A Grouping Hyper-Heuristic Framework: Application on Graph Colouring

Anas Elhag, Ender Özcan

ASAP Research Group
School of Computer Science
University of Nottingham
Jubilee Campus, NG8 1BB Nottingham, UK

Abstract

Grouping problems are hard to solve combinatorial optimisation problems which require partitioning of objects into a minimum number of subsets while a given objective is simultaneously optimized. Selection hyper-heuristics are high level general purpose search methodologies that operate on a space formed by a set of low level heuristics rather than solutions. Most of the recently proposed selection hyper-heuristics are iterative and make use of two key methods which are employed successively; heuristic selection and move acceptance. At each step, a new solution is produced after a selected heuristic is applied to the solution in hand and then the move acceptance method is used to decide whether the resultant solution replaces the current one or not. In this study, we present a selection hyper-heuristic framework including a fixed set of low level heuristics specifically designed for grouping problems. The performance of different hyper-heuristics using different components within the framework is investigated on a representative grouping problem of graph colouring. Additionally, the hyper-heuristic performing the best on graph colouring is applied to a benchmark of examination timetabling instances. The empirical results shows that the proposed grouping hyper-heuristic is not only sufficiently general, but also able to achieve high quality solutions for graph colouring and examination timetabling.

Keywords: hyper-heuristics, grouping problems, graph colouring, timetabling

1. Introduction

The task of partitioning a large set of items into a collection of mutually disjoint subsets is a common task in a variety of real-world problems. In a *grouping problem*, the goal is to optimise a given objective (cost, penalty) while achieving the minimum number of subsets (groups). Hence, grouping problems can be formulated as a multi-criteria discrete combinatorial optimisation problem, considering that there is a trade-off between minimizing cost and number of groups, as in graph colouring (Saha et al., 2013), timetabling (Qu et al., 2009) and packing (Falkenauer, 1998).

Email addresses: axe@cs.nott.ac.uk (Anas Elhag), exo@cs.nott.ac.uk (Ender Özcan)

Two crucial components in the design of *grouping algorithms* for solving grouping problems are the candidate solution representation and neighbourhood/move operator(s). A redundant representation scheme which allows equivalent solutions yielding the same grouping creates a huge search space that might impair even the most powerful search algorithm. Many grouping approaches based on genetic algorithms (GAs) have been explored in the scientific literature providing various degrees of success (Falkenauer, 1998; Korkmaz, 2010). In previous studies, it has been observed that traditional operators are rather disruptive and, in many cases, counter productive, hence special operators that are tailored for grouping problems are needed.

There is a growing number of studies on more general and reusable search methodologies applicable to multiple problem domains than the existing specifically tailored solutions to a single problem. *Hyper-heuristics* are such high level search methodologies that search the space formed by low level heuristics, instead of solutions directly for solving hard problems (Burke et al., 2013). There are different types of hyper-heuristics. The focus of this study is *selection* type of high level search methods that mix and control a pre-defined set of low level perturbative heuristics (operators) processing complete solutions at each step under a single-point based search framework.

In this study, we describe a selection hyper-heuristic framework for grouping problems. The framework embeds a set of tailored low level grouping heuristics based on a restricted version of grouping representation, referred to as the *Group Encoding* (Falkenauer, 1998). In contrast to traditional selection hyper-heuristics that use a different set of low level heuristics provided for each different problem domain, in our proposed framework the set of low level heuristics is fixed and the same framework can be used for solving various grouping problems. This adds another level of generality when compared to generic selection hyper-heuristics.

We have investigated the performance of the framework using different selection hyper-heuristic components on a set of well known graph colouring benchmark instances [1]. Additionally, we applied the same hyper-heuristic without any modification to a benchmark of examination timetabling instances in order to examine the generality of the framework. The empirical results show that a learning selection hyper-heuristic developed using the framework turns out to be indeed sufficiently general and reusable. This hyper-heuristic either beats most of the previously proposed approaches tailored for the specific problem in hand or shows that it is highly competitive.

The paper is organised as follows. Section 2 provides an overview of grouping problems, different representation schemes for grouping problems and hyper-heuristics. The details of the proposed selection hyper-heuristic framework including all low level heuristics and different components are given in Section 3. The experimental design and results are discussed in Section 4, while the last section presents concluding remarks and future work.

2. Background

2.1. Grouping Problems

Grouping problems are combinatorial optimisation problems in which a large group of n items, $U = \{x_1, x_2, x_3, ..., x_n\}$, is to be divided into a collection of k ($2 \leq k \leq n-1$) subgroups, u_i ($1 \leq i \leq k$); such that each item $x \in U$ belongs to exactly one subgroup minimizing a given objective (cost/penalty/fitness) and k. Different grouping problems have different constraints, and

[1] ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/color/

introduce different objective (cost) functions, as in graph colouring, timetabling, data clustering and packing (Falkenauer, 1998). In our formulation, we denote a cost function as a decomposable function, $f()$. For a subgroup u_i, the partial cost is denoted as $f(u_i)$, and for a complete solution $U_g = \{u_1, ..., u_k\}$, $f(U_g)$ is the total cost.

$$\textit{minimise} \qquad Z = \left(f(U_g) = \sum_{i=1}^{k} f(u_i), \;\; k \right) \tag{1}$$

$$\textit{subject to} \qquad \bigcup_{i=1}^{k} u_i = U \tag{2}$$

$$u_i \bigcap u_j = \emptyset \qquad \forall_{i,j} \text{ where } i \neq j \tag{3}$$

$$u_i \neq \emptyset \qquad \forall_i \tag{4}$$

In this study, we represent the grouping problem as a discrete two objective multi-criteria problem in which the goal is to optimize the two conflicting objectives in equation (1) above, namely the number of groups which can only take discrete values; and the cost which can take discrete or continuous values depending on the problem. Ideally, these two objectives should be *simultaneously* optimised, although they are clearly conflicting; i.e. a decrease in the number of groups k leads to an increase in the cost. In some cases, there might not be a single optimal solution. Instead, there could be multiple solutions with a trade-off from which a decision maker can choose. Those solutions are identified using the concept of *dominance* (Zitzler and Thiele, 1998) as illustrated in Figure 1.

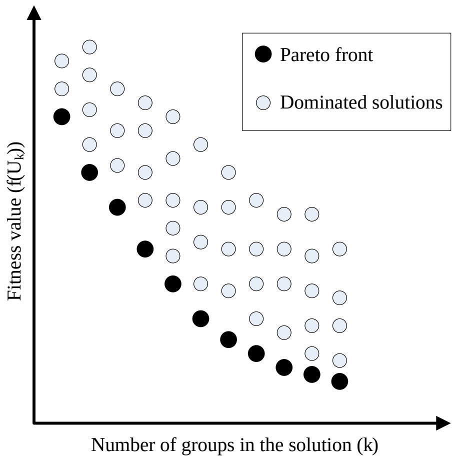

Figure 1: The Dominance concept in Multi-objective optimisation.

A solution x is considered to dominate another solution y, $(x \succ y)$ if, and only if, x is better than y in at least one objective, and x is not worse than y in any of the objectives. The set of the non-dominated solutions is known as the *Pareto optimal set*, and its image in the objective domain is known as the *Pareto optimal front*. This problem is different than a generic multi-objective problem where mostly, there is a region where the Pareto front is driven automatically via a multi-objective algorithm, however, in grouping problems the range of groups is fixed, hence the search methodology can focus on a single objective without ignoring the second one. We use some basic ideas from multi-objective optimisation, but the proposed approach is not a generic multi-objective algorithm as described in Section 3. For more on multi-objective optimisation, readers can refer to (Zitzler and Thiele, 1998; Coello et al., 2007), which is not the focus of this study.

2.1.1. Grouping Representations

Almost all previously proposed grouping approaches are genetic algorithms utilizing various encoding schemes for grouping problems. Those schemes can be classified as *fixed* and *variable* length representations. A fixed-length representation is based on an array of values associated in some way with each item in a set of objects, such as, *Group Numeric Encoding (GNE)* and *Permutation with Separators Encoding (PWS)* (Jones and Beltramo, 1991) which are widely used in the literature. Each location in the array is associated with an item and in GNE, the value at a location indicates the group that the item belongs to, whereas in PWS, that value represents the relative positions of the objects with respect to each other. However, many studies have concluded that such representations have some deficiencies. One of the crucial flaws is the redundancy in the representation. Different candidate solutions under a redundant representation could yield the same grouping of items. For example, assuming that we have 3 objects for grouping under GNE, <1, 1, 2> indicates that there are two subgroups, first and second items form a subgroup while the third item forms another subgroup. <2, 2, 1>, <3, 3, 1>, <1, 1, 3>, <3, 3, 2> and <3, 3, 1> also represent the same grouping. This type of redundancy in return creates a larger search space potentially impairing the search algorithms (Falkenauer, 1992). Also, if the traditional genetic operators are used with those representations, they could significantly damage the good solutions that are being developed during the search. For example, (Du et al., 2004) showed that the standard one-point crossover has various disadvantages and is not suitable for grouping problems.

In (Park and Song, 1998), a linkage-based representation has been proposed, known as the *Locus-Based Adjacency (LBA)* representation, which is a fixed-length representation in which each location represents one object and stores an integer value that represents a link to *another* object in the same group. This representation has been applied in many grouping problems (Handl and Knowles, 2007), but it still suffers the same issues of redundancy and lack of appropriate operators as the former representations. In (Du et al., 2004), a restricted version of the LBA known as the *Linear Linkage Encoding (LLE)* has been proposed, in which backward links are not allowed and, apart from an ending node, each node has only one node pointing to it. LLE successfully eliminates the redundancy problem that debilitates former representations, and has been applied to a variety of grouping problems such as Bin Packing, Graph Colouring and Timetabling (Ülker et al., 2006; Ülker et al., 2008). However, maintaining the LLE links during the search process is costly, particularly while using some genetic operators (Yılmaz and Korkmaz, 2010).

A variable-length representation referred to as the *Grouping Encoding (GE)* has been proposed in (Falkenauer, 1992), along with some suggested tailored genetic operators to be used with it. The basic argument behind the GE is that the grouping representations and associated operators should be carefully designed such that they are *aware* of the nature of the underlying problem; i.e constructing groups that maximise the fitness values of the solutions. Individuals in the GE are separated into two parts, $U_g = \{x_1, x_2, x_3, ..., x_n \mid u_1, u_2, ..., u_k\}$. The first part is referred to as the elements section and is only used to identify the group to which each item belongs. No genetic operators are applied to this part. The second part is referred to as the groups section, in which a list of the groups in the individual is maintained. This is the part to which the genetic operators are applied, and then, all the changes that happen in this section are reflected back on the elements section.

For example, the individual {1 4 2 3 4 4 1 3 2 2 | 1 2 3 4} shows a grouping of 10 items into 4 groups, where the actual grouping is $\{x_1, x_7\}$, $\{x_3, x_9, x_{10}\}$, $\{x_4, x_8\}$, $\{x_2, x_5, x_6\}$. Note that, for a given problem instance, the length of the elements section is fixed, whereas the length of the groups section is not. The GE has been applied in many real world grouping problems including data clustering (Agustín-Blas et al., 2012) and machine-part cell formation (Brown and Sumichrast, 2003). A useful aspect of the GE representation is its ability to address the underlying problem of constructing fit subgroups (with potentially reduced cost). However, the redundancy issue still remains, since more than one solution could represent the same grouping. For example, the solution {1 3 2 4 3 3 1 4 2 2 | 1 2 3 4} produces the same grouping as the solution provided at the beginning of this paragraph. In this study, we used a slightly modified version of the GE representation that is structured to overcome the redundancy issue. Two well-known restrictions are introduced to the standard Falkenauer's GE in order to eliminate this redundancy:

- In an encoding with k groups, subgroups are enumerated using values 1 through k, only.
- Subgroups containing items with lower indices are enumerated first.

Applying these two restrictions, the solution $\{x_1, x_7\}$, $\{x_3, x_9, x_{10}\}$, $\{x_4, x_8\}$, $\{x_2, x_5, x_6\}$ can only be represented as {1 2 3 4 2 2 1 4 3 3 | 1 2 3 4}.

2.2. Graph Colouring and Examination Timetabling

Given a an undirected graph $G = (V, E)$ with a set of vertices $V = \{x_1, x_2, x_3, ..., x_n\}$, and a set of edges E, where $e_{x_p,x_q} \in \{0, 1\}$ represents whether there is an edge between two given vertices (1) $(x_p, x_q \in V)$ or not (0), Graph Colouring Problem (GCP) requires colouring of vertices using a given number of colours such that none of the adjacent vertices (connected with an edge) are in the same colour, hence *conflict free*. The minimum number of colours that achieves a conflict free colouring is also referred to as the *chromatic number* ($\chi(G)$). Determining the chromatic number of a given graph G, and finding out whether G is k-colourable (whether k colours are sufficient to create a conflict free graph) are well-known variants of the GCP which are NP-hard and NP-complete, respectively (Saha et al., 2013).

In this study, we formulate minimum colouring variant of GCP as a multi-criteria grouping problem. Assuming that the number of colours is denoted as k, where $2 \leq k \leq n - 1$, and v_i represents a subgroup of V in which the vertices are assigned to the same colour i, $(1 \leq i \leq k)$, the definition of the grouping problem defined in section 2.1 follows, taking *colours* $\Leftrightarrow$ *groups*, $V \Leftrightarrow U$, and $v_i \Leftrightarrow u_i$. The cost can be measured using the equation 5.

$$f(u_i) = f(v_i) = \sum_{p,q} e_{x_p,x_q} \quad , \forall x_p, x_q \in v_i \ \ and \ e_{x_p,x_q} \in \{0, 1\}, where \ p < q \tag{5}$$

Graph colouring is one of the extensively studies areas of research, yet many studies have been emerging. Many techniques have been proposed for solving many variants of GCP. Exact approaches tend to fail particularly while solving *large* instances, hence many researchers have been working on heuristic approaches. For example, recursive largest fit is a well-known greedy heuristic introduced by (Leighton, 1979). Hertz and Werra (1987) presented the first tabu search implementation which outperforms another local search method, simulated annealing on random dense graph instances. Davis and Mitchell (1991) proposed a coding as an ordering of vertices which could be used in a genetic algorithm. Johnson et al. (1991) presented three simulated annealing implementations based on three neighbouring approaches. Galinier and Hao (1999) have shown that hybridisation of GAs with local search methods are more promising than GAs on their own. In such hybridisation, local search operators are used as intensification methods to explore promising areas of the search space that have found by the GA operators. Avanthay et al. (2003) proposed a variable neighbourhood search algorithm for graph colouring problem. Studies in (Ülker et al., 2006; Ülker et al., 2008; Korkmaz, 2010) and (Kirovski and Potkonjak, 1998) apply generic GAs using some of the genetic representations discussed in section 2.1.1 to solve graph colouring problem. Ülker et al. (2006) proposed special crossover operators for graph colouring, namely Lowest Index Max Crossover (LIMX), Greedy Partition Crossover Lowest Index (GPX-LI) and Greedy Partition Crossover Cardinality Based (GPX-CB). Külahçıoğlu (2007) proposed two modified versions of the LLE representation which are Linear Linkage Encoding With Ending Node Links (LLE-e) and Linear Linkage Encoding With Backward Links (LLE-b), and both of them are tested using genetic operators.

Graph colouring underpins a variety of real-world problems. Examination timetabling is one of those problems which requires allocation of periods/time-slots and other available resources to a given set of examinations taken by a number of students subject to a set of certain constraints. This problem can be formulated as a graph colouring problem, considering that the examinations are the nodes of a graph while the edges are formed using the the examinations that will be taken by each student, since those examinations taken by each student cannot be allocated the same time-slot (colour). Johnson and Trick (1996) showed that the exam timetabling problem can be reduced to graph colouring problem if the task of minimizing the number of exam periods and removing the clashes are considered.

There are many variants of examination timetabling problems. Again, many different heuristic approaches have been the focus of previous studies due to inherent difficulty in solving timetabling problems, ranging from ordering of graph colouring heuristics (Carter and Laporte, 1996) to evolutionary approaches (Burke and Newall, 1998; Paquete et al., 2001). The same formulation of an examination timetabling problem in (Carter et al., 1996) is used in this study. The relevant instances are identified as $Toronto\ a$ in (Qu et al., 2009). Caramia et al. (2001) proposed a family of local search-based timetabling algorithms that apply an optimisation step after each exam allocation attempting to minimize the number of time slots (groups) and the overall penalty simultaneously. Merlot et al. (2002) presented a hybrid algorithm in which a hill climbing phase and a simulated annealing phase are used to improve an initial solution that was developed using a constraint programming phase. Ülker et al. (2006) formulated the problem as a grouping problem and used the linear linkage encoding presented in section 2.1.1 to test a variety of evolutionary approaches on this problem. More on examination timetabling can be found in (Carter et al., 1996; Qu et al., 2009).

2.3. Selection Hyper-Heuristics

A selection hyper-heuristic is a heuristic that explores the space of heuristics formed by set of low level heuristics, each of which performs a search over the solutions while solving a given problem. Özcan et al. (2008) identified the importance of two successive tasks that a selection hyper-heuristic performs:

1. Heuristic selection: a low level heuristic is selected from the low level heuristics set and applied to the current solution, and
2. Move acceptance: a decision is made about whether to accept or reject the resulting solution.

Different combination of selection hyper-heuristic components could yield a different overall performance.

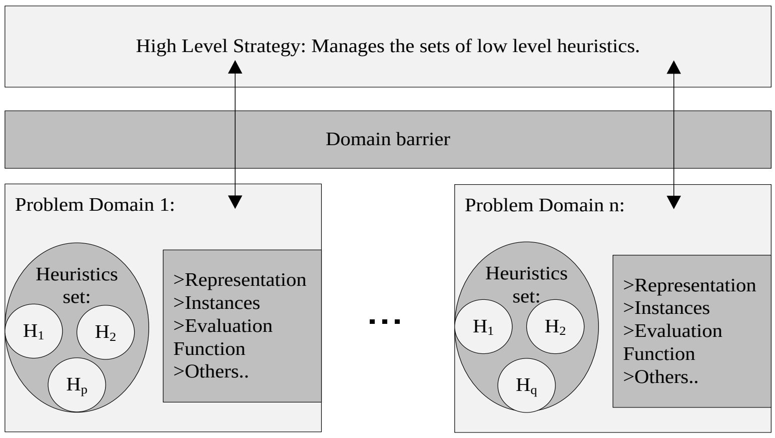

Figure 2: A typical selection hyper-heuristic framework.

There is a conceptual "domain barrier" between the high level selection hyper-heuristic and domain as illustrated in Figure 2. This barrier acts as an information filter and does not allow any problem specific information flow from the low level to high level. This feature is important, since it raises the level of generality of the designed approach operates at and supports re-usability of selection hyper-heuristics and their components, directly.

Various learning and non-learning methods have been suggested for the heuristic selection task. The simplest method used in the literature within the context of selection hyper-heuristics is based on random choice. *Simple Random (SR)* selects one low level heuristic at each iteration purely randomly; i.e. each low level heuristic has an equal chance of being selected regardless to its previous performance. There are some variants of SR. For example, *random permutation* (RP) applies the low level heuristics in a specific order that is randomly determined at the beginning of the search (Bai and Kendall, 2005; Burke et al., 2005; Cowling and Chakhlevitch, 2003).

Some heuristic selection methods utilise learning approaches that use feedback during the search process while the algorithm is running, such as the *reinforcement learning* (RL) (Burke and Soubeiga, 2003). RL assigns a score to each low level heuristic, and then rewards the low level heuristics which improve the current solution and punishes those which do not. This is achieved by means of increasing or decreasing the scores of heuristics. There are different ways of using the scores for choosing a low level heuristic (Burke et al., 2013; Chakhlevitch and Cowling, 2008). One common method is making this decision using *max* utility function. At each decision point, the low level heuristic with the highest score is chosen. If more than one heuristic share the same score, then one of them is selected at random. A modified version of the reinforcement learning (RLM) gives an extra reward to the low level heuristic, if the produced solution was accepted by the move acceptance criteria. Other than these two approaches, researchers have used more sophisticated low level heuristic selection approaches, including meta-heuristics such as GAs, tabu search and great deluge. Hyper-heuristics embedding meta-heuristic components are shown to be very effective and often beat state-of-the-art problem-tailored methods (Chakhlevitch and Cowling, 2008).

Another learning heuristic selection method is referred to as the *Adaptive Dynamic Heuristic Set* selection (DH), which was a component used under an Intelligent Hyper-heuristic Framework built as a general problem solver (Misir et al., 2013). The importance of this hyper-heuristic stems from the fact that it was the winner of an international competition: The First Cross-Domain Heuristic Search Challenge (CHeSC2011) (Burke et al., 2011) that was based on a software benchmark framework known as HyFlex (Burke et al., 2009). Hyflex is a modular Java class library that has recently been implemented to facilitate the development of hyper-heuristics and the design of cross-domain heuristic search methods. DH uses several adaptive features to eliminate subsets of the low level heuristics given, determine effective heuristic pairs and adapt the parameters of some heuristics online in order to cope with the requirements of managing different heuristic sets.

Similarly, various methods are suggested to perform move acceptance. Some of those methods are fairly simple and *deterministic* mechanisms, such as the (AM) method accepts *all moves*, (OI) accepts *only improving* moves; and (IEQ) accepts *improving or equal* moves (Bilgin et al., 2007; Cowling et al., 2000). There are other successful move acceptance methods used within hyper-heuristics which are based on meta-heuristic approaches.

The *Great Deluge (GDEL)* acceptance attempts to escape local optima by accepting moves that worsens the current solution within a range that is determined by a time-varying threshold, τ_t. This threshold starts with a certain value and decreases with time.

$$\tau_t = f_0 + \Delta F \times (1 - \frac{t}{T}) \tag{6}$$

In Equation 6, T is the maximum number of steps, ΔF is an expected range for the maximum fitness change, and f_0 is the final objective value (Dueck, 1993). The *Late Acceptance (LACC)* method is an iterative search technique that was proposed in (Burke and Bykov, 2008). In LACC, improving solutions are directly accepted, while a new worsening solution is compared to an old solution which was visited a fixed number of steps before for acceptance. If the new solution is better than that old solution, it is accepted, even if it could be worse than the current solution from which the new solution is produced. LACC maintains a queue (FIFO list) with a fixed length of solution costs. The initial queue is completely filled by replicating the initial solution's cost. After each improving solution, its cost is enqueued (inserted). Whenever a worsening solution is encountered, then the cost of the best solution is enqueued. The front item always gets deleted after an enqueue operation.

The *Iteration Limited Threshold Accepting (ILTA)* is the acceptance method used in the Intelligent Hyper-heuristic Framework (Misir et al., 2013). ILTA is a list-based threshold move acceptance mechanism that provide an adaptive diversification strategy in connection with the quality of the explored new best solutions. Instead of immediately accepting a worsening solution either within a given range of cost such as is the case in the GDEL, or by comparing it to an old solution such as in the LACC, ILTA *waits* for a predetermined number of iterations, and if no new best solutions can be found during this waiting period, then a worsening solution is accepted. Moreover, the algorithm adaptively changes the waiting times and the length of the list during the search in order to fully capture the nature of the problem being solved. More on hyper-heuristics can be found in (Burke et al., 2013; Chakhlevitch and Cowling, 2008).

3. Grouping Hyper-heuristic Framework

In this study, we describe a single-point based selection hyper-heuristic framework for grouping problems that keeps track of the non-dominated solutions, inspired from the multi-objective optimisation approaches, at each step. Figure 3 shows the layout of our framework. In addition to the common hyper-heuristic domain barrier that ensures the generality of the hyper-heuristic approach, we added the solution representation barrier at which all different types of grouping problems are encoded using a particular representation, and hence they are all treated the same way by the hyper-heuristic framework. As a result, only the definition of the cost function and the problem instances need to be fixed when applying the framework to a grouping problem. The solutions representation along with the low level heuristics remain the same across all domains.

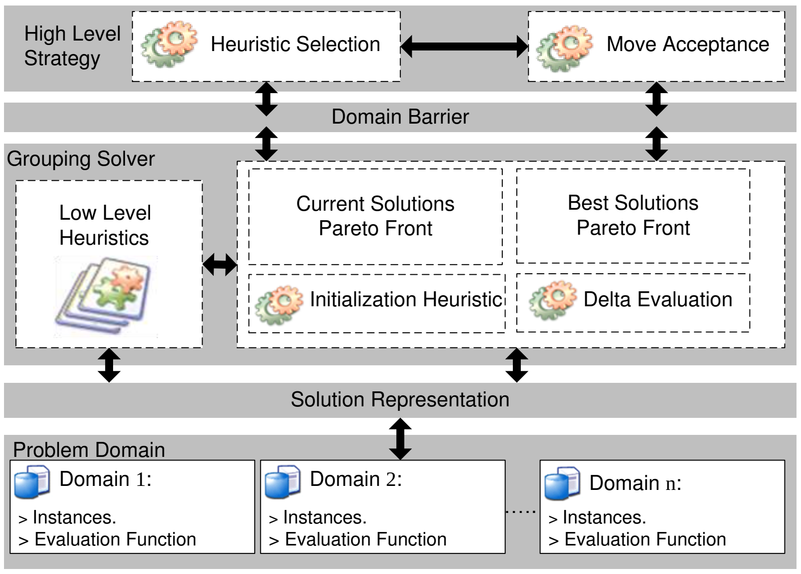

Figure 3: Framework of Hyper-heuristics for Grouping Problems.

Single point-based selection hyper-heuristic frameworks start the search process from an initial solution and deal with a single solution to the problem being solved at each step. On the other hand, most of the multi-objective optimisation algorithms are population-based approaches (Zitzler and Thiele, 1998; Coello et al., 2007). While designing our framework, we have made an attempt to exploit the bi-objective nature of the grouping problems in order to capture the best of the two worlds. The proposed framework is appropriate only for grouping problems not for multi-objective optimisation (which is not within the scope of this study).

3.1. Delta Evaluation

In the context of grouping problems as well as many similar combinatorial optimisation problems, computing the cost of new solutions is the most time consuming part of the hyper-heuristic run. At each decision point, the solution resulting from the application of the selected heuristic has to be evaluated before a decision about whether to accept or reject it is made. Typically, this evaluation is carried out using the whole solution; i.e. calculating the partial cost of each group in the given solution. However, especially with large problem instances, this process is time consuming and greatly affects the solvers with time-based stopping criteria.

In this study, we used *delta evaluation*, which requires computation of partial cost contributions of the groups that have been affected by the application of the selected low level heuristic. This would give a significant time advantage and consequently allow more iterations in each step. For example, if a *change* heuristic is applied on a solution U_{old}, and u_i and u_j are the groups that are involved in the process, the overall cost value $f(U_{new})$ of the resulting solution, U_{new}, is calculated using the equation below, where $u_i^{'}$ and $u_j^{'}$ are the resulting groups after the heuristic is applied:

$$f(U_{new}) = f(U_{old}) - (f(u_i) + f(u_j)) + (f(u_i^{'}) + f(u_j^{'}))$$

3.2. Low Level Heuristics

It has been observed that the use of crossover operators are found to be very disruptive and tends to impair the search rather than guide it for grouping problems in (Falkenauer, 1998; Korkmaz, 2010). Additionally, the proposed framework performs single-point based search, hence, crossover operators are ignored. We have considered a set of effective mutation operators that process complete solutions. Some of them are able to create reasonably large changes on a given solution, diversifying the search process. Some of them are smart operators which are enabled to make small modifications on a given solution leading to a potentially better/improved solution, intensifying the search process. Three different types of mutation operators were consequently developed: *merge*, *divide* and *change*.

Merge Heuristics. The concept of this family of heuristics is to merge two groups, u_i and u_j, into one, u_l. This operation results in *decreasing* the number of grouping in the selected solution. The cost value of the new subgroup u_l is greater than or equal to the combined cost values of u_i and u_j; i.e. $f(u_l) \geq f(u_i) + f(u_j)$. We implemented three versions of the merge heuristic in this study, which differ from each other in the way they choose the groups to be merged in a given solution.

- **M1** merges two randomly selected groups,
- **M2** merges two groups that contain the least number of items, and
- **M3** merges two groups with the lowest partial cost values.

Hence, merge heuristics can be considered as diversifying components.

Divide Heuristics. In a similar fashion, three divide heuristics were implemented in this study, each of which divides a selected group u_i into two groups, u_{i1} and u_{i2}. Applying a divide heuristic results in *increasing* the number of grouping in the selected solution. Also, some of the conflicting items in u_i may end up in different groups, which leads to the elimination of some conflicts. Consequently, the combined cost values of u_{i1} and u_{i2} is less than or equal to the cost value of u_i; i.e. $f(u_{i1}) + f(u_{i2}) \leq f(u_i)$.

- **D1** divides a randomly selected group,
- **D2** divides the group which contains the largest number of items and,
- **D3** divides the group with the highest partial cost value in the selected solution.

In all the divide heuristics, each item has equal probability of ending up in either group. Applying any of these heuristics result in increasing the number of the groups and, hopefully, decreasing the number of conflicts in the selected solution. This characteristic of divide heuristics is of interest as an intensification component and it is used to design a local search algorithm to improve the non-dominated set of solutions within our methodology (see Section 3.3).

Change Heuristics. Merge and divide heuristics produce relatively big jumps in the search space considering that they definitely influence the number of resultant groups for a given solution. However, the change heuristics attempts to make small modifications in a given solution by changing the group of a selected item while maintaining the same number of the groups after they are employed. This family of heuristics has four members.

- **C1** takes a random item x_m from a randomly selected group u_i and moves it into another randomly selected group u_j.
- **C2** finds the item that is causing the highest number of conflicts in a randomly selected group and moves it to another random group.
- **C3** finds the group with the highest number of conflicts, and from that group it takes the item that is causing the highest number of conflicts and moves it to a randomly selected group.
- **C4** is similar to the previous one, except that the item is moved to the group with the minimum number of conflicts rather than a random one.

3.3. Methodology

The pseudo-code of the proposed approach is provided in Algorithm 1.

Firstly, a random solution is created for each k in a given range $[LB, UB]$ in order to have a set of initial solutions with different number of groupings. This range is arbitrarily decided in our experiments aligned with previous work (Ülker et al., 2006). Depending on the particular grouping problem being solved, reasonable upper and lower bounds can automatically be found by using problem-specific knowledge. For instance, for graph colouring problems, these bounds can be determined using algorithms such as the fast maximal clique approximation algorithm or by finding the maximal degree of the graph (the degree of the vertex with the maximum number of neighbours), as discussed in (Ülker et al., 2006). The initialisation heuristic as a part of the problem domain implementation can be a "smart" problem-specific algorithm. However, in order to maintain a high level of generality, we have not used any such problem-specific initialisation. On the other hand, we aimed at creating a non-dominated initial set of solutions by using an initialisation heuristic embedding a problem independent "smart" move appropriate for grouping problems. Starting from $k = LB$ through $k = UB$, a population of $(UB - LB + 1)$ non-dominated solutions are produced, where each solution is created for each integer value of $k \in [LB, UB]$. In order to guarantee non-dominance, each solution for a given $k = i$ is generated at random by assigning each item to one of the groups, initially. The cost for the solution is immediately computed and compared to the cost of the solution at $k = i - 1$. Since the solution at $k = i$ is already worse in terms of the number of groups compared to the solution at $k = i - 1$, its cost value must be better. If this is not the case, that solution is discarded and re-created either randomly or by dividing one of the groups of the solution at $k = i - 1$. The final outcome of the initialisation phase is a non-dominated set of solutions (lines 1-3). Once the initialisation phase is over, the framework proceeds to select one of the low level heuristics (section 3.2) and apply it on a random solution from the set of non-dominated solutions (Algorithm 1, lines 5-8).

Maintaining the non-dominated set during the search requires that, when comparing any two solutions, the solution with the worse (better) number of groups should have the better (worse) cost value, i.e. $f(U_i) > f(U_j), \forall\, i < j, (\forall i, j \in [LB, UB])$. However, during the search (Algorithm 1, lines 5-8), it is possible that a newly created solution might violate this requirement when compared to the other solutions in the non-dominated set, i.e. the new solution might be either better or worse in both objectives when compared to some other solutions in the non-dominated set. We overcome this problem by introducing a case-based acceptance mechanism. The move acceptance of the hyper-heuristic component does not make the final call for the acceptance of a solution and can be considered as a pre-test component for final acceptance. The new solution is only accepted after passing multiple tests.

Firstly, the hyper-heuristic move acceptance criteria compares the new solution s_{new} to the current solution s_i in order to make a decision regarding whether to *consider* the new solution for acceptance or reject it immediately (Algorithm 1, line 9). This is different from how the traditional hyper-heuristic framework operate, in which the decision made by the move acceptance is final. The hyper-heuristic move acceptance methods we used in this study are non-deterministic, i.e. all improving solutions are considered for acceptance, while some of the worsening ones may or may not be considered. We differentiate between two main cases and the second case allows the use of local search to improve the non-dominated set of solutions further:

1. If s_{new} is considered for acceptance despite being worse than s_i in terms of cost value (Algorithm 1, line 10), then it is compared to s_{i-1}. s_{new} is rejected if it is worse than s_{i-1}. Otherwise, it is accepted and inserted into the non-dominated set to replace s_i (Algorithm 1, lines 11-16).
2. If s_{new} is considered for acceptance and its cost value is better than, or equal to, the cost value of s_i (Algorithm 1, line 17), then it is accepted and inserted into the non-dominated set to replace s_i. A violation to the dominance rule may occur if s_{i+1}, which is already worse than s_i in terms of the number of groups, turns out to be also worse in terms of the cost value. In order to avoid this, s_{new} is then compared to s_{i+1} (Algorithm 2, line 2), which creates two cases:
   - 2.1 If s_{new} is worse than s_{i+1}, then there are no violations in the dominance rule, and no more action is needed (Algorithm 2, lines 2-3).
   - 2.2 If s_{new} is better than s_{i+1}, then s_{i+1} is in violation of the dominance rule and consequently it is removed from the non-dominated set. A replacement solution is created by dividing a group in s_{new} using any of the divide heuristics (Algorithm 2, lines 4-8). The only remaining issue is that, this replacement solution at $i+1$ might have a better cost value than the solution at $i + 2$, hence, the for loop in Algorithm 2. In the worst case scenario, this algorithm will be applied on all the solutions in the non-dominated set between i and UB. This process can be considered as local search.

Algorithm 1 Hyper-heuristic framework for grouping problems

```
1: Generate an initial non-dominated set that contains a solution for each value of $k \in [LB, UB]$.
2: Compute the cost value of each solution in the initial non-dominated set.
3: Copy the initial non-dominated set into a an external archive to keep track of the best non-dominated solutions.
4: while (elapsedTime < maxTime) do
5:     Choose a random solution $s_j$ from the current non-dominated set by
       $j \leftarrow UniformRandom(LB, UB)$.
6:     Choose a low level heuristic, $LLH$.
7:     $s_{new} \leftarrow Apply(LLH, s_j)$ {$s_{new}$ will have $i = (j-1)$ or $j$ or $(j+1)$ number of groupings depending on the nature of LLH (merge or change or divide, respectively)}.
8:     Compute $f(s_{new})$ using delta evaluation.
9:     $result \leftarrow moveAcceptance(s_{new}, s_i)$ {Use the hyper-heuristic acceptance method to compare the cost of $s_{new}$ to the cost of $s_i$ from the current non-dominated set}.
       {Following is the case when new solution is a worsening solution which is accepted by $moveAcceptance$}
10:    if (($result$ is $ACCEPT$) and ($f(s_{new}) > f(s_i)$)) then
11:        if ($f(s_{new}) > f(s_{i-1})$) then
12:            Do nothing {$s_{new}$ is rejected}
13:        else
14:            $s_i \leftarrow s_{new}$ {$s_{new}$ is placed in the non-dominated set at grouping $i$, replacing $s_i$}.
15:        end if
16:    end if
17:    if (($result$ is $ACCEPT$) and ($f(s_{new}) \leq f(s_i)$)) then
18:        $s_i \leftarrow s_{new}$ {$s_{new}$ is placed in the non-dominated set at grouping $i$, replacing $s_i$}.
19:        $improveNonDominatedSet(i)$
20:    end if
       {if $result$ is $REJECT$ then do nothing and continue}
21: end while
```

Algorithm 2 *improveNonDominatedSet*(i): Attempts to improve upon the cost of solutions in the non-dominated set starting from i^{th} solution to UB^{th} solution using a *divide* heuristic

```
1: for ($j = i, UB$) do
2:     if ($f(s_{(j+1)}) \leq f(s_j)$) then
3:         $BREAK$ {Further improvement is not possible}
4:     else
5:         Choose a random divide heuristic, $LLDH$
6:         $s_{new} \leftarrow Apply(LLDH, s_j)$
7:         $s_{(j+1)} \leftarrow s_{new}$ {$s_{new}$ is placed in the non-dominated set at grouping $j$, replacing $s_{(j+1)}$}.
8:     end if
9: end for
```

4. Experimental Results

4.1. Experimental Design and Evaluation Criteria

In this study, we investigated the performance of nine selection hyper-heuristics, using all the combinations of the heuristic selection {SR, RL, DH} and move acceptance methods {ILTA, LACC, GDEL} on benchmark instances from the graph colouring and examination timetabling domains. RL uses increment and decrement score operations as a reward and punishment scheme, respectively. A heuristic with the maximum score is selected at each decision point. The increment and decrement scores are set to 1. The upper and lower bounds for the score of any low level heuristic are set to 40 and 0 respectively. The initial score of each one of the low level heuristics is set to ($upper\ bound - 2 * number\ of\ heuristics$). DH and ILTA are implemented using the exactly the same suggested settings as in (Misir et al., 2013). LACC uses a queue of fixed size of 50 for each $k \in [LB, UB]$. The GDEL parameters shown in equation 6 are set to the following values: T is set to the maximum duration of a trial, ΔF is set to the minimum cost value in the initial non-dominated set and f_0 is set to 0. Each experiment is repeated for 30 runs (trials), each of which stops when the best known colouring/timetable is attained or a time limit of 3600 seconds is exceeded. All initial solutions are created randomly. Experiments were conducted on 3.6GHz Intel Core $i7 - 3820$ machines with 16.0GB of memory, running Windows 7 operating system.

The proposed approach cannot be used for multi-objective optimisation, but it operates based on the dominance concept from the multi-objective optimisation. In minimum colouring, the aim is not just minimise the violations, the aim is get rid of all violation yielding 0 as the objective value. Still, in our approach we have archived a set of non-dominated solutions for a given range of number of colours/groupings. Hence, the performance of a given algorithm is evaluated based on the best (minimum) number of colours achieved with no violation and *hyper-volume* (Zitzler and Thiele, 1998) of the non-dominated solutions obtained by that algorithm. The hyper-volume is a commonly used metric to evaluate the spread of the solutions along the Pareto front, as well as the closeness of the solutions to the Pareto-optimal front. We used the cost of a fixed solution produced for the smallest allowed colouring for a given instance as a reference to compute the hyper-volume of the best non-dominated solutions obtained by each algorithm. The larger the hyper-volume, the larger the size of the space covered by the non-dominated set and better the corresponding hyper-heuristic approach.

The Wilcoxon Signed Rank test is used as a statistical test for the average pairwise performance comparison of algorithms based on the minimum colouring they obtain over 30 runs.

We have also used *success rate* (SRate%) as a performance indicator of a hyper-heuristic. SRate% indicates the percentage of 30 runs for which expected number of groups has been obtained by the given algorithm.

4.2. Experimental Data

The characteristics of the benchmark instances used during the experiments are summarised in Table 1. For graph colouring, 19 benchmark instances are used in which the number of colours, vertices, edges as well as edge densities vary. The instances in the upper half of the table are from the COLOR02 website [2], which was initially compiled for a competition. Myciel graphs are based on Mycielski transformation and are considered to be difficult to solve and the colouring

[2]http://mat.gsia.cmu.edu/COLOR02/

number increases in problem size. A queen$n.n$ graph is a graph on n^2 nodes, each represents a square on an n by n chessboard. If two squares are in the same row, column, or diagonal, then their corresponding nodes on the graph are considered to be connected. The objective is to place n sets of n queens on the board so that no two queens of the same set can capture one another, which could be achieved only if the graph has a colouring number n. In addition to these data sets, problem instances in the bottom half of the table are from the well known DIMACS[3] challenge suite. For examination timetabling, we used a subset from the Toronto benchmark suite referred to as $Toronto\ a$ instances (Qu et al., 2009). Table 1 also shows the range of k values used during the experiments for each instance.

Initial experiments were conducted to observe the behaviour of the hyper-heuristic approach considering different k values for each problem instance, as specified in Table 1, while the heuristic selection is fixed from {SR, RL, DH} and the heuristic acceptance is fixed from {ILTA, LACC, GDEL}. A thorough performance analysis of the hyper-heuristics is performed. Then the performance of the hyper-heuristic with the best mean is compared to the performance of some previously proposed approaches. The same framework using the top two hyper-heuristics, namely $RL - ILTA$ and $DH - ILTA$, are tested further on examination timetabling problem.

Table 1: The characteristics of the COLOR02, DIMACS and Toronto problem instances used during the experiments. $|V|$ represents the number of vertices, $|E|$ the number of edges, % the edge density and $k^*/\chi(G)$ represent the best known number of colours (or time-slots) and the chromatic number respectively Wu and Hao (2012). L and U represent the lower and upper bounds for the k values used during the experiments.

		Graph Colouring				Range	
	Instance	$\|V\|$	$\|E\|$	%	$k^*/\chi(G)$	L	U
COLOR02	myciel3	11	20	0.40	4/4	2	9
	myciel4	23	71	0.28	5/5	2	10
	myciel5	47	236	0.22	6/6	3	11
	queen5.5	25	160	0.53	5/5	2	10
	queen6.6	36	290	0.46	7/7	4	12
	queen7.7	49	476	0.40	7/7	2	12
	queen8.8	64	728	0.36	9/9	6	14
DIMACS	le450_25a	450	8260	0.08	25/25	20	30
	le450_25b	450	8263	0.08	25/25	20	30
	le450_25c	450	17343	0.17	25/25	20	30
	le450_25d	450	17425	0.17	25/25	20	30
	DSJC125.1	125	736	0.09	5/?	2	10
	DSJC125.5	125	3891	0.50	17/?	13	23
	DSJC125.9	125	6961	0.89	44/?	40	50
	DSJC250.1	250	3218	0.10	8/?	4	14
	DSJC250.5	250	15668	0.50	28/?	24	34
	DSJC250.9	250	27897	0.90	72/?	68	78
	DSJC500.1	500	12458	0.10	12/?	7	17
	DSJC500.5	500	62624	0.50	48/?	43	53
		Examination Timetabling				Range	
	Instance	$\|V\|$	$\|E\|$	%	k^*	L	U
TORONTO	hec92	81	1363	0.42	17	12	22
	sta83	139	1381	0.14	13	8	19
	yor83	181	4691	0.29	19	13	25
	ute92	184	1430	0.08	10	6	15
	rye93	486	8872	0.08	21	16	27

4.3. Selection Hyper-heuristics for Graph Colouring

Tables 2, 3 and 4 show the *success rate* of each hyper-heuristic approach for different values of k with each problem instance. All those results are obtained while optimising a given instance

[3]ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/color/

Table 2: Reinforcement learning based selection hyper-heuristics: The success rate (sRate%) of each hyper-heuristic on the graph colouring problem instances and the average time ($\mu_t(s)$) taken to achieve that success rate for a given number of colourings, k over 30 runs.

			RL-ILTA		RL-LACC		RL-GDEL	
	Instance	k	sRate%	$\mu_t(s)$	sRate%	$\mu_t(s)$	sRate%	$\mu_t(s)$
COLOR02	myciel3	4	100.00	0.003	100.00	0.004	93.33	0.002
	myciel4	5	100.00	0.005	100.00	0.005	100.00	0.006
	myciel5	6	100.00	0.009	100.00	0.011	100.00	0.069
	queen5.5	5	100.00	0.034	90.00	0.037	100.00	0.194
	queen6.6	7	100.00	0.786	20.00	1.449	100.00	111.831
	queen7.7	7	100.00	0.900	16.67	0.040	100.00	39.350
	queen8.8	9	100.00	4.110	3.33	0.150	100.00	74.530
DIMACS	DSJC125.1	5	96.67	113.20	6.67	6.71	100.00	295.58
		6	100.00	0.09	100.00	0.17	100.00	55.02
		7	100.00	0.03	100.00	0.04	100.00	50.18
	DSJC125.5	17	33.33	470.96	0.00	–	0.00	–
		18	100.00	5.68	60.00	115.70	66.67	655.66
		19	100.00	0.85	100.00	1.80	100.00	82.63
	DSJC125.9	44	100.00	40.02	80.00	69.40	0.00	–
		45	100.00	4.50	100.00	27.05	10.00	538.47
		46	100.00	1.58	100.00	3.90	86.67	422.53
	DSJC250.1	8	10.00	1131.10	0.00	–	0.00	–
		9	100.00	2.35	100.00	7.46	100.00	194.28
		10	100.00	0.44	100.00	0.60	100.00	175.16
	DSJC250.5	28	0.00	–	0.00	–	0.00	–
		29	20.00	2132.40	0.00	–	0.00	–
		30	100.00	288.20	20.00	1164.80	0.00	–
	DSJC250.9	72	10.00	3301.23	0.00	–	0.00	–
		73	100.00	811.32	40.00	886.41	0.00	–
		74	100.00	177.44	80.00	648.03	0.00	–
	DSJC500.1	12	0.00	–	0.00	–	0.00	–
		13	60.00	1115.87	20.00	1331.99	0.00	–
		14	100.00	26.49	100.00	43.70	0.00	–
	DSJC500.5	48	0.00	–	0.00	–	0.00	–
		49	0.00	–	0.00	–	0.00	–
		50	0.00	–	0.00	–	0.00	–
		52	10.00	1797.03	0.00	–	0.00	–
	le450_25a	25	100.00	7.85	100.00	25.33	0.00	–
		26	100.00	1.56	100.00	2.81	83.33	472.34
		27	100.00	0.82	100.00	1.94	100.00	109.32
	le450_25b	25	100.00	11.85	100.00	6.22	6.67	569.68
		26	100.00	1.19	100.00	2.55	100.00	223.61
		27	100.00	0.65	100.00	1.66	100.00	73.37
	le450_25c	25	0.00	–	0.00	–	0.00	–
		26	0.00	–	0.00	–	0.00	–
		27	76.67	565.69	40.00	873.93	0.00	–
	le450_25d	25	0.00	–	0.00	–	0.00	–
		26	0.00	–	0.00	–	0.00	–
		27	70.00	654.47	30.00	906.22	0.00	–

within the associated range as provided in Table 1. For example, the best known colouring for the DSJC500.1 problem instance is 12 and the algorithms are tested with $7 \leq k \leq 17$. From Table 2, out of the 30 runs performed using the RL-ILTA hyper-heuristic on DSJC500.1 problem instance, the percentage of the runs in which solutions with the best colourings ($k = 12$) were found is 0%, while the percentage of the runs in which solutions with $k = 13$ colours are found is 60% and the percentage of the runs in which solutions with $k = 14$ colours are found is 100%. These tables also show the average time (in seconds) taken to achieve those success rates. Only the durations of the successful runs were taken into consideration when the average times were calculated.

Most of the tested hyper-heuristic approaches successfully found the best colourings of the selected COLOR02 problem instances in a reasonable amount of time, and hence, we provide

Table 3: Adaptive dynamic heuristics set based selection hyper-heuristics: The success rate (sRate%) of each hyper-heuristic on the graph colouring problem instances and the average time ($\mu_t(s)$) taken to achieve that success rate for a given number of colourings, k over 30 runs.

			DH-ILTA		DH-LACC		DH-GDEL	
	Instance	k	sRate%	$\mu_t(s)$	sRate%	$\mu_t(s)$	sRate%	$\mu_t(s)$
COLOR02	myciel3	4	100.00	0.006	100.00	0.006	96.67	0.004
	myciel4	5	100.00	0.015	100.00	0.018	100.00	0.016
	myciel5	6	100.00	0.044	100.00	0.054	100.00	3.592
	queen5.5	5	100.00	0.167	100.00	1.053	100.00	5.532
	queen6.6	7	56.67	290.060	43.33	86.180	93.33	204.640
	queen7.7	7	50.00	304.040	10.00	432.160	86.67	152.420
	queen8.8	9	60.00	305.880	26.67	224.680	86.67	335.830
DIMACS	DSJC125.1	5	30.00	618.95	6.67	112.04	90.00	533.55
		6	100.00	0.48	100.00	12.93	100.00	39.25
		7	100.00	0.16	100.00	0.37	100.00	32.14
	DSJC125.5	17	13.33	821.69	0.00	–	0.00	–
		18	93.33	228.10	76.67	146.32	60.00	562.04
		19	100.00	14.66	100.00	12.02	96.67	226.22
	DSJC125.9	44	86.67	146.55	63.33	183.39	0.00	–
		45	100.00	29.08	93.33	32.98	10.00	537.80
		46	100.00	9.76	100.00	7.67	43.33	464.38
	DSJC250.1	8	30.00	1047.59	0.00	–	0.00	–
		9	100.00	71.23	100.00	59.16	100.00	355.09
		10	100.00	1.99	100.00	1.90	100.00	168.32
	DSJC250.5	28	0.00	–	0.00	–	0.00	–
		29	20.00	3042.47	0.00	–	0.00	–
		30	80.00	1704.74	40.00	913.88	0.00	–
	DSJC250.9	72	0.00	–	0.00	–	0.00	–
		73	60.00	2281.95	50.00	1357.03	0.00	–
		74	90.00	865.49	90.00	667.89	0.00	–
	DSJC500.1	12	0.00	–	0.00	–	0.00	–
		13	70.00	1115.84	80.00	496.77	0.00	–
		14	100.00	23.69	100.00	53.11	0.00	–
	DSJC500.5	48	0.00	–	0.00	–	0.00	–
		49	0.00	–	0.00	–	0.00	–
		50	0.00	–	0.00	–	0.00	–
		52	3.33	2553.04	0.00	–	0.00	–
	le450_25a	25	100.00	85.82	96.67	36.19	3.33	1019.41
		26	100.00	3.15	100.00	5.27	86.67	345.32
		27	100.00	1.80	100.00	3.38	100.00	147.13
	le450_25b	25	100.00	7.58	100.00	29.14	20.00	649.71
		26	100.00	2.13	100.00	3.75	100.00	211.16
		27	100.00	1.40	100.00	3.16	100.00	101.29
	le450_25c	25	0.00	–	0.00	–	0.00	–
		26	0.00	–	0.00	–	0.00	–
		27	83.33	626.85	30.0	2346.70	0.00	–
	le450_25d	25	0.00	–	0.00	–	0.00	–
		26	0.00	–	0.00	–	0.00	–
		27	63.33	704.38	36.67	665.24	0.00	–

the success rates and average times for only the best known value of k for these instances. The RL-ILTA and the SR-GDEL hyper-heuristics successfully found the best colouring for each data set in each one of the 30 runs, achieving 100.0% success rate across the board, with a maximum average time of 4.1 seconds for the RL-ILTA approach. On the other hand, hyper-heuristics with LACC acceptance failed to find the best colourings for some problem instances in most of the runs. For example, the SR-LACC and the RL-LACC hyper-heuristics found the best colourings of queen7.7 and queen8.8 problem instances respectively in only one of the 30 runs performed.

The performances of the hyper-heuristic approaches were much varied and less successful when applied on the selected DIMACS problem instances, and much longer periods of times were needed to find the best solutions in most of the cases. Generally, hyper-heuristics with ILTA acceptance have better success rates than hyper-heuristics with LACC and GDEL acceptance

Table 4: Simple random based selection hyper-heuristics: The success rate (sRate%) of each hyper-heuristic on the graph colouring problem instances and the average time ($\mu_t(s)$) taken to achieve that success rate for a given number of colourings, k over 30 runs.

			SR-ILTA		SR-LACC		SR-GDEL	
	Instance	k	sRate%	$\mu_t(s)$	sRate%	$\mu_t(s)$	sRate%	$\mu_t(s)$
COLOR02	myciel3	4	100.00	0.007	100.00	0.004	100.00	0.004
	myciel4	5	100.00	0.007	100.00	0.003	100.00	0.008
	myciel5	6	100.00	0.011	100.00	0.015	100.00	0.398
	queen5.5	5	100.00	0.016	96.67	0.017	100.00	0.687
	queen6.6	7	33.33	0.157	20.00	0.780	100.00	83.140
	queen7.7	7	16.67	0.261	3.33	0.038	100.00	43.669
	queen8.8	9	30.00	2.032	10.00	1.232	100.00	69.154
DIMACS	DSJC125.1	5	3.33	450.25	0.00	–	96.67	374.43
		6	100.00	0.13	100.00	0.14	100.00	53.92
		7	100.00	0.06	100.00	0.07	100.00	51.31
	DSJC125.5	17	20.00	756.87	0.00	–	0.00	–
		18	93.33	117.65	53.33	41.71	90.00	480.85
		19	100.00	1.06	100.00	1.88	100.00	92.52
	DSJC125.9	44	90.00	72.28	86.67	57.01	0.00	–
		45	100.00	13.79	100.00	5.28	23.33	340.94
		46	100.00	1.87	100.00	3.20	96.67	432.36
	DSJC250.1	8	0.00	–	0.00	–	0.00	–
		9	100.00	5.58	100.00	4.36	100.00	176.98
		10	100.00	0.57	100.00	0.74	100.00	175.51
	DSJC250.5	28	0.00	–	0.00	–	0.00	–
		29	0.00	–	0.00	–	0.00	–
		30	60.00	1350.55	20.00	650.08	0.00	–
	DSJC250.9	72	0.00	–	0.00	–	0.00	–
		73	50.00	2264.24	60.00	1547.57	0.00	–
		74	100.00	649.87	90.00	668.56	0.00	–
	DSJC500.1	12	0.00	–	0.00	–	0.00	–
		13	30.00	1005.16	40.00	913.15	0.00	–
		14	100.00	27.99	100.00	30.55	0.00	–
	DSJC500.5	48	0.00	–	0.00	–	0.00	–
		49	0.00	–	0.00	–	0.00	–
		50	0.00	–	0.00	–	0.00	–
		53	3.33	1944.04	0.00	–	0.00	–
	le450_25a	25	100.00	6.36	100.00	33.61	0.00	–
		26	100.00	1.52	100.00	3.34	76.67	518.65
		27	100.00	0.89	100.00	2.52	100.00	99.39
	le450_25b	25	100.00	3.10	100.00	7.70	0.00	–
		26	100.00	1.16	100.00	3.03	100.00	221.37
		27	100.00	0.67	100.00	1.95	100.00	83.17
	le450_25c	25	0.00	–	0.00	–	0.00	–
		26	0.00	–	0.00	–	0.00	–
		27	40.00	1395.13	16.67	783.49	0.00	–
	le450_25d	25	0.00	–	0.00	–	0.00	–
		26	0.00	–	0.00	–	0.00	–
		27	60.00	2326.19	30.00	2230.95	0.00	–

methods in most of the cases. ILTA based hyper-heuristics have outperformed their LACC and GDEL counterparts in all problem instances for all values of k with the exception of DSJC125.1 at $k = 5$, DSJC250.9 at $k = 73$ and DSJC500.1 at $k = 13$. In the same way, GDEL hyper-heuristics have the worst average times and success rates on most of the DIMACS instances compared to the rest of the hyper-heuristics.

Table 5 and shows the average best colouring and standard deviation for each hyper-heuristic approach on each problem instance across the 30 runs. ±0.0 standard deviation corresponds to 100.0% success rate. The row denoted 'Wins' shows the number of problem instances in which the corresponding hyper-heuristic approach achieved the best average colouring including ties with the other algorithms. From this table it can be seen that RL-ILTA hyper-heuristic has the most number of wins across all the tested approaches. Hence, we used the performance of the RL-ILTA hyper-heuristic as a reference to carry out statistical tests in order to determine how significant the differences in the best colourings found by each of the tested hyper-heuristic approaches with regard to the RL-ILTA approach are.

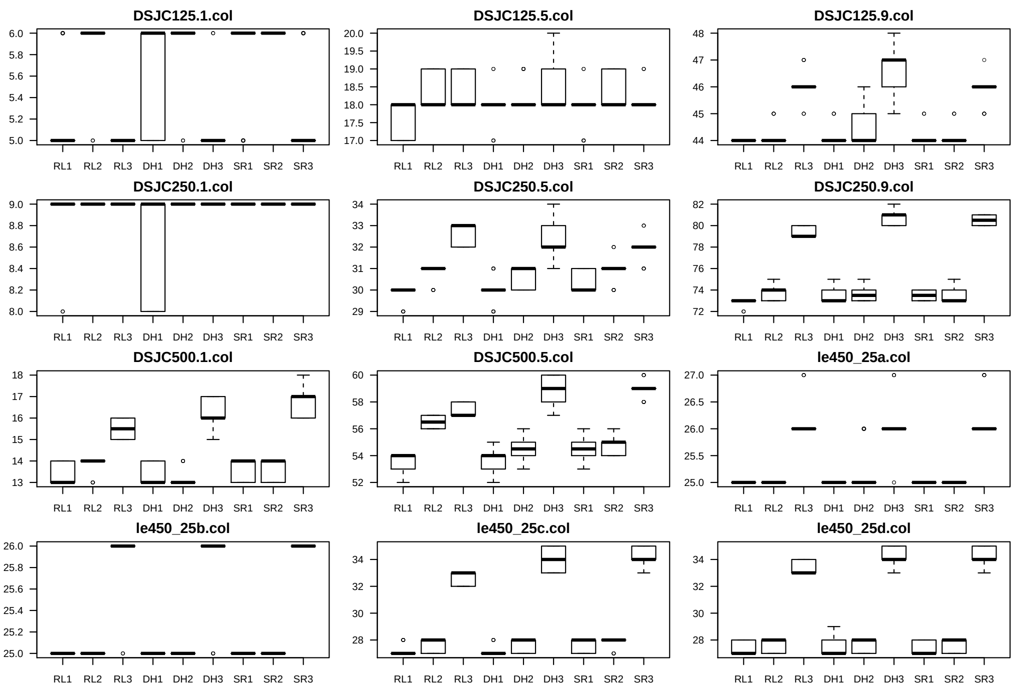

Figure 4: Box plots of the best number of colours achieved by each hyper-heuristic approach on selected DIMACS data sets. 1, 2 and 3 in the hyper-heuristic approaches names refer to ILTA, LACC and GDEL acceptance methods respectively

Table 6 shows the pairwise performance comparison of hyper-heuristics based on the Wilcoxon Signed Rank statistical test using best colourings attained by RL-ILTA hyper-heuristic as a reference. Almost all the results obtained by the GDEL based hyper-heuristics are significantly worse than the results of the RL-ILTA hyper-heuristic, with the exception of RL-GDEL on DSJC125.1, for which case, RL-GDEL performs significantly better than RL-ILTA. Similarly, the RL-ILTA hyper-heuristic performs either significantly or slightly better than the LACC based hyper-heuristics in almost all of the cases with the exception of DH-LACC on DSJC500.1, for which case DH-LACC performs significantly better than the RL-ILTA.

One of the observations is that the DH-ILTA and RL-ILTA hyper-heuristics deliver a competitive performance. On DSJC250.1, DSJC500.1 and le450_25c, DH-ILTA performs significantly better than RL-ILTA. This has been expected bearing in mind that the DH-ILTA is known to be a very powerful algorithm in cross domain search (Misir et al., 2013). However, on more instances, namely DSJC125.1, DSJC125.9, DSJC250.9, DSJC500.5 and le450_25d, RL-ILTA outperforms DH-ILTA. This performance difference is statistically significant. Additionally, on 4 other instances, RL-ILTA performs slightly better than DH-ILTA. They have a tie on six benchmark instances.

To better evaluate the differences between the performances of the 9 hyper-heuristics on the

Table 5: Average best colouring and standard deviation of each hyper-heuristic approach on each problem instance across the 30 runs.

		RL-ILTA		RL-LACC		RL-GDEL	
Instance	k^*	$\mu(k_{best})$	$\sigma(k_{best})$	$\mu(k_{best})$	$\sigma(k_{best})$	$\mu(k_{best})$	$\sigma(k_{best})$
myciel3	4	**4.0**	±0.0	**4.0**	±0.0	4.07	±0.25
myciel4	5	**5.0**	±0.0	**5.0**	±0.0	**5.0**	±0.0
myciel5	6	**6.0**	±0.0	**6.0**	±0.0	**6.0**	±0.0
queen5.5	5	**5.0**	±0.0	5.1	±0.31	**5.0**	±0.0
queen6.6	7	**7.0**	±0.0	7.8	±0.41	**7.0**	±0.0
queen7.7	7	**7.0**	±0.0	8.4	±0.77	**7.0**	±0.0
queen8.8	9	**9.0**	±0.0	9.97	±0.18	**9.0**	±0.0
DSJC125.1	5	5.03	±0.18	5.93	±0.25	**5.0**	±0.0
DSJC125.5	17	**17.67**	±0.48	18.4	±0.50	18.33	±0.48
DSJC125.9	44	**44.0**	±0.0	44.2	±0.41	46.03	±0.49
DSJC250.1	8	**8.9**	±0.30	9.0	±0.0	9.0	±0.0
DSJC250.5	28	**29.8**	±0.41	30.8	±0.41	32.6	±0.50
DSJC250.9	72	**72.9**	±0.31	73.8	±0.76	79.4	±0.50
DSJC500.1	12	**13.4**	±0.50	13.8	±0.41	15.5	±0.51
DSJC500.5	48	**53.6**	±0.67	56.5	±0.51	57.4	±0.50
le450_25a	25	**25.0**	±0.0	**25.0**	±0.0	26.17	±0.38
le450_25b	25	**25.0**	±0.0	**25.0**	±0.0	25.93	±0.25
le450_25c	25	**27.23**	±0.43	27.6	±0.50	32.6	±0.50
le450_25d	25	**27.3**	±0.47	27.7	±0.47	33.4	±0.50
Wins		**18**		5		7	
		DH-ILTA		DH-LACC		DH-GDEL	
Instance	k^*	$\mu(k_{best})$	$\sigma(k_{best})$	$\mu(k_{best})$	$\sigma(k_{best})$	$\mu(k_{best})$	$\sigma(k_{best})$
myciel3	4	**4.0**	±0.0	**4.0**	±0.0	4.1	±0.55
myciel4	5	**5.0**	±0.0	**5.0**	±0.0	**5.0**	±0.0
myciel5	6	**6.0**	±0.0	**6.0**	±0.0	**6.0**	±0.0
queen5.5	5	**5.0**	±0.0	**5.0**	±0.0	**5.0**	±0.0
queen6.6	7	7.43	±0.50	7.57	±0.50	**7.07**	±0.25
queen7.7	7	7.7	±0.79	8.37	±0.67	**7.17**	±0.46
queen8.8	9	9.4	±0.50	9.73	±0.45	**9.13**	±0.35
DSJC125.1	5	5.7	±0.47	5.93	±0.25	**5.1**	±0.31
DSJC125.5	17	**17.93**	±0.45	18.23	±0.43	18.43	±0.57
DSJC125.9	44	**44.13**	±0.35	44.43	±0.63	46.6	±0.86
DSJC250.1	8	**8.7**	±0.47	9.0	±0.0	9.0	±0.0
DSJC250.5	28	**30.0**	±0.64	30.6	±0.50	32.2	±0.89
DSJC250.9	72	**73.5**	±0.68	73.6	±0.67	80.57	±0.57
DSJC500.1	12	13.3	±0.47	**13.2**	±0.41	16.3	±0.65
DSJC500.5	48	**53.67**	±0.71	54.5	±0.82	58.77	±1.01
le450_25a	25	**25.0**	±0.0	25.03	±0.18	26.1	±0.40
le450_25b	25	**25.0**	±0.0	**25.0**	±0.0	25.8	±0.41
le450_25c	25	**27.17**	±0.38	27.7	±0.47	33.97	±0.81
le450_25d	25	**27.4**	±0.56	27.63	±0.49	34.13	±0.68
Wins		**14**		6		7	
		SR-ILTA		SR-LACC		SR-GDEL	
Instance	k^*	$\mu(k_{best})$	$\sigma(k_{best})$	$\mu(k_{best})$	$\sigma(k_{best})$	$\mu(k_{best})$	$\sigma(k_{best})$
myciel3	4	**4.0**	±0.0	**4.0**	±0.0	**4.0**	±0.0
myciel4	5	**5.0**	±0.0	**5.0**	±0.0	**5.0**	±0.0
myciel5	6	**6.0**	±0.0	**6.0**	±0.0	**6.0**	±0.0
queen5.5	5	**5.0**	±0.0	5.03	±0.18	**5.0**	±0.0
queen6.6	7	7.67	±0.48	7.8	±0.41	**7.0**	±0.0
queen7.7	7	8.23	±0.73	8.53	±0.57	**7.0**	±0.0
queen8.8	9	9.7	±0.47	9.9	±0.31	**9.0**	±0.0
DSJC125.1	5	5.97	±0.18	6.0	±0.0	**5.03**	±0.18
DSJC125.5	17	**17.87**	±0.51	18.47	±0.51	18.1	±0.31
DSJC125.9	44	**44.1**	±0.31	44.13	±0.35	45.8	±0.48
DSJC250.1	8	**9.0**	±0.0	**9.0**	±0.0	**9.0**	±0.0
DSJC250.5	28	**30.4**	±0.50	30.9	±0.55	31.9	±0.55
DSJC250.9	72	**73.5**	±0.51	**73.5**	±0.68	80.5	±0.51
DSJC500.1	12	13.7	±0.47	**13.6**	±0.50	16.9	±0.71
DSJC500.5	48	**54.57**	±0.73	54.9	±0.71	59.0	±0.64
le450_25a	25	**25.0**	±0.0	**25.0**	±0.0	26.23	±0.43
le450_25b	25	**25.0**	±0.0	**25.0**	±0.0	26.0	±0.0
le450_25c	25	**27.6**	±0.50	27.83	±0.38	34.13	±0.73
le450_25d	25	**27.4**	±0.50	27.7	±0.47	34.2	±0.61
Wins		14		8		9	

Table 6: Wilcoxon Signed Rank statistical test using RL-ILTA as a reference for the comparison: '>' ('<') denotes that RL-ILTA is significantly better (worse) than the corresponding approach in that column, '≥' means that RL-ILTA is slightly better, and '=' means that there is no difference between the two hyper-heuristic approaches across the 30 runs.

Instance	RL LACC	RL GDEL	DH ILTA	DH LACC	DH GDEL	SR ILTA	SR LACC	SR GDEL
myciel3	=	>	=	=	>	=	=	=
myciel4	=	=	=	=	=	=	=	=
myciel5	=	=	=	=	=	=	=	=
queen5.5	>	=	=	=	=	=	>	=
queen6.6	>	=	≥	>	>	>	>	=
queen7.7	>	=	>	>	>	>	>	=
queen8.8	>	=	≥	>	>	>	>	=
DSJC125.1	>	<	>	>	>	>	>	=
DSJC125.5	>	>	≥	>	>	≥	>	≥
DSJC125.9	≥	>	>	≥	>	>	>	>
DSJC250.1	>	>	<	>	>	>	>	>
DSJC250.5	>	>	≥	>	>	>	>	>
DSJC250.9	>	>	>	>	>	>	>	>
DSJC500.1	≥	>	<	<	>	≥	≥	>
DSJC500.5	>	>	>	>	>	>	>	>
le450_25a	=	>	=	>	>	=	=	>
le450_25b	=	>	=	=	>	=	=	>
le450_25c	≥	>	<	>	>	≥	>	>
le450_25d	≥	>	>	≥	>	>	≥	>

selected problem instances, we carried out another comparison using the hyper-volume measure in which the performance of different algorithms is given in terms of the size of the search space that is covered by the final Pareto front of each hyper-heuristic. Figure 5 shows the box plots of the 30 hyper-volume values produced by each hyper-heuristic on each problem instance. A quick glance at the figure exposes the fact that hyper-heuristics with GDEL acceptance cover the least size of the search space compared to the other hyper-heuristics in most of the selected data sets.

4.4. Performance Comparison to Previously Proposed Approaches

In Table 7, we compare the performance of our approach, RL-ILTA to some previously proposed approaches form the literature for graph colouring. In the table, the results denoted as RL-ILTA are the best colourings obtained by our framework. The results under M-LLE are obtained by a multi-objective genetic grouping algorithm described in (Korkmaz, 2010). Lowest Index Max Crossover (LIMX), Greedy Partition Crossover Lowest Index (GPX-LI) and Greedy Partition Crossover Cardinality Based (GPX-CB) graph colouring algorithms are proposed in (Ülker et al., 2006). Külahçıoğlu (2007) proposed two modified versions of the LLE representation which are Linear Linkage Encoding With Ending Node Links (LLE-e) and Linear Linkage Encoding With Backward Links (LLE-b), and both of them are tested using genetic operators. This last study also tested these operators with classical Linear Linkage Encoding (LLE). The results in the last two columns denoted (Kir-B) and (Kir-C) are graph colouring algorithms proposed in (Kirovski and Potkonjak, 1998). Fields marked as '–' means that the solution for that problem instance with that specific algorithm is not reported. The 'wins' row shows the number of instances in which the corresponding approach hit the best known colouring. As it can clearly be seen in the table, our proposed approach is not only competitive with the previous algorithms, but also it outperforms the previously proposed approaches almost in all cases.

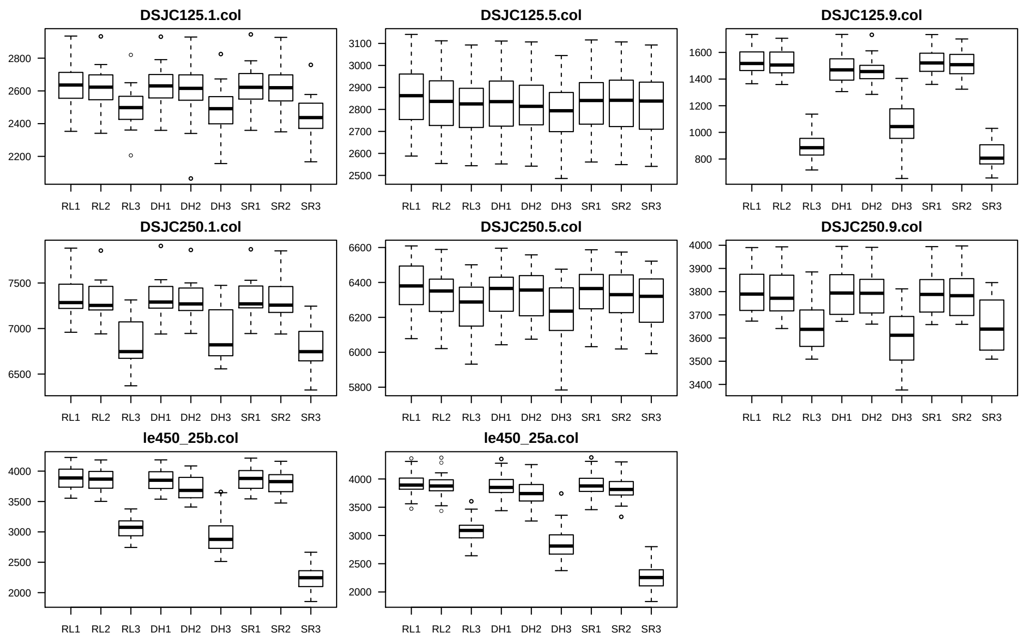

Figure 5: Box plots of the hyper-volume value of all the final Pareto fronts achieved by each hyper-heuristic approach on selected DIMACS data sets. 1, 2 and 3 in the hyper-heuristic approaches names refer to ILTA, LACC and GDEL acceptance methods respectively

Table 7: Performance comparison for different approaches based on the best result. The entries in bold indicates the best result obtained by the associated algorithm for a given instance.

Instance	k^*	RL-ILTA	M-LLE	LIMX	GPX-LI	GPX-CB	LLE-e	LLE-b	LLE	Kir-B	Kir-C
DSJC125.1	5	**5**	–	–	–	–	–	–	–	–	–
DSJC125.5	17	**17**	18	18	18	18	19	19	18	19	18
DSJC125.9	44	**44**	**44**	**44**	**44**	**44**	**44**	**44**	**44**	45	45
DSJC250.1	8	**8**	9	9	9	9	9	9	9	9	9
DSJC250.5	28	**29**	–	31	31	31	–	–	–	30	30
DSJC250.9	72	**72**	75	75	75	74	74	74	74	77	77
DSJC500.1	12	**13**	–	14	14	14	–	–	–	14	14
DSJC500.5	48	**52**	55	–	–	–	–	–	–	–	–
le450_25a	25	**25**	–	**25**	**25**	**25**	–	–	–	**25**	**25**
le450_25b	25	**25**	–	**25**	**25**	**25**	–	–	–	**25**	**25**
le450_25c	25	**27**	29	28	28	28	29	29	29	28	28
le450_25d	25	**27**	–	28	28	28	–	–	–	–	–
Wins		**12**	1	3	3	3	1	1	1	2	2

4.5. Grouping Hyper-heuristics for Examination Timetabling

Finally, in order to show that the framework is generic and can be applied to other domains of grouping problems, we tested the most successful hyper-heuristics, namely the RL-ILTA and the DH-ILTA, on selected instances from the Examination Timetabling domain. In (Johnson and Trick, 1996), it has been shown that the exam timetabling problem can be reduced to a grouping problem if the task of minimising the number of exam periods and removing the clashes are considered. The size of the selected ETT problem instances subset was kept small since the main objective of testing the ETT domain is to show the generality of the framework. Table 8 shows the results of applying the RL-ILTA and DH-ILTA hyper-heuristics. Both hyper-heuristics perform well on all instances and successfully managed to find the best colourings for each instance. Table 9 provides the average best results obtained using RL-ILTA over 30 runs. RL-ILTA achieves the best known result for all instances. Moreover, RL-ILTA detects the best number of time-slots consistently for three large size examination timetabling benchmark instances of ute92, sta83 and hec92 under a second in all runs.

Table 8: The results obtained from the application of RL-ILTA and DH-ILTA hyper-heuristics on Toronto *a* benchmark. sRate% and $\mu_t(s)$ are the success rates and the average time taken by each hyper-heuristic on each problem instance over 30 runs.

		RL-ILTA		DH-ILTA	
Instance	k	sRate%	$\mu_t(s)$	sRate%	$\mu_t(s)$
hec92	17	100.00	4.547	93.33	345.171
	18	100.00	0.328	100.00	1.911
	19	100.00	0.120	100.00	0.928
sta83	13	100.00	0.456	100.00	9.068
	14	100.00	0.089	100.00	0.448
	15	100.00	0.053	100.00	0.303
yor83	19	16.67	583.031	46.67	598.321
	20	100.00	27.724	100.00	41.306
	21	100.00	2.829	100.00	6.830
ute92	10	100.00	0.134	100.00	0.431
	11	100.00	0.059	100.00	0.206
	12	100.00	0.040	100.00	0.161
rye93	21	20.00	1320.075	50.00	1619.851
	22	100.00	82.934	100.00	103.213
	23	100.00	12.399	100.00	19.178

The performance of RL-ILTA is compared against the best results obtained by some previously proposed approaches including (Carter et al., 1996) (Carter), (Caramia et al., 2001) (Caramia) and (Merlot et al., 2002) (Merlot). The approaches in (Ülker et al., 2006) including Greedy Partition Crossover Lowest Index (GPX-LI), Greedy Partition Crossover Cardinality Based (GPX-CB) and Lowest Index Max Crossover (LIMX) algorithms are also considered in our comparison. Table 10 shows that the performance of RL-ILTA is competitive. It performs as good as the Carter and Caramia approaches. Moreover, it is better than the previously proposed population based grouping algorithms, including LIMX, GPX-LI and GPX-CB on most of the instances, particularly yor83 and rye93.

5. Conclusions

Designing an automated intelligent search methodology which can be applied to the unseen problem instances with different characteristics without requiring a change is an extremely challenging task. Selection hyper-heuristics have emerged as such flexible search methodologies

Table 9: Average best colouring and associated standard deviation obtained by RL-ILTA over 30 runs.

		RL-ILTA	
Instance	k^*	$\mu(k_{best})$	$\sigma(k_{best})$
hec92	17.0	17.0	±0.0
sta83	13.0	13.0	±0.0
yor83	19.0	19.83	±0.38
ute92	10.0	10.0	±0.0
rye93	21.0	21.8	±0.41

Table 10: Performance comparison for different approaches based on the best result. The entries in bold indicates the best result obtained by the associated algorithm for a given instance.

Instance	k	RL-ILTA	LIMX	GPX-LI	GPX-CB	Carter	Caramia	Merlot
hec92	17	**17**	**17**	**17**	**17**	**17**	**17**	18
sta83	13	**13**	**13**	14	14	**13**	**13**	**13**
yor83	19	**19**	20	20	20	**19**	**19**	23
ute92	10	**10**	**10**	**10**	**10**	**10**	**10**	11
rye93	21	**21**	23	23	23	**21**	**21**	22
Wins		**5**	3	2	2	**5**	**5**	1

supporting re-usability and component based development (Burke et al., 2013). Most of the previously proposed general purpose hyper-heuristics contain two main reusable components: heuristic selection and move acceptance. This study describes a general hyper-heuristic framework for solving grouping problems, employing generic components as well as a fixed set of low level heuristics and a slightly modified version of the grouping representation in (Falkenauer, 1998)[4].

A performance comparison of nine different hyper-heuristics using different components under this framework is presented on various graph colouring benchmark instances. The results indicates the success of an online learning hyper-heuristic which uses feedback during the search process. The selection hyper-heuristic combining the reinforcement learning (RL) heuristic selection and ILTA move acceptance (Misir et al., 2013) methods even outperforms some previously proposed approaches. RL-ILTA is further tested on an examination timetabling benchmark. This hyper-heuristic without requiring any change again yielded successful results. The proposed framework is indeed flexible allowing different hyper-heuristic components to be brought together and sufficiently general. The RL-ILTA hyper-heuristic implemented under the proposed framework performs slightly better than the selection hyper-heuristic which won a hyper-heuristic competition (Burke et al., 2011). This previously proposed method contains many parameters which were tuned and complicated subcomponents for heuristic selection. The RL-ILTA hyper-heuristic implemented under the proposed framework conforms to one of the crucial properties of a reusable hyper-heuristic, that is the simplicity. The heuristic selection component has only one parameter, that is the memory length and that is set to a fixed value as suggested in (Burke and Soubeiga, 2003).

Our observations in this study are consistent with the previous findings from the literature (Burke et al., 2012; Özcan et al., 2009; Özcan et al., 2010). The use of a different component in a hyper-heuristic could lead to a different performance of the overall algorithm. Learning during the heuristic selection process definitely helps, and the move acceptance plays a major role in the performance of hyper-heuristics. It is crucial to employ compatible and synergistic components yielding an improved performance and RL performed well with ILTA under the proposed grouping hyper-heuristic framework.

Acknowledgements: This work was funded in part by the EPSRC grant EP/F033613/1.

[4]The grouping hyper-heuristic tool will be made publicly available from `http://www.cs.nott.ac.uk/~axe/`

References

Agustín-Blas, L., Salcedo-Sanz, S., Jimnez-Fernndez, S., Carro-Calvo, L., Ser, J.D., Portilla-Figueras, J., 2012. A new grouping genetic algorithm for clustering problems. Expert Systems with Applications 39, 9695 – 9703. doi:http://dx.doi.org/10.1016/j.eswa.2012.02.149.

Avanthay, C., Hertz, A., Zufferey, N., 2003. A variable neighborhood search for graph coloring. European Journal of Operational Research 151, 379–388.

Bai, R., Kendall, G., 2005. An investigation of automated planograms using a simulated annealing based hyper-heuristics, in: Ibaraki, T., Nonobe, K., Yagiura, M. (Eds.), Metaheuristics: Progress as Real Problem Solver - (Operations Research/Computer Science Interface Serices, Vol.32). Springer, pp. 87–108.

Bilgin, B., Özcan, E., Korkmaz, E.E., 2007. An experimental study on hyper-heuristics and exam timetabling, in: Proceedings of the 6th international conference on Practice and theory of automated timetabling VI, Springer-Verlag, Berlin, Heidelberg. pp. 394–412.

Brown, C.E., Sumichrast, R.T., 2003. Impact of the replacement heuristic in a grouping genetic algorithm. Computers & OR 30, 1575–1593.

Burke, E.K., Bykov, Y., 2008. A Late Acceptance Strategy in Hill-Climbing for Exam Timetabling Problems, in: PATAT '08 Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling. URL: http://w1.cirrelt.ca/\~{}patat2008/PATAT_7_PROCEEDINGS/Papers/Bykov-HC2a.pdf.

Burke, E.K., Curtois, T., Hyde, M., Kendall, G., Ochoa, G., Petrovic, S., Vazquez-Rodriguez, J., 2009. Hyflex: A flexible framework for the design and analysis of hyper-heuristics., in: Proceedings of the Multidisciplinary International Scheduling Conference (MISTA09), pp. 790–797.

Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., McCollum, B., Ochoa, G., Parkes, A., Petrovic, S., 2011. The cross-domain heuristic search challenge - an international research competition, in: Yao, X., Coello, C.A.C. (Eds.), Proceedings of Learning and Intelligent Optmization (LION5), pp. 631–634.

Burke, E.K., Gendreau, M., Hyde, M.R., Kendall, G., Ochoa, G., Özcan, E., Qu, R., 2013. Hyper-heuristics: a survey of the state of the art. JORS 64, 1695–1724.

Burke, E.K., Kendall, G., Misir, M., Özcan, E., 2012. Monte carlo hyper-heuristics for examination timetabling. Annals of Operations Research 196, 73–90. doi:10.1007/s10479-010-0782-2.

Burke, E.K., Landa-Silva, J.D., Soubeiga, E., 2005. Multi-objective hyper-heuristic approaches for space allocation and timetabling, in: Ibaraki, T., Nonobe, K., Yagiura, M. (Eds.), Meta-heuristics: Progress as Real Problem Solvers. Springer. 5th Metaheuristics International Conference (MIC 2003), pp. 129–158.

Burke, E.K., Newall, J.P., 1998. A multi-stage evolutionary algorithm for the timetable problem. IEEE Transactions on Evolutionary Computation .

Burke, E.K., Soubeiga, E., 2003. Scheduling nurses using a tabu-search hyperheuristic, in: Proc. of the 1st MISTA, pp. 197–218.

Caramia, M., Dell'Olmo, P., Italiano, G.F., 2001. New algorithms for examination timetabling, in: Algorithm Engineering 4th International Workshop 2000, Springer-Verlag, Berlin Heidelberg New York. pp. 230–241.

Carter, M.W., Laporte, G., 1996. Recent developments in practical examination timetabling, in: Practice and Theory of Automated Timetabling, pp. 3–21.

Carter, M.W., Laporte, G., Lee, S.T., 1996. Examination timetabling: algorithmic strategies and applications. Journal of the Operational Research Society 47, 373–383.

Chakhlevitch, K., Cowling, P.I., 2008. Hyperheuristics: Recent developments, in: Cotta, C., Sevaux, M., Sörensen, K. (Eds.), Adaptive and Multilevel Metaheuristics. Springer. volume 136 of *Studies in Computational Intelligence*, pp. 3–29.

Coello, C.C., Lamont, G., van Veldhuizen, D., 2007. Evolutionary Algorithms for Solving Multi-Objective Problems. Genetic and Evolutionary Computation. 2nd ed., Springer, Berlin, Heidelberg.

Cowling, P., Chakhlevitch, K., 2003. Hyperheuristics for managing a large collection of low level heuristics to schedule personnel, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC2003), IEEE Computer Society Press, Canberra, Australia. pp. 1214–1221.

Cowling, P., Kendall, G., Soubeiga, E., 2000. A hyperheuristic approach to scheduling a sales summit, in: Selected Papers of the Third International Conference on the Practice And Theory of Automated Timetabling, PATAT 2000, Springer, Konstanz, Germany. pp. 176–190.

Davis, L.D., Mitchell, M., 1991. Handbook of Genetic Algorithms. Van Nostrand Reinhold.

Du, J., Korkmaz, E.E., Alhajj, R., Barker, K., 2004. Novel clustering that employs genetic algorithm with new representation scheme and multiple objectives., in: Kambayashi, Y., Mohania, M.K., W, W. (Eds.), DaWaK, Springer. pp. 219–228.

Dueck, G., 1993. New optimization heuristics: The great deluge algorithm and the record-to-record travel. Journal of Computational Physics 104, 86–92. doi:10.1006/jcph.1993.1010.

Falkenauer, E., 1992. The grouping genetic algorithms: Widening the scope of the GAs. Belgian Journal of Operations Research, Statistics and Computer Science (JORBEL) 33, 79–102.

Falkenauer, E., 1998. Genetic Algorithms and Grouping Problems. John Wiley & Sons, Inc., New York, NY, USA.

Galinier, P., Hao, J.K., 1999. Hybrid evolutionary algorithms for graph coloring. J. Comb. Optim. 3, 379–397.

Handl, J., Knowles, J.D., 2007. An evolutionary approach to multiobjective clustering. IEEE Trans. Evolutionary Computation 11, 56–76.

Hertz, A., Werra, D.D., 1987. Using tabu search techniques for graph coloring. Computing 39, 345–351. URL: `http://dx.doi.org/10.1007/BF02239976`, doi:`10.1007/BF02239976`.

Johnson, D.J., Trick, M.A. (Eds.), 1996. Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993. American Mathematical Society, Boston, MA, USA.

Johnson, D.S., Aragon, C.R., Mcgeoctt, L.A., Schevon, C., 1991. Optimization by simulated annealing: an experimental evaluation; part ii. Operational Research .

Jones, D.R., Beltramo, M.A., 1991. Solving partitioning problems with genetic algorithms, in: Belew, R.K., Booker, L.B. (Eds.), ICGA, Morgan Kaufmann. pp. 442–449.

Kirovski, D., Potkonjak, M., 1998. Efficient coloring of a large spectrum of graphs, in: 35th Design Automation Conference Proceedings, pp. 427–432.

Korkmaz, E.E., 2010. Multi-objective genetic algorithms for grouping problems. Appl. Intell. 33, 179–192.

Külahçıoğlu, B., 2007. Multiobjective Hyperheuristic for Data Clustering and Linear Linkage Encoding. Master's thesis. Yeditepe University.

Leighton, F.T., 1979. A graph coloring algorithm for large scheduling problems, in: Journal of Reasearch of the National Bureau of Standards, pp. 489–506.

Merlot, L.T.G., Boland, N., Hughes, B.D., Stuckey, P.J., 2002. A hybrid algorithm for the examination timetabling problem., in: Burke, E.K., De Causmaecker, P. (Eds.), Proceedings of Practice and Theory of Automated Timetabling, Fourth International Conference, Gent, Belgium. pp. 348–371.

Misir, M., Verbeeck, K., Causmaecker, P.D., Berghe, G.V., 2013. A new hyper-heuristic as a general problem solver: an implementation in hyflex. J. Scheduling 16, 291–311.

Özcan, E., Bilgin, B., Korkmaz, E., 2008. A comprehensive analysis of hyperheuristics. Intelligent Data Analysis 12, 1–21.

Özcan, E., Bykov, Y., Birben, M., Burke, E.K., 2009. Examination timetabling using late acceptance hyper-heuristics, in: Proceedings of IEEE Congress on Evolutionary Computation (CEC 2009), pp. 997–1004.

Özcan, E., Mısır, M., Ochoa, G., Burke, E.K., 2010. A reinforcement learning - great-deluge hyper-heuristic for examination timetabling. Int. J. of Applied Metaheuristic Computing 1, 39–59.

Paquete, F.L., Fortseca, C.M., Pt, E.L., 2001. A study of examination timetabling with multiobjective evolutionary algorithms, in: Proc. of the 4th Metaheuristics International Conference (MIC 2001, pp. 149–154.

Park, Y.J., Song, M.S., 1998. A genetic algorithm for clustering problems, in: Koza, J.R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M.H., Goldberg, D.E., Iba, H., Riolo, R. (Eds.), Genetic Programming 1998: Proceedings of the Third Annual Conference, Morgan Kaufmann, University of Wisconsin, Madison, Wisconsin, USA. pp. 568–575.

Qu, R., Burke, E.K., McCollum, B., Merlot, L., Lee, S., 2009. A survey of search methodologies and automated system development for examination timetabling. Journal of Scheduling 12, 55–89.

Saha, S., Kumar, R., Baboo, G., 2013. Characterization of graph properties for improved pareto fronts using heuristics and EA for bi-objective graph coloring problem. Applied Soft Computing 13, 2812 – 2822. URL: `http://www.sciencedirect.com/science/article/pii/S1568494612003018`, doi:`http://dx.doi.org/10.1016/j.asoc.2012.06.021`.

Ülker, O., Korkmaz, E.E., Özcan, E., 2008. A grouping genetic algorithm using linear linkage encoding for bin packing, in: Parallel Problem Solving from Nature, pp. 1140–1149.

Ülker, Ö., Özcan, E., Korkmaz, E.E., 2006. Linear linkage encoding in grouping problems: Applications on graph coloring and timetabling, in: Burke, E.K., Rudová, H. (Eds.), PATAT, Springer. pp. 347–363.

Wu, Q., Hao, J.K., 2012. An effective heuristic algorithm for sum coloring of graphs. Computers & OR 39, 1593–1600.

Yılmaz, B., Korkmaz, E.E., 2010. Representation issue in graph coloring, in: ISDA, IEEE. pp. 1171–1176.

Zitzler, E., Thiele, L., 1998. Multiobjective optimization using evolutionary algorithms - a comparative case study, in: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.P. (Eds.), PPSN, Springer. pp. 292–304.