Hyperheuristic Approaches for Multiobjective Optimisation

E.K. Burke J.D. Landa Silva E. Soubeiga

Automated Scheduling, Optimisation and Planning Research Group
School of Computer Science and IT, University of Nottingham, UK
{ekb,jds,exs}@cs.nott.ac.uk

1 Introduction

In multiobjective optimisation the aim is to find solutions that represent a compromise between the various (sometimes conflicting) criteria used to evaluate the quality of solutions. A solution *x* is said to be non-dominated with respect to a set of solutions *S* if there is no other solution in *S* that is, as good as *x* in all the criteria and better than *x* in at least one of the criteria. In Pareto optimisation the goal is to find a set of solutions that is representative of the whole trade-off surface, i.e. non-dominated solutions that are a good approximation to the Pareto optimal front [5]. The present work proposes the use of hyperheuristics to improve the ability of local search-based metaheuristics to produce non-dominated fronts that are uniformly distributed over the desired trade-off surface. A hyperheuristic can be thought at as, basically, a heuristic that manages the application of a set of heuristics in order to solve an optimisation problem [1]. By using a hyperheuristic approach, the neighbourhood exploration can be targeted in order to guide the search towards the desired regions of the trade-off surface. This strategy takes into consideration the localization of the current solution(s) in the objective space and the ability of each neighbourhood exploration heuristic to achieve improvements on each of the objectives. That is, a hyperheuristic systematically tries to apply the neighbourhood exploration heuristic that improves on 'poor' objectives while maintaining the quality of 'rich' objectives on a given solution. This is a novel approach for tackling the problem of achieving a good coverage of the desired trade-off surface in multiobjective combinatorial optimisation.

2 Techniques for Improving the Distribution of Non-Dominated Sets

One of the issues of major concern when developing metaheuristics for Pareto optimisation is how to ensure that the algorithm produces a uniformly distributed non-dominated front at the end of the search. Several strategies that aim to improve the distribution of non-dominated solutions have been proposed. For example, the search can be directed towards the desired area of the trade-off surface by tuning weights [eg. 12]. Clustering or niching methods attempt to achieve a good distribution by assigning fitness to solutions based on the density of solutions in a given area [5]. Fitness sharing is a clustering technique that reduces the fitness of solutions in proportion to the number of solutions that are close together [5]. Cellular structures and adaptive grids are also clustering techniques that aim to uniformly distribute the solutions over the trade-off surface [15,17]. Restricted mating sets the probability of recombining two solutions according to the degree of similarity between these solutions in order to avoid the generation of new solutions that are 'too-similar' to the recombined solutions [5]. Relaxed forms of the dominance relation [2,14] and entropy metrics [9] have also been proposed to improve the ability of multiobjective metaheuristics to achieve a good coverage of the trade-off surface. Also, fuzzy logic has been used to provide different degrees of Pareto optimality within non-dominated sets [8]. Most of the above techniques attempt to 'restrict' the likelihood of generating solutions in 'crowded' regions of the trade-off surface and 'boost' the likelihood of generating solutions in 'under populated' regions. From these techniques, the specification of the search direction by tuning weights is the method that directly attempts to 'push' the current solution(s) towards the desired region of the trade-off surface. The hyperheuristic approach proposed here follows this same strategy, but attempts to do it in a more 'intelligent' way by applying the neighbourhood search heuristic that is more likely to 'push' the solution in the desired direction.

3 Hyperheuristics for Multiobjective Optimisation

It has been shown that the use of various 'simple' neighbourhood heuristics (eg. variable neighbourhood search [10] and cooperative teams of heuristics [16]) can be beneficial when tackling complex combinatorial optimisation problems. Hyperheuristics have been described as strategies designed to control the application of a set of heuristics during the search process [1]. At each time during the search, the selection of the next heuristic to be used is based on the past performance that each of them has exhibited. An important feature of hyperheuristics is that the set of heuristics that are applied during the search can be simple neighbourhood exploration heuristics (similar to [10] and [16]) or more elaborate algorithms such as metaheuristics. Various hyperheuristic approaches have been developed over the past ten years or so (eg. [6,7,11]).

The performance of a given neighbourhood exploration heuristic may depend on the problem domain, the particular instance of the problem domain and the present conditions of the search process. For some multiobjective combinatorial optimisation problems, a set of simple neighbourhood exploration heuristics can be developed. Then, the approach proposed here selects the most appropriate neighbourhood heuristic at certain points during the search in order to 'push' the solution in the desired direction towards the Pareto optimal front. An 'intelligent' way to do this is by 'learning' how well each simple heuristic achieves improvements on each of the objectives of a given solution. Then, having a hyperheuristic that systematically chooses the best strategy to explore the neighbourhood of the current solution(s), can help to maintain a uniformly distributed set of non-dominated solutions. The ‘learning’ mechanism can be implemented using a simple choice function that rewards and/or penalises each simple heuristic according to its performance on the optimisation of each objective [7]. Another method (described later in section 4) to store this *historical* knowledge is to maintain a list of the heuristics and control their application during the search following the principles of tabu search.

Our method measures the performance of each simple neighbourhood exploration heuristic and adapts it according to the knowledge gained during the search and during previous runs of the algorithm. The aim of this adaptation is to approximate the trade-off surface in a more efficient way by using those moves that are more promising according to the current quality of the various objectives and the historical knowledge stored. This is illustrated in figure 1, where a two-objective minimisation problem is considered. The desired trade-off surface and three non-dominated solutions are shown. Solutions *A* and *B* are 'good' with respect to the objective *u* but are 'bad' with respect to the objective *v*. On the other hand, solution *C* is 'good' with respect to objective *v* but it is 'bad' with respect to the objective *u*. The region of the trade-off surface enclosed in the rectangle can be considered to be under populated because no solutions have been found in that area. Table 1 shows how each of eight neighbourhood heuristics could perform with respect to each objective. Then, in order to aim a better coverage of the trade-off surface, the following strategy can be used: maintain solutions *A* and *C* in their current locations and 'push' solution *B* towards the under populated region. This can be achieved by applying heuristics H_2 and H_3 to solutions *A* and *C* and applying heuristic H_5 to solution *B*.

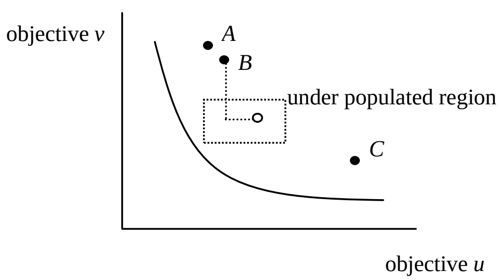

Figure 1: Towards a better coverage of the trade-off surface. Solutions in crowded regions of the trade-off surface (such as solution *B*) are pushed towards the under populated regions (such as the region enclosed by the rectangle).

		improves *v*		**deteriorates *v***	
		small	large	small	large
improves *u*	small	H_2 H_3	-----	-----	H_6
	large	-----	H_1 H_8	-----	H_7
deteriorates *u*	small	-----	H_5	-----	-----
	large	-----	H_4	-----	-----

Table 1: Directed neighbourhood search for a better coverage of the trade-off surface. The adequate heuristics can be applied to push one solution from one region of the trade-off surface to another region (see figure 1).

4 Preliminary Results and Ongoing Research

We have applied the ideas described above to develop hyperheuristic approaches for the two-objective space allocation problem. This problem consists of distributing the available room space among a set of entities (staff, research students, computer rooms, lecture rooms, etc.) in such a way that the misuse of room space (represented by *F1*) and the violation of soft constraints (represented by *F2*) are minimised. Soft constraints are restrictions that limit the ways in which the entities can be allocated to the rooms (eg. entities that should not share a room, entities that should be allocated together, etc.) and that are penalised if violated. See [4] for a more detailed description of this problem. Several neighbourhood exploration heuristics have been designed based on three moves: *relocate*, *swap* and *interchange*. A *relocate* move changes one entity from one room to another. In a *swap* move, the assigned rooms between two entities are swapped. The third move interchanges all the entities between two rooms. There are nine neighbourhood exploration heuristics (three for each neighbourhood structure or move). For example, there are three heuristics based on the *relocate* move. **RelocateRndRnd** selects an allocated entity and a room at random. **RelocateRnd-BestRnd** selects an allocated entity at random, then explores a number of randomly selected rooms evaluating the suitability of each of them to relocate the selected entity. Then, the chosen entity is allocated to the best of the subset of explored rooms. In the **RelocatePnty-BestRnd** heuristic the allocated entities are sorted in non-increasing order of their individual penalties (violation of soft constraints). In each iteration, the allocated entity with the highest penalty is selected and the room in which to relocate this entity is chosen with the same procedure as in the heuristic **RelocateRnd-BestRnd**.

Experiments were carried out in order to assess the performance of each neighbourhood search heuristic with respect to the two objectives. Three test instances: **nottl**, **nott1b** and **trent1** (described in detail and available at http://www.cs.nott.ac.uk/~jds/research/spacedata.html) were used in these experiments. For each test instance, three populations (of size 20) P_U, P_L and P_C were generated. To initialise one solution, an unallocated entity is chosen at random. Then, a number of randomly selected rooms are evaluated and the best of this set is chosen to allocate the entity. The P_U population contained solutions with low values of *F1* and high values of *F2* (i.e. solutions near to the upper part of the trade-off surface). The P_L population contained solutions with high values of *F1* and low values of *F2*. The P_C population contained solutions with moderate values of *F1* and *F2*. Each of the neighbourhood exploration heuristics was applied on its own (in a simple iterative improvement strategy) to each of the solutions in the three populations of the tests instances. It was observed that the improvements that each heuristic achieves in each of the two objectives depend not only on the problem instance but also on the localization of the current solution. For example, for the **nottl** instance, the heuristics based on the swap move achieve large improvements in both objectives for solution in P_L. These heuristics achieve negligible improvements in both objectives for solutions in P_C or P_U. The same group of swap heuristics perform differently on the **trentl** instance. For solutions in P_L, large improvements are achieved in both objectives. For solutions in P_C, moderate improvements are achieved in *F2* and negligible improvements are obtained in *F1*. If the solutions are near to the upper part of the trade-off surface, i.e. in P_U, the improvements produced in *F2* are very small while *F1* is not improved at all or it is even worsened.

A simple hyperheuristic approach was designed to choose the strategy for exploring the neighbourhood of a given solution during the search. The approach first generates a set of initial solutions and then, iteratively improves these solutions by selecting the most appropriate heuristic for neighbourhood exploration in order to aim for a good coverage of the trade-off surface. The results obtained so far in our experiments show that this technique is suitable for achieving a uniformly distributed set of non-dominated solutions. At present, we are carrying out experiments using other multiple-criteria combinatorial optimisation problems such as timetabling and personnel scheduling in order to produce more evidence of this. We are also investigating the application of a more elaborate hyperheuristic approach based on the tabu search metaheuristic. In this tabu search hyperheuristic there is a set of n neighbourhood search heuristics and each of them competes against the others in order to be selected [3]. The competitions rules are inspired from the principles of reinforcement learning [13]. At the beginning of the search each simple heuristic has a score of zero points associated to each of the k objectives in the multiobjective optimisation problem. When a heuristic has been applied, a change Δi may be observed in the value of the i^{th} objective ($i = 1,..,k$) from the previous solution to the new one. If this change is positive, i.e. an improvement in the i^{th} objective was obtained, the corresponding score of the heuristic is increased. If the change is negative reflecting a detriment in the i^{th} objective, the corresponding score is decreased. In addition, a tabu list of simple neighbourhood search heuristics is maintained, which excludes some heuristics from the competition for a certain duration. The purpose of this tabu list of heuristics is to prevent a heuristic that did not perform well from being chosen too soon (even if it has a high rank).

5 Summary and Final Remarks

The problem of obtaining a uniformly distributed set of non-dominated solutions is of great concern in Pareto optimisation. This work proposes the application of hyperheuristics for achieving a good coverage of the trade-off surface in Pareto optimisation. The central idea is to develop a strategy that selects the most promising neighbourhood search heuristic in order to guide the search towards the desired areas of the trade-off surface. This technique has the advantage that it can be applied to single-solution and population based algorithms because no population statistics are required like in some clustering techniques. Experiments have been carried out in the space allocation problem and other timetabling and personnel scheduling problems. The results obtained show that hyperheuristic approaches are capable of obtaining non-dominated sets that represent a good coverage of the trade-off surface.

References

[1] Burke E.K., Kendall G., Newall J., Hart E., Ross P., Schulemburg S., Hyper-heuristics: an Emerging Direction in Modern Search Technology, In: Glover F.W., Kochenberger G.A. (eds.), Handbook of Metaheuristics, Kluwer, 2003.

[2] Burke E.K., Landa Silva J.D., Improving the Performance of Multiobjective Optimisers by Using Relaxed Dominance, Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), Singapore, pp. 203-207, 2002.

[3] Burke E.K., Soubeiga E., Scheduling Nurses Using a Tabu-Search Hyperheuristic, to appear in proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and

Applications (MISTA 2003), Nottingham UK, August 2003.
[4] Burke E.K., Varley D.B., Space Allocation: An Analysis of Higher Education Requirements, The Practice and Theory of Automated Timetabling II: Selected Papers from the 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT 97), LNCS, Springer, Vol. 1408, pp. 20-33, 1998.
[5] Coello Coello C.A., Van Veldhuizen D.A., Lamont G.B., Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer, 2002.
[6] Cowling P., Kendall G., Han L., An Investigation of a Hyperheuristic Genetic Algorithm Applied to a Trainer Scheduling Problem, Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), pp. 1185-1190, 2002.
[7] Cowling P., Kendall G., Soubeiga E., Hyperheuristics: A Robust Optimisation Method Applied to Nurse Scheduling, Proceedings of the VII Parallel Problem Solving From Nature (PPSN VlI), LNCS, Vol. 2439, Springer, pp. 7-11, 2002.
[8] Farina M., Amato P., Fuzzy Optimality and Evolutionary Multiobjective Optimization, Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003), LNCS, Vol. 2632, Springer, pp. 58-72, 2003.
[9] Gunawan S., Farhang A., Azarm 5., Multi-level Multi-objective Genetic Algorithm Using Entropy to Preserve Diversity, Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003), LNCS, Vol. 2632, Springer, pp. 148-161, 2003.
[10] Hansen P., Mladenovic N., Variable Neighbourhood Search: Principles and Applications, European Journal of Operational Research, Vol. 130, No. 3, pp. 449-467, 2001.
[11] Hart E., Ross P., A Heuristic Combination Method for Solving Job-shop Scheduling Problems, Proceedings of the V Parallel Problem Solving From Nature (PPSN V), LNCS, Vol. 1498, Springer, pp. 845-854, 1998.
[12] Ishibuchi H., Yoshida T., Murata T., Selection of Initial Solutions for Local Search in Multiobjective Genetic Local Search, Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), Hawaii USA, pp. 950-955, 2002.
[13] Kaelbling L.P., Littman M.L., Moore A.W., Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research, Vol. 4, pp. 237-285, 1996.
[14] Laumams M., Thiele L., Deb K., Zitzler E., Combining Convergence and Diversity in Evolutionary Multiobjective Optimization. Evolutionary Computation, Vol. 10, No. 3, pp. 263-282, 2002.
[15] Murata T., Ishibuchi H., Gen M., Specification of Genetic Search Directions in Cellular Multi-objective Genetic Algorithms, Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001), LNCS, Vol. 1993, Springer, pp. 82-95, 2001.
[16] Salman F.S., Kalagnaman JR., Murthy S., Davenport A., Cooperative Strategies for Solving Bicriteria Sparse Multiple Knapsack Problem, Journal of Heuristics, Vol. 8, pp. 215-239, 2002.
[17] Toscano Pulido G.., Coello Coello C.A., The Micro Genetic Algorithm 2: Towards Online Adaptation in Evolutionary Multiobjective Optimization, Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003), LNCS, Vol. 2632, Springer, pp. 252-266, 2003.