A Grouping Hyper-heuristic Framework based on Linear Linkage Encoding For Graph Coloring

Anas Elhag and Ender Özcan
ASAP Research Group, School of Computer Science
University of Nottingham, Jubilee Campus
Nottingham, NG8 1BB, UK
Email: {axe, exo}@cs.nott.ac.uk

***Abstract*—Grouping problems are a class of combinatorial optimization problems in which the task is to search for the best partition of a set of objects into a collection of mutually disjoint subsets while satisfying a given set of constraints. Typical examples include data clustering, graph coloring and exam timetabling problems. Selection hyper-heuristics based on iterative search frameworks are high level general problem solving methodologies which operate on a set of low level heuristics to improve an initially generated solution via heuristic selection and move acceptance. In this paper, we describe a selection hyper-heuristic framework based on an efficient representation referred to as linear linkage encoding for multi-objective grouping problems. This framework provides the implementation of a fixed set of low level heuristics that can work on all grouping problems where a trade-off between a given objective and number of groups is sought. The empirical results on graph coloring problem indicates that the proposed grouping hyper-heuristic framework can indeed provide high quality solutions.**

I. Introduction

Grouping problems comprise a set of very well known combinatorial optimisation problems, such as data clustering, timetabling, and graph colouring [1]. Grouping problems require the common task of partitioning a set of objects U into a collection of mutually disjoint subsets u_i of U such that each object is in exactly one subset (Equation 1).

$$\begin{aligned} \bigcup u_i &= U \quad \forall i \\ u_i \bigcap u_j &= \emptyset \quad \forall i, \forall j \quad where\ i \neq j \\ u_i &\neq \emptyset \quad \forall i \end{aligned} \tag{1}$$

Such grouping (partitioning) problem arises in many real world and industrial problems such as packing and timetabling, most of which are known to be NP-hard [1]. Using an exact algorithm to solve a grouping problem might not be attainable in such cases, requiring alternative solution methods such as heuristics. Most of the grouping problems have been tackled individually; and many problem specific solution methodologies for a given problem are available in the literature. However, the vast majority of these methodologies are not reusable, since they have been specifically tailored to fit particular problem domains.

Different grouping problems have different constraints and so they introduce different objectives. Consequently, not all groupings are feasible for all grouping problems. For instance, in graph colouring problem [2], assuming that each color represnts a group, no two connected nodes are allowed to be in the same group. Mostly, the overall fitness f of a grouping problem solution $U_g = \{u_1, ..., u_i, ..., u_k\}$ can be measured using an evaluation function which adds up the partial contribution from each group as in Equation 2.

$$f(U_g) = \sum_{i=1}^{k} f(u_i) \tag{2}$$

An important common feature of the grouping problems as defined is that they require the number of groups k to be minimized as well as the combined fitness of the groups $f(U_g)$. Consequently, grouping problems are considered to be multi-objective optimisation problems [3]; or more precisely, bi-objective optimisation problems. In the graph colouring problem, the minimum number of colours causing the least conflicts is searched. These objectives are conflicting objectives; i.e optimising one of them will cause the other to deteriorate. Solving a grouping search problem will, hopefully, yield a set of optimal solutions at the end of the search process. Each one of these solutions is better than the others in terms of one of the objectives, and worse in terms of the other. This set of optimal solutions is known as the *pareto-optimal front*. Traditionally, the pareto-optimal front is approximated by a set of what is called non-dominated solutions. A solution U_1 dominates another solution U_2 if

- U_1 is no worse than U_2 in any of the objectives, and
- U_1 is better than U_2 in at least one objective.

Hyper-heuristics are methodologies that select or generate heuristics during the search process for solving computationally hard problems [4], [5]. Hyper-heuristics are designed to solve as many optimization problems as possible, by exploring search spaces of heuristics rather than problem solutions. In order to facilitate the development of hyper-heuristics, a software benchmark framework known as *HyFlex* has recently been developed to support the design of cross-domain heuristic search methods [6]. It provides a common interface to the algorithmic components for six selected hard combinatorial problems. Researchers only need to provide the high-level strategy that will manage and configure the provided algorithmic components. HyFlex has been used to support The First Cross-Domain Heuristic Search Challenge (CHeSC 2011) [7], which thenceforth became a benchmarking tool for selection hyper-heuristics across different problem domains.

Typically, different sets of low level heuristics are provided to the hyper-heuristics with different problem domains. However, we are aiming to achieve a further level of generality in this study, by designing a selection hyperheuristic framework with a fixed set of low level heuristics that is designed to work on an efficient representation, referred to as Linear Linkage Encoding (LLE), in order to allow the same set of low level heuristics to be used with all grouping problems. Our aim is not to beat the state of the art techniques which are designed and tuned for a particular problem domain or even a particular instance in a given domain. Our aim is to provide a single selection hyper-heuristic framework for all grouping problems. We tested our framework on instances from the graph coloring problem domain, in this study.

II. Background

A. Selection Hyper-Heuristics

A selection hyper-heuristic explores a search space of heuristics by *choosing* a heuristic from a set of problem specific heuristics in order to transform the problem at each decision point. [5] identified the importance of two successive processes that a selection hyper-heuristic employs:

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 the resulting solution.

The heuristic selection step is carried out using a variety of learning and non-learning approaches. *Simple random* (SR) chooses a random low level heuristic at each step, while *random permutation* (RP) applies the low level heuristics in a specific order that is randomly determined at the beginning of the search [8], [9], [10]. Some hyperheuristics apply learning approaches that use some online feedback from the search process, such as the *reinforcement learning* (RL) [11] which 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, by means of increasing and decreasing their scores or probabilities of being selected in the future. At each decision point, the low level heuristic with the highest score is selected. Other hyperheuristics use more sophisticated selection approaches, including meta-heuristics such as GAs, tabu search and great deluge. Also, various methods are used to perform the move acceptance step. Some of these methods are fairly simple such as the (IEQ) method which accepts *improving or equal* moves [12], [13]. Advanced methods, such as *great deluge* (GD) and *simulated annealing* (SA), accept all the improving moves and use some probability-based criteria to decide whether to accept the worsening ones. A selection hyperheuristic is created by combining any of the heuristic selection methods with any of the move acceptance methods.

B. Hyper-heuristics for Multi-objective Optimization Problems

Hyperheuristics for multi-objective optimization problems (HHMO) is a new area of research in Evolutionary Computation and Operational Research [4], [5]. There are a few studies on multi-objective hyper-heuristics in the literature. A multi-objective hyper-heuristic based on tabu search was proposed in [9]. The key feature of this study is the use of tabu search as a high-level search strategy to choose a suitable heuristic at each iteration for solving space allocation and timetabling problems. Another approach proposed in [14] comprises of two phases. The first phase aims to produce an efficient Pareto front (this may be of low quality based on the density), while the second phase aims to deal with a given problem in a flexible way to drive a subset of the population to the desired Pareto front. This approach was evaluated on the multi-objective traveling salesman problems using eleven low level heuristics.

A comparison to other multi-objective approaches from the literature reveals that the proposed approach generates good quality solutions, however, future work is still needed to improve the methodology. An online learning Markov chain based selection hyper-heuristic was described in [15]. The Markov chains guide the selection of low level heuristics and online reinforcement learning is used to adapt the transition weights between heuristics. This approach was applied over some benchmark instances, and compared to a (1+1) Evolution Strategy meta-heuristic, a random hyper-heuristic and a multi-objective hyper-heuristic [9]. The comparison shows the efficacy of the proposed approach in terms of Pareto converge and learning ability to select good heuristics combinations. [16] investigated hyper-heuristics and their hybridization with multi-objective evolutionary algorithms. A choice function-all moves hyper-heuristic was proposed in this study. The choice function acts as a high level search strategy. Three multi-objective evolutionary algorithms act as low level heuristics. The experimental results over a set of benchmark problem instances show that this approach can benefit from the strengths of the low level heuristics and avoid their weaknesses. In the majority of cases, the choice function-all moves hyper-heuristic outperforms the other multi-objective evolutionary algorithms when used in isolation. Moreover, the choice function hyper-heuristic has the capability to intelligently adapt to calling combinations of low-level heuristics. In this study, we keep the same heuristic selection method along with the low level heuristics while investigate the performance of the same framework when great deluge is used as a move acceptance criterion.

C. Grouping Representation

In this study, we used a powerfull grouping representation referred to as *linear linkage encoding* (LLE) [17], which is a restricted form of the more general *linkage encoding* scheme. Each gene in the LLE representation represents one object, and stores an integer value (index) which represents a link between that object and another object belonging to the same group. Starting from an object in a given group, the links are used to reach to all of the other objects within that group. The membership of a given object can be identified using either the starting or the ending node of the cluster.

LLE representation can be implemented using a constant length array for a given grouping problem instance. Formally, an LLE solution should satisfy the following two conditions [17]:

1) The integer value stored in each gene is greater than or equal to its index but less than or equal to the number of the objects.
2) Other than the index of an ending node, no two genes can have the same value.

D. Graph Coloring Problem

The task in the graph coloring problem is to assign a color to each vertex of an undirected graph such that no two connected vertices have the same color. The objective is to minimise the number of colors used, where each vertex is assigned to exactly one color. The minimum number of colors that can be used to solve a graph coloring problem is known as the *chromatic number* [18]. Given a graph $G = (V, E)$ with vertex set V and edge set E, and given an integer k, a *k-coloring* of G is a function $c : V \rightarrow 1, ..., k$. The value of $c(x)$ of a vertex is called the color of x. The vertices with color $r(1 \leq r \leq k)$ define a color class, denoted V_r. If two connected vertices x and y have the same color r, x and y are conflicting vertices, and the edge E_x, y is called a conflicting edge. If there are no conflicting edges, then the color classes are all independent sets and the *k-coloring* is valid. The graph coloring problem is to determine the minimum integer k (the chromatic number of $G - x(G)$) such that there exists a legal *k-coloring* of G [19]. There are many studies on graph coloring. Recursive largest fit is a well-known greedy heuristic introduced by [20]. [21] proposed a coding as an ordering of vertices which could be used in a genetic algorithm. [22] presented the first tabu search implementation which outperforms another local search method, simulated annealing on random dense graph instances. [23] presented three simulated annealing implementations based on three neighboring approaches. [19] proposed a variable neighborhood search algorithm for graph coloring problem.

III. A Grouping Selection Hyper-heuristic Framework

A. Framework Pareto Sets

For each instance, a range of expected number of groups $k_{min} - k_{max}$ is selected around the known best value of k. Then, throughout the search, a current pareto front maintains a number of solutions specified by the set_size (m) for each value of k in the selected range. The initial pareto front is built by randomly creating a set of m solutions for each value of k in the range. Firstly, m random solutions are created for $k = k_{min}$ and the fitness values for these solutions are calculated and maintained in an ascending order. Then, m solutions are created for the next values of $k \leq k_{max}$, one by one. At each point, if the fitness of the created solution is worse than the best fitness for the previous value of k, then this solution is discarded and a new solution is created instead of it. This is necessary to maintain the definition of the pareto front for the initial solutions; i.e. all the solutions in a given set at a particular k are better than the best solution in the set at the previous value of k. When the initialization step is over, the best solution at each k in the range is maintained in the best pareto front. The solutions in the current pareto front are then iteratively improved using a selection hyperheuristic that perturbs the one of the current solutions generating a new one using a chosen low level heuristic and then decides whether to accept or reject the new solution. The accpted solution replaces the worst solution in the corresponding set in the current pareto front.

B. Low Level Heuristics

We designed one crossover and five mutation heuristics as part of the framework.

- Merge and Merge tournament heuristics select 2 groups to be merged. These heuristics reduce the number of groups in the selected solution.
- Divide and Divide tournament heuristics select 2 groups to be divided, and consequently, increase the number of groups in the selected solution.
- Swap heurisitc exchanges 2 items between 2 randomly selected groups in the selected solution, and does not affect the number of groups in the solution.

The crossover operator works in the following manner:

- It selects a random group from the first parent and transfers it to the offspring. Then, it selects a group from the other parent and transfers it.
- The items that have already been transferred to the offspring are deleted from the group before it is copied into the offspring.
- If all the items in a specific group have already been transferred, then the whole group is discarded.
- This process continues until all the items are transferred to the offspring.
- At that point, if the number of the groups have exceeded k_{max}, then a random number of merge operations are performed until the number of groups in the resulting solution is in the range of $kmin - k_{max}$.

C. Delta Evaluation

The fitness evaluation of new solutions was found to be a very time consuming part of the hyperheuristic framework. Since the time is used as the stopping critertia for the experiments, we had to find a way to work around the fitness evaluation problem. Typically, this evaluation is carried out using the whole solution; i.e. calculating the fitness of every single group in the solution. Alternatively, the delta evaluation requires the computation of partial fitness contributions of the groups that are modified by a given low level heuristic, before and after that modification. For instance, if *swap* heuristic is applied and u_1 and u_2 are the groups that swap nodes n_i and

n_j, the overall fitness f of the resulting solution is calculated using equation 3 below:

$$f(U_{new}) = f(U_{old}) - (f(u_1) + f(u_2)) + (f(u_1^{'}) + f(u_2^{'})) \quad (3)$$

IV. Experiments and Results

A. Experimental Setup

We tested our grouping hyper-heuristic framework using randomly selected instances from the DIMACS challenge suite [24]. The characteristics of the selected instances are presented in Table I. The framework is designed to work with any combination of selection and acceptance methods. In this study, we implemented a hyper-heuristic that is composed of Simple Random (SR) as a selection method, and Improving or Equal (IEQ) as an acceptance method. The experiments were carried out on a 3.6GHz Intel Core i7-3820 Windows 7 machines with 16GB memory. Each hyper-heuristic was run 31 times with each dataset; and each run was terminated after 600 seconds.

TABLE I
The characteristics of the problem instances from the DIMACS benchmark suite. $|V|$ is the number of vertices, $|E|$ is the number of edges, % is the edge density and $\chi(G)$ is the chromatic number.

Instance	$\|V\|$	$\|E\|$	%	$\chi(G)$
zeroin.1.col	211	4100	0.19	49
zeroin.2.col	211	3541	0.16	30
zeroin.3.col	206	3540	0.17	30
flat300_20	300	21375	0.48	20
flat300_26	300	21633	0.48	26
flat300_28	300	21695	0.48	28
le450_15a	450	8168	0.08	15
le450_15b	450	8169	0.08	15
le450_15c	450	16680	0.17	15
le450_15d	450	16750	0.17	15
le450_25a	450	8260	0.08	25
le450_25b	450	8263	0.08	25
le450_25c	450	16680	0.17	25
le450_25d	450	16750	0.17	25

B. Results

As shown in Table II, the *best* solutions obtained by our framework, denoted as HH-LLE, are found to be superior to solutions obtained in previous studies. Greedy Partition Crossover Lowest Index (GPX-LI) and Greedy Partition Crossover Cardinality Based (GPX-CB) are graph coloring algorithms presented in [25]. (LLE-e) which is a modified version of the LLE representation referred to as Linear Linkage Encoding With Ending Node Links (LLE-e) is presented in [26] along side a classical Linear Linkage Encoding (LLE), and both of these are tested using genetic operators. The fields marked " – " in the table means that the solution for that problem instance with that specific method is not reported. From the table, it is clear that the grouping hyper-heuristic framework we proposed in this study succeeded to find the expected best quality solutions in all problem instances. The best results obtained are competitive with the previous studies, and outperform the old studies in some of the problem instances. We also compared our framework to a similar framework that uses only one point for each value of k in the current pareto front and only mutation operators as low level heuristics [27]. We found that our proposed framework pushes the initial pareto front much further towards the zero than the previous framework. Yet, the same best-of-runs results are obtained. Figure 1 shows a comparison between the initial and the final pareto fronts obtained by SRAN-IEQ for the zeroin.i.3.col data set. Figure 2 shows a comparison between the ratios of calls, acceptance and acceptance for best made to each low level heuristic by SRAN-IEQ hyper-heuristic on the same data set. More details of the multi-objective framework hyper-heuristic approach as well as more results on some other grouping problem domains will be provided at the conference.

TABLE II
Graph Coloring best colorings comparison

Instance	HH-LLE	GPX-LI	GPX-CB	LLE-e	LLE
zeroin.1.col	**49**	**49**	**49**	–	–
zeroin.2.col	**30**	31	31	–	–
zeroin.3.col	**30**	**30**	31	–	–
flat300_20	**20**	27	32	**20**	33
flat300_26	**26**	34	34	–	–
flat300_28	**28**	34	34	–	–
le450_15a	**15**	16	16	–	–
le450_15b	**15**	16	16	17	17
le450_15c	**15**	23	23	–	–
le450_15d	**15**	23	23	–	–
le450_25a	**25**	**25**	**25**	–	–
le450_25b	**25**	**25**	**25**	–	–
le450_25c	**25**	28	28	29	29
le450_25d	**25**	28	28	–	–
Wins/Draws	**9/5**	0/4	0/3	0/1	0/0s

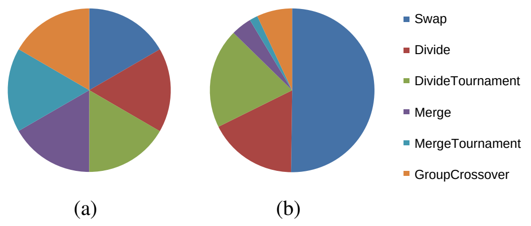

Fig. 2. The ratios of low level heuristics (a) selection (total of 3,544,363 decisions) and (b) acceptance (total of 762,047 decisions) by SRAN-IEQ on zeroin.i.3.col data set

V. Conclusion

In this study, we presented a general and effective Pareto-set-based grouping selection hyperheuristic framework that is provided with a fixed set of low level heuristics for solving any given grouping problem as described based on an efficient representation scheme; namely linear linkage encoding.

The proposed framework was tested using instances from the DIMACS challenge suite. When the experimental results

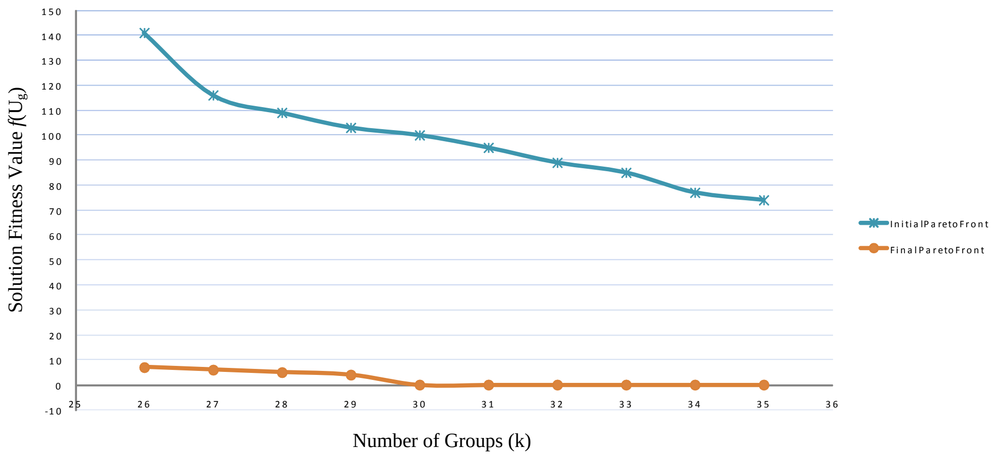

Fig. 1. Comparison between an initial and a final pareto fronts obtained by the SRAN-IEQ hyper-heuristic on zeroin.i.3.col data set.

are compared to the previously proposed approaches, they are found to be very competitive and promising. The framework succeeded to find the expected best solutions for the benchmark problem instances. In some cases, the framework outperform some of the problem-specific approaches. The MGIHH is found to be slightly better than the SRAN hyperheuristics. This is consistent with the previous findings ([28], [29], [30]). Learning during the heuristic selection process definitely helps, but move acceptance plays a major role in the performance of hyperheuristics.

In the future, a better approach to select the $k_{min} - k_{max}$ range can be researched. Also, more low-level heuristics can be tested along with new hyperheuristic methodologies.

References

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

[2] M. Chiarandini and T. Stützle, "An analysis of heuristics for vertex colouring," in *Experimental Algorithms, Proceedings of the 9th International Symposium, (SEA 2010)*, ser. Lecture Notes in Computer Science, P. Festa, Ed., vol. 6049. Springer, May 2010, pp. 326–337.

[3] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput*, vol. 8, no. 2, pp. 173–195, 2000.

[4] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and Q. Rong, "Hyper-heuristics: A survey of the state of the art, to appear," *Journal of the Operational Research Society*, 2013.

[5] E. Özcan, B. Bilgin, and E. Korkmaz, "A comprehensive analysis of hyperheuristics," *Intelligent Data Analysis*, vol. 12, no. 1, pp. 1–21, January 2008.

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

[7] E. Burke, M. Gendreau, M. Hyde, G. Kendall, B. McCollum, G. Ochoa, A. Parkes, and S. Petrovic, "The cross-domain heuristic search challenge - an international research competition," in *Proceedings of Learning and Intelligent Optmization (LION5)*, ser. Lecture Notes in Computer Science, X. Yao and C. A. C. Coello, Eds., vol. 6683, Jan 17-21, 2011, Rome, Italy 2011, pp. 631–634.

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

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

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

[11] E. K. Burke and E. Soubeiga, "Scheduling nurses using a tabu-search hyperheuristic," in *Proc. of the 1st MISTA*, 2003, pp. 197–218.

[12] B. Bilgin, E. Özcan, and E. E. Korkmaz, "An experimental study on hyper-heuristics and exam timetabling," in *Proceedings of the 6th international conference on Practice and theory of automated timetabling VI*, ser. PATAT'06. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 394–412.

[13] P. Cowling, G. Kendall, and E. Soubeiga, "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*, ser. Lecture Notes in Computer Science. Konstanz, Germany: Springer, August 2000, pp. 176–190.

[14] N. Veerapen, D. Landa-Silva, and X. Gandibleux, "Hyperheuristic as component of a multi-objective metaheuristic," in *SLS-DS 2009 : Doctoral Symposium on Engineering Stochastic Local Search Algorithms*, Bruxelles, Belgium, 2009-08, technical Report TR/IRIDIA/2009-024, IRIDIA, Universite Libre de Bruxelles, Brussels, Belgium.

[15] K. McClymont and E. C. Keedwell, "Markov chain hyper-heuristic (mchh): an online selective hyper-heuristic for multi-objective continuous problems," in *GECCO*, 2011.

[16] M. Maashi, G. Kendall, and E. Özcan, "A choice function based hyperheuristic for multi-objective optimization, to appear in jors." *Evolutionary Computation*, 2012.

[17] J. Du, E. E. Korkmaz, R. Alhajj, and K. Barker, "Novel clustering that employs genetic algorithm with new representation scheme and multiple objectives." in *DaWaK*, ser. Lecture Notes in Computer Science, Y. Kambayashi, M. K. Mohania, and W. W, Eds., vol. 3181. Springer, 2004, pp. 219–228.

[18] T. R. Jensen and B. Toft, *Graph Coloring Problems*. 1 edition, 1994.

[19] C. Avanthay, A. Hertz, and N. Zufferey, "A variable neighborhood search for graph coloring," *European Journal of Operational Research*, vol. 151, pp. 379–388, 2003.

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

[21] L. D. Davis and M. Mitchell, *Handbook of Genetic Algorithms*, 1991.

[22] A. Hertz and D. D. Werra, "Using tabu search techniques for graph coloring," *Computing*, 1987.

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

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

[25] O. Ülker, E. Özcan, and E. E. Korkmaz, "Linear linkage encoding in grouping problems: Applications on graph coloring and timetabling," in *International Conference on the Practice and Theory of Automated Timetabling*, 2006.

[26] B. Külahçıoğlu, "Multiobjective hyperheuristic for data clustering and linear linkage encoding," Master's thesis, Yeditepe University, 2007.

[27] M. Birben, "Solving grouping problems using a selection hyper-heuristic framework based on linear linkage encoding," Yeditepe University, Computer Engineering, Tech. Rep., 2011.

[28] E. K. Burke, G. Kendall, M. M. sır, and E. Özcan, "Monte carlo hyper-heuristics for examination timetabling," in *Conference on the Practice and Theory of Automated Timetabling (PATAT 2008)*, 2008.

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

[30] E. Özcan, M. Mısır, G. Ochoa, and E. K. Burke, "A reinforcement learning - great-deluge hyper-heuristic for examination timetabling," *Int. J. of Applied Metaheuristic Computing*, vol. 1, no. 1, pp. 39–59, 2010.