Chapter 3
Selection Perturbative Hyper-Heuristics

3.1 Introduction

Selection perturbative hyper-heuristics select which low-level perturbative heuristic to apply at each point of improvement to a given initial complete solution to a problem. The initial solution is usually created either randomly or using a constructive low-level heuristic. It is usually iteratively refined by applying a perturbative low-level heuristic until there is no further improvement, measured using problem specific criteria such as the objective value of the perturbed solution. Starting from the initial problem state (solution), the application of each low-level perturbative heuristic results in moving from one problem state to the next until a final problem state, which cannot be improved further, is reached. A formal definition of selection perturbative hyper-heuristics is given in Definition 3.1.

Definition 3.1. Given a problem instance *p*, an initial solution s_0 and a set of low-level perturbative heuristics $L = \{L_0, L_1, ..., L_n\}$ for the problem domain, a selection perturbative hyper-heuristic *SPH* improves the solution s_0 by selecting and applying a perturbative heuristic L_i from L to get from one problem state s' to the next s'' until a problem state resulting in no further improvement of the solution s_i is reached.

As in the case of low-level constructive heuristics, the low-level perturbative heuristics are problem dependent. For example, in the case of solving the examination timetabling problem, a perturbative heuristic will swap the examinations of two timetable periods, while in the case of the travelling salesman problem a perturbative heuristic inserts a subset of cities at a new position in the route.

Selection perturbative hyper-heuristics employ *single-point* or *multipoint* search to select the low-level perturbative heuristics, as discussed further in Section 3.2 and Section 3.3, respectively. In single-point selection perturbative hyper-heuristics, two decisions are usually made, namely *heuristic selection* and *move acceptance* [30]. Multipoint selection perturbative hyper-heuristics employ population-based methods such as evolutionary algorithms to search the space of perturbative heuristics. By their nature, these search techniques perform both heuristic selection and

N. Pillay, R. Qu, *Hyper-Heuristics: Theory and Applications*,
Natural Computing Series, https://doi.org/10.1007/978-3-319-96514-7_3

move acceptance, and hence separate components for these functions are not needed [5, 45, 60, 73, 157].

3.2 Single-Point Search Selection Perturbative Hyper-Heuristics

Algorithm 7 depicts the general algorithm employed by single-point search selection perturbative hyper-heuristics.

Algorithm 7 Selection perturbative hyper-heuristic algorithm

```
1:  procedure SELECTIONPERTURBATIVEHYPERHEURISTIC($p$, $L$)
2:      create an initial solution $s_0$ using a random or constructive heuristic
3:      repeat
4:          use the heuristic selection technique $h$ to select a perturbative heuristic $L_{\mathrm{i}}$ from $L$
5:          apply $L_{\mathrm{i}}$ to solution $s_i$ to produce the perturbed solution $s_{i+1}$
6:          use the move acceptance technique $M$ to accept the move or not
7:          if the move is accepted $s_i$=$s_{i+1}$
8:      until the termination criterion
9:      return $s_i$
10: end procedure
```

A termination criterion commonly used is that there is no further improvement in the solution s_i. Alternatively, the processes of heuristic selection and move acceptance can be performed for a set number of iterations. The following sections provide an overview of heuristic selection and move acceptance techniques.

3.2.1 Heuristic Selection Techniques

This section provides an overview of early and commonly used techniques for heuristic selection. It is by no means exhaustive as the list of techniques employed by selection perturbative techniques is rapidly growing.

The simplest heuristic selection technique is *random selection* [3, 99, 118], which randomly selects a perturbative heuristic from the available heuristics and applies it to the current solution s_i. A variation of the *random selection* technique is *random gradient* [3, 30], which selects a heuristic randomly and applies it iteratively, beginning with solution s_i, until there is no further improvement. *Random permutation* selects a sequence of perturbative heuristics randomly, they are applied in order [3, 30]. *Greedy* [30] applies all the perturbative heuristics in L and selects the heuristic producing the solution s_{i+1} with the best objective value.

Evolutionary algorithms have also been used for purposes of heuristic selection. Two methods used for heuristic selection are *tournament selection* and *fitness proportionate selection* [30]. These methods select a heuristic from the set L of available low-level perturbative heuristics. The fitness of each perturbative heuristic L_j is a problem specific measure, such as the objective value of the perturbed solution s_{i+1} resulting from applying L_j to s_i. In the case of tournament selection, a set of heuristics of fixed size is randomly selected from L, and the heuristic producing the solution with the best objective value is selected. On the other hand fitness proportionate selection creates a pool of heuristics based on the fitness of each heuristic L_i in L, and a heuristic is randomly selected from this pool.

The concept of a *choice function* was introduced for heuristic selection [30, 55, 89, 99]. A choice function calculates a rank for each heuristic L_j in L based on its performance, i.e. the improvements it has produced thus far and when it was last applied during the process. The heuristic with the best rank is selected and applied. The rank for each h_i is calculated using the following formulae [55, 89, 99]:

$$f(h_i) = \alpha f_1(h_i) + \beta f_2(h_i) + \delta f_3(h_i) \tag{3.1}$$

$$f_1(h_i) = \sum_n \alpha^{n-1} \frac{I_n(h_i)}{T_n(h_i)} \tag{3.2}$$

$$f_2(h_j, h_i) = \sum_n \beta^{n-1} \frac{I_n(h_j, h_i)}{T_n(h_j, h_i)} \tag{3.3}$$

$$f_3(h_i) = \tau(h_i) \tag{3.4}$$

f_1 in equation 3.2 is a measure of the recent performance of heuristic h_i over its previous n invocations. $I_n(h_i)$ is the change in objective value from the last invocation. Similarly, $T_n(h_i)$ is the difference in the time since the last invocation of the heuristic h_i. f_2 in equation 3.3 is a measure of the pairwise performance of h_i with all other heuristics h_j over n invocations of successive application of h_j and h_i. $I_n(h_j, h_i)$ is the difference in objective values from one successive application of h_j and h_i to the next. $T_n(h_j, h_i)$ is the difference in time since the last successive invocation of h_j and h_i. f_3 indicated in equation 3.4 is a measure of the time taken in CPU seconds since the heuristic was last applied during the improvement process. The parameters α, $\beta \in [0,1]$ set the importance of the recent performance of heuristic h_i. δ is a real-valued parameter used to maintain diversity. The basis of the choice function is reinforcement learning.

Reinforcement learning has also been successfully used for heuristic selection in selection perturbative hyper-heuristics [55, 85, 131, 132]. It assigns a score to each heuristic in the set L based on its performance during the improvement process. At the beginning of the improvement process all the heuristics are assigned the same score. During the process if a low-level heuristic L_i results in an improvement of a candidate solution its score is increased whereas if it results in a worse solution the score is decreased. The heuristic with the best score at the particular point of improvement is selected and applied.

A tabu list [70] has also been used as part of the heuristic selection component [36, 90] to prevent a poorly performing heuristic from being reused or to prevent the use of the same heuristic for a number of iterations in the improvement process.

Stochastic methods such as Markov chains are also used for heuristic selection [91]. Each chain is composed of low-level heuristics from L. The heuristics are not applied in sequences, a transition probability is associated with each heuristic and is used to decide which heuristic to apply next. Roulette wheel selection is used to choose the next heuristic to apply based on the transition probability.

Depending on the technique employed by the heuristic selection component, *learning* may or may not take place in selecting a heuristic. For example, in the case of random selection or random gradient, no learning is performed. However, reinforcement learning involves learning in terms of the performance of the low-level heuristics on previous iterations of the improvement process.

3.2.2 Move Acceptance Techniques

This section provides an account of early and commonly used move acceptance techniques. As in the case of heuristic selection techniques, the list of move acceptance methods successfully employed by selection perturbative hyper-heuristics is growing and the overview provided is not exhaustive. Move acceptance criteria are used to decide whether to accept the solution s_{i+1} produced by applying a perturbative heuristic L_j to s_i. The outcome of the move acceptance method is to either accept s_{i+1}, in which case it replaces s_i, or reject s_{i+1}, in which case s_i remains unchanged. The quality of solutions is measured in terms of the objective value of s_{i+1}.

The simplest move acceptance approach *accept all moves* [30] accepts the resulting solution produced by applying heuristic L_j irrespective of the quality of the perturbed solution s_{i+1} produced. A variation of this method accepts heuristics producing a worse perturbed solution s_{i+1} with a specified probability [91]. An extended technique *accept improving moves* [55, 99] only accepts the application of the heuristic L_j to s_i if there is an improvement in quality of the resulting solution s_{i+1}. Similarly, *accept equal and improving* [118, 131] accepts moves producing solutions of the same or better quality. Misir et al. [118] extend this idea and introduce two new move acceptance approaches, namely, Iteration Limited Threshold Accepting (ILTA) and Adapted Iteration Limited Threshold Accepting (AILTA). ILTA generally accepts improving and equal moves, and moves producing worse solutions if the current iteration exceeds the specified iteration limit and the fitness of the produced solution is less than a factor, specified by the threshold, of the best fitness obtained thus far. Both the iteration limit and the threshold value are parameters. AILTA is similar to ILTA but allows for the threshold value to be adapted if there is no improvement in the solutions perturbed.

Local search techniques have also been used for the purposes of move acceptance. These include simulated annealing [3, 55, 85], late-acceptance hill climbing

[55] and great deluge [3, 132]. This involves using the mechanism employed by the local search to accept a move to decide whether to accept the solution resulting from the application of the heuristic. The heuristic selected by the heuristic selection component is applied to solution s_i to produce the perturbed solution s_{i+1}. The acceptance mechanism of the local search is used to determine whether to accept or reject s_{i+1}. If s_{i+1} is an improvement over s_i it is accepted; otherwise the criterion specific to the local search for accepting worsening moves is applied. For example, in the case of simulated annealing a worse perturbed solution is accepted based on the temperature value.

3.3 Multipoint Search Selection Perturbative Hyper-Heuristics

Multipoint search selection perturbative hyper-heuristics employ population-based methods such as genetic algorithms [73, 156, 157], particle swarm optimization [5] and ant colonization [45, 60], to explore the heuristic space. The hyper-heuristic produces a heuristic [5] or a sequence of heuristics [5, 45, 73, 157] to improve an initial solution created randomly or using a constructive heuristic. When using particle swarm optimization each particle represents a heuristic or sequence of heuristics [5]. Similarly, in the study employing ant colonization [45] each ant chooses the next heuristic to apply. Genetic algorithms explore the space of low-level heuristic sequences [73, 156, 157]. In the case of a single heuristic this is applied to improve the initial solution, while a heuristic sequence is applied iteratively with each heuristic of the sequence applied in order to improve the initial solution. This process is depicted in Algorithm 8.

Algorithm 8 Applying a perturbative heuristic sequence

```
1: Given an initial solution $s_0$ and heuristic combination $h = h_1 ... h_n$
2: for $i \leftarrow 1, n$ do
3:     Apply the chosen $h_i$ to $s_i$ to create $s_{i+1}$
4: end for
5: Report $s_n$
```

Genetic algorithms have been the most popular multipoint search method employed by selection perturbative hyper-heuristics. Each element of the population, i.e. a chromosome, is a sequence of heuristics [73, 156, 157]. Research has shown that variable-length chromosomes are more effective than fixed-length chromosomes in genetic algorithm selection perturbative hyper-heuristics [73]. Each sequence is created by randomly selecting low-level perturbative heuristics from the available set of perturbative heuristics for the problem domain. An initial solution is created randomly or using a constructive heuristic and is used to calculate the fitness of each chromosome on each generation. The fitness of the chromosome is determined by applying it to the initial solution using Algorithm 8. The fitness of the chromosome is a function of the objective value of the resulting perturbed solution s_n.

3.4 Discussion

The majority of the research on selection perturbuative hyper-heuristics has focused on single-point search combining heuristic selection techniques and move acceptance criteria. The heuristic selection techniques used range from simple techniques with no learning that randomly select a perturbative heuristic, to approaches with learning and that select the heuristic based on its performance in previous iterations during the improvement process. The simplest move acceptance techniques are deterministic and accept all moves or improving and/or equal moves only. Variations include approaches that will also accept worsening moves based on certain criteria, e.g. a specified threshold value or the iteration of the improvement process. The acceptance mechanisms of local searches such as simulated annealing, great deluge and late-acceptance hill climbing have also proven to be effective for the purposes of move acceptance.

More recently multipoint search techniques such as genetic algorithms and particle swarm optimization have been employed by selection perturbative hyper-heuristics. An area that needs further investigation is comparative studies of single-point search and multipoint selection perturbative hyper-heuristics for different problem domains. Tables 3.1 and 3.2 list some of the application domains that single-point search and multipoint search hyper-heuristics have been applied to, respectively.

Table 3.1 Single-point search selection perturbative hyper-heuristic applications

Problem domain	**Heuristic Selection**	**Move Acceptance**
Maximum satisfiability [99]	Random, choice function	Improving only
Nurse rostering [36]	Reinforcement learning with tabu list	Accept all moves
University course timetabling [36]	Reinforcement learning with tabu list	Accept all
Multidimensional knapsack problem [36]	Random, choice function, reinforcement learning	Accept all, improving only, late acceptance, simulated annealing
Examination timetabling [90]	Tabu list	Accept all
Examination timetabling [132]	Reinforcement learning	Great deluge
Examination timetabling [131]	Reinforcement learning	Improving or equal
Home care scheduling [118]	Random	Improving or equal, iteration limited threshold accepting, adaptive iteration limited threshold accepting

As the field is developing, the range of techniques employed by selection perturbative hyper-heuristics is growing. Burke et al. [34] present a Monte Carlo selection perturbative hyper-heuristic framework that can be used with different approaches

Table 3.2 Multipoint Search Selection Perturbative Hyper-Heuristic Applications

Problem domain	**Multipoint Search Hyper-Heuristic**
Geographically distributed course timetabling problem	Genetic algorithm [73]
Student project presentation problem	Genetic algorithm [73]
Travelling tournament problem	Ant colonization [45]
Nurse rostering problem	Genetic algorithm [156]
School timetabling	Genetic algorithm [157]
Resource scheduling in a grid environment	Particle swarm optimization [5]
Set covering problem	Ant colonization [60]

for heuristic selection and move acceptance. Three Monte Carlo move acceptance approaches are used in this framework, namely, simulated annealing, simulated annealing with reheating and exponential Monte Carlo. In [91] Markov chains are used for heuristic selection. Misir et al. [121] have introduced the use of an automaton for heuristic selection.

Along with the research developments in meta-heuristics and evolutionary algorithms, a range of different perturbative move operators and acceptance criteria have been developed in various combinatorial optimization problems. The hybridizations of these with other techniques showed to be effective when adapted in selection perturbative hyper-heuristics. For example, in vehicle routing problems (see Chapter 7), both perturbative and constructive heuristics are selected by the high-level search to construct and improve solutions. In nurse rostering (see Chapter 8), low-level perturbative heuristics and acceptance criteria are selected as pairs at each decision point during the solution improvement process.