Proceedings of the 6th WSEAS Int. Conf. on EVOLUTIONARY COMPUTING, Lisbon, Portugal, June 16-18, 2005 (pp43-48)

Two-phase multiobjective optimization

ALBERTO CANCELA, JULIÁN DORADO, JUAN R. RABUÑAL, ALEJANDRO PAZOS
Departamento de las Tecnologías de la Información y las Comunicaciones
Universidade da Coruña
Facultade de Informática, Campus de Elviña 15071, A Coruña
SPAIN

Abstract: - This work proposes a genetic algorithm (GA) based approach for the search of the Pareto optimal set of a multiobjective optimization problem. First the global population is divided into various subpopulations. The algorithm operation consists of two phases: firstly each subpopulation tries to optimize a different objective; later the algorithm searches for good compromise solutions between objectives. Information is exchanged by means of the migration of individuals during the second phase. A weighted sum is used for fitness calculation. Weight vectors are randomly generated for each selection event, which creates a wide range of search directions. The good behaviour of the proposed algorithm becomes visible in its application to some continuous problems.

Key-Words: - Multiobjective, optimization, genetic algorithm, subpopulation, migration, memetic

1 Introduction

In multiobjective optimization, obtaining the Pareto-optimal front by means of GAs [1] implies two main objectives: finding solutions that are close to the front, and obtaining a uniform distribution of the points. At present, there is no complete theoretical basis that sustains the use of GAs for multiobjective optimization. However, various methods have been proposed [1] [2] and studies have been carried out to guarantee the convergence of the obtained non-dominated solutions [1] [2] to the Pareto optimal set and, more recently, to maintain the diversity [3].

This work presents a new approach that tries to improve the convergence towards the optimal front. Section 2 contains a description of the proposed two-phase method. An algorithm following the two-phase approach is presented in section 3. The efficiency of the algorithm is compared with other multiobjective genetic algorithms by applying it to various continuous problems. The results obtained are summarized in section 4.

2 The two-phase procedure

Usually it is much more difficult to optimize a function with many objectives than a function with only one objective. The interactions between the different objectives, mainly when not lineal, greatly increase the difficulty of multiobjective problems. The proposed approach is based on the 'Divide and Conquer' paradigm. The approach can be divided in two phases.

During the first phase the main objective is to find good solutions for each of the objectives independently of the other. To achieve this goal we propose the division of the global population into subpopulations, each optimizing one objective. The separation of the individuals into smaller groups allows a greater convergence speed in each subpopulation. Also, if there exists certain independence between the subpopulations, each of them can converge towards a different region of the search space, helping to maintain some degree of diversity. Another benefit of the use of subpopulations is that it facilitates the implementation of a parallel GA: just processing each subpopulation in a different machine. Other approaches based on the use of subpopulations have been published [4, 5]. A metapopulation evolutionary algorithm (MEA) for multiobjective optimisation problems was recently proposed [5]. In that work elements from landscape ecology and population dynamics are used to develop an algorithm that combines 'diffusion' and 'island' properties. During the second phase the search is centered on finding good compromise solutions between the objectives. At this stage, each subpopulation is allowed to periodically export its best individuals, so that, when combined with those of other subpopulations, they can produce good compromise solutions of the Pareto optimal set.

A similar approach that also uses two phases, although not a GA, was already proposed [6] but only for the bi-ojective case. The two phases of this procedure are to (i) generate an initial solution by optimizing only one single objective, and then (ii) , using some heuristics, different aggregates of the objectives are optimized, increasing in each step the importance of the second objective. Although this approach showed interesting results, test problems

used had clustered solutions that for similar weighted objectives optimal solutions were very similar and the algorithm took advantage of that property.

In this paper we propose an algorithm that uses the two phase approach. The algorithm is based on the one proposed by Ishibuchi and Murata: *Multi-Objective Genetic Local Search Genetic Algorithm* (MOGLS) [7]. MOGLS uses elitism and an external set of non-dominated solutions so according to Coello [8] it could be considered as a second generation multiobjective evolutionary algorithm. Besides it has shown promising results [2] and it takes under consideration the use of a local search that can improve the convergence towards the Pareto optimal front. The main reason of choosing MOGLS is that it is easy to implement although not the most effective. Therefore, good experimental results can be justified by the use of the two-phase approach and not to the algorithm used as the basis. It should be noted that weighted sum method cannot capture points on the non-convex portion of a Pareto curve [9].

3 An algorithm with two phases

As stated in the previous section, the proposed algorithm applies a division into subpopulations. Following the nomenclature of Ishibuchi and Murata, it could be called *Multi-Objective Genetic Local Search with Subpopulations* (MOGLSS). The division of the population is carried out by a division operator (cf. section 3.1). The MOGLSS uses a tentative set of non-dominated solutions for each subpopulation, which is the set of all non-dominated solutions found during the execution of the algorithm in the subpopulation. A migration operator was added to deal with the problem of information interchange between different subpopulations (cf. section 3.2).

Similarly to the MOGLS, the MOGLSS uses a weighted sum of the *n* objectives [9] to obtain the fitness value of each solution:

$$f(x) = w_1 f_1(x) + w_2 f_2(x) + \ldots + w_n f_n(x)$$

The weight values w_1, …, w_n are not constant but randomly specified whenever a pair of parent solutions are selected for a crossover operation. The weights distribution varies according to the execution phase of the algorithm (cf. section 3.1).

The execution of the proposed algorithm is divided in two phases. During the first phase, each subpopulation tries to optimize one of the objectives without exchanging information with the other subpopulations. During the second phase, each subpopulation periodically exports copies of its best individuals, so that, when combined with those of other subpopulations, they can produce good compromise solutions.

MOGLS applies a local search to each solution generated by the genetic operators. The search consists in examining *k* (user specified number) neighbour solutions of each individual *i*. If any is better than *i* then *i* is replaced with it. The use of a local search is maintained in MOGLSS. The different steps of this algorithm are outlined in figure 1.

Algorithm
N_p: size of the population.
N_e: number of non-dominated solutions that are added to the population in each generation
N_s: number of subpopulations.
Step 0 (Initialization): Generate an initial population of N_p solutions. Initialize the non-dominated solution sets as the empty sets.
Step 1 (Evaluation): Calculate the values of the objective functions for each solution of the current population.
Step 2 (Division): Divide the population and update the tentative sets of non-dominated solutions.
Steps 3 to 8 are executed out for each subpopulation
Step 3 (Change of phase): If a given number of generations has passed, switch to the second phase.
Step 4 (Selection): Repeat until $(N_p/N_s - N_e)/2$ pairs of parent solutions are selected in each subpopulation: Randomly generate a normalized weight vector (according to the actual phase) and select a pair of solutions using the vector obtained.
Step 5 (Crossover and mutation): Apply crossover and mutation operations and evaluate generated solutions.
Step 6 (Elitist strategy): Randomly select in each subpopulation N_e individuals of the tentative set of non-dominated solutions of the subpopulation, and add them to those obtained in step 5.
Step 7 (Local search): Apply a local search process to the individuals obtained in step 6. Update the tentative set of non-dominated solutions of the actual subpopulation.
Step 8 (Migration): If the algorithm is in the second phase, apply the migration operator.
Step 9 (Finalization Test): If stopping conditions are not met return to step 3.

Fig. 1. MOGLSS algorithm.

3.1 Subpopulations and weight generation

The objective of the first phase, as has already been stated, is to optimize each objective separately. This is achieved by creating, at the start of the execution of the algorithm, as many subpopulations as objectives has the problem that is being solved. Henceforth, we shall suppose (without loss of generality) that the nth subpopulation searches optima for the nth objective. The initial separation of individuals is done through the assignment of the solutions that best optimize the nth objective to the nth subpopulation.

To help each subpopulation to search its corresponding optima, during the first phase, the weight vectors generated do not follow an uniform *m*-dimensional distribution (with *m* being the number of objectives of the problem). We favour the algorithm to generate higher values in the component associated to the objective that is being optimized. To achieve this goal, weights for the objective we are trying to optimize are generated with an average value six times higher than for the rest of the objectives. Other values were tested but better results were obtained for values between five and ten. As a result, in the nth subpopulation vectors are created with a higher average value for the nth component. However, non zero weight values are allowed for the rest of the components of the vector. This allows a search for solutions in the complete Pareto optimal front, especially in the zones close to the optimum of each objective.

The operation during the first phase helps the subpopulations to place themselves in different regions, since the individuals that best optimize each of the objectives are usually widely divergent. This means that the generation of vectors of different weights should facilitate the maintenance of the population diversity. Another reason to not search exclusively in the direction of the canonical vectors, is that in that case the method would degenerate into a VEGA shape [10]. This method does not provide an uniform distribution of solutions, since it focuses on the extremes of the Pareto Optimal front [7].

The objective of the second phase is to find good compromise solutions between various objectives. To this effect, the weight vectors generated during the second phase follow an uniform distribution as in MOGLS. This allows a search in a wide range of directions.

3.2 Migration

The information exchange between different populations occurs only in the last phase of the execution. Given the fact that during the first phase each subpopulation focuses on optimizing one single objective, the solutions that are obtained in that population will be not so good at the rest of the objectives. To avoid the interference in the search of the other subpopulations, we avoid sending copies of individuals during the first phase by preventing any migrations to occur until it has finished.

During the second phase, however, at fixed intervals that can be modified, synchronous interchanges of individuals [11] do take place between subpopulations. For each migration, the average adjustment is calculated for the nth objective of the individuals of the nth subpopulation. After that, those individuals that are better than the calculated average are selected and their copies are sent to the other subpopulations. The copies of other subpopulations are received subsequently. Finally, when new individuals are received due to migration, we must provide a replacement mechanism that preserves the size of the population. The algorithm used in this work is shown in figure 2. This replacement operator allows to conserve the best individuals already in the subpopulation while allowing the most promising individuals from other subpopulations to be added.

Replacement
f: set of individuals that reach the subpopulation. *g*: set of individuals of the subpopulation. f_b: variable to store the immigrating individual that has the best weighted fitness. g_w: variable to store the individual of the subpopulation that has the worst weighted fitness.
1. Order the individuals of *f* and *g* according to their weighted fitness, using for each case the weight vector that was used during its selection. 2. Select, among the previously non-elected individuals of *f*, the best individual and assign it to f_b. Select the worst individual of *g* among the previously non-elected ones, and assign it to g_w. If f_b dominates g_w or if the weighted fitness of f_b is better than that of g_p, replace g_w with f_b. 3. Repeat step 2 until all the possible replacements have been carried out.

Fig. 2. Replacement algorithm.

4 Results

The proposed algorithm was applied to standard minimization functions: SPH-2, SPH-3 [1] [12], ZDT6 [1] [12] and QV [12] [13]. The expression of those functions is now shown:

SPH-*m*: $f_j(x) = \sum_{1<i<n, i\neq j} (x_i)^2 + (x_j - 1)^2$

$1 \le j \le m, m = 2,3$

Domain $[-10^3, 10^3]^n$,

$n = 100$

ZDT6: $f_1(x) = 1 - \exp(-4x_1)\sin(6\pi x_1)$

$f_2(x) = g(x)\left(1 - \left(f_1(x)/g(x)\right)^2\right)$

$g(x) = 1 + 9\left(\left(\sum_{i=2}^{n} x_i\right)/(n-1)\right)^{\frac{1}{4}}$

Domain: $[0,1]^n$,

$n = 100$

QV: $f_1(x) = \left(\frac{1}{n} \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right) \right)^{\frac{1}{4}}$

$f_2(x) = \left(\frac{1}{n} \sum_{i=1}^{n} \left((x_i - 1.5)^2 - 10\cos(2\pi(x_i - 1.5)) + 10 \right) \right)^{\frac{1}{4}}$

Domain: $[-5,5]^n$,
$n = 100$

MOGLSS performance for these problems is compared to the most used multiobjective evolutionary algorithms: SPEA, SPEA2, NSGA-2 and PESA. The results for those algorithms are those obtained by Zitzler [12] and taken from his web page [14]. MOGLSS parameters, for this comparison, where chosen to make one million evaluations, same number as in the work by Zitzler [12] [14] and 75% of the generations were used during the first phase of the algorithm. The number of local searches for individual was set to 3. The one million evaluations include those from the local search. For each test function 30 runs with different random seeds were carried out. The global population of MOGLSS was set to 80, two point crossover with a probability of 0.6 was used and a tournament selection with three participants was chosen. The mutation rate was 0.01.

SPH-2	C(M,P)	C(M,N)	C(M,S2)	C(M,S)
Mean	0.631	0.620	0.610	1.000
Std.dev.	0.395	0.392	0.392	0.000
	C(P,M)	C(N,M)	C(S2,M)	C(S,M)
Mean	0.204	0.232	0.245	0.000
Std.dev.	0.232	0.265	0.274	0.000

Table. 1. Coverage results for SPH-2[1].

SPH-3	C(M,P)	C(M,N)	C(M,S2)	C(M,S)
Mean	0.900	0.803	0.854	1.000
Std. dev.	0.187	0.216	0.201	0.000
	C(P,M)	C(N,M)	C(S2,M)	C(S,M)
Mean	0.000	0.002	0.000	0.000
Std. dev.	0.000	0.010	0.000	0.000

Table. 2. Coverage results for SPH-3[1].

The crossover operator chosen does not improve the exploration capacities of the algorithm, nevertheless the exploration can be enhanced by the use of the local search. The neighbour solutions for the local search were generated using the following instructions:

If 40% of generations have not passed, modify the value of one of the variables by an amount with an absolute value lower than half the length of the variable domain. In other case, modify it with a percentage of the actual value of the variable. This percentage was randomly generated with an absolute value lower than 85%. This value was empirically chosen after simulations with values of 100%, 85%, 70%, 50% and 25%.

The initial idea was to modify the value of the variable based on its current value. This idea had an important drawback; if a variable had a value near zero it would be difficult to generate values in distant zones. That is the reason to modify the values according to the length of the interval during the first stages of the execution. During the later generations it could be assumed that the values of those variables whose optimum was not near zero had a value distant enough from zero.

The metric used to compare the performance was the coverage C(A,B) [1]:

$$C(A,B) = \frac{\left| \{ a \in A; b \in B; a \text{ weakly dominates } b \} \right|}{|B|}$$

QV	C(M,P)	C(M,N)	C(M,S2)	C(M,S)
Mean	0.011	0.127	0.160	0.288
Std. dev.	0.041	0.134	0.184	0.106
	C(P,M)	C(N,M)	C(S2,M)	C(S,M)
Mean	0.210	0.171	0.300	0.111
Std. dev.	0.000	0.139	0.210	0.074

Table. 3. Coverage results for QV[1].

ZDT6	C(M,P)	C(M,N)	C(M,S2)	C(M,S)
Mean	0.006	0.004	0.005	0.010
Std. dev.	0.012	0.012	0.011	0.018
	C(P,M)	C(N,M)	C(S2,M)	C(S,M)
Mean	0.939	0.957	0.958	0.938
Std. dev.	0.080	0.088	0.010	0.080

Table. 4. Coverage results for ZDT6[1].

The results are shown in tables 2, 3, 4 and 5, where M, N, P, S and S2 denote MOGLSS, NSGA-2, PESA, SPEA and SPEA2 respectively. Tables 1 and 2 show MOGLSS outperforms the rest of the algorithms for SPH-2 and SPH-3. According to the values shown in table 1 and 2 MOGLSS has better scalability with respect to the number of objectives, at least with respect to SPH. Besides, comparable results are obtained for QV (table 3). In this case MOGLSS seems to outperform Spea but it is outperformed by Pesa and Spea2. Results for QV are quite similar for Nsga2 and MOGLSS. The worst obtained results, although acceptable, were those for ZDT6 (table 4). In this case, although the diversity and the spread of the solutions was good, the algorithm failed to converge as near to the Pareto front as the other algorithms. This is caused by the non-convexity of the front [1]. ZDT6 front points

[1] M=Moglss, N=Nsga2, S= Spea, S2=Spea2

have all variables equal to zero except for the first variable. The MOGLSS only managed to get values in the interval (10^{-8}, 10^{-12}) approximately.

Another experiment was carried out to gain knowledge of the influence of the duration of each phase in the final results. We fixed all parameters as in the previous experiment and used a first phase lenght of 25%, 50%, 75% and 90% of the total number of generations. 30 simulations were carried out for each combination of problem and phase lenght. Results are shown in tables 6, 7, 8 and 9. In each cell of those tables we show the set coverage for the corresponding row and column, Coverage(row, column). Besides, for a better performance measure, hypervolume [1] was calculated using normalized objective function values. The results for the hypervolume metric are represented in boxplots in figures 3 and 4.

First phase	*25%*	*50%*	*75%*	*90%*
25%	-	0.229	0.077	0.140
50%	0.290	-	0.142	0.118
75%	0.445	0.359	-	0.186
90%	0.551	0.426	0.358	-

Table. 5. Set Coverage results for SPH-2.

First phase	*25%*	*50%*	*75%*	*90%*
25%	-	0.091	0.050	0.086
50%	0.110	-	0.067	0.047
75%	0.194	0.241	-	0.128
90%	0.288	0.233	0.141	-

Table. 6. Set Coverage results for SPH-3.

First phase	*25%*	*50%*	*75%*	*90%*
25%	-	0.066	0.058	0.216
50%	0.470	-	0.141	0.333
75%	0.529	0.315	-	0.402
90%	0.401	0.234	0.156	-

Table. 7. Set Coverage results for QV.

First phase	*25%*	*50%*	*75%*	*90%*
25%	-	0.225	0.195	0.393
50%	0.588	-	0.356	0.639
75%	0.625	0.429	-	0.668
90%	0.360	0.185	0.170	-

Table. 8. Set Coverage results for ZDT6

From the results of the simulations we can recommend that about 75% of time should be used by the first phase. This behaviour is more evident in QV and ZDT6 test functions (tables 7 and 8 and figure 4). SPH-2 and SPH-3 have very similar optima for each objective. Pareto optimal solutions only change in the first variables while the rest have value zero. During the first phase, search is not disturbed by the migration of individuals. Spending a lot of time in the first phase is not worthless in those cases as the search is not disturbed and the result of the search should contain most of the correct values for most of the Pareto optimal front (97-98 variables out of 100 in these cases). This justifies that similar results were obtained for 75% and 90% first phase length for SPH, even 90% obtained better results. Nevertheless the results for ZDT6 (table 8 and figure 4) are rather different, although it is a similar function from the point view of Pareto optimal points similarity, results were better for 75%. This different behaviour should be a consequence of the non-convexity of the problem.

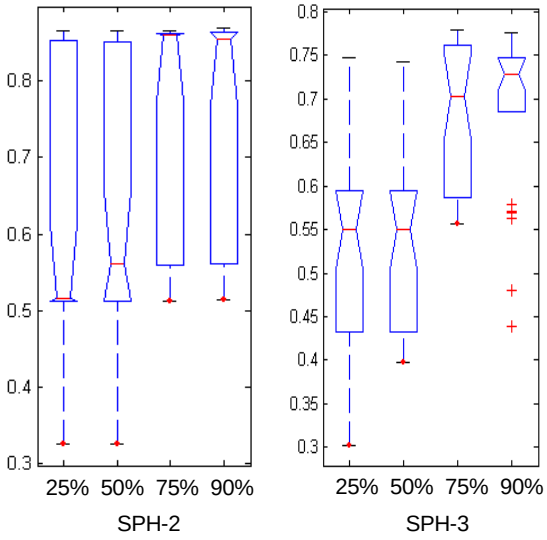

Fig. 3. Hypervolume boxplots for SPH-2 and SPH-3 for first phase different lengths.

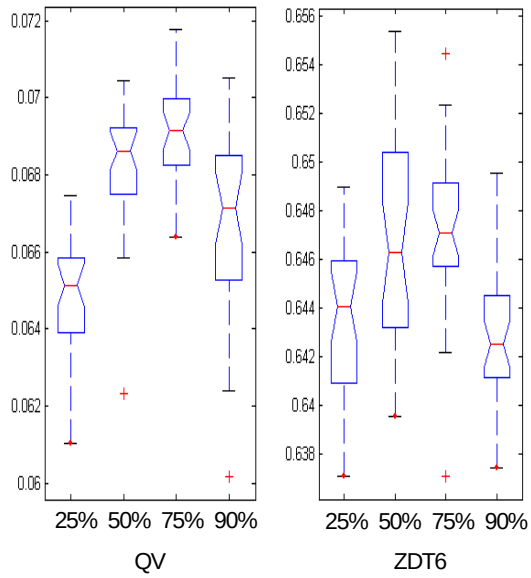

Fig. 4. Hypervolume boxplots for QV and ZDT6 for first phase different lengths.

The results showed that spending little time in the first phase makes difficult to obtain good solutions in only one of the objectives. Nevertheless a short second phase may make difficult to find good compromise solutions. Therefore can be easily deduced that if the user is more concerned about compromise solutions a longer second phase should be used. On the other hand if the user is interested in 'extreme' solutions a shorter second phase is recommended.

5 Conclusion

This paper proposes a new genetic algorithm for multiobjective optimization that is based on a two phases procedure. An implementation based on the multiobjective algorithm with local search [7] is used for performance test. The efficiency of the algorithm is shown through comparison with extensively used test functions and with recent algorithms. Besides, some indications are given about the recommended length of each phase.

It should be also taken under consideration that the distributions on which the weight vectors generation was based could be modified during the execution of the algorithm, according to the preferences of the user. This modification would provide a method with a progressive articulation of preferences which would allow the incorporation of user preferentes, which is one of the future challenges of the multiobjective evolutionary algorithms [8].

The parallel implementation of the algorithm will be developed in future works. Besides the proposed approach should be tested with other multiobjective algorithms like SPEA2, NSGA2 and PESA.

Acknowledgments:

This work was supported by the Spanish Department of Science and Technology (MCYT) (TIC2003-07593) and by the Instituto de la Salud Carlos III (Red nº: G03/160).

References:

[1] Deb K. *Multi-Objective Optimization Using Evolutionary algorithms*. John Wiley & Sons, 2001

[2] Coello C. A. A Comprehensive Survey if Evolutionary-Based Multiobjective Optimization Techniques, *Knowledge and Information Systems. Knowledge and Information Systems*, Vol. 1, No. 3, pp. 269-308, 1999.

[3] Laumanns M. *et al*. *On the Convergence and Diversity-Preservation Properties of Multi-Objective Evolutionary Algorithms*. TIK-Report No. 108. Institut für Technische Informatik und Kommunikationsnetze, 2001.

[4] Whitley D. and Starkweather T. *Genitor II: a Distributed Genetic Algorithm*. Journal Expt. Theor. Artif. Intell., 2:189-214. 1990.

[5] Kirley M. "MEA: A metapopulation evolutionary algorithm for multi-objective optimisation problems". *Proceedints of the 2001 Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Service Center, may 2001, vol. 2, pp. 949-956.

[6] Paquete L. and Stützle. "A two-phase local search for the biobjective traveling salesman problem". In C. Fonseca, P. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Proceedings of the Evolutionary Multi-criterion Optimization (EMO 2003)*, pp. 479-493. Lecture Notes in Computer Science 2632. Springer Verlag, 2003.

[7] Ishibuchi H. and Murata T. "Multi-Objective Genetic Local Search Algorithm", *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, May, Nagoya, Japan, Institute of Electrical and Electronics Engineers, Piscataway, NJ, 119-124, 1996.

[8] Coello C. A. "Evolutionary Multiobjective Optimization: Current and Future Challenges". *Advances in Soft Computing Engineering, Design and Manufacturing*. pp. 243-256, Springer-Verlag, 2003.

[9] Marler R. T. and Arora J. S. Revies of Multi-Objective Optimization Concepts and Algorithms for Engineering. University of Iowa, Optimal Design Laboratory. Technical Report Number ODL-01.03. 2003.

[10] Schaffer J. D. "Multi-Objective Optimization with Vector Evaluated Genetic Algorithms". *Proceedings of 1st ICGA*, p. 93-100, 1985.

[11] Cantú-Paz E. *A Survey of Parallel Genetic Algorithms*, 1998.

[12] Zitzler E., Laumanns M. and Thiele L. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization*. Technical Report no. 103, Computer Engineering and Communication Networks Lab (TIK) Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, May, 2001.

[13] Quagliarella D. and Vincini A. "Coupling genetic algorithms and gradient based optimization techniques". *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science: Recent advances and industial applications*, pp. 289-309. Wiley, Chichester, 1997.

[14] Zitzler E. Test Problem Suite web page. http://www.tik.ee.ethz.ch/~zitzler/testdata.html