Combining Hybrid Metaheuristics and Populations for the Multiobjective Optimisation of Space Allocation Problems

E.K. Burke
ekb@cs.nott.ac.uk
Dept. of Computer Science
University of Nottingham
Nottingham, NG8 1BB, UK

P. Cowling
pic@cs.nott.ac.uk
Dept. of Computer Science
University of Nottingham

J.D. Landa Silva
jds@cs.nott.ac.uk
Dept. of Computer Science
University of Nottingham

S. Petrovic
sxp@cs.nott.ac.uk
Dept. of Computer Science
University of Nottingham

Abstract

Some recent successful techniques to solve multiobjective optimisation problems are based on variants of evolutionary algorithms and use recombination and self-adaptation to evolve the population. We present an approach that incorporates a population of solutions into a hybrid metaheuristic with no recombination. The population is evolved using self-adaptation, a mutation operator and an information-sharing mechanism. Since the main component in our approach is a simulated annealing algorithm, the cooling schedule for the whole population becomes critical. A common cooling schedule for the whole population is determined based on an evolutionary process. Results are presented using a real-world multiobjective combinatorial optimisation problem, namely space allocation with two conflicting criteria. These results suggest that this approach is a suitable alternative not only for combinatorial multiobjective optimisation problems, but also for obtaining a population of locally optima solutions in single-objective optimisation problems.

1 INTRODUCTION

In this paper, we present a population-based technique for multiobjective combinatorial optimisation problems. The approach is based in a hybrid metaheuristic that uses hill-climbing, simulated annealing, tabu lists and a mutation operator. This technique has been extended to solve multiobjective combinatorial optimisation problems, by incorporating a population of solutions. The population of feasible solutions is initialised and improved using hill-climbing. Self-adaptation is achieved using simulated annealing with a common cooling schedule for all individuals. The mutation operator is used to disrupt a solution so that a different area of the search space can be explored. Cooperation between individuals is induced using lists of tabu and attractive moves.

The proposed approach has the ability to provide a diverse set of high quality solutions for a combinatorial optimisation problem. In our experiments, we consider both an aggregating function and the dominance relation to evaluate the solution fitness. Although we present and discuss the results of applying this technique to a specific multiobjective combinatorial optimisation problem, we believe it can also be used for other multiobjective optimisation problems. This technique can also be used to find a diverse set of local optima in single-objective optimisation problems.

There are a considerable number of papers about multiobjective optimisation including: theoretical studies, surveys, experimental comparative studies, test problem sets, analysis of future trends, and others. In the next section, we provide a brief summary of the recent work in this area. Our intention with this is to clarify the main concepts, identify the most important aspects of interest in multiobjective optimisation and define the scope of our contribution. Our approach is described in section 3. In section 4, the experiments and results are presented and discussed. Conclusions are established in section 5.

2 MULTIOBJECTIVE OPTIMISATION

2.1 EVOLUTIONARY APPROACHES

It has been stated by Hertz & Klober (2000) that there is not a clear and widely accepted definition of an evolutionary algorithm. However, they suggest that in a strict sense, an evolutionary algorithm handles a population of solutions, evolves this population by means of cooperation (recombination) and self-adaptation (mutation) and uses a coded representation of the solutions. They introduce a framework to describe evolutionary algorithms. Some guidelines to classify evolutionary algorithms are proposed by Calegari et.al. (1999).

Why begin with evolutionary algorithms? Some of the recent successful techniques proposed to solve multiobjective optimisation problems are of this type. For a comprehensive overview refer to Fonseca & Fleming (1995), Van Valdhuizen & Lamont (2000b) and Zitzler (1999). Additional surveys and comparative studies are presented by Coello Coello (1999), Coello Coello (2001), Horn (1997), Van Valdhuizen & Lamont (2000) and Zydallis et.al. (2001). Recent advances on evolutionary multiobjective optimisation are reported in Zitzler et.al. (2001).

2.2 ALTERNATIVE APPROACHES

Many authors use non-evolutionary approaches to solve multiobjective optimisation problems. An overview of some of these techniques is available in Miettinen (2001). Even some methods that have been classified as evolutionary by their creators may not be labelled as such if we use a strict definition of evolutionary algorithms (some of them do not use recombination). It is not our intention to argue whether an approach should be called evolutionary or not. In this section, we refer to some algorithms that use self-adaptation as the main tool to evolve the population rather than recombination.

In recent years, the interest in these different approaches has become greater. For example, Menczer et.al. (2000) proposed the Evolutionary Local Search Algorithm in which they use agents with a local selection scheme. The multiobjective A* is an approach for multiobjective combinatorial optimisation problems proposed by Stewart & White (1991) based on the A* algorithm. This multiobjective A* technique was extended by Dasgupta et.al. (1999) for those combinatorial problems that can be represented as search trees. The multiobjective simulated annealing algorithm by Tuyttens et.al. (2000) is an adaptation of simulated annealing to multicriteria problems. Their algorithm uses a vector of weights to induce a privileged direction in the search. An extension of the multiobjective simulated annealing algorithm was presented by Teghem et.al. (2000) for large-scale problems. In their technique, interaction with the decision-maker is required to adjust preference settings during the search. An approach called the Pareto Archived Evolution was recently introduced by Knowles & Corne (2000). This technique basically uses local search and makes comparisons to select among mutated individuals keeping an archive of previously seen solutions.

2.3 FUNDAMENTALS

A multiobjective optimisation problem can be thought of as a number of decision variables, a set of objectives and a number of constraints. In many real-world optimisation problems, the various objectives are conflicting and often incommensurable. In some cases, it is desirable to obtain those solutions that represent a tradeoff between the different objectives. These solutions are known as non-inferior or non-dominated solutions. To be more precise we can define d*ominance* as follows (where we aim to maximise each objective):

Let $V = (v_1, v_2, ..., v_k)$ and $U = (u_1, u_2, ..., u_k)$ be two distinct k-dimensional vectors of objective function values for a k-objective problem where we aim to maximise each objective, then

- V strictly dominates U if $v_i > u_i$, for $i = 1,2,..,k$
- V loosely dominates U if $v_i \geq u_i$, for $i = 1,2,..,k$
- V and U are incomparable if neither V dominates U nor U dominates V

For minimisation objectives these definitions are altered in the obvious way. Given a set S of feasible solutions, a solution is said to be non-inferior or non-dominated if its k-dimensional vector containing the values for the k-objectives is non-dominated by any other vector in S. Then the Pareto optimal front is the set of non-dominated solutions in the whole solution space. Refer to Van Valdhuizen & Lamont (2000b) for a more detailed description of Pareto concepts.

Three goals can be used to evaluate the effectiveness of a given multiobjective optimisation technique that attempts to find the Pareto optimal front (Zitzler,1999):

- distance from the resulting non-dominated set of solutions to the true Pareto optimal front should be minimised
- a good distribution of the set of solutions found is desirable, i.e. in any given non-dominated set the solutions should not be clustered together
- the spread of the non-dominated set should be as large as possible, i.e. the set of solutions should cover as much of the non-dominated front as possible

Three main problems arise when using and designing a multiobjective optimisation technique: how to combine search and decision-making, how to perform fitness evaluation to guide the search and how to maintain a diverse population (Horn,1997; Zitzler,1999). For the first problem, there are three alternative ways to combine the searching and decision-making processes:

1. to perform decision-making before the search
2. to search before decision-making
3. to perform decision-making during the search

In the first case, the relative importance of each criterion is established before the searching process. In the second alternative, after the searching process is completed a set of possible solutions is proposed and the decision-making process takes place. Finally, the last method refers to interactively modifying user preferences while the searching process is being performed.

For the problem of fitness evaluation, many different solutions have been proposed such as: aggregating functions, Pareto-based selection or switching objectives (Coello Coello,1999). Aggregating functions combine multiple objectives into a single-objective, for example using a weighted sum of objectives. Pareto-based selection uses the concept of dominance to find the set of those solutions that represent a tradeoff among the multiple objectives. Switching objectives refers to the optimisation of one objective at a time while imposing constraints on the others.

The problem of population diversity refers to the possibility of premature convergence of the population. A review of some alternatives to avoid this problem and accomplish the goal of maintaining a diverse population can be found in Zitzler (1999).

In this paper, we analyse the performance of our approach upon a bi-criteria combinatorial optimisation problem: a real instance of the space allocation problem (Burke et.al.,2001). We show how both objectives are conflicting so that we must find a tradeoff between them. The algorithm is tested using different forms of fitness evaluation to guide the search: an aggregating function and dominance-ranking. From the results we make some observations about the set of solutions obtained using each form of fitness evaluation. Our approach allows us to produce a set of good solutions at the expense of more computation time, or a single high quality solution in a shorter time.

3 THE ALGORITHM

3.1 GENERAL DESCRIPTION

There are three main components in the method: selection of parameters, heuristic hill-climbing initialisation and the hybrid simulated annealing algorithm (Fig. 1). The local search heuristic is problem dependent. This heuristic is part of both components, the initialisation (hill-climbing) and the evolution phase (simulated annealing). Therefore, the selection of parameters for the local search heuristic is also dependent on the specific problem. More details about the local search heuristic and corresponding parameters for our specific problem domain are in Burke et.al. (2000).

```
1. Heuristic_Parameters_Selection
2. For Individual = 1 To Population_Size Do
    2.1. Construct an initial Current_Solution
    2.2. Heuristic_Hill_Climbing on Individual
    2.3. Add Individual to Current_Population
3. Best_Population = Current_Population
4. While Termination_Criterion Not Satisfied Do
    4.1  For Individual = 1 To Population_Size Do
        4.1.1 Apply Hybrid_Simulated_Annealing
    4.2 Adjust Global Parameters
    4.3 Update Best_Population
```

Fig. 1. The Hybrid Population-Based Metaheuristic.

In the initialisation phase (step 2, Fig. 1) feasible individuals are constructed and improved using heuristic hill-climbing. The population is evolved using our hybrid simulated annealing algorithm until a termination criterion is satisfied. After each iteration, the best solution achieved by each individual is updated in the best population. The final best population is composed of the best solutions achieved by each individual in the population during the evolution process. The feasibility of the individuals and the neighbourhood exploration are defined by the particular problem. For example in space allocation (the test problem is described in section 4.1) the neighbourhood is explored by making a `move`. A `move` is any change in the allocation of an object and by exploring `moves` new solutions can be constructed (Burke et.al.,2001; Burke et.al.,2000). Intensification is provided by our hybrid simulated annealing heuristic when the temperature is zero or near to zero. Diversification is achieved in the initialisation phase and since no direct recombination is used, premature convergence of the population is unlikely to occur.

Given the features of multiobjective optimisation techniques (section 2.1), we can define our approach as a hybrid population-based metaheuristic. The hybridisation of metaheuristics has been studied for some time. For example Pirlot (1996) presents a brief tutorial on simulated annealing, tabu search and genetic algorithms and a survey on hybridisations of these techniques for solving single objective search problems. Other approaches using hybridisations can be found in Aarts et.al. (1997), Osman & Kelly (1996), Rayward-Smith et.al. (1996) and Reeves (1995). Since in our approach the local search heuristic and the fitness evaluation technique can be modified, it is possible to adapat this technique to other optimisation problems.

3.2 THE HYBRID SIMULATED ANNEALING COMPONENT

After presenting an overview of the algorithm, we use the pseudocode in Fig. 2 to describe how the hybrid simulated annealing component (step 4.1.1. in Fig. 1) operates upon the population. Each individual in the population has three attributes: `BestIndividual`, `ReHeatCounter` and `NoImprovesCounter`. `BestIndividual` is the memorised best instance of each individual found during the execution of the algorithm. `ReHeatCounter` and `NoImprovesCounter` are both initialised to zero.

```
If First Individual in the Population
    If Temperature is zero                                              (1)
        If GlobalReHeatCounter ≥ ReHeatInterval
            Temperature = InitialTemperature
            GlobalReHeatCounter  = 0
            For Individual = 1 To Population_Size Do
                Individual.ReHeatCounter  = 0
    Else
        If GlobalIntervalCounter  < Interval
            GlobalIntervalCounter = GlobalIntervalCounter + 1
        Else
            Temperature = Temperature – Decrement
            GlobalIntervalCounter = 1
SearchNewIndividual
If NewIndividual better than Individual                                 (2)
    If  NewIndividual better than Individual.BestIndividual
        Individual.BestIndividual = NewIndividual
        Individual.ReHeatCounter = 0
    Else
        Increment Individual.NoImprovesCounter
        If Temperature is zero
            Increment Individual.ReHeatCounter
Else
        Increment Individual.NoImprovesCounter
        If Temperature is zero
            AcceptProbability  = 0
            Increment Individual.ReheatCounter
        Else  AcceptProbability = e^-( Δ / Temperature )
        If AcceptProbability > Random   Accept  NewIndividual
        Else   Reject NewIndividual
GlobalReHeatCounter = max of all ReHeatCounters                         (3)
If SingleHighQualitySolution
    GlobalImprovesCounter = max of all NoImprovesCounter
If MultipleGoodSolutions
    GlobalImprovesCounter = min of all NoImprovesCounter
```

Fig. 2 In the Hybrid Simulated Annealing Phase, the Lists of Tabu and Attractive Moves and the Mutation Operator are Incorporated in `SearchNewIndividual`.

There are four global parameters for the population in our hybrid simulated annealing algorithm: `Temperature`, `GlobalImprovesCounter`, `GlobalIntervalCounter` and `GlobalReHeatCounter`. The first three are initialised to zero, while `Temperature` is initialised to a value of `InitialTemperature` (values for our experiments are given in section 4.3).

3.3 INFORMATION-SHARING AND THE MUTATION OPERATOR

The neighbourhood of each individual is explored by using a local search heuristic (`SearchNewIndividual`). During this exploration to produce a `NewIndividual` from the current one, lists of `moves` are maintained and shared among all individuals. These lists contain tabu and attractive `moves` so that information-sharing within the population is encouraged. In our heuristic for the space

allocation problem, one `move` is selected and carried out for the current individual. In Fig. 2 we can observe that once the `NewIndividual` is evaluated, it is compared with the current individual and with the best solution achieved by the individual so far (`BestIndividual`). Those `moves` that produced a `NewIndividual` which outperforms the `BestIndividual` are inserted in the list of attractive `moves`. The `moves` that generated a `NewIndividual` that worsens the current solution are inserted in the list of tabu `moves`. During the neighbourhood exploration, if a `move` is selected and it is in the list of tabu `moves`, then we reject the `move` and a new search in the neighbourhood is started. The list of attractive `moves` is used to select a `move` when it has not been possible to find a `move` that leads to a feasible `NewIndividual`. The size of both lists is determined by the size of the problem (Burke et.al.,2001). The last tabu or attractive `move` replaces the `move` that has been in the corresponding list for the longest number of iterations. If while exploring the neighbourhood, no feasible `NewIndividual` is found, the solution is modified through a mutation operator. The mutation operator modifies the current individual so that new `moves` are explored and a feasible `NewIndividual` can be found. In our problem, the modification consists of removing some allocated objects from their assigned room. Note that the local search heuristic can be modified according to the specific optimisation problem and hence, specific strategies to implement information-sharing and mutation can also be designed.

3.4 GLOBAL TEMPERATURE - DISTRIBUTED COOLING SCHEDULE

This population-based simulated annealing algorithm controls the cooling schedule by sensing the performance of each individual in the population. Note in code (1) of Fig. 2, that this control over the global `Temperature` is done only at the beginning of each iteration, i.e. before the algorithm is executed for all the individuals in the population. `Interval` is the number of iterations after which there is a decrement in the `Temperature` parameter. `ReHeatInterval` is the number of iterations after which if there is no improvement in the current solution, the `Temperature` is raised again to the value of `InitialTemperature`. These parameters are set at the beginning of the whole process and are defined by the size and type of problem (Burke et.al.,2001). If the process is being cooled, then the common `Temperature` is decremented constantly after `Interval` number of iterations. When the `Temperature` is equal to zero, the value for `GlobalReHeatCounter` indicates whether or not to reheat the process.

In code (2) of Fig. 2, if after searching and finding a `NewIndividual`, this new solution improves the current one, then a second comparison is made with the best recorded performance so far for that individual. If this `NewIndividual` overcomes the best recorded performance for the present individual then the second population (the population containing the best solutions achieved by each individual) is updated. When the `NewIndividual` is not better than the current one, the current `Temperature` determines its acceptance. If this global `Temperature` is zero the solution is rejected since only improvements are accepted. But if the global `Temperature` is not zero, the new solution is accepted only if a calculated probability is greater than a random number chosen from the uniform distribution on the interval [0,1]. The probability is defined by $-e^{-(\Delta / Temperature)}$ where Δ is the fitness variation from the current to the new solution.

At the end of each iteration in code (3) of Fig. 2, `GlobalReHeatCounter` is set to the highest `ReHeatCounter` of all individuals. This means that as soon as one of the individuals cannot be improved for a `ReHeatInterval` iterations, the common `Temperature` is raised again. The effect of this strategy is that while one (maybe more) individual is stuck in the improvement process, the others are not yet. Then switching to the random phase of the simulated annealing algorithm (`Temperature` not zero) the exploration of the search space can continue. Finally, we can see that there are two ways of setting `GlobalImprovesCounter`. This value is used as a termination criterion for the whole process. If the maximum of `NoImprovesCounter` over all individuals is used, the individual for which the best performance has not been improved for the longest number of iterations, determines the termination of the process. This quickly yields a varied population of solutions, although it might be possible to improve some of the solutions. On the other hand, if the minimum value of `NoImprovesCounter` is used, then all the individuals have been improved considerably. In this case a set of more uniform (in terms of fitness) solutions is obtained at the expense of longer computation time.

To summarise, this hybrid population-based simulated annealing algorithm has a global `Temperature` for the whole population, but the control of this parameter is distributed over all individuals in the population. Additionally, deciding which individual determines the termination criterion, a single high quality solution or a group of good solutions can be achieved in less or more computation time respectively.

4 EXPERIMENTS AND RESULTS

4.1 THE SPACE ALLOCATION PROBLEM

Results of our experiments using a real instance of the space allocation problem are presented in this section. This is a complex real-world combinatorial optimisation problem. It consists of the allocation of a number of objects with different sizes to a number of areas of space, subject to certain constraints. More formally, the space allocation problem can be described as follows:

Given a set $X = \{ x_1, x_2, \ldots, x_n \}$ of n objects and a set of m available areas of space $Y = \{ y_1, y_2, \ldots, y_m \}$, find the optimal allocation of the n objects into the m available areas of space, given by $h : X \rightarrow Y$ where,

$h(x_i) = y_j$, if object x_i has been allocated to area y_j

$h(x_i) = 0$, if object x_i has not been allocated

to minimise the k functions $f_1, f_2, f_3, \ldots, f_k$ subject to a number of constraints.

The instance of the space allocation problem that we have considered is the distribution of people to rooms. Given a number of rooms with different sizes, the problem is to distribute all people in the set of rooms. Constraints are

set on the people and the size of rooms that they should be allocated to. In addition to specific space requirements, each person may have other requirements. These other requirements are expressed as constraints. Examples of constraints are: people that need to be allocated in certain rooms, people that need to be located in adjacent rooms, people that should not share their room with another person, people that need to be located far away from certain areas, etc. There are two objectives in this problem. The first objective is to minimise the misuse of the space so that people are allocated to areas of space that are neither too large nor too small. The second objective is to minimise the penalty for the violation of soft constraints in the problem. Given the examples of constraints mentioned before, a hard constraint is a requirement that must be satisfied so that the solution can be considered feasible. Soft constraints are those requirements that need not be satisfied but in that case should be penalised. A feasible solution in our space allocation problem must have all resources allocated and all hard constraints satisfied.

In our test problem there are 55 human resources with different requirements of space (test data is available in Burke et.al.,2001). There are 55 rooms with different sizes, 15 hard constraints and 37 soft constraints. The types of constraints are for example: people that must be allocated in certain areas or in certain rooms; people that must be together, adjacent or close to each other; rooms that must not be empty or overused; and other similar constraints.

4.2 THE BI-CRITERIA NATURE OF THE TEST PROBLEM

The objectives "minimise the misuse of the space" and "minimise the violation of soft constraints" are often conflicting and incommensurable. For example, to achieve a better utilisation of the space, some constraints may need to be violated and vice versa. Space utilisation is expressed in terms of area while the constraints are expressed in terms of satisfaction or non-satisfaction and penalties. This means that it is complicated to evaluate and compare both objectives by combining them into a single objective. Let f_1 and f_2 be two functions that represent objectives and are defined as follows:

- f_1 is the function that represents the misuse of space which we wish to minimise
- f_2 is the function that represents the violation of soft constraints which we wish to minimise

In order to show that both objectives in this problem are conflicting, we analysed the behaviour of each objective while the other was subject to improvement. We executed 10 runs for each case, and in all runs the behaviour is similar. For clarity, we show three of these runs. In Fig. 3 we observe that as a result of improvement on the space utilisation, the satisfaction of constraints is worsened. In the second one, Fig. 4, we notice that the effect is also present when the satisfaction of soft constraints is being improved. For these graphs, the numbers in the axis for f_1 are a measure of the amount of space misused, i.e. the sum of total space wasted and overused in the allocation. The numbers in the axis for f_2 are a measure of the total penalty due to the violation of soft constraints in the allocation.

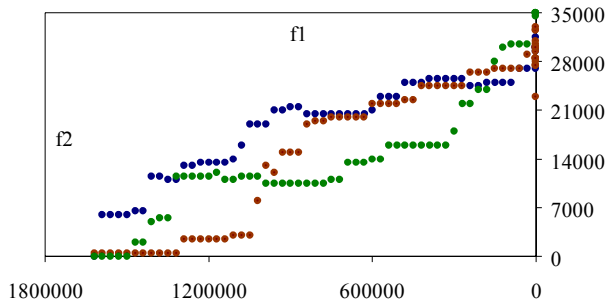

Fig. 3. Minimising Space Misuse.

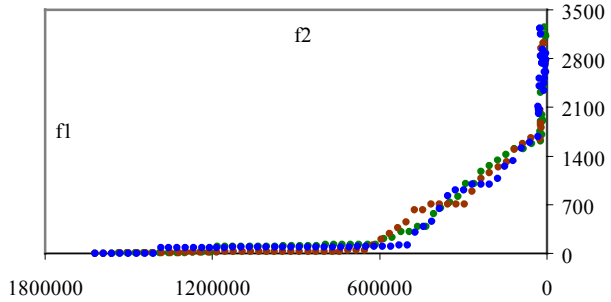

Fig. 4. Minimising Soft Constraints Violation.

Minimising the misuse of space affects the second objective but minimising the violation of soft constraints seems to have a lighter effect on the first objective. If we apply an approach in which the switching of objectives is used (see section 2.3), then we can obtain a deeper analysis of the correlation between the objectives in a multiobjective optimisation problem.

4.3 MINIMISING BOTH OBJECTIVES SIMULTANEOUSLY

Now we present the results of applying our approach to solve the test problem described in section 4.1. We used a population of 10 individuals. In reference to the pseudocode presented in Fig. 2, we used these parameters in our experiments: `InitialTemperature` equal to 1000, `Decrement` set to 200, `ReHeatInterval` equal to 550 and `Interval` set to 55. Two ways of assigning fitness to the solutions were examined: aggregating functions and dominance-ranking (see section 2.3). For the first method we add the values of both f_1 and f_2 into one scalar value that represents the solution fitness. In the second method, the solution fitness is represented by a 2-dimensional vector that contains the values for f_1 and f_2.

For example, suppose we have three solutions A, B and C with the objective values shown in Table 1. If we use the aggregated function approach then the sum given by f_1+f_2 is an evaluation of the total penalty for each solution. In the other case the solutions are ranked using the dominance concept defined in section 2.3. For the minimisation example shown in Table 1, if we use the aggregated function then solutions B and C are preferred over solution A (98<135). But with dominance-ranking, only solution B is preferred over solution A ([43,55] strictly dominates [45,90]), while C is incomparable with respect to A and B.

Table 1. Comparing Two Ways of Evaluating Objectives: Aggregating Function and Dominance-Ranking.

	A	B	C
f_1	45	43	5
f_2	90	55	93
$f_1 + f_2$	135	98	98
dominance	[45,90]	[43,55]	[5,93]

The aggregating function used in our experiments is given by equation (1) below.

$$f_1 + f_2 = \sum_{i=1}^{m} [WP(A_i) + OP(A_i)] + \sum_{i=1}^{n} SCP(O_i) \qquad (1)$$

Here, n is the number of objects (resources), m is the number or areas of space (rooms), WP is the penalty for area A_i if there is space wastage, OP is the penalty for area A_i if there is space overuse, SCP is the penalty for violating a soft constraint for object O_i.

All the data for the test problem used here is available in Burke et.al. (2001), including the penalty parameters. In our experiments, we executed 200 runs using the aggregating function and dominance-ranking for fitness evaluation. We also used both strategies explained in section 3.4: single high quality solution and multiple good solutions and two termination criteria: no improvement after a maximum number of generations (1700 generations) and a fixed execution time (1200 seconds of CPU time). Table 2 below shows the distribution of these 200 runs in our experiments.

Table 2. Distribution of Runs. 100 Runs Using Single High Quality Solution Strategy (SS) and 100 Runs Using Multiple Good Solutions Strategy (MS).

	Aggregating Function		Dominance Ranking	
	SS	MS	SS	MS
No Improvement	25	25	25	25
Execution Time	25	25	25	25

4.4 PERFORMANCE USING THE AGGREGATING FUNCTION

In Fig. 5 below, we show the best populations obtained in each set of 25 runs when using the aggregating function to combine the two objectives. We can compare the populations obtained with the single high quality solution strategy (SS) and with the multiple good solutions strategy (MS). The comparison is made using both termination criteria mentioned in the last section. In each population, the best and worst solutions are marked with a solid and a blank bar respectively.

The first observation we make is the effect of the control over the temperature for the hybrid simulated annealing algorithm as explained in section 3.4, and the termination criterion. The decision on whether to search for one good solution or a population of them produces quite different results. The single high quality solution strategy (SS) consumes less computation time and the rest of the population tend to be of poor quality. If we want a set of good solutions then we require a population where the sum of the objective values tends to be uniform among the individuals, this is produced by the multiple good solutions strategy (MS). The drawback is the increase in computation time. Note that the effect of deciding between the single solution and the multiple solution strategies holds over a fixed execution time. When both strategies are executed for a similar time, a much better solution is achieved by the single solution strategy, at the expense of the rest of the population being relatively poor. In this case, the population produced by the multiple solution strategy is uniform in terms of the objective values.

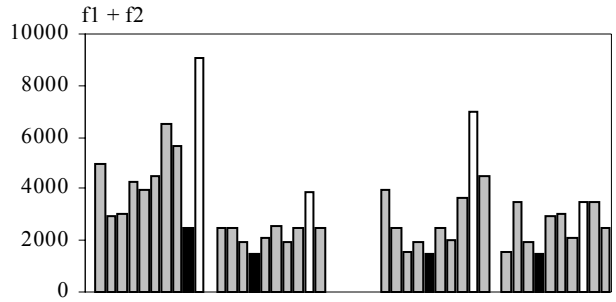

Fig. 5. Populations Using the Aggregating Function.

In Table 3 we show the statistics for the best and worst populations (best and worst runs) obtained in each set of 25 runs when using the aggregating function. The values shown are: best individual, worst individual, the average for the whole population and the total CPU time in seconds. The small difference between the best and worst runs with each strategy suggests that the algorithm is reliable. However, further experiments are required to provide more evidence of this.

Table 3. Best and Worst Populations Obtained Using the Aggregated Function to Combine Both Objectives.

Best and Worst Runs Using Aggregating Function				
Population Statistics	Single Solution Runs		Multiple Solutions Runs	
	Best	Worst	Best	Worst
Average	4744.24	6807.94	2393.54	3500.13
Best	2495.44	4604.55	1495.44	1976.33
Worst	9082.24	9123.66	3905.44	6405.44
CPU Time	252 secs	149 secs	1573 secs	1174 secs

4.5 PERFORMANCE USING DOMINANCE-RANKING

The best populations obtained in each set of 25 runs when using dominance-ranking to evaluate the objectives, are shown in Fig. 6. Again, we can compare the populations produced with the single high quality solution strategy (SS) and with the multiple good solutions strategy (MM). The comparison is made using both termination criteria: no improvement for a maximum of 1700 generations and a fixed CPU execution time equal to 1200 seconds. Note that each population in Fig. 6 is a non-dominated set and no solution dominates the rest of the population. These non-dominated sets of solutions represent a tradeoff between both objectives, the minimisation of space misuse (f_1) and the minimisation of soft constraints violation (f_2).

We can observe that the two sets of non-dominated solutions produced by the strategy single high quality solution (SS) contain a very good solution, but the rest of the population has many other solutions with low quality. The non-dominated sets obtained with the multiple good solutions strategy (MS) contain more solutions with a good quality (but certainly no identical solutions). While the solutions produced by the multiple solutions strategy are closer to each other, the solutions produced by the single solution strategy have a bigger distance between them. Again, even with identical execution time, both strategies maintain the behaviour described here.

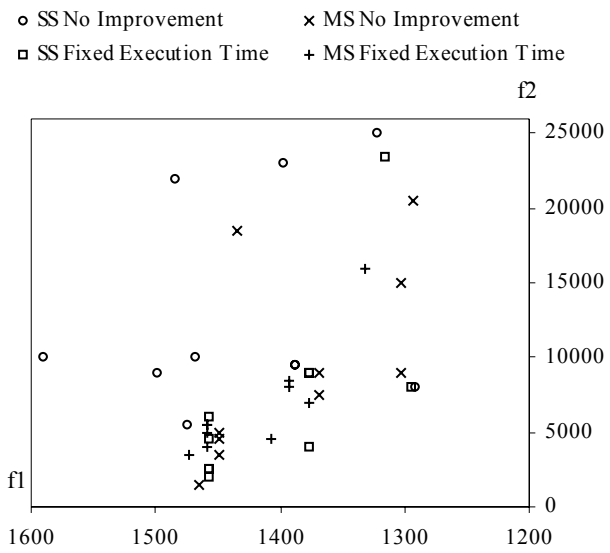

Fig. 6. Populations Using Dominance-Ranking.

The final set of solutions is the result of the best performance achieved by each individual in the population independently from rest of the population. Then, there is no guarantee that the final set of solutions will always be non-dominated. It may occur that one or more individuals dominate the rest of the population. But as shown here, the resulting set is diverse and it represents a tradeoff between the objectives. To obtain a non-dominated front (not the Pareto-optimal front) we can select those solutions in the final population that are non-dominated.

In Table 4 we present the statistics for the best and worst populations (best and worst runs) obtained in each set of 25 runs when using dominance-ranking. An interesting aspect is that the computation time is improved because it is not necessary to calculate the aggregating function. Again, the sets of solutions obtained with both strategies reveal the effect of the control over the common temperature and the termination criterion discussed in section 3.4. As we said before, there is no solution dominating the rest of the population. Then the best individual in Table 4 refers to the solution in the population than dominates most of the other individuals. The worst individual refers to the solution that is dominated by most of the individuals in the population. The populations produced with the single solution strategy contain a very good solution, but the overall quality of the population is low. With the multiple solution strategy, the overall quality is better at the expense of longer computation time.

Table 4. Cost Vectors for the Set of Solutions Obtained Using the Concept of Dominance for Fitness Evaluation.

Best and Worst Runs Using Dominance-Ranking				
Population Statistics	Single Solution Runs		Multiple Solutions Runs	
	Best	Worst	Best	Worst
Best	[1474,5500]	[1479,13500]	[1464,1500]	[1372,7500]
Worst	[1322,25000]	[1489,32000]	[1292,20500]	[1225,29000]
CPU Time	121 secs	118 secs	637 secs	297 secs

These preliminary experiments and results suggest that our population-based metaheuristic effectively produces good solutions for space allocation problems. There is of course much more work to do in order to establish our approach as a competitive multiobjective optimisation technique. Nevertheless, this insight into the application to this type of problem encourages us to continue the investigation. We believe that the combination of metaheuristics, heuristic initialisation techniques and populations may produce good results in the area of multiobjective optimisation for other real-world problems.

5 CONCLUSIONS

We have described a population-based approach for multiobjective optimisation resulting from a combination of metaheuristics. Our approach uses self-adaptation, information-sharing between individuals and mutation as a basis to develop the population. It also permits us to obtain a single high quality solution or a set of good solutions by deciding the strategy to control the common cooling schedule and termination criterion.

Our experiments show the ability of our technique to maintain a diverse population. We believe this is because instead of recombination, we use self-adaptation (including mutation) to evolve the individuals and share information concerning the current state of the solution process between individuals. The experiments and results presented here suggest that this technique may be an effective approach to tackle combinatorial multiobjective optimisation problems. It could also be used on single-objective optimisation problems where it is necessary to produce several good solutions simultaneously. This method may have potential use for a wider range of optimisation problems.

More work is necessary to validate the effectiveness of our approach. For example, we will test our approach using a range of test problems like those proposed by Deb (1999), Zitzler (1999) and Knowles & Corne (2000). It is important to carry out a theoretical analysis of the role that the common temperature has in this algorithm and to investigate the effect of parameters such as population size. One interesting future direction is the evaluation of individuals using different strategies. For example we could evaluate some individuals in the population using dominance, other individuals using an aggregated function and so on.

Acknowledgments

J.D. Landa Silva acknowledges support from Universidad Autónoma de Chihuahua and PROMEP in Mexico.

References

E.H.L Aarts, J.H.M. Korst, P.J.M. Van Laarhoven, (1997), Simulated Annealing in Aarts E., Lenstra J.K. (Eds.), Local Search in Combinatorial Optimization, Wiley.

E.K. Burke, P. Cowling, J.D. Landa Silva, (2001), The Space Allocation Problem, [Online], Available at web site:http://www.asap.cs.nott.ac.uk/ASAP/space/spacealloc ation.html.

E.K. Burke, P. Cowling, J.D. Landa Silva, B. McCollum, (2000), Three Methods to Automate the Space Allocation Process in UK Universities, Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling, PATAT 2000, Konstanz, Germany, pp.374-393.

P. Calegari, G. Coray, A. Hertz, D. Klober, P. Kuonen, (1999), A Taxonomy of Evolutionary Algorithms in Combinatorial Optimization, Journal of Heuristics, Vol. 5, No. 2, pp.145-158.

C.A. Coello Coello, (1999), A Comprehensive Survey of Evolutionary–Based Multiobjective Optimization Techniques, Knowledge and Information Systems, Vol. 1, No. 3, pp.269-308.

C.A. Coello Coello, (2001), A Short Tutorial on Evolutionary Multiobjective Optimization, Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization, EMO 2001, Lecture Notes in Computer Science Vol. 1993, pp.21-40, Springer, Zurich, Switzerland.

P. Dasgupta, P.P. Chakrabarti, S.C. DeSarkar, (1999), Multiobjective Heuristic Search: An introduction to Intelligent Search Methods for Multicriteria Optimization, Vieweg.

K. Deb, (1999), Multi–Objective Genetic Algorithms : Problem Difficulties and Construction of Tests Problems, Evolutionary Computation, Vol. 7, No. 3, pp.205-230.

C.M. Fonseca, P.J. Fleming, (1995), An Overview of Evolutionary Algorithms in Multiobjective Optimization, Evolutionary Computation, Vol. 3, No. 1, pp.1-16.

A. Hertz, D. Klober, (2000), A Framework for the Description of Evolutionary Algorithms, European Journal of Operational Research, Vol. 126, No.1, pp.1-12.

J. Horn, (1997), Multicriteria Decision Making and Evolutionary Computation in T. Bäck, D.B. Fogel, Z. Michalewicz (Eds.), Handbook of Evolutionary Computation, Institute of Physics Publishing, Bristol, UK.

J. Knowles, D.C. Corne, (2000), Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy, Evolutionary Computation, Vol. 8, No. 2, pp.149-172.

F. Menczer, M. Degeratu, W.N. Street, (2000), Efficient and Scalable Pareto Optimization by Evolutionary Local Selection Algorithms, Evolutionary Computation, Vol. 8, No. 2, pp.223-247.

K Miettinen, (2001), Some Methods for Nonlinear Multi-Objective Optimization, Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization, EMO 2001, Lecture Notes in Computer Science Vol. 1993, pp.1-20, Springer, Zurich, Switzerland.

I.H. Osman, J.P. Kelly (Eds.), (1996), Meta–Heuristics: Theory & Applications, Kluwer Academic Publishers.

M. Pirlot, (1996), General Local Search Methods, European Journal of Operational Research, Vol. 92, No. 3, pp.493-511.

V.J. Rayward–Smith, I.H. Osman, C.R. Reeves, G.D. Smith (Eds.), (1996), Modern Heuristic Search Methods, Wiley.

C.R. Reeves (Ed.), (1995), Modern Heuristic Techniques for Combinatorial Problems, McGraw–Hill, UK.

B.S. Stewart, C.C. White, (1991), Multiobjective A*, Journal of the ACM, Vol. 38, No. 4, pp.775-814.

J. Teghem, D. Tuyttens, E.L. Ulungu, (2000), An Interactive Heuristic Method, for Multi–Objective Combinatorial Optimization, Computers & Operations Research, Vol. 27, No. 7–8, pp.621-634.

D. Tuyttens, J. Teghem, P.H. Fortemps, K. Van Nieuwenhuyze, (2000), Performance of the MOSA Method for the Bicriteria Assignments Problem, Journal of Heuristics, Vol. 6, No. 3, pp. 295-310.

D.A. Van Valdhuizen, G.B. Lamont, (2000) On Measuring Multiobjective Evolutionary Algorithms Performance, Proceedings of the 2000 Congress on Evolutionary Computation, Vol. 1, pp. 204-211.

D.A. Van Valdhuizen, G.B. Lamont, (2000b), Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art, Evolutionary Computation, Vol. 8, No. 2, pp.25-147.

E. Zitzler, (1999), Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications, Thesis submitted to the Swiss Federal Institute of Technology Zurich, Shaker Verlag, Germany.

E. Zitzler, K. Deb, L. Thiele, C.A. Coello Coello, D. Corne (eds.), (2001), Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization, EMO 2001, Lecture Notes in Computer Science Vol. 1993, Springer, Zurich, Switzerland.

J.B. Zydallis, D.A. Van Valdhuizen, G.B. Lamont, (2001), A Statistical Comparison of Multiobjective Evolutionary Algorithms Including the MOMGA-II, Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization, EMO 2001, Lecture Notes in Computer Science Vol. 1993, pp.226-240, Springer, Zurich, Switzerland.