1

Hyperheuristics: Recent Developments

Konstantin Chakhlevitch[1] and Peter Cowling[2]

[1] CASS Business School, City University, London EC1Y 8TZ, UK
Konstantin.Chakhlevitch.1@city.ac.uk
[2] Department of Computing, University of Bradford, Bradford BD7 1DP, UK
P.I.Cowling@Bradford.ac.

1.1 Introduction

Given their economic importance, there is continuing research interest in providing better and better solutions to real-world scheduling problems. The models for such problems are increasingly complex and exhaustive search for optimal solutions is usually impractical. Moreover, difficulty in accurately modelling the problems means that mathematically "optimal" solutions may not actually be the best possible solutions in practice. Therefore heuristic methods are often used, which do not guarantee optimal or even near optimal solutions. The main goal of heuristics is to produce solutions of acceptable quality in reasonable time. The problem owners often prefer simple, easy to implement heuristic approaches which do not require significant amount of resources for their development and implementation [12]. However, such individual heuristics do not always perform well for the variety of problem instances which may be encountered in practice. There is a wide range of modern heuristics known from the literature which are specifically designed and tuned to solve certain classes of optimisation problems. These methods are based on the partial search of the solution space and often referred as *metaheuristics*.

Although tailor-made metaheuristic algorithms have proved to be very effective for solving various combinatorial optimisation problems, their application is usually limited to particular problem domains. Metaheuristics incorporate information specific for the problem and "require extensive knowledge in both the problem domain and appropriate heuristic techniques" [21]. Therefore such methods are often quite expensive to implement. Metaheuristic approaches that perform well on a particular real-world problem, may not work at all or may produce very poor solutions for other problems or even for other instances of the same problem. Such limitations can become especially critical in situations when problem data and business requirements change frequently over time. This can make a metaheuristic even more expensive because it should be properly maintained.

Burke et al. in [12] note that many businesses are interested in "good enough-soon enough-cheap enough" solutions to their problems provided by easy-to-use and robust heuristic approaches rather than optimal or near optimal solutions achieved at the expense of the development of customised problem-specific methods such as metaheuristics and possibly using greatly simplified models. This is a primary motivation for development of generalised, domain-independent heuristic search techniques which have recently become known as *hyperheuristics* and have received an increased attention in the research community. The purpose of hyperheuristics is not to compete with state-of-the-art problem-specific approaches, but to provide a general framework able to deliver solutions of a good quality for a wide range of optimisation problems.

Another motivation for development of hyperheuristics comes from the fact that performance of different heuristics may vary significantly depending on the specific characteristics of the problem and problem instance under consideration. Moreover, individual heuristics may be particularly effective at certain stages of the solution process (i.e. when certain areas of the solution space are being explored) while performing poorly at any other stages. Therefore, it is fair to expect that several heuristics combined in a proper way may produce better solutions then if they are applied separately. A hyperheuristic can be defined as a *heuristic which chooses heuristics* [61]. In other words, a hyperheuristic operates in a *space of heuristics* choosing and applying one low level heuristic from a given *set* at each decision point. This is where the fundamental difference between hyperheuristics and metaheuristics lies since a metaheuristic is a heuristic which controls the search in a *space of solutions* performed by a single low level heuristic..

The term "hyperheuristic" was first introduced by Cowling et al. in [21]. They defined a hyperheuristic as an "approach that operates at a higher level of abstraction than metaheuristics and manages the choice of which low-level heuristic method should be applied at any given time, depending upon the characteristics of the region of the solution space currently under exploration". This means that the hyperheuristic itself does not search for a better solution to the problem. Instead, it selects at each step of the solution process the most promising simple low-level heuristic (or combination of heuristics) which is potentially able to improve the solution. On the other hand, if there is no improvement, i.e., a locally optimal solution is found, the hyperheuristic diversifies the search to another area of the solution space by selecting appropriate heuristics from the given set. Low-level heuristics usually represent the simple local search neighbourhoods or the rules used by human experts for constructing solutions. However, it is also possible that more complex heuristics such as metaheuristics can be considered at a lower level. All domain-specific information is concentrated in the set of low-level heuristics and the objective function. Hyperheuristics do not require knowledge of how each low-level heuristic works or the contents of the objective function of the problem (other than the value returned). It only needs to know the direction

of the optimisation process (maximising or minimising) and analyses the value of one or more objective functions and, sometimes, the amount of CPU time required to perturb the solution, which are returned by the low-level heuristic after its call.

Hyperheuristics have received much attention over the last 5 years or so and will likely remain a hot research topic for the near future. The first paper on hyperheuristics was presented by Fisher and Thompson [30] in 1961, but there was no other work in this area published between then and the 1990's, when a few related approaches were developed. The latter methods were mainly based on genetic algorithms which used indirect chromosome representation so that the chromosome encoded the method to solve a problem instead of the solution of a problem. However, the motivation for such methods was rather to overcome the difficulties related to solution encoding and maintaining the solutions feasibility than to develop general solution techniques able to tackle different optimisation problems. Nevertheless, the ideas used in these approaches created the basis for recent developments in hyperheuristics. We shall discuss them later in this chapter.

Based on the the original definition introduced by Cowling et al. [21], we shall use the following criteria to define a hyperheuristic:

1. A *hyperheuristic* is a *higher level heuristic* which manages a *set* of *low level heuristics*, of cardinality greater than one.
2. A hyperheuristic searches for a good *method* to solve the problem rather than for a good solution.
3. A hyperheuristic uses only *limited* problem-specific information (ideally this information includes only the number of low level heuristics for the problem and objective function(s) to be maximised or minimised).

The third criterion is the most crucial one since it defines the level of generality of the hyperheuristic approach as well as its potential robustness across different problem domains. Many techniques considered in this chapter match to the first two statements and fail to comply to the third one.

From our point of view, hyperheuristic approaches developed so far can be classified into the following categories: hyperheuristics based on the random choice of low level heuristics (Section 1.2), greedy and peckish hyperheuristics (Section 1.3), metaheuristic-based hyperheuristics (Section 1.4) and hyperheuristics employing learning mechanisms to manage low level heuristics (Section 1.5). We will also consider other generic solving methods which are closely related to hyperheuristics (Section 1.6).

1.2 Hyperheuristics based on random selection

Hyperheuristics based on the random choice of low level heuristics from a given set have been widely represented in the literature. Table 1.1 provides a list of the approaches which fall in this class.

Table 1.1. Hyperheuristic methods based on random selection of low level heuristics

Approach	Papers	Details
Pure random	Cowling et al. [21], [23], [24] Kendall and Mohamad [46] Bai and Kendall [4] Burke et al. [9] Cowling and Chakhlevitch [18] Storer et al. [62]	Uniform selection of LLH*; either all LLH or only improving LLH are accepted
Random descent	Cowling et al. [21], [22] Soubeiga [61]	Improving LLH is applied repeatedly until it does not improve the solution
Unbiased random process	Fisher and Thompson [30], [31]	Multistart process where probabilities of LLH selection are adjusted after each run
Monte Carlo	Ayob and Kendall [3] Bai and Kendall [4] Soubeiga [61] Chakhlevitch [16]	Probability of accepting LLH is a function of the difference between old and new objective values; includes simulated annealing
Random with deterministic acceptance	Kendall and Mohamad [45], [46]	Acceptance rules are based on the distance between two solutions; includes Great Deluge and record-to record-travel methods

*LLH stands for low level heuristic(s)

The "random" hyperheuristic is the oldest, the simplest, and easiest to implement of the family of hyperheuristics. It randomly selects one low level heuristic from a given set at each decision point. The selected low level heuristic is always applied, even if it does not produce any improvement or worsens the current solution of the problem.

Pure random hyperheuristics have been tested by many researchers for different optimisation problems including the sales summit scheduling problem [21], the project presentation scheduling problem [23], the nurse rostering problem [24], the channel assignment problem in mobile communications [46], the shelf space allocation problem [4], university course timetabling problem [9], the trainer scheduling problem [18] and job shop scheduling problem [62]. Note that pure random approach is usually used as a point of comparison for the performance of individual low level heuristics (where it is usually better) or other, more intelligent hyperheuristic methods (where it is usually worse).

The main disadvantage of a purely random approach is that the quality of the solution obtained depends on the chance of selecting a "good" sequence of low level heuristics. In order to avoid the search becoming trapped into poor regions of the solution space, modifications to the pure random approach are needed. These modifications may concern the rule of accepting non-improving low level heuristics during the hyperheuristic run or the way of applying low level heuristic at each decision point. In [21] and [22], Cowling et al. compare different versions of random hyperheuristics applied to a practical problem of

scheduling meetings for a sales summit. One version of hyperheuristic applies a randomly selected low level heuristic only once at each iteration (simple random hyperheuristic), another one conducts simple local search by applying a randomly selected low level heuristic in a descent fashion, i.e. the low level heuristic is reapplied as long as it continues to produce an improvement to the current solution (descent hyperheuristic). The results of experiments show that descent hyperheuristics perform better than simple versions. In addition, two acceptance rules are considered: accept all low level heuristics or only improving ones. Note that the "only improving" strategy may cause the hyperheuristic to get stuck in a local optimum at the early stages of the search when there is no a single low level heuristic in the set which is able to improve the current solution. This situation is likely to happen when the number of low level heuristics used is small. The natural way to overcome the problems related to both extreme acceptance strategies is to allow non-improving low level heuristics to be applied with some probability.

In their seminal work, Fisher and Thompson [30], [31] use the term "unbiased random process" for their random hyperheuristic. They consider a classical job shop scheduling problem [7] and just two simple priority rules to select the next job to be scheduled at each machine. In [31], the sequences of rules constructed by the unbiased random process, produced better schedules than both rules in the set applied separately. Fisher and Thompson [31] use reinforcement learning techniques [63] in order to determine the probabilities of selecting specific decision rules at any point of the scheduling process. Each time a new sequence of rules is generated, the resulting schedule produced by this sequence is compared to the standard schedule (which is usually the best schedule found so far). If the current schedule is better than the standard, it becomes a new standard, otherwise the standard schedule remains the same. Then the points of difference in the sequences of rules for both schedules are determined and the probabilities of selecting rules at these points are adjusted and used for the next schedule generation. The results reported in [31] show an improved average performance of the method with learning if compared to a purely random approach. Fisher and Thompson conclude that probabilistic learning approach might be more effective if a larger number of decision rules is combined rather than just two rules considered in [31].

A number of hyperheuristics recently presented in the literature are based on Monte Carlo methods. A general Monte Carlo method [33] uses probability for accepting a new non-improving solution which decreases when the difference δ in objective values between the new and the current (best) solutions increases (for minimisation problem). In the context of a hyperheuristic, the probability of accepting non-improving low level heuristic is considered. This probability can be computed in different ways. Ayob and Kendall in [3] apply a Monte Carlo hyperheuristic to optimise electrical component placement on a printed circuit board. The set of 6 low level heuristics represents different versions of simple swap moves. The authors consider linear and exponential functions of δ to define acceptance probabilities for non-improving low level

heuristics. The best results are achieved for an exponential function where time factors are taken into account for calculating probabilities, i.e. an acceptance probability at any decision point depends not only on δ, but on the time elapsed since a start of a hyperheuristic and on the number of consecutive non-improving iterations. Such a method for calculating probabilities is similar to that used in another approach from the Monte Carlo family, simulated annealing [1], where the acceptance probability is a function of δ and control parameter (temperature) which gradually decreases as the number of iterations grows. Bai and Kendall in [4] develop a simulated annealing based hyperheuristic to solve a shelf space allocation problem. Combining 12 low level heuristics, their hyperheuristic approach outperforms two versions of simulated annealing metaheuristic and produce high quality results for different problem instances. Strong performance of simulated annealing based hyperheuristics for a sales summit scheduling problem and for a trainer scheduling problem are reported by Soubeiga in [61] and Chakhlevitch in [16], respectively.

Another example of random hyperheuristics are hyperheuristics based on the variants of Great Deluge algorithm [27], considered by Kendall and Mohamad in [45] and [46]. A Great Deluge hyperheuristic randomly selects a low level heuristic at each iteration and applies it if the objective value returned by the low level heuristic is better than some specified threshold level. The level is initially set to the objective value of the starting solution and then adjusted after each iteration, i.e. it decreases (for minimisation problems) at a fixed rate [45]. This strategy allows non-improving moves to be accepted frequently at the early stages of a hyperheuristic run and very occasionally towards the end. In [46], another method to control accepting non-improving low level heuristics is used: a low level heuristic is accepted only if its returned objective value is reasonably close to the objective value of the current solution. This is implemented by introducing a parameter which represents the maximum possible distance between two solutions. Both hyperheuristics produce results of a good quality for the channel assignment problem (see [45] and [46]).

Random hyperheuristics are simple and fast, and can be easily implemented and applied to any optimisation problem. The results achieved by basic random hyperheuristics can be used as benchmarks for evaluating other hyperheuristic approaches. Hybridisation with more advanced techniques for accepting low level heuristics make random hyperheuristics competitive with other approaches. This is probably the case since the outcome of applying each low level heuristic depends on effectively random factors and their behaviour at different decision points is difficult to predict.

1.3 Greedy and peckish hyperheuristics

A basic greedy hyperheuristic simply selects and applies at each decision point the low level heuristic which produces the largest improvement to the current objective value (or the smallest deterioration if no improving low level heuristic exists at some iteration). Two versions of the greedy strategy can be used: one accepts only improving low level heuristics and another allows non-improving low level heuristics to be applied. The second version is advantageous since it prevents a hyperheuristic from stopping too early when no improving low level heuristic is available in the set. Note that a greedy hyperheuristic requires preliminary evaluation of each low level heuristic in the set in order to select the best one which makes it much slower than a hyperheuristic based on the random choice. Greedy hyperheuristics are considered by Cowling et al. [21] – [24] and by Cowling and Chakhlevitch [18], [19] and their results are mainly used as benchmarks for other methods.

The main disadvantage of a greedy hyperheuristic is its limited ability to effectively explore the search space leaving many regions with potentially strong solutions unvisited. To overcome problems associated with local optima, Cowling and Chakhlevitch [18], [19] develop a group of "peckish" hyperheuristics which combine greedy and random mechanisms for managing the choice of low level heuristics. A peckish hyperheuristic randomly selects a low level heuristic from the candidate list of the "best" (not necessarily improving) ones. The length of the candidate list may be adjusted in order to achieve a good ratio between intensification and diversification in the search. The authors consider four versions of peckish hyperheuristics using both static and dynamic candidate lists and apply them to the trainer scheduling problem

Note that peckish hyperheuristics may be particularly useful when a large set of low level heuristics is used since they present a scalable method capable of handling any number of low level heuristics. In [18] and [19], the authors present a generic idea for constructing a large set of low level heuristics for complex little-studied optimisation problems. For such problems, there is no obvious choice of low level heuristics and traditional neighbourhoods (swap, insert or replace moves) are not easily applicable. Tackling the real-world trainer scheduling problem, Cowling and Chakhlevitch show how a set of low level heuristics can be formed by combining simple selection rules for events and resources. See also [16] for more details.

Greedy and peckish hyperheuristics can be readily applied for different optimisation problems due to their simplicity and high level of generality. However, their slow speed makes them unfavourable for the problems where the time to construct solutions is a crucial factor.

1.4 Metaheuristic-based hyperheuristics

A conventional metaheuristic is a local search based method which operates in a *solution* space of the problem and employs some strategy to escape local

optima. Taking into account a proven record of successful applications of metaheuristics to solving complex real-world optimisation problems, the question of how effectively metaheuristics can perform the search over a *heuristic* space is of a great research interest. Various metaheuristic approaches and their hybrids have been tested as a high level heuristic selectors in the last few years and we refer to them as metaheuristic-based hyperheuristics in this section. We start with hyperheuristics based on genetic algorithms (GAs) which have created the foundation for current research in hyperheuristics.

1.4.1 GA-based hyperheuristics

Early efforts to search for an effective solution method for a problem rather than for a good solution are related to the development of GAs with indirect chromosome encoding. A chromosome in a traditional GA encodes a solution to the problem directly (by means of binary strings, permutations, etc.). However, the solutions to many real-world problems have a very complex structure which makes the direct encoding extremely difficult. The other disadvantages of direct encoding are a large length of the chromosomes for large problems and the need of specific repair operators to maintain the feasibility of solutions. A number of indirect GAs developed in 1990s were aimed to overcome these limitations.

In indirect GAs each chromosome represents the way a solution is constructed rather than the solution itself. In [64], the chromosome represents a sequence of heuristics to be applied to the initial solution. The ith gene of the chromosome encodes the heuristic number in the set of possible heuristics and indicates that this heuristic will be applied at the ith step of generating a new solution. Note that a chromosome in an indirect GA may also encode the order in which the (single) heuristic is applied, but we do not consider such an approach in this review. Table 1.2 provides a summary of the methods which can be classified as GA-based hyperheuristics.

The idea of indirect encoding was first implemented by Norenkov [54] for a scheduling problem connected with the CAD system hardware design and by Fang et al. [28] for an open shop scheduling problem. Fang et al. [28] use a set of eight simple dispatching rules as low level heuristics. The chromosome is organized as a sequence of pairs of genes. The first gene in each pair represents the heuristic and the second one represents the uncompleted job whose operation will be scheduled by applying this heuristic. The results produced by the GA are very close to the best previously found for the most of the benchmark open shop problems and even better for some instances.

Another successful application of indirect GA is reported by Hart et al. [42], [43]. They present a GA-based approach to tackle the real-world scheduling problem of a chicken processing company. The problem is heavily constrained and required several days of work of a human expert to produce a practical schedule. The goal is to schedule chicken catching squads and lorries to deliver a set of orders to the factories to ensure that the factories will

Table 1.2. Hyperheuristics based on genetic algorithms

Papers	Details
Fang et al. [28] Hart et al. [42], [43]	Indirect GA with a block structure of the chromosome; each block contains combination of LLH and domain-specific information
Norenkov and Goodman [55] Dorndorf and Pesch [25] Hart and Ross [41]	The length (or dimensions, in case of matrix representation) of a chromosome is determined by the value(s) of problem-specific parameter(s)
Terashima-Marín et al. [64] Hart and Ross [41]	LLHs performing different actions are combined in the chromosome
Cowling et al. [20] Han et al. [39] Han and Kendall [37], [38]	A chromosome determines a sequence of LLHs and the order of their application; the length of the chromosome is either fixed or adaptively adjusted
Ross et al. [59]	A chromosome encodes characteristics of the problem instances together with associated LLHs

be supplied with the birds continuously throughout the day. The problem is decomposed into two stages and two separate GAs are implemented in each stage respectively. We mention here only the first one, an *assignment GA*, which performs the assignment of tasks to squads. An assignment GA uses an indirect chromosome representation thus evolving an assignment strategy and then applying that strategy to construct a schedule. The strategy incorporates the combinations of two heuristics for each order: one heuristic for splitting the order into tasks and the second heuristic for assigning the tasks to squads. The chromosome consists of four sections. The first section contains problem specific information, expressing certain fixed criteria which must be satisfied by every solution. Including such information into the chromosome allows the reduction of the search space. The second section contains the permutation of the factory orders, i.e. the sequence in which the orders will be processed by the strategy. The third and the fourth sections of the chromosome specify for each order the splitting and the assignment heuristics respectively. The GA uses specially designed crossover and mutation operators for each section of the chromosome. An assignment GA in [42], [43] produces practical schedules with almost all constraints satisfied.

Norenkov and Goodman in [55] further develop the approach introduced in [54] and refer to it as a Heuristic Combination Method (HCM). In [55], HCM is applied to solve multistage job-shop scheduling problems. The authors consider two parts of a process of schedule synthesis, specifically job ordering and the assignment of the jobs to servers, and define a set of simple heuristic rules for each part. The composition of the rules for both parts forms the set of heuristics, and the objective is to find the optimal sequence of application of these heuristics. The chromosome is represented as a matrix of size $q * N$,where N is the number of jobs and q is the number of successive service stages each job passes during its processing. The schedule is constructed con-

secutively for each service stage by adding one job in each step. The element (i, j) of the matrix refers to the heuristic which is applied on the jth step of schedule synthesis at the ith stage of service. The jth step here means that $j-1$ jobs have been already scheduled and job j is due to be placed into the schedule. Given such a representation, the authors define specific horizontal and vertical crossover operators (crossover applied to rows and columns of the matrices respectively). They present several evolutionary algorithms based on the above chromosome representation which have been successfully applied to some benchmark job-shop scheduling problems.

A GA developed by Dorndorf and Pesch [25] evolves the sequence of low level heuristics for minimising makespan in job shop scheduling problems. The set of low level heuristics consists of 12 well-known priority rules. A chromosome consists of $n-1$ genes, where n is the number of operations to be scheduled, and each gene encodes the priority rule to be applied to schedule one operation. The crossover operator exchanges substrings of priority rules in two chromosomes and the mutation operator replaces the rule in the randomly selected position of the chromosome with another, randomly selected rule. Although the indirect GA loses to other heuristic methods (such as the Shifting Bottleneck heuristic [2], a conventional GA and simulated annealing) both in solution quality and in speed, it is easy to implement and it shows robustness to problem changes. Hart and Ross [41] extend the approach presented in [25] and develop the method they call HGA (heuristically-guided GA) to solve a dynamic job shop problem. Each gene of a chromosome in HGA now encodes a pair (*Method, Heuristic*) where *Method* represents one of the two algorithms used to calculate the conflicting set of operations at each iteration (Dorndorf and Pesch in [25] consider only one such algorithm) and *Heuristic* is one of the 12 priority rules used to select an operation from the conflicting set. Hart and Ross show that switching between two scheduling methods during schedule construction is beneficial and their method outperforms other heuristics for many instances.

Terashima-Marín et al. [64] investigate the effectiveness of an indirect GA applied to a real-world examination timetabling problem. The authors consider the performance of Brelaz's well-known graph-colouring algorithm [6] combined with heuristics for handling different types of problem constraints. The performance varies for the different problems from the test set and depends on the heuristics chosen. Since there is no evidence which combination of heuristics will be the most suitable for solving any particular problem, Terashima-Marín et al. develop an indirect GA to evolve combinations of heuristics and find the best one for any instance. They specify three different algorithms for solving graph colouring problems and two sets of heuristics for decision making. At the first decision point the nodes of the graph are ordered for the colouring algorithm (variable ordering) and at the second decision point, the order algorithm will select the colours for a node (value ordering). The chromosome encoding is the 10-position array of characters which represent two combinations of graph colouring method with variable ordering and

value ordering heuristics, the condition for switching from the first combination to the second, the parameter for the specific condition, and the indicator of the method of handling the constraints. The graph colouring methods include Brelaz's algorithm [6] and two procedures which involve backtracking and forward checking mechanisms respectively. The purpose of the two combinations of the methods and the heuristics in the chromosome is to handle the situations when the first combination becomes inefficient (performs too many backtracking steps, exceeds time limit) or just to mix two combinations in the hope of obtaining a better solution. The solutions obtained for all tested problems by applying GA with such chromosome representation are superior to those produced by Brelaz's graph colouring algorithm (see [64]).

The GA-based approaches considered so far are rather domain-specific. Indirect GAs are designed to solve different instances of the particular problems and are shown to be highly efficient. Since some portion of problem-specific information is usually injected into a chromosome (which, in turn, leads to problem-specific genetic operators), such methods can not be used for different problems. However, the indirect GA approaches described above are generalised in some recent work.

In [20], Cowling et al. develop a GA-based hyperheuristic approach, called *hyper-GA*, and test it on a simple model for a real-world trainer scheduling problem. An indirect GA operates at a higher level and evolves a sequence of low-level heuristics from a given set. The low-level heuristics are then applied in the order they appear in the sequence to find a good solution of the problem instance. The set of low-level heuristics contains twelve problem-specific heuristics based on combinations of adding, swapping, and deleting events in the schedule. The chromosome for a hyper-GA represents a sequence of integers corresponding to low-level heuristics. The length of the chromosome is equal to the number of low-level heuristics in the set so that each heuristic could be possibly present exactly once in the chromosome. Thus, each individual in a hyper-GA population encodes a sequence of low-level heuristics and indicates which heuristics to apply and in what order. A hyper-GA uses a one-point crossover operator and a mutation operator which replaces the low level heuristics in randomly selected positions of the chromosome by other low level heuristics from the set. The hyper-GA presented in [20] produces significantly better solutions than individual low-level heuristics and outperforms the direct GA and memetic algorithm for all 5 instances of the problem both in the quality of the solutions and CPU time used. The analysis of the behaviour of hyper-GA reveals that the hyperheuristic tends to change the range of low-level heuristics in chromosomes as the search progresses selecting more often the heuristics which lead to improved solutions.

Han et al. further improve hyper-GA in [39]. Since the optimal length of the chromosome for hyper-GA is unknown, they developed a mechanism that adaptively changes the chromosome length during the search. This mechanism allows hyper-GA to evolve the best combinations of low-level heuristics which may contain different number of heuristics. Indeed, in some cases it is reason-

able to remove from the sequence the heuristic (or heuristics) which worsen the current solution hence making the chromosome shorter. In other situations, insertion of "good" heuristics into the chromosome may be necessary so that the chromosome becomes longer. The idea of the adaptive length chromosome in hyper-GA is embodied in [39] by introducing specific crossover and mutation operators which operate with groups of genes. A similar approach with variable length of the chromosomes is implemented by Han and Kendall in [37], where they develop a strategy which decides whether to make the chromosome longer or shorter (by means of applying different mutation operators) in order to maintain its length consistent with the average length of the chromosomes over previous generations. Another version of hyper-GA is considered in [38], where the length of the chromosome is regulated by making poorly performing genes tabu for a number of generations. Note that all versions of hyper-GA maintain a high level of generality and have the potential to be applicable to a range of problems with only minor modifications. However, hyper-GA has been applied only to a simplified version of the trainer scheduling problem and no results have been reported for other problems.

Ross et al. [59] propose an interesting GA-based hyperheuristic approach where the fitness of a chromosome is determined by its ability to successfully solve different instances of the same problem. The approach is implemented for a one-dimensional bin-packing problem where many benchmark instances are available from the literature. A chromosome consists of a number of blocks (genes). Each block contains information about the instance of the problem state and low level heuristic associated with this instance. For a bin-packing problem, the information related to the problem state represents the proportions of the items of different sizes remaining to be packed. The genetic operators perform crossovers and mutations either at block level or inside blocks. Each chromosome from the population is tested on different problems from a *training* set in order to calculate its fitness. At every stage of a bin-packing process, the current state of the problem is compared against the instances encoded in the blocks of the chromosome, thus determining the block representing the closest instance. The low level heuristic associated with the latter instance is then applied to the actual problem state. The fittest chromosome after a specified number of generations is used to solve problems from a *test* set. Ross et al. [59] report that their GA-based approach achieves optimal solutions for most of the problem instances considered and outperforms each low level heuristic applied separately. Although the idea used in [59] can be applied when considering other problems, the approach has some significant limitations. First, it requires many problem instances to be included into training and test sets which are often unavailable for real-world problems. Second, it can be much more difficult to encode the problem state instance for problems with complex structures than for a relatively simple bin-packing problem, as well as to define a measure of distance between different problem states. Finally, the approach might be expected to be very slow for a range of complex real-world optimisation problems.

1.4.2 Other metaheuristic-based hyperheuristics

A proven record of successful applications of GA-based hyperheuristics to various problems has founded an interest in using other metaheuristics as higher level heuristic selectors. Most of the relevant approaches have been developed over very recent years. The exception is an early publication of Storer et al. [62] where the authors study the effects of performing the search in two different search spaces which are alternatives to the commonly used solution space. The list of papers discussing hyperheuristics based on popular metaheuristic methods is given in Table 1.3.

In [62], Storer et al. consider minimising makespan in a job shop scheduling environment. They define two search spaces namely problem space and heuristic space as a basis for local search algorithms. The idea of the search in a problem space is to apply a base heuristic (for example, simple SPT dispatching rule for job shop problem) to a perturbed versions of the original problem (where processing times for operations are slightly modified) in order to generate alternative sequences of scheduled jobs. These solution sequences are evaluated using original data and the best solution is recorded. The main point of our interest in this work, however, is the concept of *heuristic space*, which is the basis for any hyperheuristic. In [62], the heuristic space contains strings of dispatching rules of a specified length, selected from the set of 6 rules commonly used in machine scheduling. The string of rules defines which rules and in which order should be called by a base heuristic (schedule generator) in the process of schedule construction when a decision about the operation to be scheduled next is required. Apart from random, hill-climbing and steepest descent methods, Storer et al. [62] study the performance of basic versions of popular metaheuristics in heuristic space. Simulated annealing, tabu search and genetic algorithm are applied for searching heuristic space and tested on a range of hard job shop scheduling problems. The authors report the consistency and high quality of results produced by these metaheuristic-based hyperheuristics and conclude that heuristic space search can be very "useful in providing fast solutions to very large problems".

Simulated annealing [1] and tabu search [32] approaches can be used to control the search in heuristic space in a similar manner as they manage the neighbourhoods of problem solutions in conventional Metaheuristics. In the context of a hyperheuristic, both algorithms decide at each iteration whether to accept or to reject the application of a particular low level heuristic to the current solution of the problem, depending on the objective value which would result after applying the low level heuristic.

Chakhlevitch in [16] demonstrates that a simulated annealing hyperheuristic produces more consistent results across a range of instances of a relatively detailed model of a real-world trainer scheduling problem than a problem-specific version of a simulated annealing metaheuristic. Additionally, the hyperheuristic approach is less sensitive to the choice of initial solution for the problem than its metaheuristic counterpart. Other examples of simulated an-

Table 1.3. Hyperheuristic based on non-GA metaheuristics

Approach	Papers	Details
Simulated annealing	Bai and Kendall in [4]	Random selection of LLH; probabilistic acceptance criteria
	Storer et al. [62]	
	Soubeiga [61]	
	Chakhlevitch [16]	
Tabu search	Storer et al. [62]	Basic version
	Kendall and Mohd Hussin [47]	Basic version
	Kendall and Mohd Hussin [48]	Hybrids with hill-climbing and great deluge methods; random tabu durations
	Burke et al. [8], [10]	Constructive version
	Burke and Soubeiga [15]	Combines tabu search and reinforcement learning; variable tabu list size
	Burke et al. [13], [9]	
	Dowsland et al. [26]	
	Cowling and Chakhlevitch [18], [19]	Methods with different tabu list contents; tabu list size is either fixed or automatically adjusted
VNS	Qu and Burke [57]	Neighbourhoods of LLH sequences of different lengths are explored

nealing based hyperheuristics can be found in [4] and [61]. We also refer to discussion of these methods in Section 1.2.

Hyperheuristics based on the tabu search metaheuristic have received increasing attention in recent publications. Kendall and Mohd Hussin [48] consider a simple tabu search based hyperheuristic for solving examination timetabling problem. Their hyperheuristic manages a set of 13 low level heuristics based on adding, moving, swapping and removing exams in the timetable. A low level heuristic becomes tabu as soon as it is applied to the current solution irrespective of whether it improves the solution or not. The tabu duration for each low level heuristic is short and fixed and there is no aspiration criterion. This means that a low level heuristic can not be applied while it remains tabu, even if it leads to the largest improvement among all low level heuristics. The best non-tabu low level heuristic is applied instead. Although such a simplified approach is never able to beat the best known results for a range of benchmark timetabling problems, it consistently produces good quality outcomes provided a sufficient amount of CPU time is available. In [47], Kendall and Mohd Hussin consider two more advanced versions of tabu search based hyperheuristic developed in [48]. In the first version, a low level heuristic which improves the previous best solution is applied repeatedly and becomes tabu only when it does not produce further improvements (tabu search hyperheuristic with hill climbing). The second version accepts the best non-improving and non-tabu low level heuristic only if it updates the solution within a certain boundary (the idea used in the great deluge algorithm, see [27]). In addition, random tabu durations from a given range are consid-

ered for both versions. The authors report further improvements to results obtained in [48]. Another tabu search based hyperheuristic approach for tackling examination timetabling problems is proposed by Burke et al. [8]. Instead of starting the search from the previously constructed initial solution, their method starts from a blank timetable and generates the sequences of low level heuristics which are used for step-by-step timetable construction. They use only two low level heuristics which represent two different ordering strategies widely used in examination timetabling. Tabu search is performed over a space of permutations of these two heuristics. The hyperheuristic outperforms both low level heuristics applied separately, losing, however, to problem-specific approaches on benchmark timetabling problems. A similar approach with six low level heuristics is considered in [10].

Burke and Soubeiga [15] employ tabu lists of poorly performing low level heuristics in a hyperheuristic approach to solving the nurse rostering problem. In their method, low level heuristics compete against each other using rules based on the principles of reinforcement learning. There are 9 low level heuristics in the set which are ranked according to their performance during the hyperheuristic run. At the beginning of the search each low level heuristic receives zero score and the scores are dynamically changed as search progresses. Note that similar idea is used in the hyperheuristic developed by Nareyek [53] and in choice function hyperheuristics ([21] – [24]) which will be discussed in the next section. If the applied low level heuristic yields improvement to the current solution, its score is incremented (positive reinforcement), otherwise it is decreased on a specified number of points (negative reinforcement). However, such a scheme has a disadvantage of repetitive calls of the poorly performing low level heuristics with the highest scores (until the scores become low enough). The highest scores for such low level heuristics have been achieved due to improvements produced in the earlier stages of the search. In order to overcome this problem, each applied non-improving low level heuristic immediately becomes tabu and is released from the tabu list as soon as the current solution is changed by some other low level heuristic. Therefore, the size of the tabu list is variable and depends on the number of low level heuristics applied before the current solution is changed. The authors present results of a high quality for different instances of the nurse scheduling problem. They also claim the robustness of their approach across a range of instances of different problems. This claim is supported in [13] where the hyperheuristic is applied to the university course timetabling problem, outperforming two problem-tailored metaheuristics in terms of feasibility of solutions and showing competitiveness in terms of solution quality. In [9], tabu search based hyperheuristic approaches developed in [15] are adapted to solving multiobjective optimisation problems of space allocation and course timetabling. Finally, similar approach is used within simulated annealing framework to solve a complex shipper rationalisation problem (see [26]).

Cowling and Chakhlevitch [18], [19] present different versions of tabu search based hyperheuristics for the trainer scheduling problem. These hy-

perheuristics are designed to manage a large collection of (nearly 100) low level heuristics. The basic hyperheuristic employs a tabu list of recently called low level heuristics which have not improved the objective function. The algorithm greedily selects the best low level heuristic at each iteration of the search. If such a heuristic leads to an improved objective function value, it is always accepted and released from the tabu list if present; a non-improving heuristic is chosen only if it is not in the tabu list and immediately becomes tabu after its application. The authors test several versions of hyperheuristics with fixed and dynamically changed tabu list sizes as well as with different contents of tabu list such as recently applied non-improving low level heuristics and recently modified events. The results reported for tabu search hyperheuristics are advantageous to those obtained for other hyperheuristic methods considered in [18] and [19].

Qu and Burke in [57] develop a variable neighbourhood search (VNS) [40] hyperheuristic for the examination timetabling problem. As in [10], a timetable is generated by consecutively applying constructive low level heuristics in an order specified by their sequence to schedule exams. The search is performed over a space of all possible sequences of low level heuristics of a given length. In [57], the neighbourhoods are defined by random replacement of two, three, four and five low level heuristics in a sequence. The hyperheuristic explores each neighbourhood for a specified number of iterations before switching to another neighbourhood. However, the results of [57] do not demonstrate this approach to be advantageous when compared to other hyperheuristics which use a single neighbourhood.

To conclude this section, we note that metaheuristic-based hyperheuristics have been tested on different real-world problems and shown to be very effective, even beating state-of-the-art problem-tailored methods on occasion. However, like traditional metaheuristic approaches, such hyperheuristics require fine tuning of parameters (temperature for simulated annealing, tabu list length or tabu tenure for tabu search, crossover and mutation rates for GA, etc.). Although hyperheuristics are often less sensitive to changes of these parameters, there is no guarantee that a hyperheuristic will work equally well for different problems using the same parameter settings. Recall that one of the main goals of a hyperheuristic is to provide a general framework for quickly producing solutions of a good quality for problems from different domains. The ideal hyperheuristic should be parameter-free (or nearly parameter-free) and easily applicable to a new problem without significant modifications and tuning. A few efforts have been undertaken to develop such methods by means of embedding learning techniques into hyperheuristics. We review hyperheuristics with learning in the next section.

Table 1.4. Hyperheuristics with learning

Approach	Papers	Details
Reinforcement learning	Nareyek [53]	Weight adaptation methods
	Burke and Soubeiga [15]	LLH score adjustment within a tabu search framework
	Fisher and Thompson [31]	Adjustment of LLH selection probabilities
Choice function	Cowling et al. [21]-[24] Kendall et al. [49] Soubeiga [61]	Three-component choice functions are used to keep track of the historical performance of LLHs
Learning subsets of LLH	Chakhlevitch and Cowling [17] Chakhlevitch [16]	Different learning criteria and strategies are used to choose effective LLH from a large set
Learning classifier system	Ross et al. [60]	Learning effective combinations of problem states and LLHs for their solving
Case based reasoning	Burke et al. [11]	LLHs suitable for modifying partial solutions of the problem are retrieved from the case base

1.5 Hyperheuristics with learning

Hyperheuristics from this group employ various techniques for learning the historical performance of low level heuristics. A hyperheuristic selects a promising low level heuristic at each decision point based on the information about the effectiveness of each low level heuristic accumulated in earlier stages of its run (or in previous runs). Table 1.4 provides a list of publications together with a brief details of of the techniques used in this area.

One popular learning mechanism which has been employed in a hyperheuristic framework is based on the principles of reinforcement learning [44]. The general idea of such a technique is to "reward" improving low level heuristics at each iteration of the search and "punish" poorly performing ones by means of respectively increasing and decreasing their weights (scores) or probabilities of being selected. The weights of low level heuristics are adaptively changed as the search progresses and reflect the effectiveness of low level heuristics at any stage of the search.

Nareyek in [53] presents a weight adaptation method based on reinforcement learning. He investigates different schemes of selecting the promising heuristics from the set of alternatives during the search. Each heuristic has a weight assigned to it. The weight of a heuristic is changed as soon as a heuristic has been called and its performance has been evaluated. If the choice of the particular heuristic leads to improvement of the objective function, the weight of this heuristic increases, otherwise the weight decreases. The weights are bounded from above and from below. Nareyek considers different schemes for weight adaptation during the search and separates these schemes for the cases of improvement and deterioration. The current values of the weights express the information about the past experience of using the corresponding

heuristics and depend on the region of the search space under exploration. The author presents two methods of selection of the heuristics based on their weights. The first one is the roulette-wheel approach where the heuristic is randomly selected with the probability proportional to its weight. The second method simply selects the heuristic with the maximum weight. A learning strategy (i.e. a hyperheuristic) combines three components: the weight adaptation scheme for the case of improvement, the scheme for the case of non-improvement, and heuristic selection method. The results of applying different strategies to two real-world optimisation problems (*Orc Quest* problem and the *Logistics Domain*) are reported. The hyperheuristic with the weight adaptation mechanism outperforms the stationary expert strategy even when the latter has a carefully selected combination of weights.

Other examples of hyperheuristics using principles of reinforcement learning include methods of Fisher and Thompson [31] and Burke and Soubeiga [15] (see also [13] and [9]) discussed in previous sections. Note that the former approach employs learning to adjust probabilities of selecting low level heuristics, while the latter uses the learning schemes similar to those in [53].

Cowling et al. in [21] introduce a hyperheuristic approach based on statistical ranking of low level heuristics. In this method, historical information about the recent performance of low level heuristics is accumulated in a *choice function*. The selection of low level heuristic at each decision point depends on the current value of the corresponding choice function. They define the choice function as a "key to capturing the nature of the region of the solution space currently under exploration and deciding which neighbourhood (low-level heuristic) to call next, based on the historical performance of each neighbourhood". In [21], the choice function represents the weighted sum of the three components which reflect recent performance of each low-level heuristic, recent effectiveness of consecutive pairs of low-level heuristics, and the amount of time since the heuristic was last called, respectively. The first two components provide the intensification of the search while the third one is included for diversification. A good balance between intensification and diversification factors allows the hyperheuristic to explore the search space effectively. The weights of the components (denoted by α, β, δ respectively) express their relative importance in the choice function. The choice functions for low-level heuristics are recalculated at each iteration of the hyperheuristic. The general idea of the choice function is that the choice of an effective low-level heuristic at any given time may be stipulated by the recent successful application of the heuristic or by the effectiveness of this heuristic in combination with another heuristic, or, if the local optimum is reached, by the opportunity to redirect the search to a new region of the solution space. Choice function based hyperheuristics produce significantly better results for a simplified model of a real-world sales summit scheduling problem than those provided by the currently used scheduling system.

The limitations of the approach mentioned above are that it requires a warm-up period during which the heuristics should be selected randomly in

order to initialise the values of the choice functions and that the weights α, β,and δ of individual components in the choice function should be manually tuned to achieve the best results. To overcome these limitations, Cowling et al. have developed an adaptive procedure that automatically adjusts the choice function's parameters during the search [22]. The method of parameter adjustment is to "reward" the improving heuristics and to "penalise" the non-improving heuristics ensuring that the best heuristics will be selected frequently and the worst ones will not be chosen very often. Such an adaptive procedure of parameter adjustment makes the hyperheuristic essentially parameter-free. The parameter-free hyperheuristic approach provides further improvements in the quality of the solutions of the sales summit scheduling problem (see [22]). The effectiveness and robustness of the approach are further investigated in [23] and [49] for the project presentation problem and in [24] for the nurse rostering problem. A detailed discussion and analysis of the choice function based hyperheuristics can be also found in [61].

Chakhlevitch and Cowling in [17] and Chakhlevitch in [16] consider the trainer scheduling problem and employ learning strategies embedded into peckish and tabu search based hyperheuristics in order to identify the subsets of the most effective low level heuristics in a large set. One of the reasons for introducing learning techniques is that, given a particular instance of the problem and a large collection of low level heuristics, it is difficult to predict in advance the behaviour of different heuristics. Some low level heuristics may be particularly useful while other ones may bring no or little contribution to the solution process. Moreover, reducing the number of low level heuristics in the set may significantly speed up the search for a better solution. The authors consider two learning strategies. According to the first strategy, a hyperheuristic removes a certain number of the weakest low level heuristics after a fixed number of iterations and then continues its run with a reduced set. In the second strategy, low level heuristics with a poor performance are eliminated continuously during the hyperheuristic run until a required number of the best ones remains in the set. In [17] and [16], several selection criteria for low level heuristic ranking based on their ability to make changes to the current solution, frequency of calls by a hyperheuristic, frequency and magnitude of improvements are tested. The results of the experiments suggest that hyperheuristics with embedded learning strategies outperform hyperheuristics without learning given similar CPU time.

In [60], Ross et al. use a learning classifier system [65] to learn a set of rules for solving one-dimensional bin-packing problems. As in [59], the rule is a combination of a problem state and an associated low level heuristic (see discussion in subsection 1.4.1 of the GA-based method used in [59]). The learning classifier system works on binary representation of rules. The set of benchmark bin-packing problems is divided on two subsets used for training the learning classifier system and for testing the learned rules respectively. The method achieves similar results and has similar disadvantages to the GA-based approach in [59].

Burke et al. in [11] develop a hyperheuristic approach employing case based reasoning for low level heuristic selection and apply it to the examination timetabling problem. The approach has similarities to that presented in [60] which uses the learning classifier system. The timetable is constructed step-by-step by applying a low level heuristic to the partial solution at each step. The appropriate low level heuristic is retrieved from a *case base*. The case base is a collection of cases where each case describes possible partial solution and suggests a low level heuristic which has been previously found to be effective in dealing with such a partial solution. The cases in a base are picked up in a process of solving the problems from a training set. The partial solution in each case is represented by a list of features (properties) which is determined earlier at a knowledge discovery stage by a specific tabu search procedure. After the case base has been formed, it is tested on another set of problems (test set). At each step of timetable construction, the hyperheuristic identifies the case which is the closest to the current partial solution and applies the low level heuristic recorded in this case. The authors demonstrate in [11] that a hyperheuristic with case based low level heuristic selection consistently outperforms individual low level heuristics for a range of timetabling problems.

1.6 Other generic problem solving techniques

In this section we consider a range of AI approaches which are closely related to hyperheuristics. These approaches are aimed to providing a general methodology for solving various instances from a selected problem domain. The main goal is usually either to automatically select a good (ideally, the best) problem solving method (heuristic) from a list of possible alternatives or to learn a good strategy (combination of heuristics) which performs well over a distribution of problem instances. The final choice of the solution method or strategy is based on the historical performance of the alternatives on a training set of problems. Table 1.5 summarises recent developments in this area.

Gratch et al. [35] and Gratch and Chien [34] consider a statistical approach to adaptively solve the real-world problem of scheduling satellite communications. They develop a machine learning system that performs a hill-climbing search over a space of possible combinations of heuristic methods (called control strategies) and returns the most effective combination for the given problem domain. The performance of each control strategy is evaluated by means of statistical techniques. The heuristic scheduling algorithm makes its control decisions applying the corresponding heuristic from the control strategy at each decision point. The authors specify 5 decision points during the process of solving each problem instance with a set of possible low level heuristics to try at each decision point. The sample of the problems for each run of the system is formed by random selection of the specified number of problems from the distribution. The best strategy is determined from several runs of

Table 1.5. Generic methods related to hyperheuristics

Papers	Details
Gratch and Chien [34] Gratch et al. [35]	A statistical approach used to identify the best strategy (combination of LLHs) for solving problems from a given distribution
Minton [51], [52]	Expert system which generates efficient computer programs to solve constraint satisfaction problems
Fink [29] Gupta et al. [36] Petrovic and Qu [56] Burke et al. [14] Lagoudakis and Littman [50]	Various techniques to select a single heuristic from a set of possible alternatives
Randall and Abramson [58]	A generic problem solver based on the linked-list formulation of the problems

the system since the random selection of training problems may result in different learned strategies on different runs. The selected strategy is then used to solve all the problems from a given distribution. Gratch and Chien [34] report a significant improvement in the performance of their adaptive learning approach in comparison to the system which employs a human expert strategy. The learned control strategies outperform the expert strategy both in terms of CPU time required to produce a feasible schedule and the number of problems from the problem distribution solved within a specified resource bound.

Note that although the idea of the adaptive problem solving used in [35] and [34] is applicable to other problem domains, the approach may require significant modifications. Indeed, the adaptive problem solver employs domain-specific knowledge which is expressed not only in the structure of the control strategy, but in the method of exploration of the control strategy space as well. Another limitation of the solver is the problem of the local maxima due to the nature of the hill-climbing search. Finally, the statistical approach to adaptive problem solving implemented in [34] is computationally expensive since it requires a large number of training examples in order to evaluate the performance of the strategy.

Minton in [52] and [51] describes the expert system, Multi-Tac, for solving constraint satisfaction problems [5]. The objective of the system is to produce an efficient Lisp program tailored to particular problem and instance distribution. Since many combinatorial optimisation problems can be formulated as constraint satisfaction problems, the system can be used for solving problems of various natures. Multi-Tac specialises a set of generic heuristics for constraint satisfaction problems for a particular application and performs the search for the best combination of the domain-specific versions of these heuristics. This combination is then used to generate the problem-specific program. As for [34], Minton uses hill-climbing search over the space of heuristic com-

binations is performed and each combination is evaluated on a set of training instances. The system is tested on two well-known NP-hard problems and the resulting synthesised programs are shown to be competitive with the programs developed by human experts.

Fink [29] develops a statistical technique for automatic selection among available problem solving methods (heuristics) for a given problem instance. This technique is implemented in the framework of a sophisticated AI planning system and combines knowledge acquired from the past performance of the methods, for solving other problem instances from the same distribution, with exploration of new alternatives. Incremental learning is used for selection of the solving method. If the past performance data for all methods are available, a weighted random selection among the methods is performed, where a weight for each method represents the probability that the method is the best for a given problem instance (exploitation). If there is no previous data for some method, it is immediately selected and applied (exploration). The technique is tested on a large set of transportation problems and proved to be effective.

Several other approaches have been developed to learn a single best heuristic (strategy) from the set of possible alternatives for solving a range of problems. For example, Gupta et al. in [36] study the application of neural networks to selecting the best heuristic algorithm for a flowshop scheduling problem. Petrovic and Qu [56] and Burke et al. [14] employ case based reasoning to select a heuristic solving method for course timetabling problems. An approach based on reinforcement learning is proposed by Lagoudakis and Littman [50] to select the most efficient algorithm for solving large instances of simple problems of order statistic selection and sorting.

Randall and Abramson in [58] develop a general problem solver for combinatorial optimisation problems. Their solver is based on simulated annealing and tabu search metaheuristics. The crucial point of their system is a modelling representation for combinatorial optimisation problems. The authors introduce an alternative representation based on dynamic data structures, specifically multi-level linked lists. The linked list representation is well suited for many combinatorial optimisation problems. List modelling also allows for the elimination of many constraints which would typically appear in a traditional integer linear programming formulation of the problem since the range of possible values can be defined for the elements of the list. Finally, traditional local search moves like swapping, adding and repositioning components of the solution can be easily implemented within the list structure in terms of inserting and deleting elements in the list.

The problem solver has been tested by the authors on many instances of the benchmark combinatorial optimisation problems. The system has been able to produce optimal or close to optimal solutions in most occasions in a reasonable time. However, the approach has been applied only to relatively easy problems, which require only one level of sublists for their list-based formulation. Note that the majority of real-world problems are much more complicated and multi-level list structures may be needed for their representation. There is no

evidence in [58] that the solver would perform well on such problems. Another significant drawback of the solver is a substantial amount of computer time required for parameter tuning.

1.7 Conclusions

Hyperheuristics represent an interesting direction in the development of generic solving techniques for combinatorial optimisation problems. Although general problem solving methods have received increased attention among researchers over the last two decades, most of the approaches presented in the literature have been designed to solve various instances of a particular problem rather than to be applied to a range of different problems. One of the main advantages of hyperheuristics over traditional problem-tailored approaches is their reapplicability and robustness across different problem domains. Development of problem-independent hyperheuristic approaches is an important and challenging task and research to date provides only a few promising initial steps in this direction.

In this chapter we have reviewed a wide spectrum of techniques which can be classified as hyperheuristics as well as a range of approaches closely related to them. In our review we have opted for a detailed analysis of the ideas lying behind different hyperheuristic approaches in order to identify their advantages and drawbacks. We can mention the following common limitations.

- Some hyperheuristic techniques make use of additional problem specific knowledge. For example, such knowledge can be used to describe the current state of the problem in order to select a suitable low level heuristic in hyperheuristics employing learning classifier systems. In indirect GAs, a portion of problem-specific information is often injected into the chromosome.
- For many hyperheuristics, a significant amount of parameter tuning is required in order to find good parameter settings for a given problem.
- A large number of problem instances may be required for training and testing of the method in order to accumulate enough knowledge to make the right choice of low level heuristics. However, for many real-world problems the problem data are not easily available and randomly generated instances may not adequately represent the real distribution.
- Many hyperheuristic methods are only tested on a relatively simple benchmark problems for which the best solutions (often optimal) as well as effective low level heuristics are known in advance. There is no evidence that such hyperheuristics would be effective in more complex real-world situations.

Future research efforts in the area of hyperheuristics should be undertaken in order to overcome these limitations. Development of effective parameter-free hyperheuristics, methods for automatic parameter tuning in hyperheuristics,

and creating new techniques with a clear boundaries between the hyperheuristic (higher level) and problem-specific (lower level) components are important research directions. Hyperheuristics should be tested on a wider range of real-world optimisation problems of different nature which would be the key to proving their suitability as a fundamental component of future generic optimisation software.

References

1. E. H. L. Aarts, J. H. M. Korst, and P. J. M. van Laarhoven. Simulated annealing. In E. H. L. Aarts and J. K. Lenstra (eds.). *Local Search in Combinatorial Optimisation.* John Wiley & Sons (1997) 91-120
2. J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science* 34 (1988) 391-401
3. M. Ayob and G. Kendall. A Monte Carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. In *Proceedings of the 2003 International Conference on Intelligent Technologies (InTech2003).* Thailand (2003) 132-141
4. R. Bai and G. Kendall. An investigation of automated planograms using a simulated annealing based hyper-heuristic. In *Proceedings of the 5th Metaheuristics International Conference (MIC2003).* Kyoto, Japan, 23-25 August 2003
5. S. Brailsford, C. Potts, and B. Smith. Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research* 119 (1999) 557-581
6. D. Brelaz, New methods to colour the vertices of the graph. *Communications of the ACM* 22 (1979) 251-256
7. P. Brucker. *Scheduling Algorithms.* Springer-Verlag (1995)
8. E. Burke, M. Dror, S. Petrovic, and R. Qu. Hybrid graph heuristics within a hyper-heuristic approach to exam timetabling problems. In: B. L. Golden, S. Raghavan, and E. A. Wasil (eds.). *The Next Wave in Computing, Optimisation and Decision Technologies.* Conference Volume of the 9th INFORMS Computing Society Conference. Springer (2005) 79-91
9. E. K. Burke, J. D. Landa Silva, and E. Soubeiga. Multi-objective hyper-heuristic approaches for space allocation and timetabling. In: T. Ibaraki, K. Nonobe, and M. Yagiura (eds.). *Metaheuristics: Progress as Real Problem Solvers.* Selected Papers from the 5th Metaheuristics International Conference (MIC2003). Operations Research/Computer Science Interfaces Series, Vol. 32, Springer (2005) 129-158
10. E. Burke, A. Meisels, S. Petrovic, and R. Qu. A graph-based hyper heuristic for timetabling problems. *Technical Report NOTTCS-TR-2004-9*, School of Computer Science and Information Technology, University of Nottingham (2004)
11. E. Burke, S. Petrovic, and R. Qu. Case based heuristic selection for examination timetabling. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02).* Orchid Country Club, Singapore (2002) 277-281
12. E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg. Hyperheuristics: an emerging direction in modern search technology. In: F. Glover

and G. A. Kochenberger (eds.). *Handbook of Metaheuristics*. Kluwer Academic Publishers (2003) 457-474
13. E. Burke, G. Kendall, and E. Soubeiga. A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics* 9 (2003) 451-470
14. E. K. Burke, B. L. MacCarthy, S. Petrovic, and R. Qu. Knowledge discovery in a hyper-heuristic for course timetabling using case-based reasoning. In: *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT2002)*. Ghent, Belgium (2002) 90-103
15. E. Burke and E. Soubeiga. Scheduling nurses using a tabu-search hyperheuristic. In: G. Kendall, E. Burke, and S. Petrovic (eds.). *Proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA2003)*. Nottingham, UK (2003) 197-218
16. K. Chakhlevitch. A hyperheuristic methodology for real-world scheduling. *PhD Thesis*, Department of Computing, University of Bradford, UK (2006)
17. K. Chakhlevitch and P. Cowling. Choosing the fittest subset of low level heuristics in a hyperheuristic framework. In: G. R. Raidl, J. Gottlieb (eds.). *Evolutionary Computation in Combinatorial Optimization: Proceedings of the 5th European Conference, EvoCOP 2005*. Springer Lecture Notes in Computer Science, vol. 3448. Springer-Verlag (2005) 23-33
18. P. Cowling and K. Chakhlevitch. Hyperheuristics for managing a large collection of low level heuristics to schedule personnel. In *Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC'2003)*. IEEE Press (2003) 1214-1221
19. P. Cowling and K. Chakhlevitch. Using a large set of low level heuristics in a hyperheuristic approach to personnel scheduling. To appear in: K. Dahal, K. C. Tan, and P. I. Cowling (eds.). *Evolutionary Scheduling*. Springer-Verlag (2007)
20. P. Cowling, G. Kendall, and L. Han. An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In *Proceedings of 2002 Congress on Evolutionary Computation (CEC2002)*. IEEE Computer Society Press, Honolulu, USA (2002) 1185-1190
21. P. Cowling, G. Kendall, and E. Soubeiga. A hyperheuristic approach to scheduling a sales summit. In: E. Burke and W. Erben (eds.). *Practice and Theory of Automated Timetabling III: PATAT2000*. Lecture Notes in Computer Science, vol. 2079. Springer-Verlag, Berlin Heidelberg New York (2000) 176-190
22. P. Cowling, G. Kendall, and E. Soubeiga. A parameter-free hyperheuristic for scheduling a sales summit. In *Proceedings of the Third Metaheuristic International Conference (MIC'2001)*. Porto, Portugal (2001) 127-131
23. P. Cowling, G. Kendall, and E. Soubeiga. Hyperheuristics: a tool for rapid prototyping in scheduling and optimisation. In: S. Cagoni, J. Gottlieb, E. Hart, M. Middendorf, and G. Raidl (eds.). *Applications of Evolutionary Computing: Proceedings of Evo Workshops 2002*. Lecture Notes in Computer Science , Vol. 2279 Springer-Verlag, Berlin Heidelberg New York (2002) 1-10
24. P. Cowling, G. Kendall, and E. Soubeiga. Hyperheuristics: a robust optimisation method applied to nurse scheduling. In: J. J. Merelo Guervos et al. (eds.). *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN2002)*. Lecture Notes in Computer Science, Vol. 2439, Springer-Verlag, Berlin Heidelberg New York (2002) 851-860
25. U. Dorndorf and E. Pesch. Evolution based learning in a job shop scheduling environment. *Computers and Operations Research* 22 (1995) 25-40

26. K. Dowsland, E. Soubeiga, and E. Burke. Solving a shipper rationalisation problem with a simulated annealing based hyperheuristic. *Technical Report NOTTCSTR-2004-1*, School of Computer Science and Information Technology, University of Nottingham (2004)
27. G. Dueck. New optimisation heuristics: the great deluge algorithm and the record-to-record travel. *Journal of Computational Physics* 104 (1993) 86-92
28. H.-L. Fang, P. Ross, and D. Corne. A promising hybrid GA/heuristic approach for open-shop scheduling problems. In: A. Cohn (ed.). *Proceedings of ECAI 94: 11th European Conference on Artificial Intelligence*. John Wiley & Sons (1994) 590-594
29. E. Fink. How to solve it automatically: selection among problem-solving methods. In *Proceedings of the 4th International Conference of AI Planning Systems*. AAAI Press (1998) 128-136
30. H. Fisher and G. L. Thompson. Probabilistic learning combinations of local job-shop scheduling rules. Presented at the *Factory Scheduling Conference*, Carnegie Institute of Technology, May 10-12, 1961
31. H. Fisher and G. L. Thompson. Probabilistic learning combinations of local jobshop scheduling rules. In: J. F. Muth, G. L. Thompson (eds.) *Industrial Scheduling*. Prentice Hall, Englewood Cliffs (1963) 225-251
32. F. Glover and M. Laguna. *Tabu search*. Kluwer Academic Publishers, Norwell, MA (1997)
33. F. Glover and M. Laguna. Tabu search. In C. R. Reeves (ed.). *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publications (1993) 70-150
34. J. Gratch and S. Chien. Adaptive problem-solving for large-scale scheduling problems: a case study. *Journal of Artificial Intelligence Research* 4 (1996) 365-396
35. J. Gratch, S. Chien, and G. DeJong. Learning search control knowledge for deep space network scheduling. In *Proceedings of the 10th International Conference on Machine Learning*. Amherst, USA (1993) 135-142
36. J. N. D. Gupta, R. S. Sexton, and E. A. Tunc. Selecting scheduling heuristics using neural networks. *INFORMS Journal on Computing* 12 (2000) 150-162
37. L. Han and G. Kendall. Guided operators for a hyper-heuristic genetic algorithm. In: T. D. Gedeon and L. C. C. Fung (eds.). *Proceedings of AI-2003: Advances in Artificial Intelligence. The 16th Australian Conference on Artificial Intelligence (AI'03)*. Perth, Australia (2003) 807-820
38. L. Han and G. Kendall. An investigation of a tabu assisted hyper-heuristic genetic algorithm. In: *Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC'2003)*. IEEE Computer Society Press, Canberra, Australia (2003) 2230-2237
39. L. Han, G. Kendall, and P. Cowling. An adaptive length chromosome hyperheuristic genetic algorithm for a trainer scheduling problem. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*. Orchid Country Club, Singapore (2002) 267-271
40. P. Hansen and N. Mladenović. Variable neighbourhood search: Principles and applications. *European Journal of Operational Research* 130 (2001) 449-467
41. E. Hart and P. Ross. A heuristic combination method for solving job-shop scheduling problems. In: A. E. Eiben, T. Back, M. Schoenauer, H.-P. Schwefel (eds.) *Parallel Problem Solving from Nature V*. Lecture Notes in Computer Science, Vol. 1498. Springer-Verlag, Berlin Heidelberg New York (1998) 845-854

42. E. Hart, P. Ross, and J. Nelson. Solving a real-world problem using an evolving heuristically driven schedule builder. *Evolutionary Computation* 6 (1998) 61-80
43. E. Hart, P. Ross, and J. Nelson. Scheduling chicken catching – An investigation into the success of a genetic algorithm on a real-world scheduling problem. *Annals of Operations Research* 92 (1999) 363-380
44. L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research* 4 (1996) 237-285
45. G. Kendall and M. Mohamad. Channel assignment in cellular communication using a Great Deluge hyper-heuristic. In *Proceedings of the 2004 IEEE International Conference on Networks (ICON2004)*. Singapore, 16-19 November 2004
46. G. Kendall and M. Mohamad. Channel assignment optimisation using a hyperheuristic. In *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS2004)*. Singapore, 1-3 December, 2004
47. G. Kendall and N. Mohd Hussin. Tabu search hyper-heuristic approach to the examination timetabling problem at University of Technology MARA. In: E. Burke and M. Trick (eds.). *Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT2004)*. Pittsburgh, USA (2004) 199-217
48. G. Kendall and N. Mohd Hussin. An investigation of a tabu search based hyperheuristic for examination timetabling. In G. Kendall, E. Burke, S. Petrovic, and M. Gendreau (eds.). *Multidisciplinary Scheduling: Theory and Applications*. Selected papers from the 1^{st} Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA2003). Springer (2005) 309-328
49. G. Kendall, E. Soubeiga, and P. Cowling. Choice function and random hyperheuristics. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*. Orchid Country Club, Singapore (2002) 667-671
50. M. G. Lagoudakis and M. L. Littman. Algorithm selection using reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning* (2000) 511-518
51. S. Minton. Integrating heuristics for constraint satisfaction problems: a case study. In *AAAI Proceedings* (1993)
52. S. Minton. An analytic learning system for specializing heuristics. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence* (1993)
53. A. Nareyek. Choosing search heuristics by non-stationary reinforcement learning. In: M. Resende and J. de Sousa (eds.). *Metaheuristics: Computer decision-making*. Kluwer Academic Publishers (2003) 523-544
54. I. Norenkov. Scheduling and allocation for simulation and synthesis of CAD system hardware. In *Proceedings of EWITD94, East-West International Conference*. Moscow, ICSTI (1994) 20-24
55. I. Norenkov and E. Goodman, Solving scheduling problems via evolutionary methods for rule sequence optimisation, *Second World Conference on Soft Computing (WSC2)*, June 1997
56. S. Petrovic and R. Qu. Case-based reasoning as a heuristic selector in a hyperheuristic for course timetabling problems. In *Proceedings of the 6th International Conference on Knowledge-Based Intelligent Information Engineering Systems and Allied Technologies (KES'02)*. Crema, Italy (2002) 336-340
57. R. Qu and E. Burke. Hybrid variable neighbourhood hyperheuristics for exam timetabling problems. In *Proceedings of the 6th Metaheuristics International Conference (MIC2005)*. Vienna, Austria (2005)

58. M. Randall and D. Abramson. A general meta-heuristic based solver for combinatorial optimisation problems. *Computational Optimisation and Applications* 20 (2001) 185-210
59. P. Ross, J. G. Marín-Blázquez, S. Schulenburg, and E. Hart. Learning a procedure that can solve hard bin-packing problems: a new GA-based approach to hyper-heuristics. In *Proceedings of the 2003 Genetic and Evolutionary Computation Conference (GECCO2003)*. Berlin, Germany (2003) 1295-1306
60. P. Ross, S. Schulenburg, J. G. Marín-Blázquez, and E. Hart. Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*. Morgan Kauffmann (2002) 942-948
61. E. Soubeiga. Development and application of hyperheuristics to personnel scheduling. *PhD Thesis*, Department of Computer Science, University of Nottingham, UK (2003)
62. R. H. Storer, S. D. Wu, and R. Vaccari. Problem and heuristic search space strategies for job shop scheduling. *ORSA Journal on Computing* 7 (1995) 453-467
63. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press (1998)
64. H. Terashima-Marín, P. Ross, and M. Valenzuela-Rendón. Evolution of constraint satisfaction strategies in examination timetabling. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO99)*. Morgan Kaufmann (1999) 635-642
65. S. W.Wilson. Classifier systems based on accuracy. *Evolutionary Computation* 3 (1995) 149-175