, 2013.

[20] 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. IEEE, 2014.

[21] Andrei Lissovoi, Pietro S Oliveto, and John Alasdair Warwicker. On the runtime analysis of generalised selection hyper-heuristics for pseudo-boolean optimisation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 849–856. ACM, 2017.

[22] Benjamin Doerr, Andrei Lissovoi, Pietro S Oliveto, and John Alasdair Warwicker. On the runtime analysis of selection hyper-heuristics with adaptive learning periods. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1015–1022. ACM, 2018.

[23] Andrei Lissovoi, Pietro S Oliveto, and John Alasdair Warwicker. On the time complexity of algorithm selection hyper-heuristics for multimodal optimisation. In *the AAAI Conference on Artificial Intelligence*, 2019.

[24] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002.