Evolutionary Multiobjective Optimization with a Segment-Based External Memory Support for the Multiobjective Quadratic Assignment Problem

Adnan Acan
Eastern Mediterranean University
Computer Engineering Department
Gazimagusa, TRNC
Mersin 10, TURKEY
adnan.acan@emu.edu.tr

Ahmet Ünveren
Eastern Mediterranean University
Computer Engineering Department
Gazimagusa, TRNC
Mersin 10, TURKEY
ahmet.unveren@emu.edu.tr

Abstract - **Multiobjective evolutionary optimization has been demonstrated to be an efficient method for some difficult multiobjective optimization problems; particularly the quadratic assignment problem which is a provably difficult NP-complete problem with a multitude of real-world applications. This paper introduces the use of a segment-based external memory in evolutionary multiobjective optimization. In principle, variable-size solution segments taken from a number of previously promising solutions are stored in an external memory whose elements are used in the construction of new solutions. In the construction of a solution, a solution segment is retrieved from the external memory and used in the construction of complete solutions through evolutionary recombination operators. The aim is to provide further intensification around promising solutions without weakening the exploration capabilities. Different instances of the multiobjective quadratic assignment problem are used for performance evaluations and, almost in all trials, the proposed external memory strategy provided significantly better results than the multiobjective genetic algorithm (MOGA).**

1 Introduction

Multiobjective optimization (MOO) framework provides more realistic formulation of many real-life problems since a set of solutions, rather than a single solution, exhibiting different forms of concession among multiple and often conflicting objectives is provided as result of the optimization process. Such a set of solutions is commonly known as a Pareto-optimal set in which Pareto-optimality is defined in terms of a dominance relation between two solutions as follows: given two solutions u and v, $u \neq v$, u is said to dominate v if u is not worse than v in all objectives and u strictly is better than v for at least one objective. For example, for a maximization problem,

$$\text{Max } f(x)=(f_1(x), f_2(x),\ldots,f_K(x)) \qquad (1)$$
$$x=(x_1,x_2,\ldots,x_n) \in R^n$$

solution vector u is better than solution vector v with respect to objective i, if $f_i(u) \geq f_i(v)$, and u is said to dominate v, denoted as $u \succ v$, if and only if

$$f_i(u) \geq f_i(v) \text{ for } i=1,2,\ldots,j-1,j+1,\ldots K \qquad (2)$$
$$f_j(u) > f_j(v) \text{ for at least one } 1 \leq j \leq K.$$

A common difficulty with multi-objective optimization problems is the presence of a number of conflicting objectives and, in general, none of the feasible solutions allow simultaneous optimality for all objectives. Hence, any favorable Pareto-optimum provides a solution exhibiting a subjective compromise between the problem objectives. In order to find such a solution, classical methods transform the multiobjective optimization problem into a single-objective one through different scalarization and objective combination methods that include serious drawbacks in terms of appropriate representation of the real-world problem and the resulting solution qualities [1].

In order to have better mathematical models for real-world problems and increase the efficiency of search within arbitrarily complex solution spaces through providing a set of solutions rather than a single solution, some advanced MOO techniques have been proposed in the last few years [2]. These techniques are generally based on some metaheuristics such as Simulated Annealing, Evolutionary Algorithms, Tabu Search, Particle Swarm Optimization, Artificial Immune Systems, Cultural Algorithms and Ant Colony Optimization algorithms. In this study, the use of an external memory implementation in evolutionary multiobjective optimization is introduced towards improving their search capability and solution quality. For this purpose, a well-known evolutionary multiobjective GA, namely MOGA, is taken as the base search strategy and integrated with a gene segment-based external memory. The details of the proposed strategy are given in Section 4.

2 Multiobjective Genetic Algorithms

Studies on multiobjective GAs started with the pioneering work by Schaffer in 1984 [3] and, since then, there has been a number of different types of multiobjective genetic algorithms. A fundamental motivation behind these studies is due to the population-based search ability of GAs in which a population of individuals captures multiple Pareto-optimal solutions in a single run.

In most of the literature surveys and comparative studies, multiobjective GAs are divided into non-Pareto and Pareto-based approaches [4,5]. In the category of non-Pareto GAs, the first multiobjective GA was VEGA (Vector Evaluating GA) by Schaffer [3]. Basically, it uses one population and associated selection metric for each individual objective function. This method generally results in a poor coverage of Pareto frontier. Later Fourman and Kurasawe introduced multiobjective GAs based on binary tournaments where one selected objective is used as the selection metric within a single population [6,7]. Different objective selection methods and diversity maintenance mechanisms are tried within these approaches, however, they tend to converge to a subset of the Pareto-frontier while they leave a large part of it unexplored [8].

The Pareto-based approaches are mainly motivated by a suggestion of a non-dominated GA by D. Goldberg [9]. According to this approach, first non-dominated individuals within the population are identified, they are given the rank 1, and removed from the population. Then, the non-dominated individuals within the reduced population are identified and given the rank 2, followed by their removal from the population. This procedure is repeated until the whole population is ranked. Srinivas and Deb [10] implemented non-dominated sorting GA (NSGA) based on Goldberg's suggestions and sharing techniques between individuals with the same ranking. NSGA and an improved version of it NSGA-II [11] have been proved to be quite successful in maintaining diversity and exploring the Pareto-front in several benchmark numerical multiobjective optimization problems.

Fonseca and Fleming introduced a multiobjective GA (MOGA) [12,13], in which each individual is ranked according to their degree of dominance defined in terms of the number of individuals that dominate a given individual. Accordingly, individuals on the Pareto-front are given rank 1 since they are non-dominated. Both sharing and mating restrictions are employed to maintain population diversity. We have compared our proposed approach with MOGA in the solution of a provably difficult combinatorial optimization problem and have demonstrated that the proposed approach beats MOGA for several problem instances.

The strengthened Pareto evolutionary algorithm (SPEA) presented by Zitzler and Thiele [14] uses two populations P and P' such that copies of all non-dominated individuals are stored in P'. The fitness of the individuals in P' is calculated based on how many individuals in P they dominate and fitness of solution in P is computed as a sum of the fitness values of individuals in P' that dominate it. In addition, selection is performed using binary tournaments from both populations. In the niched Pareto GA (NPGA) of Horn et al. [15], individuals are selected using Pareto-domination tournaments where a subset of the population is used as a basis to form the tournament. Accordingly, if an individual in the subset dominates its contestant, it is selected to survive, else selection is based on the niche count of similar individuals. Both SPEA and NPGA are used for numerical multi-objective optimization problems and proved to be successful with respect to other Pareto-based approached. Comparative studies on multiobjective GAs can be found in [14].

3 Multi-Objective Quadratic Assignment Problem

The quadratic assignment problem (QAP) is a model for many practical problems like backboard wiring, campus and hospital layout, and scheduling. Intuitively, QAP can best be described as the problem of assigning a set of facilities to a set of locations with given distances between the locations and given flows between facilities. The goal then is to place the facilities on locations in such a way that the sum-of-product of flows and distances is minimal [16]. More formally, given n facilities and n locations, two $n \times n$ matrices $A = [a_{ij}]$ and $B = [b_{rs}]$, here a_{ij} is the distance between location i and location j and b_{rs} is the flow between facility r and facility s. The QAP can be stated as follows:

$$\min_{\Psi \in S(n)} J_{\Psi} = \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} a_{\Psi_i \Psi_j} \qquad (3)$$

where $S(n)$ is the set of all permutations (corresponding to the assignments) of the set of integers $\{1, \ldots, n\}$, and Ψ_i gives the location of facility i in the current solution $\Psi \in S(n)$. Here the product $b_{ij} a_{\Psi_i \Psi_j}$ describes the cost contribution of simultaneously assigning facility i to location Ψ_i and facility j to location Ψ_j [17].

The multi-objective QAP (mQAP) with multiple flow matrices naturally models any facility layout problem that is concerned with the flow of more than one type of items or agents. For example, in a hospital layout problem we may be concerned with simultaneously minimizing the flow/distance products of doctors on their rounds, of patients, of hospital visitors, and of pharmaceuticals and other equipment. The mQAP proposed by Knowles and

Corne [17] uses different flow matrices and keeps the same distance matrix. Given *n* facilities and *n* locations, a $n \times n$ matrix *A*, where a_{ij} is the distance between locations *i* and *j*, and *T* number of $n \times n$ matrices B^t, $t = 1,...,T$, where b_{rs}^t is the t^{th} flow between facilities *r* and *s*, the mQAP can be stated as follows:

$$\min_{\Psi \in S(n)} \begin{cases} \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b^1_{\Psi_i \Psi_j} \\ \vdots \\ \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b^T_{\Psi_i \Psi_j} \end{cases} \quad (4)$$

where min refers to the notion of Pareto optimality [18].

4 The Proposed Segment-Based External Memory Strategy

The basic inspiration behind the use of a gene-segment memory is to store some allelic values that result in non-dominated solutions within the current population so that they can be reused in future generations for providing further intensification around non-dominated solutions. Storing gene-segments from non-dominated individuals of previous generations will give a chance to these individuals to take part in recombination operations over multiple future generations, which is obvoiusly an effective method of intensification around the set of non-dominated solutions.

Generation of a gene-segment from a given parent chromosome is illustrated in Figure 1. Simply, a randomly located and random-length gene-segment is selected from the parent chromosome and stored within the external memory. The stored parental objective function values are used in memory update procedures.

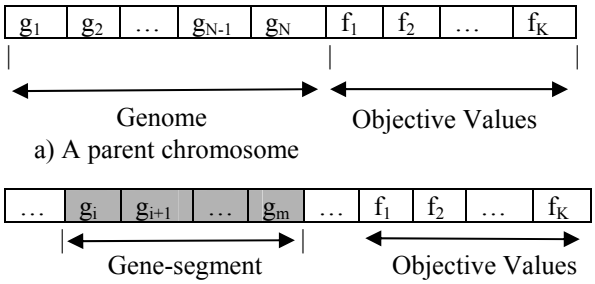

Figure 1. a) A parent chromosome and b) a randomly located random-length gene-segment stored in the external memory.

Based on the above descriptions, an external memory of variable-size solution segments from a number of non-dominated solutions of previous iterations is maintained. Initially the memory is empty and a number classical MOGA iterations is performed to fill in the memory. In this respect, after every iteration, the set of non-dominated solutions are determined and randomly positioned variable-size segments are cut from these individuals, one segment per solution, and stored in the memory. Each segment is also associated with a vector of objective function values that will be used in updating the memory. The memory size *M* is fixed and normal MOGA iterations are repeated until the memory is full.

After the initial phase, MOGA algorithm works in conjunction with the implemented external memory as follows: in order to construct an offspring, one parent is selected from the current population based on the MOGA's ranking methodology. Then, a solution segment is retrieved from the memory using a uniformly random selection strategy and inserted into the parent chromosome similar to a reciprocal translocation operation. The location of the segment within the selected parent is also determined randomly. This operation also plays the role of the crossover where the recombination is carried out between a non-dominated individual and a solution segment of a previously non-dominated solution. This procedure is followed by a mutation of the offspring with a very small probability. Based on these descriptions, the proposed external-memory based MOGA method can be put into an algorithmic form as described in Figure 2.

In memory update procedure, first the set of non-dominated solutions within the offspring population is determined. Then, each individual within this set is compared with the objective vectors of memory elements and in cases where a memory element is dominated by the newly developed solution; the memory element is replaced by a new one cut from the dominating solution. One randomly-positioned and random-length segment is cut from dominating individual within the new population. The results that are obtained by the proposed MOGA strategy are listed in the following section.

5 Results

In all experiments, integer-valued chromosomes representing a permutation of facilities assigned to *N* locations are used for problem representations. The selection method used is the tournament selection with elitism. Uniform crossover is used for conventional MOGAs, whereas there is no crossover operator in the proposed approach since recombination is carried out by inserting selected gene segments into parent chromosomes. A mutation operator is used to modify those parent-only genes, while genes of the inserted sequences remain unmodified, and the mutation rate is 0.06. Experiments are carried out using a population size

of 400 individuals for conventional MOGAs. For the proposed approach, the total number of individuals in the population and the gene- segments library is also taken as 400, i.e. 200 individuals in each. This way, total number of individuals in conventional MOGAs and the proposed strategy are kept the same.

Algorithm: MOGA integrated with an external memory

1. Initialize the population.
2. Evaluate the initial population
3. Repeat
3.1 Perform one MOGA generation
3.2 Determine the set of non-dominated solutions.
3.3 Cut one randomly located random-length gene segment form each non-dominated solution and insert into the external memory. Also, store the length of the segment and the objective vector its parent.
4. Until the memory is full.
5. ITER=1
6. Repeat
6.1 Rank the individuals according to their degree of dominance.
6.2 For i=1 to |Pop|
6.3 Parent=Selection(Pop,Ranking)
6.4 Gene_Segment= Select a memory element uniformly randomly.
6.5 Offspring=Insert(Parent, Gene_Segment)
6.6 Mutate Offspring.
6.7 New_Pop(i)=Offspring.
6.8 EndFor
6.9 Evaluate the new population.
6.10 Update memory using new population.
6.11 ITER=ITER+1.
6.12 Until (Termination Condition == TRUE)

Figure 2. MOGA with an external solution-segment memory.

The developed Algorithm was tested on eleven real-life like instances and fifteen uniform instances making a total of twenty six different mQAP instances. All instances used in experimental evaluations are taken from [17, 19]. The results of the most difficult 2-objective QAP's instances with 10 and 20 locations are given in the following figures.

In Figure 3, non-dominated solution sets provided by the proposed algorithm SBEM (SBEM- Segment-Based Extended Memory algorithm with MOGA) and MOGA when applied to KC10-2fl-1uni uniform instance with 10 locations are presented. According to these results SBEM returned almost the same Pareto points as the given Pareto Front of this problem and found more pareto points than MOGA. In this respect, the number of points on the Pareto front is 12, MOGA found only 7 non-dominated solutions while the number of non-dominated solutions found by SBEM is 13.

In Figure 4, non-dominated solution sets provided by SBEM and MOGA when applied to KC10-2fl-3uni uniform instance with 10 locations are shown. In this case, the non-dominated solutins found by MOGA and SBEM are very close to the ones on the Pareto front, hovewer, the supriority in the success of SBEM is best seen when the number of non-dominated solutions by algorithms is considered. The number of solutions on the provided Pareto front is 64, the number non-dominated solutions found by MOGA is 15, and the number of non-dominated solutions computed by SBEM is 120. This numerical picture makes the additional power gained through the use of the proposed approach much more clear.

In Figure 5, and Figure 6, non-dominated solution sets provided by SBEM and MOGA when applied to KC20-2fl-1uni and KC20-2fl-2uni uniform instances with 20 locations, respectively, are illustrated. In both cases Pareto points of SBEM algorithm clearly dominates the Pareto points of the MOGA algorithm.

Figure 7 exhibits the non-dominated solution sets provided by SBEM and MOGA when applied to KC20-2fl-3uni, uniform instance, with 20 location. Resutlts shows that most of the Pareto points obtained by SBEM algorithm dominates the pareto points obtained by MOGA. For this particular problem instance, the number of Pareto solutions extracted by SBEM and MOGA are 66 and 60, respectively.

In Figure 8, the Pareto fronts provided by SBEM and MOGA when applied to KC10-2fl-5rl real-life like instance with 10 locations are shown and these results demonstrate that SBEM return almost the same Pareto points with MOGA but finds more pareto points than MOGA. The number of points on the provided Pareto front is 48. The number of Pareto points computed by SBEM and MOGA are 33 and 14, respectively. These results demonstrate that SBEM improves the performance of MOGA more than a factor of 2.

Figure 9 shows non-dominated solution sets provided by SBEM and MOGA when applied to KC20-2fl-5rl real-life like instance with 20 locations. Results show that Pareto points of SBEM algorithm outperforms MOGA in the sense that all Pareto points of MOGA algorithm are dominated by those of SBEM. Considering the performances in terms of the number of Pareto solutions computed, the score of SBEM is 105 while that of MOGA is 41.

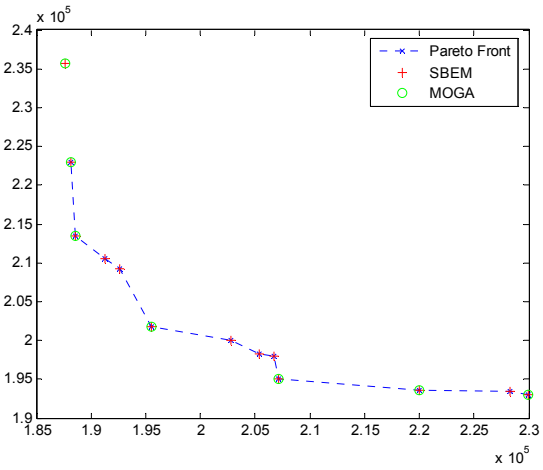

Figure 3. KC10-2fl-1uni: 2-objective QAP instant with 10 locations.

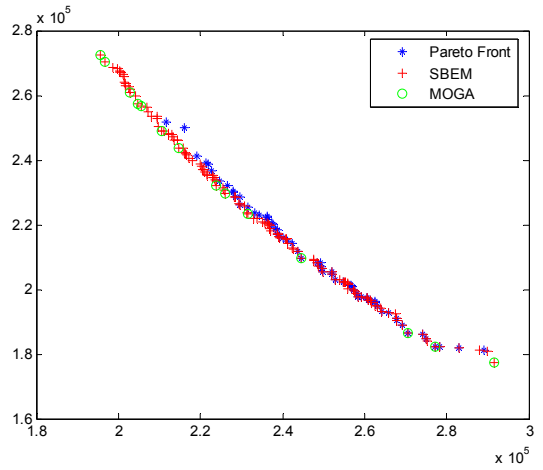

Figure 4. KC10-2fl-3uni: 2-objective QAP instant with 10 locations.

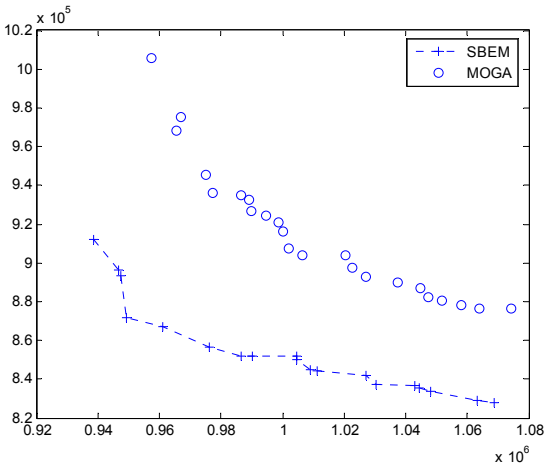

Figure 5. KC20-2fl-1uni: 2-objective QAP instant with 20 locations.

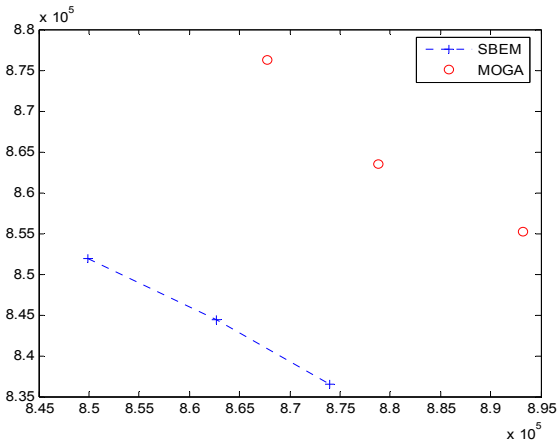

Figure 6. KC20-2fl-2uni: 2-objective QAP instant with 20 locations.

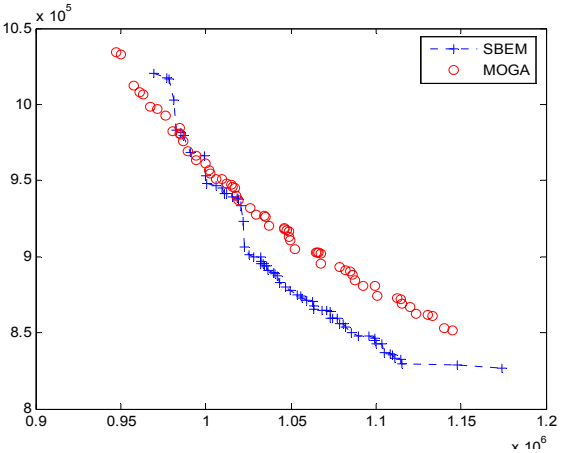

Figure 7. KC20-2fl-3uni: 2-objective QAP instant with 20 locations.

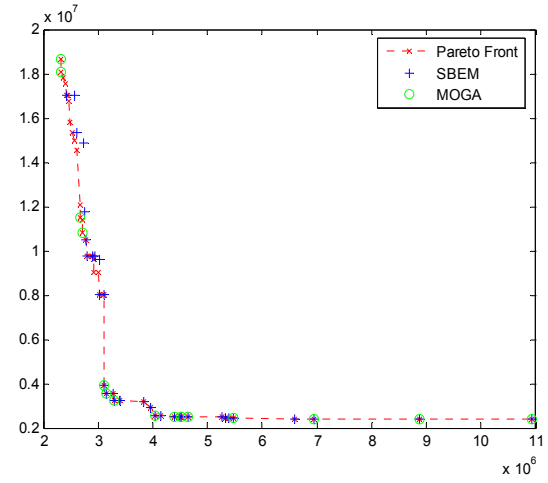

Figure 8. KC10-2fl-5rl: 2-objective QAP instant with 10 locations.

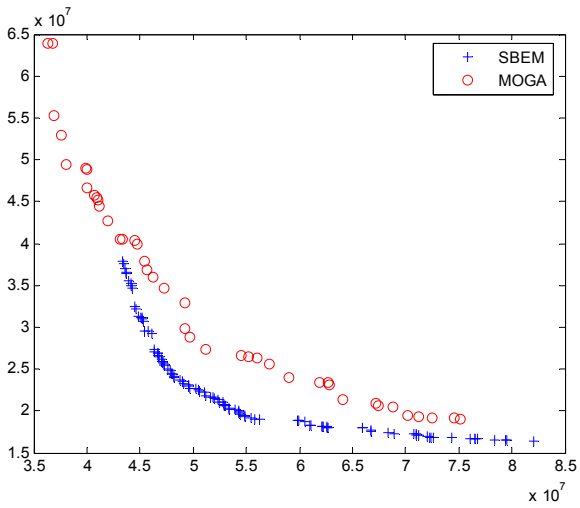

Figure 9. KC20-2fl-5rl: 2-objective QAP instant with 20 locations.

6 Conclusions

In this paper a novel external memory-based multiobjective genetic algorithms strategy employing a gene segments memory including gene segments from potentially promising solutions of previous generations is introduced. The implemented strategy is used to solve a provably difficult combinatorial optimization problem, namely the multiobjective quadratic assignment problem, and its performance is compared with a well known multiobjective genetic algorithm MOGA.

For all test cases, for the same population size, it is concluded that the proposed stragey outperforms MOGA in terms of the number of non-dominated Pareto solution computed. Also, for all problems instances handled in experimental evaluations, the performance SBEM in terms of the closeness to the provided Pareto front is either equally well or significantly better than that of MOGA. These results are important in demonstrating the usefulness of memory-based approaches in evolutionary multiobjective optimization. The presented approach is an intial but successful step in this respect, as demonsrated by the presented results.

This work requires further investigation from following point of views: supporting SBEM with multiple chromosome libraries with different chromosome lifetimes and memory update strategies, use of internal memory-based methods in MOO, and integration with other well-known evolutionary multiobjective optimization algorithms.

Bibliography

[1] C. A. Coello, D. A. Van Veldhuizen, G. B. Lamant, Evolutionary Algorithms for Solving Multiobjective Problems, *Kluwer*, 2002.

[2] X. Gandibleux, M. Sevaux, K. Sörensen, V. T'kindt (Eds.), Metaheuristics for Mutiobjective Optimisation, Lecture Notes in Economics and Mathematical Systems 535, Springer Verlag, 2004.

[3] C. Fonseca C., P. Fleming, An overview of evolutionary algorithms in multiobjective optimization, Evolutionary Computation, vol. 3, pp. 1-18, 1995.

[4] H. Tamaki, H. Kita, and S. Kobayashi, Multi-objective optimization by genetic algorithms: a review, 1996 IEEE International Conference on Evolutionary Computation, ICEC'96, Nagoya, Japan, 1996.

[5] E. Zitzler, L. Thiele, Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, IEEE Transaction on evolutionary computation, vol. 3, pp. 257-271, 1999.

[6] M.P. Fourman, Compaction of symbolic layout using genetic algorithms, 1st Int. Conference on Genetic Algorithms, Pittsburgh, 1985.

[7] F. Kurasawe, A variant of evolution strategies for vector optimization, Parallel Problem Solving from Nature, Lecture Notes in Computer Science 496, Berlin, Springer Verlag, pp. 193-7, 1991.

[8] G. Harik, Finding multimodal solutions using restricted tournament selection, Sixth International Conference on Genetic ALgorithms, 1995.

[9] D. Goldberg, Genetic Algorithms in Search and Machine Learning. Reading, Addison Wesley, 1989.

[10] N. Srinivas, K. Deb, Multiobjective optimization using nondominated sorting in genetic algorithms, Evolutionary Computation, vol. 2, pp. 221-248, 1995.

[11] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multiobjective optimization: NSAGA-II, Parallel Problem Solving From Nature VI, pp. 849-858, 2000.

[12] C.M. Fonseca, P.J. Fleming, Multiobjective optimization and multiple constraint handling with evolutionary algorithms - Part I: a unified formulation, IEEE Transactions on Systems, Man, & Cybernetics Part A: Systems & Humans, vol. 28, pp. 26-37, 1998.

[13] C.M. Fonseca, P.J. Fleming, Multiobjective genetic algorithms made easy: Selection, sharing and mating restriction, 1st IEE/IEEE International Conference on

Genetic Algorithms in Engineering Systems, Sheffield, England, 1995.

[14] E. Zitzler, L. Thiele, Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, IEEE Transaction on evolutionary computation, vol. 3, pp. 257-271, 1999.

[15] J. Horn, N. Nafpliotis, Multiobjective Optimization Using the Niched Pareto Genetic Algorithm, Technical Report 93005, Illinois Genetic Algorithm Laboratory, Dept. of General Engineering, University of Illinois at Urbana- Champaign, Urbana, USA, 1993.

[16] T. Stützle and M. Dorigo, 2001. Local Search and Metaheuristics for the Quadratic Assignment Problem. Technical Report AIDA-01-01, Intellectics Group, Darmstadt University of Technology, Germany.

[17] J. Knowles and D. Corne. Instance generators and test suites for the multiobjective quadratic assignment problem. In C. M. Fonseca et al., editors, *Proc. Of EMO'03*, LNCS 2632, pages 295–310. Springer Verlag, 2003.

[18] M. López-Ibáňez, L. Paquete, T. Stützle, On the Design of ACO for the Biobjective Quadratic Assignment Problem. In M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Montada, and T.Stützle (Eds.), Proc. Of the Fourth International Workshop on Ant Colony Optimization (ANTS2004), *Lecture Notes in Computer Science, Springer Verlag*, 2004.

[19] The multi-objective Quadratic Assignment Problem (mQAP), 2003, http://dbkweb.ch.umist.ac.uk/knowles/mQAP/.