Chapter 6
Theoretical Aspect—A Formal Definition

6.1 Introduction

Along with the continuous developments in hyper-heuristic (HH), various descriptive definitions for HH have emerged, leading to classifications of HH. Initially, hyper-heuristics have been defined as a search technique "to decide (select) at a higher abstraction level which low-level heuristics to apply" [51], "to combine simple heuristics" [162], or recently as a search method or learning mechanism for selecting or generating heuristics to solve computational search problems [30]. HH is thus categorized into four classifications, namely, selection perturbative / constructive, generation perturbative / constructive (see Chapters 3, 2, 5 and 4). Some attempts have also been made to generalize these classifications of HH, to allow both selection / generation and offline / online learning to interoperate within a repository [180]. It has also been proposed that the "domain barrier" in the HH definition should be moved so more knowledge can be easily incorporated in a more expressive HH for inexperienced practitioners [179].

This chapter presents a formal definition of HH based on the existing conceptual definitions in the literature [35]. Within the two-level framework of HH, two search spaces, namely the heuristic space and the solution space, are under consideration. Some fundamental issues are then discussed within this framework. In addition to the different encoding and search operations, various objective functions are defined in both spaces to evaluate searches on heuristics and direct solutions, respectively.

Within the two-level framework of HH, a selection constructive hyper-heuristic is then demonstrated to illustrate the inter-relationship between the two search spaces. A landscape analysis on the heuristic space reveals interesting characteristics for designing more effective hyper-heuristics. Future potential research developments are finally presented based on existing research advances addressing the theoretical aspects of HH.

© Springer Nature Switzerland AG 2018

N. Pillay, R. Qu, *Hyper-Heuristics: Theory and Applications*,
Natural Computing Series, https://doi.org/10.1007/978-3-319-96514-7_6

6.2 A Formal Definition of Hyper-Heuristics

A hyper-heuristic HH can be defined as a search algorithm for solving an optimization problem P, whose decision variables are heuristics, rather than direct solution variables in the optimization problem p under consideration. To solve P, HH explores at a higher level a heuristic space H of heuristic configurations h, which at a lower level generate direct solutions s in the solution space S for problem p. Two search spaces can thus be defined, namely a heuristic space H of P and a solution space S of p, each associated with an objective function, within the two-level framework [155].

Definition 6.1. Within a two-level framework, a hyper-heuristic HH explores heuristic configurations $h \in H$ in the heuristic space H at a high level. The performance of HH is measured using $F(h) \to R$. At the low level, an objective function $f(s) \to R$ evaluates the direct solutions $s \in S$ in the solution space S for the optimization problem p under consideration.

Solution s is obtained by using a corresponding heuristic configuration $h \in H$, i.e. $h \to s$. Let M be a mapping function M: $f(s) \to F(h)$. The objective of HH is to search in H for the optimal heuristic configuration $h*$, which generates the optimal solution(s) $s*$, so that $F(h*)$ is optimized:

$$F(h* \mid h* \to s*, h* \in H) \leftarrow f(s*,\ s* \in S) = \mathbf{min}\{f(s), s \in S\} \tag{6.1}$$

The following terminologies are defined in the formal HH definition [155].

- Problem p: an optimization problem under consideration, whose direct solutions $s \in S$ are evaluated against objective function $f(s)$.
- Problem P: an optimization problem considered by HH, whose decision variables are heuristic configurations $h \in H$ evaluated against objective function $F(h)$.
- Solutions s: direct solutions for p.
- Heuristic configurations h: configurations upon low-level heuristics in L for P.
- Solution space S: consists of s for p, obtained by using h, i.e. $h \to s$.
- Heuristic space H: consists of h for P, explored by high-level heuristic algorithms HLH in HH.
- Low-level heuristics L: a given set of domain specific heuristics configured by HLH at the low level to compose h, i.e. L contains the set of domain values for the decision variables in h.
- High-level heuristics HLH: search algorithms or configuration methods at the high level upon L to search $h \in H$ for P.
- Objective function f: fitness evaluation for p, i.e. $f(s) \to R$ evaluates $s \in S$, s obtained using $h \in H$.
- Objective function F: fitness evaluation for P, i.e. $F(h) \to R$ evaluates $h \in H$ explored by HLH. The objective is to find the optimal $h*$, which obtains the optimal solution $s*$ for p, i.e. $h* \to s*$.
- Mapping function M: $F(h) \leftarrow f(s)$: each h maps an s, thus the performance of HLH upon h is measured based on the evaluation of its mapping s. Note that

F(h) may not be the same as *f(s)* although it is the case in most of the existing *HH* literature.

In the HH literature, different optimization problems p can be solved by plugging in a problem specific set L at the low level. The design of HH can thus be focused on the design of the high-level *HLH*. Solving different p thus can be transferred to solving a general optimization problem P; the latter can usually be encoded with lower-dimension representation and is easier to explore [155]. The generality of HH is also raised, as problem specific details and constraint handling are left with the direct solutions s obtained at the low level for p. Due to the above definition of the two search spaces at the two levels, the burdens of designing problem specific algorithms are also eased, focusing on the high-level configurations of heuristics. HH showed to be easy to implement, and has been successfully applied to a wide range of combinatorial optimization problems [30].

In [151], a formal definition of selection constructive HH based on graph colouring is presented. The above formal definition is extended to define both types of selection and generation HH with constructive and pertubative L as classified in [31]. Note that the p at hand may be either continuous or discrete. In most of the current HH literature only combinatorial optimization problems are investigated. The formal HH definition can also be extended to define continuous optimization problems, which represents a new line of interesting future research directions.

6.2.1 Two Search Spaces Within the Formal Hyper-Heuristic Framework

HH solves p by indirectly configuring and exploring h in H at the higher level, which then used to search for direct solutions s in S. Therefore, it is necessary to distinguish between the heuristic space H for P and the solution space S for p. In the current literature, most HH mainly explore $h \in H$, aiming to find $h*$ that maps to (near-)optimal $s* \in S$, with less focus on the low-level S. Note, however, that within S, search can also be conducted by applying standard meta-heuristics to directly search s for p [151]. Table 6.1 compares the characteristics of the two search spaces based on terminologies defined for HH in Section 6.2.

Table 6.1 Characteristics of the two search spaces in the formal hyper-heuristic framework

Search Space	**Heuristic Space *H***	**Solution Space *S***
Encoding	Heuristic configurations h	Direct solutions s
Operation	High-level methods *HLH* upon the given L to configure h	Move or evolutionary operators on s
Objective Function	Evaluation function $F(h)$ upon h for P, $F(h) \leftarrow f(s)$	Objective function $f(s)$ on s for p

In selection hyper-heuristics, operations in HH often employ evolutionary algorithms or local search algorithms to configure h based on L [30]. Other configuration methods are also studied, including choice functions and case-based reasoning [30], see Chapters 2 and 3. In generation hyper-heuristics (see Chapters 4 and 5), genetic programming and its variants [11, 193] are often used to generate h, which can act as new problem specific heuristics to produce $s \in S$. Each h maps an s, thus the process of configuring or searching $h \in H$ simulates a search process exploring the mapping $s \in S$.

In most of the HH literature, $F(h) = f(s)$ [30]. Different evaluation functions, however, can be used in H and S, respectively. For example, in [17, 52, 119], a reward is used as F for a choice function to assess L and configure h at the high level, and a different problem specific evaluation function is used to evaluate the mapping s. Further in-depth studies may explore different M: $F(h) \leftarrow f(s)$ in the two-level framework, to design effective selection or generation HH with different high-level configuration methods and problem specific L.

HH indirectly searches $s \in S$ by exploring $h \in H$, thus may not directly explore from s towards the (local) optimal solutions $s^* \in S$, evaluated against f, as standard meta-heuristics usually do. Depending on the type of L in HH, neighbourhood solutions s' explored in S mapped by h' may or may not be the neighbourhood solutions from their precedent s mapped by h.

- In most HH employing perturbative L, individual low-level heuristics in h operate consecutively on complete direct solutions s, thus s can be seen as explored directly by the high-level search towards the (near)-optimal solutions s^* in S, guided by f on direct solutions s.
- In HH employing constructive L, solutions s and s' are constructed by h and h', thus s' obtained indirectly in S by h' may not be the neighbours of s, even if its corresponding h' is the neighbour of h. This is because during the solution construction using h', any different values assigned to variables in a partial solution using a different low-level heuristic in h', compared to h, are likely to lead to a different complete solution s'. Thus s' produced by the successive h' explored from h may not be neighbours of s in S.

Figure 6.1 presents the relationship between $h \in H$ and $s \in S$ within HH. In H, h_2 and h_3 are two successors of h_1 using an operation at the high level. In an HH that employs constructive L, their mapping corresponding solutions s_2 and s_3 in S may not be neighbours as defined using different (or even the same) operations upon the direct solution s_1 in S, obtained from h_1.

Given the characteristics in Table 6.1, it is noted that the size of H is very likely to be different from the size of S. In particular, S consists of all the possible direct solutions s for p, while H consists of heuristic configurations h for P. However, depending on how encoding and operation are defined, some of the s may not be obtained from any $h \in H$. This is reflected in Figure 6.1: s_4 may be a neighbour of $s_1 \in S$ using a specific operation; however, it may not have any corresponding $h \in H$ depending on how h is configured at the high level. In the example presented

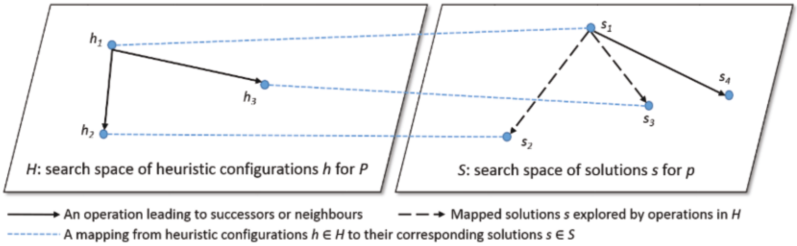

Fig. 6.1 Search in the two spaces H and S

in Section 6.3, this interesting issue has been investigated to explore search within both P and S to reach all $s \in S$ for p.

6.2.2 Fitness Landscape of the Heuristic Space in the Hyper-Heuristic Framework

In meta-heuristics, the concept of fitness landscape has been adapted to analyze the search space of solutions [115], revealing useful characteristics for designing more effective algorithms. For example, analysis of the landscape of the travelling salesman problem reveals an interesting feature called a "big valley", indicating a positive correlation between solutions and their fitnesses with the optimal $s*$, i.e. solutions closer to $s*$ are of better quality [128]. This observation may be used to design effective encodings and operators to guide the search towards $s*$ in TSP and other problems with similar features in the landscape.

Based on the definitions of fitness landscape in state space theory [115], the fitness landscape of H for the optimization problem P in the HH framework (Section 6.2) can then be defined with three factors, namely an encoding using some finite alphabet in L to represent all possible heuristic configurations h; a successor operator to define how $h \in H$ are connected (explored), and a fitness function $F(h) \rightarrow R$ that assigns a fitness value R to each $h \in H$.

In some HH where h are encoded as one-dimensional sequences of low-level heuristics in L, it is possible and useful to conduct landscape analysis on H, whose spatial structure can be defined using the operation and a distance metric D on h. This proved to be very difficult, if not impossible, for $s \in S$ for many complex combinatorial optimization problems with n-dimensional solutions, n $\geq$ 2. In the literature, fitness distance correlation *fdc* has been mostly used to analyze landscape properties and measure problem difficulty. Given a set of encodings h_1, h_2, ..., h_n and their fitness F, and the distance of $h \in H$ to their nearest optimum $h_{opt} \in H$, the *fdc* coefficient is defined as follows [87]:

$$fdc: \quad \sigma(F, D_{opt}) = Cov(F, D_{opt}) / \sigma(F)\sigma(D_{opt}) \tag{6.2}$$

where $Cov(.,.)$ denotes the covariance of two random variables and $\sigma(.)$ the standard deviation. In the literature, the optimal h_{opt} is estimated by the h which leads to the best solution $s*$ found for p. For h_i, fdc thus indicates how closely their F and D are related to that of h_{opt}. A value of σ = 1.0 (σ = -1.0) for maximization (minimization) problems indicates F and D are perfectly correlated to h_{opt} [87], and thus provide perfect guidance to h_{opt}; thus P is an easy problem in HH. In a landscape analysis, a value of $fdc \leq 0.5$ ($fdc \geq 0.5$) for maximization (minimization) problems usually indicates an easy P in HH.

More landscape analysis could be conducted using other measures including auto-correlation [192], which calculates the fitness correlation of a series of h recorded along a random walk over a time series T. The longer the time lag between two correlated h in the random walk, the less rugged is the landscape of H thus the easier the problem for HH. This also indicates from another aspect the difficulty of search problem P in H.

6.3 Example: A Selection Constructive Hyper-Heuristic for Timetabling Problems

A graph-based selection constructive hyper-heuristic in [32] is re-defined in this section based on the formal definitions of HH in Section 6.2 for educational timetabling problems. Based on an analysis of the two search spaces, a hybrid HH [151] is demonstrated, together with a landscape analysis on H in this HH framework [127]. More details of the work can be found in the original papers [32, 127, 151, 155].

6.3.1 A Graph-Based Selection Hyper-Heuristic (GHH) Framework

In timetabling, graph colouring heuristics (see more details in Section 10.2) are constructive heuristics that order the events using some difficulty measure strategies. The ordered events are then assigned, one by one starting with the most difficult ones, to construct complete timetable solutions. The basic assumption is that the most difficult events need to be scheduled earlier to avoid problems at a later stage. For example, if *SD* (Saturation Degree) is used in an exam timetabling problem, the exams are ordered by the number of remaining valid slots in the partial timetable during the solution construction, and the most difficult one is scheduled first to avoid the problem of no valid slots left at a later stage.

A graph-based selection constructive hyper-heuristic (GHH) is defined as follows: On the high-level space H, a local search algorithm as the high-level heuristic HLH explores heuristic sequences $h \in H$ using the low-level graph colouring constructive heuristics in $L = \{LD, LWD, SD, LE, CD\}$, as explained in Section 10.2.

Each $h = \{h_1, ..., h_n\}$, $h_i \in L$, is evaluated by $F(h) \rightarrow R$. n is the problem size, i.e. the number of decision variables in p.

At the lower level of GHH, a timetable solution $s \in S$ is constructed iteratively by using an $h \in H$, considering constraints and f for the timetabling problem p (see Appendix B.4). At iteration i, $h_i \in h$ is employed to order the events not yet scheduled in p using its corresponding ordering strategy. The first event in the ordering (i.e. the most difficult one using h_i) is then scheduled in s. In the next iteration, h_{i+1} in h is used to reorder and schedule the most difficult remaining events. This process is repeated until a complete s is constructed. Any h that leads to infeasible solutions is discarded. The objective function $f(s) \rightarrow R$ evaluates $s \in S$ for p (see Appendix B.4).

The mapping function is defined as M: $F(h) = f(s)$, $h \rightarrow s$. The optimization problem P in the HH framework is thus to search for h^* of L at a higher level in H which constructs (near-)optimal solution(s) s^*.

6.3.2 Analysis of Two Search Spaces in the GHH Framework

In the GHH defined above, different local search algorithms are employed at the high level [151] to search for $h \in H$, and a greedy steepest descent method is used at the low level to exploit local optima from $s \in S$; s is obtained using the corresponding h. Thus search has been conducted within both H and S, with characteristics given in Table 6.2. Note that different meta-heuristics can be employed at both levels, and the objective functions at the two levels can be different.

Table 6.2 Characteristics of the two search spaces in the GHH framework

Search Space	**Heuristic Space *H***	**Solution Space *S***
Encoding	Sequences of heuristics h	Direct timetable solutions s
Upper Bound of the Search Space	n^e (e: length of h; n: size of L, i.e. $\lvert L \rvert$)	t^e (t: no. of slots; e: no. of events)
Operator	Randomly change two h_i in h	Move events in s to new slots
Objective Function	Cost of s constructed by the new h	Cost of the new neighbour s

Within GHH, the high-level search explores h rather than direct solutions s. As an s is constructed by an h step by step, similar neighbouring h in H may construct quite different s, likely to be widely distributed in S. As illustrated in Figure 6.1, by making local neighbourhood moves from h_1 to h_2 or h_1 to h_3 at the high level in H, GHH can explore s_2 or s_3 across very different regions in S. A local search at the low level upon s in S, on the other hand, usually generates similar local solutions, i.e. s_3 to s_4. The GHH search thus can be seen as exploring much larger neighbourhood regions in S using a local search in H, similarly to the search behaviour of large-neighbourhood search algorithms.

In [151], a fast steepest descent method is hybridized at the low level within GHH to further exploit a local optimum from $s \in S$. The motivation is twofold: First, the steepest descent in S can exploit local areas around s_3 to reach local optima quickly; Second, GHH thus is able to explore the whole search space S including s_4, which may not be reached by any $h \in H$.

6.3.3 Performance Evaluation of GHH

In [151], four different local search algorithms, namely steepest descent, tabu search, variable-neighbourhood search and iterated local search, have been used as the high-level search to explore h for both the course and exam timetabling problems, employing the same L, as presented in Section 6.3.1.

It was found that although variable-neighbourhood search and iterated local search performed slightly better, in general high-level search within GHH did not play a crucial role. This may be because that at the high level, the h are not concerned with the actual assignments of decision variables in s for p, but are indirect configurations of constructive heuristics, which are then used to build s. s sampled by h at the high level tend to *jump* within S; thus s and s' from the neighbouring h and h' are not successive neighbours. The different search methods used in H thus did not directly lead to different performance of HH upon S.

When employing steepest descent at the low level in S on each complete s constructed by h, GHH obtained significantly better results. Although the local optimum h in H at the high level might not map a local optimum s in S, the steepest descent upon s further explores S, leading to locally optimal solutions for p. Within GHH, the role of the high-level local search in H can thus be seen as to explore S indirectly, while the steepest descent search at the low level is to exploit local regions in S.

Penalties of timetable solutions obtained by GHH using iterated local search on the exam timetabling problems (see Appendix B.4) are presented in Table 6.3, compared against existing algorithms. In [151] exactly the same GHH is applied to both the exam and course timetabling problems. The only difference is $f(s) \rightarrow R$ on $s \in S$ for different p. Note that some of the existing approaches in Table 6.3 are not hyper-heuristics, and are specially designed for solving the specific problem under consideration, thus may not have been applied to solve both problems.

The overall idea of the exploration in H and exploitation in S using search at two levels in GHH is similar to that of memetic algorithms or genetic local search, where genetic operators applied to the population of solutions in S facilitate global exploration, while the local search on solutions in the population conducts exploitation within local regions. The difference is that GHH explores S by indirectly searching H, at a high level, in the manner of local search. The hybrid GHH is much simpler yet is capable of exploring and exploiting the search space S at two levels.

HH aims to increase the level of generality in solving multiple problems and problem instances, while most of the existing HH approaches have been applied to one problem domain, or evaluated by specific objective functions for different

Table 6.3 Penalties of timetable solutions by GHH on benchmark exam timetabling problems against existing approaches; details of the problem and penalty function can be found in Appendix B.4

	car91	car92	ear83 I	hec92 I	kfu93	lse91	sta83 I	tre92	ute92	uta93 I	yok83 I
GHH	5.3	4.77	38.39	12.01	15.09	12.72	159.2	8.74	30.32	3.42	40.24
LNS [2]	5.21	4.36	34.87	10.28	13.46	10.24	159.2	8.7	26	3.63	36.2
Fuzzy [6]	5.2	4.52	37.02	11.78	15.81	12.09	160.4	8.67	27.78	3.57	40.66
Adaptive [39]	4.6	4.0	37.05	11.54	13.9	10.82	168.7	8.35	25.83	3.2	36.8
Local search[25]	4.8	4.2	35.4	10.8	13.7	10.4	159.1	8.3	25.7	3.4	36.7
Hybrid [42]	6.6	6.0	29.3	9.2	13.8	9.6	158.2	9.4	24.4	3.5	36.2
Heuristics [43]	7.1	6.2	36.4	10.8	14.0	10.5	161.5	9.6	25.8	3.5	41.7
Tabu Search [67]	6.2	5.2	45.7	12.4	18.0	15.5	160.8	10.0	29.0	4.2	42.0
Hybrid [114]	5.1	4.3	35.1	10.6	13.5	10.5	157.3	8.4	25.1	3.5	37.4

problems, respectively. Thus the generality of HH approaches have not yet been assessed using a uniform or consistent measure. In recent research, a performance assessment for HH, at four different levels of generality, has been proposed [147]. Such study shows a welcome attempt to address the fundamental aspects of further research developments in HH.

6.3.4 Fitness Landscape Analysis on GHH

In the literature some landscape analysis has been conducted using measures such as *fdc* and auto-correlation, both indicating from different aspects the difficulty of search in H. An example analysis using *fdc* as explained in Section 6.2 to analyze H in GHH is presented in this section. More details can be found in [127].

Based on a variant of GHH using two low-level heuristics, *LWD* and *SD* as defined in Section 10.2, in [127] the landscape of H in GHH has been analyzed to gain insight into the global structure of H. As in the literature, the best known h obtained, h_{opt}, is used as an estimation of the optimal solution in the *fdc* analysis (Equation 6.2). $h \in H$ can thus be represented by binary strings, whose distance D is measured using Hamming distance.

In the *fdc* analysis, locally optimal h are measured against h_{opt} to reveal landscape features of H, indicated by the correlations between their distances and costs. A set of h is first randomly generated, one of each distance j, $j = 1, ..., l$, away from h_{opt}, l is the length of h_{opt}. A non-deterministic steepest descent search using one-flip neighbourhood moves is then applied 10 times to these binary h to generate 10 locally optimal h for each j. In total $LO = 10 \times l$ locally optimal h are thus obtained and their correlations with h_{opt} using (Eq. 6.2) are calculated. More details can be found in [127].

The fitness values of these LO for two example timetabling instances, *hec92 I* and *sta83 I*, are plotted in Figure 6.2, ordered increasingly by their costs. The plots show a number of local optima of the same cost especially for *sta83 I*, demonstrating several plateaus in H.

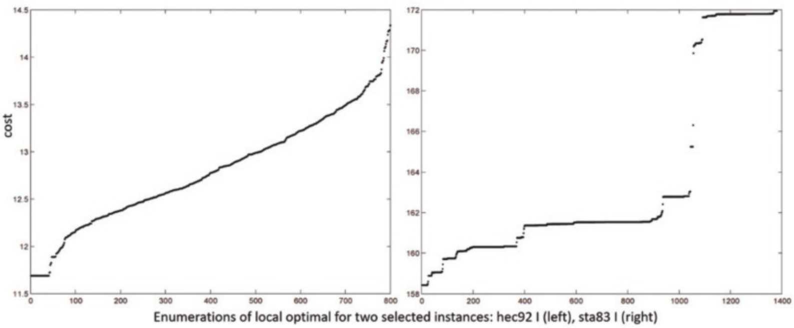

Fig. 6.2 Cost of local optimal h in GHH [127].

Figure 6.3 shows the best 10% of these LO local optima. Some interesting patterns can be extracted especially at the beginning of h. For example, values at certain positions h_i in the top h are fixed, i.e. the first four positions in h for *hec92 I* are always *LWD*. It is not surprising to see that random patterns appear at the end of h, as the last steps of solution construction tend to make less impact on the quality of s. No obvious patterns can be observed on lower-quality h.

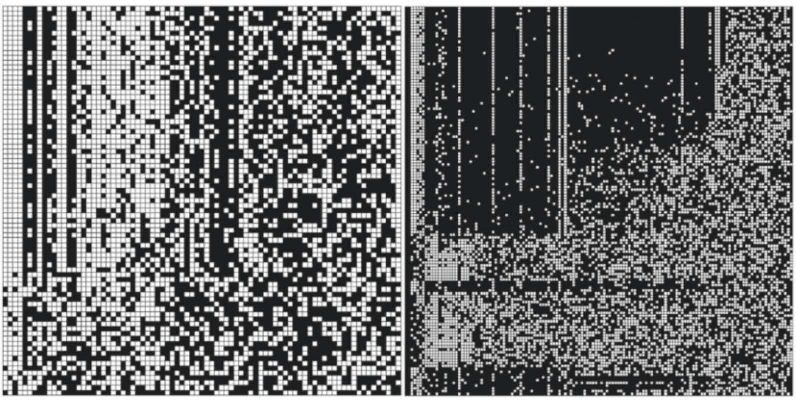

Fig. 6.3 The best 10% local optimal h in GHH: hec92 I (left), sta83 I (right). Note: at each horizontal line white and black plots indicate low-level heuristics *LWD* and *SD*, respectively

The scatter plots between $F(h)$, $h \in LO$, and their distances to h_{opt} indicate a moderate-to-high positive correlation (in the range of 0.51 to 0.64). This is a very useful "big valley" feature in the landscape of H, similar to that of TSP observed in the literature [128], meaning better local optima are closer to h_{opt} in H. This also indicates that search in H is probably easier, as $F(h)$ of locally optimal h provides a useful indication of how close they are to h_{opt}.

Other patterns can also be observed, revealing some interesting features in the landscape of H. Although presenting similar positive correlation, the scatter plot for instance *sta83 I* shows several wide plateaus of the same cost f at different levels. In addition, those h with a cost below 38 (around $l/2$ bits away from $h*$) are of low-quality, and show no clear correlation i.e. they are randomly located in H. More details can be found in [127].

Due to the simple one-dimensional structure of h, it is possible to conduct landscape analysis on H. This showed to be very difficult, if not impossible, for s in S for some problems investigated including the timetabling, vehicle routing and nurse rostering problems.

6.4 Discussion

Along with more recent advances in HH research addressing different theoretical aspects, more interesting research issues have emerged that require a formal definition of different types of HH in the literature. Based on the existing different conceptual definitions, a formal definition of HH [155] is presented in this chapter as an *optimization problem* to provide a unified fundamental basis for further explorations of emerging research directions in future HH research.

To demonstrate the formal HH framework, an existing selection constructive hyper-heuristic employing high-level local search algorithms [151] has been redefined, along with a landscape analysis for educational timetabling problems. Within the two-level HH framework, two search spaces, namely, the heuristic space H and solution space S can be explored separately, each with its own objective function. Landscape of the high-level search space H with one-dimension sequences of h showed the "big valley" feature by using the *fdc* analysis. The relation / mapping between the two search spaces is worth further investigation under the formal HH framework for the four types of HH [31] described in Chapters 2, 3, 4 and 5.

Based on this HH framework, several future research directions are worth further exploration.

HH aims to raise the generality of algorithms for solving multiple problems. This poses an interesting research question, namely how the No Free Lunch Theorem (NFL) applies to this new type of search algorithms. Some interesting discussions in [150] analyzed the conditions under which the NFL applies to HH. Based on the statement, if a set of fitness functions associated with problems p are *closed under permutation* [194], it would make no sense to find a solver for such p. However, such a set of problems usually represents a small fraction of the whole, thus there may be a free lunch developing HH approaches for not too large a set of problems. It would be interesting to carry out more in-depth analysis on NFL within the formal definition of the HH framework, to further explore the scope of p HH may address.

In [100], a runtime analysis is conducted on a selection HH using a randomized local search. It is shown that configuring a set of neighbourhood operators with an appropriate distribution is crucial, and is also problem dependent. It also shows that online reinforcement learning on configuring operators may perform more poorly than fixed distribution of operators in selection perturbative hyper-heuristics. A further investigation within the formal HH framework for both selection and generation HH with constructive and perturbative low-level heuristics would establish the theoretical foundations of HH as that of runtime analysis for evolutionary algorithms [100].

Performance measures on most of the current HH approaches in the literature are problem specific, even where multiple problems are concerned, when assessing the performance of HH. Some progress has been made in [147] to devise a new generality performance measure for HH of different levels of generality for different problems. With the aim of raising the generality of search algorithms, this new performance measure can be associated with the formal HH framework, to provide evaluations of different HH across various problems.

A unifying mathematical formulation for hyper-heuristics is proposed in [180], where by using a high-level controller, elements of heuristic design (both constructive and perturbative heuristic activities) compete for resources within a shared repository workspace to configure better heuristics. There is thus no distinction between online and offline activities, and heuristic activities interoperate based on information shared from other heuristics. The formal HH framework defined in this chapter could be integrated with this unifying framework, where the heuristic space H of P is explored by the high-level controller to solve multiple problems p.

Recent research on landscape analysis observes that multiple "sub-valleys" exist in a single big valley for TSP [128]. Similar analysis of the high-level H in the HH framework, where h is encoded as one-dimensional strings or sequences, might reveal more insights into the high-level landscape in HH, and inspire the design of more effective HH across different problems p.