Data Clustering Using Grouping Hyper-heuristics

Anas Elhag and Ender Özcan

University of Nottingham, School of Computer Science,
ASAP Research Group, Nottingham, UK
Anas.Abdalla@outlook.com,Ender.Ozcan@nottingham.ac.uk

Abstract. Grouping problems represent a class of computationally hard to solve problems requiring optimal partitioning of a given set of items with respect to multiple criteria varying dependent on the domain. A recent work proposed a general-purpose selection hyper-heuristic search framework with reusable components, designed for rapid development of grouping hyper-heuristics to solve grouping problems. The framework was tested only on the graph colouring problem domain. Extending the previous work, this study compares the performance of selection hyper-heuristics implemented using the framework, pairing up various heuristic/operator selection and move acceptance methods for data clustering. The selection hyper-heuristic performs the search processing a single solution at any decision point and controls a fixed set of generic low level heuristics specifically designed for the grouping problems based on a bi-objective formulation. An archive of high quality solutions, capturing the trade-off between the number of clusters and overall error of clustering, is maintained during the search process. The empirical results verify the effectiveness of a successful selection hyper-heuristic, winner of a recent hyper-heuristic challenge for data clustering on a set of benchmark problem instances.

Keywords: heuristic, multiobjective optimisation, reinforcement learning, adaptive move acceptance

1 Introduction

Solving a grouping problem which is an NP-hard combinatorial optimisation problem [1] requires partitioning of set of objects/items into a minimal collection of mutually disjoint subsets. Different grouping problems impose different problem specific constraints, and introduce different objectives to optimise. Consequently, not all groupings are allowed for all problems, since a solution must satisfy the problem specific constraints. These problem specific constraints/objectives forbid the 'trivial' solutions which consist of placing all the objects into one group.

Data clustering is a grouping problem which requires partitioning a given set of data items or property vectors into a minimal number of disjoint clusters/groups, such that the items in each group are *close* to each other with respect to a given similarity measure, and are *distant* from the items in the other groups with respect to the same measure. Data clustering plays an important role in many disciplines where there is a need to learn the inherent grouping structure of data such as data miningand bioinformatics [2–4].

Selection hyper-heuristics have emerged as metaheuristics searching the space formed by operators/heuristics for solving combinatorial optimisation problems [5]. In this study, we use a bi-objective formulation of data clustering as a grouping problem with the goal of simultaneously optimising the number of clusters and error/cost. [6] proposed a grouping hyper-heuristic framework for solving grouping problems, exploiting the bi-objective nature of the grouping problems to capture the best of the two worlds. The results showed the potential of selection hyper-heuristics based on the framework for graph colouring. In here, we extend that previous work, apply and compare the performance of selection hyper-heuristics implemented based on the grouping hyper-heuristic framework for data clustering on a set of well-known benchmarks.

Section 2 discusses commonly used encoding schemes to represent solutions to grouping problems and related work on data clustering. Section 3 describes the selection hyper-heuristic framework for grouping problems. The details of the experimental design, including the benchmark instances, tested selection hyper-heuristics, parameter settings and evaluation criteria used for performance comparison of algorithms, are given in Section 4. Section 4.4 presents the empirical results and finally, conclusion is provided in Section 5.

2 Related Work

2.1 Solution Representation in Grouping Problems

A variety of encoding schemes and population-based operators have been proposed and applied to various grouping problems [1]. Examples include the objects membership representation [7] which uses a constant-length encoding in which each position corresponds to one object; the Locus-Based Adjacency (LBA) representation [8, 4] and the Linear Linkage Encoding (LLE) [9], which are linkage-based fixed-length encodings in which each location represents one object and stores an integer value that represents a link to *another* object in the same group. However, most of these schemes violate one or more of the six design principles for constructing useful representations [10]. For instance, the NE and LBA representations allow multiple chromosomes in the search space to represent the same solution [11], and hence violate the "minimal redundancy" principle, which states that each solution should be represented by as few distinct points in that search space as possible; ideally one point only.

In this study we implemented a modified version of the Grouping Genetic Algorithm Encoding (GGAE) representation which was proposed as part of a genetic algorithm that was heavily modified to suit the structure of the grouping problems [12], and has successfully been applied in many real world problems such as data clustering [2] and machine-part cell formation [13]. The encoding in the GGAE consists of two parts. The object membership part is used only to identify which objects are in which groups. No operators are applied to this part. The groups part encodes the groups on a 'one gene per group' basis. The group part is written after the standard NE part, and the two are separated by a colon. For example, a solution that puts the first three items in one group and the last two items in a different group is represented as follows: $\{D, D, D, C, C : D, C\}$. GGAE operators are applied only to the groups part, and might lead to increasing or decreasing the number of groups in the given solution. This approach implies that the length of the GGAE encoding is not fixed and can not be known in advance. Further discussion of grouping representations could be found in [6, 1].

2.2 Data Clustering

Given a set X of n vectors in a given feature space S, X=$\{x_1$, x_2, x_3, ..., $x_n\}$, the data clustering problem requires finding an optimal partition U=$\{u_1$, u_2, u_3, ..., $u_k\}$ of X, where u_i is the i^{th} cluster/group of U, such that an overall similarity measure between the vectors that belong to the same group, or an overall dissimilarity measure of the vectors that belong to different groups, is maximized, in terms of a given cost function $f(U)$.

There are different supervised and unsupervised measures that are used to evaluate the clustering results [14], including the Sum of Quadratic Errors (SSE) [2], the Silhouette Width (SW) [15], the Davies-Bouldin Index (DB) [16] and the Rand Index (R) [17], among others. The most well-known evaluation method in the data clustering literature is based on the Euclidean distance which is used to give an overall measure of the error of the clustering. In this study, a well-known unsupervised distance measure known as the 'sum of quadratic errors' (SSE) is adopted. Assuming that each data item has p properties, the SSE calculates a centroid vector for each cluster. The resulting centroid is also composed of p values, one for each property. The sum of the distances; i.e. errors; of each item's properties corresponds to the distances to the property values of the centroid of the cluster to which the item belongs.

$$error = \sum_{l=1}^{k} \sum_{i=1}^{n} W_{il} \sum_{j=1}^{p} (x_{ij} - u_{lj})^2 \qquad (1)$$

$k \equiv$ the number of clusters, $n \equiv$ the number of items, $p \equiv$ the number of properties, $W_{il} = 1$ if the i^{th} item is in k^{th} cluster, and 0 otherwise, $x_{ij} \equiv$ the i^{th} item's j^{th} property, and $u_{lj} \equiv$ the center of the j^{th} property of the k^{th} cluster. Traditionally, clustering approaches are classified to partitional, hierarchical and density-based approaches [18]. [8] and [19] categorize the clustering approaches into 3 and 4 groups, respectively, based on the clustering criterion being optimized. The first group in both studies consists of the clustering algorithms that look for compact clusters by optimizing intra-cluster variations, such as variations between items that belong to the same cluster or between the items and cluster representatives. Well-known algorithms such as the k-means [20], model-based clustering [21], average link agglomerative clustering [22] and self-organizing maps [23] belong to this category. The second group, consists of the clustering algorithms that strive for connected clusters by grouping neighboring data into the same cluster. Classical clustering techniques such as single link agglomerative clustering [22] and density-based methods [24] belong to this group.

The third group according to [8] consists of clustering algorithms that look for spatially-separated clusters. However, this objective on its own may result in clustering the oultiers individually while merging the rest of the data items into one big cluster. Clustering objective such as the Dunn Index and the Davies-Bouldin Index [14] combine this objective with other clustering objectives, such as compactness and connectedness, in order to improve the resulting solution. In contrast to the first two groups, this objective has not been used in any specialized clustering algorithm. According to [19], the third group includes simultaneous row-column clustering techniques known as bi-clustering algorithms [25]. Finally, the the fourth group according to [19] includes the multi-objective clustering algorithms that seek to optimize different characteristics of the given data set [3] along with the clustering ensembles approaches [26].

3 A Grouping Hyper-heuristic Approach to Solve Grouping Problems

Algorithm 1 describes the steps of the grouping approach used in this study. Firstly, an initial set of $(UB - LB + 1)$ non-dominated solutions is generated such that there is exactly 1 solution for each value of $k \in [LB, UB]$ (steps 1-3 of Algorithm 1). Consequently, one of the low level heuristics is selected (using the hyper-heuristic selection method) and applied on a solution that is randomly selected from the current set of solutions (steps 5-8 of Algorithm 1). It is vital to ensure that none of the solutions in the non-dominated set that is maintained by the framework break the dominance requirement throughout the search process. To this end, an adaptive acceptance mechanism that involves multiple tests is introduced. Traditionally, the hyper-heuristic's move acceptance makes the final decision regarding the acceptance of a solution. In our approach however, this component acts only as a pre-test for the final acceptance. New solutions are accepted only after successfully passing multiple tests. The decision made by the traditional hyper-heuristic move acceptance only indicates whether the new solution s_{new} is to be considered for acceptance by the grouping framework (step 9 of Algorithm 1). s_{new} could still be a worsening solution based on the nature of the traditional move acceptance used. At this point, two main possibilities, one of which involves the application of local search to further improve the set of non-dominated solutions:

1. If a worsening solution s_{new} passes the traditional move acceptance pre-test, it does not immediately replace the solution s_i in the non-dominated set (steps 10 of Algorithm 1). Instead, s_{new} is compared to s_{i-1}. Only if the cost

value of s_{new} is better than s_{i-1}, it gets to replace s_i in the non-dominated set. Otherwise, s_{new} is rejected, despite having passed the pre-test (steps 11-16 of Algorithm 1).

2. If an improving solution s_{new} passes the traditional move acceptance pre-test, it replaces the solution s_i in the non-dominated set immediately without further tests (step 17 of Algorithm 1). However, this replacement might lead to a violation of the dominance rule if s_{new}, which has fewer groups than s_{i+1}, also has a better cost value than s_{i+1} (step 2 of Algorithm 2). This situation leads to two cases:

2.1 If the cost value of s_{i+1} is better than that of s_{new}, then no violations have occurred to the dominance rule, and no further action is required (steps 2-3 of Algorithm 2).

2.2 If the cost value of s_{i+1} is worse than that of s_{new}, then s_{i+1} violates the dominance rule and hence the framework removes it from the set of non-dominated solutions being maintained. The framework then applies one of the divide heuristics on s_{new} in order to generate a new solution to replace the solution that has been removed (steps 4-8 of Algorithm 2). The for loop in Algorithm 2 is to repeat this process for solutions at $i+1$ and $i+2$. In the worst case scenario, all the solutions between i and UB will get replaced. This process can be considered as local search.

3.1 Low Level Heuristics

Three types of low level heuristics were implemented in this study. *Merge Heuristics* merge two groups, u_i and u_j, into one, u_l, and decrease the number of grouping in the selected solution. 3 merge heuristics were developed. In **M1**, the 2 groups to be merged are selected at random. In **M2**, the 2 groups containing the fewest items are merged. In **M3**, the 2 groups with the smallest partial costs are merged. These heuristics yields big jumps in the search space, and hence, are regarded to be diversifying components.

Similarly, *Divide Heuristics* divide a selected group u_i into two, u_{i1} and u_{i2}, and increase the number of groups in the given solution. 3 versions of the divide heuristic were developed. In **D1**, a group that is selected at random is divided. In **D2**, the group to be divided is the group with the most number of items. In **D3**, the group with the biggest partial cost value is divided. These heuristics can be considered as intensifying components.

Change Heuristics attempt to make small alterations in a selected solution by moving selected items between different groups while preserving the original number of the groupings in the selected solution. 4 change heuristics have been developed. In **C1**, a randomly selected item is moved to a randomly selected group. In **C2**, the item with the largest number of conflicts in a group is moved into a randomly selected group. **C3** and **C4** find the item with the largest number of conflicts in the group with the largest number of conflicts. C3 moves this item into a randomly selected group, while C4 moves it into the group with the minimum number of conflicts.

Algorithm 1 A Grouping Hyper-heuristic Framework

```
1: Create an initial set of non-dominated solutions, containing 1 solution for each
   value of $k \in [LB, UB]$.
2: Calculate the cost values of all solutions in the solutions set.
3: Keep an external archive copy of the solutions set in order to keep track of the best
   solutions found.
4: while (elapsedTime < maxTime) do
5:    Randomly select a solution $s_j$ from the current set of non-dominated solutions
      $j \leftarrow UniformRandom(LB, UB)$.
6:    Select one of the low level heuristics, $LLH$.
7:    $s_{new} \leftarrow Apply(LLH, s_j)$ {$s_{new}$ contains $i = (j-1)$ or $j$ or $(j+1)$ groups based
      on the applied LLH}.
8:    Calculate $f(s_{new})$.
9:    $result \leftarrow moveAcceptance(s_{new}, s_i)$. // Compare the cost value of $s_{new}$ to the
      cost value of $s_i$ from the current non-dominated set using the move acceptance
      method returning $ACCEPT$ or $REJECT$
      // When a worsening solution passes the traditional moveAcceptance pre-test,
      it is handled as follows
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:         $s_i \leftarrow s_{new}$. // $s_i$ is replaced by $s_{new}$ in the non-dominated set
15:      end if
16:   end if
17:   if (($result$ is $ACCEPT$) and ($f(s_{new}) \leq f(s_i)$)) then
18:      $s_i \leftarrow s_{new}$. // $s_i$ is replaced by $s_{new}$ in the non-dominated set
19:      $improveNonDominatedSet(i)$
20:   end if
      // if $result$ is $REJECT$ continue
21: end while
```

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

```
1: for ($j = i, UB$) do
2:    if ($f(s_{(j+1)}) \leq f(s_j)$) then
3:       $BREAK$. // No further improvement is possible
4:    else
5:       Randomly select a divide heuristic, $LLDH$.
6:       $s_{new} \leftarrow Apply(LLDH, s_j)$.
7:       $s_{(j+1)} \leftarrow s_{new}$. // $s_{(j+1)}$ is replaced by $s_{new}$ in the non-dominated set
8:    end if
9: end for
```

3.2 Selection Hyper-heuristic Components

An investigation of the performance of the grouping hyper-heuristic framework over a set of selected benchmark instances from the data clustering problem domain is carried out using different selection hyper-heuristic implementations. A total of 9 selection hyper-heuristics is generated using the combinations of the Simple Random (SR), Reinforcement Learning (RL) and Adaptive Dynamic Heuristic Set (ADHS) heuristic selection methods, and the Late Acceptance (LACC), Great Deluge (GDEL) and Iteration Limited Threshold Accepting (ILTA) move acceptance methods. From this point onward, a selection hyper-heuristic will be denoted as *heuristics selection-move acceptance*. For example, SR-GDEL is the hyper-heuristic that combines simple random selection method with great deluge move acceptance criterion.

SR chooses a low level heuristic at random. RL [27] maintains a utility score for each low level heuristic. If a selected heuristic generates an improved solution then its score is increased by one, otherwise it is decreased by one. At each decision point, the heuristic with the maximum score is selected. LACC [28] accepts all improving moves, however a worsening current solution is compared to a previous solution which was visited at a fixed number of steps prior during the search. If the current solution's objective value is better than that previous solution's objective value, it is accepted. Hence, a fixed size list containing previous solutions is maintained and this list gets updated at each step. A slightly modified version of GDEL is used in here and multiple lists are maintained, where a list is formed for each active group number (number of clusters). GDEL [29] sets a target objective value and accepts all solutions whether improving or worsening as long as the objective value of the current solution is better than the target value. The target values is often taken as the objective value of the initial solution and it is decreased linearly in time towards a minimum expected objective value. ADHS-ILTA [30] is one of the best performing hyper-heuristics in the literature. This elaborate online learning hyper-heuristic won the CHeSC 2011 competition across six hard computational problem domains [31]. The learning heuristic selection method consists of various mechanisms, such as for creating new heuristics vi relay hybridisation or for excluding the low level heuristics with poor performance. The move acceptance component is adaptive threshold method. The readers can refer to [30] for more details on this hyper-heuristic.

4 Application of Grouping Hyper-heuristics to Data Clustering

In this section, we provide the performance comparison of nine selection hyper-heuristics formed by the combination of {SR, RL, DH} heuristic selection and {ILTA, LACC, GDEL} move acceptance methods for data clustering. The performance of the approaches proposed based on the developed framework are further compared to previous approaches from the literature.

4.1 Experimental Data

16 data clustering problem instances that have different properties and sizes were used in this study. The first 12 of these instances, shown in Table 1 and Figure 1, were taken from [8]. The top 6 instances are 2-D hand-crafted problem instances that contain interesting data properties, such as different cluster sizes and high degrees of overlap between the clusters. Instances Square1, Square4 and Sizes5 contain four clusters each. The main difference between these instances is that clusters in Square1 and Square4 are of equal size, whereas the clusters in Sizes5 are not. On the other hand, data instance Long1 consists of two well-connected long clusters. Problem instances Twenty and Forty exhibit a mixture of the properties discusses above. The 6 problem instances on the bottom half of Figure

Table 1. The characteristics of the synthetic, Gaussian and real-world data clustering problem instances used during the experiments. N is the number of items, D is the number of dimensions/attributes and k^* is the best number of clusters [8]. L and U are the lower and upper bounds for the k values used during the experiments.

	Data Clustering				Range (k)	
	Instance	N	D	k^*	LB	UB
Synthetic	Square1	1000	2	4	2	9
	Square4	1000	2	4	2	9
	Sizes5	1000	2	4	2	9
	Long1	1000	2	4	2	9
	Twenty	1000	2	20	16	24
	Fourty	1000	2	40	36	44
Gaussian	2D-4c	1623	2	4	2	9
	2D-10c	2525	2	10	6	14
	2D-20c	1517	2	20	16	24
	2D-40c	2563	2	40	36	44
	10D-4c	958	10	4	2	9
	10D-10c	3565	10	10	6	14
Real	Zoo	101	16	7	3	11
	Iris	150	4	3	2	7
	Dermatology	366	34	6	2	10
	Breast-cancer	699	9	2	2	7

1 are randomly generated instances that were created using the Gaussian cluster generator described in [8]. Instances 2D-4c, 2D-10c, 2D-20c and 2D-40c are all 2 dimensional instances containing 4, 10, 20 and 40 clusters respectively. Similarly, instances 10D-4c and 10D-10c are 10 dimensional instances that contain 4 and 10 clusters respectively.

Additionally, 4 real world problem instances were used in this study. These are taken from the UCI Machine Learning Repository [32], which maintains multiple data sets as a service to the machine learning community. These selected real-world instances differ from each other in many ways, such as in the number of clusters, number of dimensions as well as the data type of the values

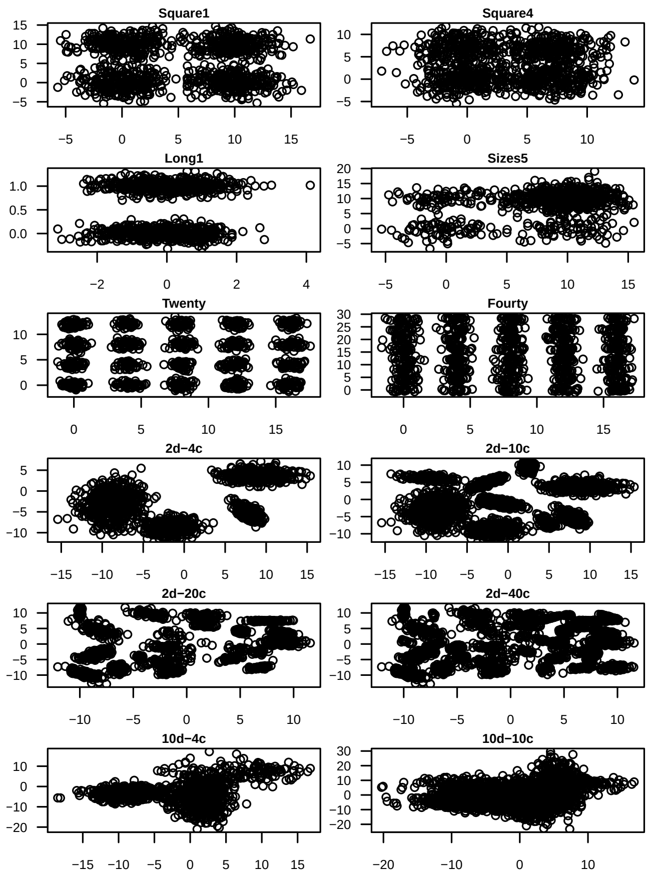

Fig. 1. The synthetic data clustering problem instances used in the experiments.

in each dimension. For example, Iris instance contains 3 equal-sized clusters of 50 data items each, and the data type of the values of each dimension is continuous, whereas Dermatology instance contains 6 clusters of different sizes of $(112, 61, 72, 49, 52, 20)$ data items, and the data type of each dimension is integer.

4.2 Trials and Parameters Settings and CPU Specifications

Based on initial experiments, the initial score value of all the LLHs in the RL selection method were set to (upper score bound - 2 * number of heuristics). The upper and lower score bounds of each one of the LLHs are set to 40 and 0 respectively, and the score increments and decrements values are both set to 1. The GDEL parameters 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 [29]. A LACC approach that uses k separate lists of equal lengths, one for each value of $k \in [LB, UB]$, is adopted based on the results of some initial experiments. A separate list of 50 previous solutions is maintained for each one of the solutions in the current set of non-dominated solutions. The parameters of the ADHS and ILTA followed the same settings as suggested in the literature [30].

30 initial solutions are created randomly for each one of the problem instances. In order to avoid initialization bias, all hyper-heuristic approaches operated on the same 30 initial solutions for each problem instance. Each experiment was repeated 30 times, for 600 seconds each. Experiments were conducted on 3.6GHz Intel Core $i7-3820$ machines with 16.0GB of memory, running on "Windows 7 OS".

4.3 Evaluation Criteria

In each experiment, each one of the 9 different hyper-heuristics used in this study starts from an initial set of non-dominated solutions and tries to find the best non-dominated set that can be found in the given period of time allowed for each run. The overall success of a hyper-heuristic is evaluated using *success rate*, denoted as ($sRate\%$) which indicates the percentage of the runs in which the expected (best) number of groups/corner points has been successfully found by a given algorithm; and the average time (in seconds) taken to achieve those success rates which is calculated using the duration of the successful runs only.

4.4 Experimental Results and Remarks

The actual values for the different dimensions within each one of the instances described above were measured on different scales. Consequently, some data processing was carried out before applying the grouping hyper-heuristics on these clustering instances. In this pre-processing, the real data is normalized such that the mean is equal to 0 and the standard deviation is equal to 1 in each dimension.

Initial experiments were conducted to observe the behavior of the grouping hyper-heuristic framework 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 corner point is compared to the performance of some previously proposed approaches.

The results are summarised in Tables 2, 3 and 4, showing the average best grouping, the standard deviation and the success rate for each of the grouping hyper-heuristics for each of the instances over 30 runs. The bottom row of each table, titled 'wins', shows the number of times each grouping selection hyper-heuristic has achieved the best average grouping including the ties with the other algorithms. A standard deviation of ± 0.0 for a particular algorithm indicates that this algorithm succeeded to find the best grouping for the particular instance over the 30 runs and achieved a 100% success rate. In general, hyper-heuristics using the ILTA selection method performs better than the others. Grouping hyper-heuristics which use LACC or GDEL as acceptance method achieved low

Table 2. The Performance of Reinforcement Learning Selection Hyper-heuristics: the success rate (*sRate*%), the average best number of clusters ($\mu(k_{best})$) and the standard deviation ($\sigma(k_{best})$) of each hyper-heuristic approach on the data clustering problem instances over the 30 runs.

			RL-ILTA			RL-LACC			RL-GDEL		
	Instance	k^*	sRate%	$\mu(k_{best})$	$\sigma(k_{best})$	sRate%	$\mu(k_{best})$	$\sigma(k_{best})$	sRate%	$\mu(k_{best})$	$\sigma(k_{best})$
Synthetic	Square1	4	**100.00**	**4.0**	±0.0	86.67	4.13	±0.35	93.33	4.07	±0.25
	Square4	4	**96.67**	**4.03**	±0.18	93.33	4.07	±0.25	86.67	4.13	±0.35
	Sizes5	4	**83.33**	3.9	±0.40	70.00	**4.03**	±0.56	73.33	**4.03**	±0.61
	Long1	4	**13.33**	**5.53**	±2.19	3.33	6.07	±2.07	3.33	5.83	±1.98
	Twenty	20	**23.33**	**20.87**	±1.36	16.67	20.9	±1.37	6.67	**20.87**	±1.38
	Fourty	40	**20.00**	**41.07**	±1.72	16.67	41.43	±1.73	10.00	41.33	±2.04
Gaussian	2D-4c	4	**16.67**	3.83	±0.59	6.67	**3.90**	±0.96	10.00	3.83	±0.83
	2D-10c	10	**13.33**	8.83	±1.21	3.33	8.97	±1.27	10.00	**9.17**	±1.46
	2D-20c	20	**63.33**	18.0	±2.30	40.00	18.23	±2.33	50.00	**18.73**	±2.35
	2D-40c	40	**20.00**	32.6	±3.24	10.00	32.4	±3.41	13.33	**32.77**	±3.21
	10D-4c	4	**13.33**	3.63	±1.07	6.67	3.77	±1.43	3.33	**4.13**	±1.72
	10D-10c	10	0.00	8.57	±1.74	0.00	8.5	±1.59	0.00	**8.9**	±1.86
Real	Zoo	7	**36.67**	**7.30**	±1.49	13.33	7.6	±1.61	16.67	7.67	±1.60
	Iris	3	**20.00**	**4.47**	±1.83	6.67	4.83	±1.84	3.33	4.90	±1.92
	Dermatology	6	**20.00**	**6.37**	±1.50	10.00	**6.37**	±1.52	3.33	6.57	±1.77
	Breast-cancer	2	**16.67**	**3.33**	±0.92	6.67	3.43	±0.90	6.67	3.43	±0.82
	Wins		**15**	**9**	–	0	3	–	0	7	–

Table 3. The Performance of Adaptive Dynamic Heuristics Set (ADHS) Selection Hyper-heuristics: the success rate (*sRate*%), the average best number of clusters ($\mu(k_{best})$) and the standard deviation ($\sigma(k_{best})$) of each hyper-heuristic approach on the data clustering problem instances over the 30 runs.

			ADHS-ILTA			ADHS-LACC			ADHS-GDEL		
	Instance	k^*	sRate%	$\mu(k_{best})$	$\sigma(k_{best})$	sRate%	$\mu(k_{best})$	$\sigma(k_{best})$	sRate%	$\mu(k_{best})$	$\sigma(k_{best})$
Synthetic	Square1	4	**100.00**	**4.0**	±0.0	83.33	4.23	±0.57	100.00	4.0	±0.0
	Square4	4	**100.00**	**4.0**	±0.0	76.67	4.33	±0.66	93.33	4.07	±0.25
	Sizes5	4	**80.00**	3.87	±0.43	70.00	**4.0**	±0.64	73.33	4.13	±0.68
	Long1	4	**16.67**	**5.60**	±2.22	3.33	6.0	±2.05	3.33	6.27	±2.24
	Twenty	20	**36.67**	20.93	±1.50	30.00	**20.90**	±1.56	30.00	21.0	±1.49
	Fourty	40	**23.33**	**41.03**	±1.69	6.67	41.6	±1.90	3.33	41.6	±2.08
Gaussian	2D-4c	4	**23.33**	3.87	±0.51	13.33	**4.07**	±0.94	6.67	4.3	±1.39
	2D-10c	10	**20.00**	**8.97**	±1.22	3.33	8.77	±1.43	10.00	8.9	±1.35
	2D-20c	20	**73.33**	18.23	±2.24	43.33	18.57	±2.21	23.33	**18.63**	±2.58
	2D-40c	40	**26.67**	**33.6**	±3.80	10.00	32.57	±3.33	6.67	32.63	±3.37
	10D-4c	4	**20.00**	3.8	±0.961	6.67	**4.03**	±1.35	6.67	4.17	±1.84
	10D-10c	10	**10.00**	8.83	±1.62	0.00	8.93	±1.87	0.00	**9.1**	±1.90
Real	Zoo	7	**56.67**	**7.30**	±1.47	36.67	7.37	±1.63	3.33	7.73	±1.72
	Iris	3	**26.67**	**4.5**	±1.68	3.33	5.23	±1.45	3.33	5.3	±1.32
	Dermatology	6	**26.67**	**6.37**	±1.45	10.00	6.6	±1.57	6.67	6.43	±1.74
	Breast-cancer	2	**13.33**	**3.30**	±0.95	3.33	3.67	±0.99	0.00	3.70	±1.06
	Wins		**16**	**10**	–	0	4	–	0	2	–

Table 4. The Performance of Simple Random (SR) Selection Hyper-heuristics: the success rate (*sRate%*), the average best number of clusters ($\mu(k_{best})$) and the standard deviation ($\sigma(k_{best})$) of each hyper-heuristic approach on the data clustering problem instances over the 30 runs.

			SR-ILTA			SR-LACC			SR-GDEL		
	Instance	k^*	sRate%	$\mu(k_{best})$	$\sigma(k_{best})$	sRate%	$\mu(k_{best})$	$\sigma(k_{best})$	sRate%	$\mu(k_{best})$	$\sigma(k_{best})$
Synthetic	Square1	4	**100.00**	**4.0**	±0.0	80.00	4.27	±0.58	86.66	4.17	±0.46
	Square4	4	**86.67**	**4.13**	±0.35	83.33	4.3	±.75	73.33	4.57	±1.10
	Sizes5	4	**80.00**	**4.13**	±0.63	63.33	4.33	±0.76	70.00	**4.13**	±0.78
	Long1	4	**6.67**	**5.73**	±2.24	**6.67**	5.90	±2.16	**6.67**	5.93	±2.35
	Twenty	20	**26.67**	**20.93**	±1.39	10.00	21.37	±1.69	16.67	21.27	±1.62
	Fourty	40	**3.33**	**41.27**	±1.70	**3.33**	41.67	±1.94	0.00	41.47	±1.82
Gaussian	2D-4c	4	**16.67**	**4.03**	±0.93	3.33	4.3	±1.34	6.67	4.43	±1.38
	2D-10c	10	**10.00**	8.87	±1.28	3.33	8.73	±1.17	6.67	**9.03**	±1.30
	2D-20c	20	**46.67**	18.77	±2.47	26.67	18.93	±2.49	6.67	**19.3**	±2.47
	2D-40c	40	**23.33**	**33.6**	±3.71	3.33	32.6	±3.39	6.67	32.77	±3.51
	10D-4c	4	3.33	3.67	±0.88	0.00	3.6	±1.30	**10.00**	**3.83**	±1.49
	10D-10c	10	0.00	8.77	±1.72	0.00	8.7	±1.73	0.00	**9.03**	±1.73
Real	Zoo	7	**36.67**	**7.33**	±1.56	10.00	7.53	±1.55	6.67	7.90	±1.67
	Iris	3	**20.00**	**5.23**	±1.41	6.67	5.4	±1.28	10.00	5.47	±1.25
	Dermatology	6	**13.33**	**6.60**	±1.61	3.33	6.73	±1.78	3.33	6.73	±1.76
	Breast-cancer	2	**6.67**	**4.03**	±1.45	0.00	4.3	±1.62	0.00	4.27	±1.60
	Wins		**14**	**12**	–	2	0	–	2	5	–

success rates in most of the instances, scoring a success rate that is less than 40% in most of the real and Gaussian instances. ADHS-ILTA hyper-heuristic receives the most number of wins across all the tested approaches, while RL-ILTA and SR-ILTA follow it in that order, respectively, as could be seen in the tables. ADHS-ILTA delivers the best success rate on all Gaussian instances, and most of the remaining instances. On average, ADHS-ILTA performs better than the other grouping hyper-heuristics on 50% of instances and the standard deviation associated with the average values is the lowest in most of the cases. The performance of ADHS-ILTA is, therefore, taken for further comparison with other known clustering algorithms as shown in Table 5 including "k-means" [20], "Mock" [8], "Ensemble" [26], "Av. Link" [22], and "S. Link" [22]. The comparison results shown in Table 5 are based on the average number of clusters. The 'wins' row at the bottom of the table indicates the number of winning times each approach achieved over all the Gaussian and the Synthetic instances. Given this table, it is observed that the performance of ADHS-ILTA lies on the top of the other approaches equally with MOCK algorithm scoring five wins, and outperforming the other competing algorithms. ADHS-ILTA scored the best average distinctively in four instances including, Twenty, Forty, 2D-4c and 2D-20c.

Table 5. Comparing the performances of different approaches on data clustering problem instances based on the average best number of clusters. The entries in bold indicate the best result obtained by the associated algorithm for the given instance.

	Instance	k^*	ADHS-ILTA	k-means	MOCK	Ensemble	Av. link	S. link
Synthetic	Square1	4	**4.00**	**4.00**	**4.00**	**4.00**	**4.00**	2.72
	Square4	4	**4.00**	**4.00**	**4.00**	4.04	4.26	2.00
	Sizes5	4	**3.87**	3.74	**3.87**	3.70	3.76	2.44
	Long1	4	5.60	8.32	8.34	**4.92**	7.78	2.02
	Twenty	20	20.93	–	–	–	–	–
	Fourty	40	41.03	–	–	–	–	–
Gaussian	2D-4c	4	**3.87**	3.69	3.70	2.23	4.50	4.40
	2D-10c	10	8.97	10.66	**9.65**	4.50	15.20	8.00
	2D-20c	20	**18.23**	21.87	17.31	1.24	16.30	16.3
	2D-40c	40	33.60	30.63	**35.26**	2.27	30.90	27.7
	10D-4c	4	3.80	3.59	3.60	5.37	**4.00**	2.00
	10D-10c	10	8.83	**9.02**	8.88	3.86	5.30	2.00
	Wins		**5**	3	**5**	2	2	0

5 Conclusion

In this study, the grouping hyper-heuristic framework, previously applied to graph colouring, is extended to handle the data clustering problem. The performances of various selection hyper-heuristics are compared using a set of benchmark instances which vary in terms of the number of items, groups as well as number and nature of dimensions. This investigation is carried out using different pairwise combinations of the 'simple random', the 'reinforcement learning' and the 'adaptive dynamic heuristic set' heuristic selection methods and the 'late acceptance', 'great deluge' and 'iteration limited threshold accepting' move acceptance methods. The genetic grouping algorithm encoding is used as a solution representation. The best heuristic selection and move acceptance turns out to be the Adaptive Dynamic Heuristic Set and Iteration Limited Threshold Accepting methods. This selection hyper-heuristic, winner of a hyper-heuristic challenge performing well across six different problem domains, is sufficiently general and very effective considering that it still ranks the best algorithm for data clustering as well.

The empirical results show that the proposed framework is indeed sufficiently general and reusable. Also, although the ultimate goal of the grouping framework is not to beat the state of the art techniques that are designed and tuned for specific problems, the results obtained by learning-based hyper-heuristics which uses feedback during the search process turned out to be very competitive when compared to previous approaches from the literature.

References

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

2. Agustın-Blas, L.E., Salcedo-Sanz, S., Jiménez-Fernández, S., Carro-Calvo, L., Del Ser, J., Portilla-Figueras, J.A.: A new grouping genetic algorithm for clustering problems. Expert Systems with Applications **39**(10) (August 2012) 9695–9703
3. Mitra, S., Banka, H.: Multi-objective evolutionary biclustering of gene expression data. Pattern Recognition **39**(12) (2006) 2464–2477
4. Park, Y.J., Song, M.S.: 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, University of Wisconsin, Madison, Wisconsin, USA, Morgan Kaufmann (22-25 July 1998) 568–575
5. Burke, E.K., Gendreau, M., Hyde, M.R., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: a survey of the state of the art. JORS **64**(12) (2013) 1695–1724
6. Elhag, A., Özcan, E.: A grouping hyper-heuristic framework: Application on graph colouring. Expert Systems with Applications **42**(13) (2015) 5491 – 5507
7. Talbi, E.G., Bessiere, P.: A parallel genetic algorithm for the graph partitioning problem. In: Proceedings of the 5th international conference on Supercomputing, ACM (1991) 312–320
8. Handl, J., Knowles, J.D.: An evolutionary approach to multiobjective clustering. IEEE Trans. Evolutionary Computation **11**(1) (2007) 56–76
9. Ülker, Ö., Özcan, E., Korkmaz, E.E.: Linear linkage encoding in grouping problems: Applications on graph coloring and timetabling. In Burke, E.K., Rudová, H., eds.: PATAT. Volume 3867 of Lecture Notes in Computer Science., Springer (2006) 347–363
10. Radcliffe, N.J.: Formal analysis and random respectful recombination. In: Proceedings of the 4th International Conference on Genetic Algorithm. (1991) 222–229
11. Radcliffe, N.J., Surry, P.D.: Fitness variance of formae and performance prediction. In Whitley, L.D., Vose, M.D., eds.: FOGA, Morgan Kaufmann Publishers Inc. (1994) 51–72
12. Falkenauer, E.: The grouping genetic algorithms: Widening the scope of the GAs. Belgian Journal of Operations Research, Statistics and Computer Science (JORBEL) **33**(1-2) (1992) 79–102
13. Brown, C.E., Sumichrast, R.T.: Impact of the replacement heuristic in a grouping genetic algorithm. Computers & OR **30**(11) (2003) 1575–1593
14. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of Intelligent Information Systems **17** (2001) 107–145
15. Rousseeuw, P.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**(1) (November 1987) 53–65
16. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. **1**(2) (February 1979) 224–227
17. Rand, W.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association **66**(336) (1971) 846–850
18. Jain, A.K., Dubes, R.C.: Algorithms for clustering data. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1988)
19. Chang, D.X., Zhang, X.D., Zheng, C.W.: A genetic algorithm with gene rearrangement for k-means clustering. Pattern Recognition **42**(7) (2009) 1210 – 1222
20. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Number 14 in 1, California, USA (1967) 281–297

21. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B **39**(1) (1977) 1–38
22. Voorhees, E.M.: The effectiveness and efficiency of agglomerative hierarchical clustering in document retrieval. PhD thesis (1985)
23. Kohonen, T.: The self-organizing map. Proceedings of the IEEE **78**(9) (1990) 1464–1480
24. Ankerst, M., Breunig, M.M., peter Kriegel, H., Sander, J.: Optics: Ordering points to identify the clustering structure. In Delis, A., Faloutsos, C., Ghandeharizadeh, S., eds.: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data. SIGMOD '99, ACM Press (1999) 49–60
25. Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: a survey. IEEE/ACM Transactions on Computational Biology and Bioinformatics **1** (2004) 24–45
26. Hong, Y., Kwong, S., Chang, Y., Ren, Q.: Unsupervised feature selection using clustering ensembles and population based incremental learning algorithm. Pattern Recognition **41**(9) (September 2008) 2742–2756
27. Özcan, E., Misir, M., Ochoa, G., Burke, E.K.: A reinforcement learning-great-deluge hyper-heuristic for examination timetabling. Int. J. Appl. Metaheuristic Comput. **1**(1) (January 2010) 39–59
28. Burke, E.K., Bykov, Y.: The late acceptance hill-climbing heuristic. European Journal of Operational Research **258**(1) (2017) 70 – 78
29. Dueck, G.: New optimization heuristics: The great deluge algorithm and the record-to-record travel. Journal of Computational Physics **104** (January 1993) 86–92
30. Misir, M., Verbeeck, K., Causmaecker, P.D., Berghe, G.V.: A new hyper-heuristic as a general problem solver: an implementation in hyflex. J. Scheduling **16**(3) (2013) 291–311
31. Burke, E., Curtois, T., Hyde, M., Kendall, G., Ochoa, G., Petrovic, S., Vazquez-Rodriguez, J.: Hyflex: A flexible framework for the design and analysis of hyper-heuristics. In: Proceedings of the Multidisciplinary International Scheduling Conference (MISTA09). (2009) 790–797
32. Bache, K., Lichman, M.: UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Science (2013)