

A Min-Max Genetic Algorithm with Alternating Multiple Sorting for Solving Constrained Problems

Timo Mantere

Department of Electrical Engineering and Automation

University of Vaasa

FIN-65101 Vaasa

timan@uwasa.fi

Abstract

This paper introduces a min-max genetic algorithm that can naturally be applied to the min-max problems. A min-max genetic algorithm was originally designed for simultaneous minimization and maximization of the same object function during the same optimization run. In this paper we apply the method with multiple sorting for optimizing constrained functions. The other end of the genetic algorithm population minimizes the constraint violations and the other end maximizes the values of feasible solutions. According the results this method reaches the feasible area reasonably fast and consistently and produces relatively good results.

1 Introduction

This paper studies if the use of min-max genetic algorithm (MMGA) is beneficial when optimizing constrained functions. The MMGA means genetic algorithm where the elitism is applied to the both ends of the population fitness scale. In other words the number of individuals with the lowest fitness value and number of individuals with the highest fitness value will survive to the next generation. In practice this means that we are simultaneously minimizing and maximizing the same object function.

In an elitist steady-state genetic algorithm (GA) the optimization direction depends on the elitism. If some individuals with the highest fitness values will survive to the next generation we are maximizing the target function and vice versa if a number of individuals with lowest fitness values are we are minimizing the problem. If we preserve both individuals with the highest fitness value and individuals with the lowest fitness values we have MMGA, which does simultaneous minimization and maximization of the same object function. This multi-objective, or more precisely bi-objective, optimization method could also be called two-headed GA or reciprocal GA (Mantere, 2004).

This method was first introduced in papers (Mantere and Alander, 2001 and 2002). However, the application therein was relatively special, so the somewhat promising result with that application could not be generalized. In paper (Mantere, 2004) this method is represented more precisely with a set of unconstrained optimization problems. The paper reached the conclusion that this method is beneficial if the problem is difficult, multimodal and non-serialisable. The most important requirement seemed to be that two or more of the problem parameters had must joint effect and influence together to the component of the fitness value sum. With completely separable problems like onemax where each parameter alone directly produces one component of the total fitness value sum, the use of MMGA was not beneficial.

The findings in (Mantere, 2004) also seem to support the claim that MMGA is beneficial when one needs to find both the minimum and maximum of a problem simultaneously. There is a set of problems called min-max problems where this kind of simultaneous minimizations and maximizations are needed.

In this paper we apply the min-max GA method to a collection of constrained test functions in order see if this method works with them. We treat

constrained maximization problem as a multi-objective problems and MMGA is applied so that the same genetic algorithm maximizes the object function and minimize the constraint violations. This is achieved by multiple sorting, where the feasible solutions are first sorted to the other end of fitness scale according to the fitness value. Thereafter the infeasible solutions are sorted to the other end, according to the constraint violations.

This way we can have simultaneous optimization of feasible solutions and the minimization of constraint violations among the infeasible solutions in the same GA population.

1.1 Genetic Algorithms

Genetic algorithms (Holland, 1992) are computer based optimization methods that uses the Darwinian evolution of nature as a model. The solution base of the problem is encoded as individuals that are chromosomes consisting of several genes. The GAs are simplified models derived from the natural genetics, *e.g.* in GAs the individual (phenotype) is usually exactly derived from the chromosome (genotype), whereas in the nature phenotype is not directly derived genotype, but the age, living conditions, diseases, accidents etc. of an individual have effect to the phenotype. In GAs the virtual individuals are tested against the problem represented as a fitness function. The better the fitness value individual gets the better chance it has to be selected to be a parent for new individuals. The worst individuals are removed from the population in order to make room for the new generation. Using crossover and mutation operations GA creates new individuals. In crossover we select the genes for a new chromosome from each parent using some reselected practice, *e.g.* one-point, two-point or uniform crossover. In mutation we change random genes of the chromosome either randomly or using some predefined strategy. The GA strategy is usually elitist and follows the “survival of the fittest” principles of Darwinian evolution.

1.2 Related work

The MMGA method is relatively simple, but we have not yet found other studies suggesting similar method. The book by Bäck *et al* (2000) introduces several ways of enhancing the performance of evolutionary algorithms, but do not represent a report of similar experiments as ours.

That might be due that there is a very limited set of problems where this kind of simultaneous minimization and maximization of the same function is necessary or profitable. Perhaps, someone has

tried this kind of method earlier with the problems it is not well suited for, but results have not been good enough to be published (Alander, 2004).

This method has some similarity to co-evolution (Brodie, 1999, Bäck *et al*, 2000), but in this case we have only one species, with two races, that evolves simultaneously towards both “bad” and “good” solutions (= minimum and maximum) for the target problem.

In multi-elitist GA methods (Bellamo *et al*, 2002) elitism is applied so that in multi-modal problem the best individual from each peak is preserved to the next population. In our method we preserve peak and valley.

Obviously other kind of methods of multi-objective optimizations (Bäck *et al*, 2000, Deb *et al*, 2002) can be seen as related work to this one, since we also have two objectives here.

1.3 The benchmark problems

The set of constrained benchmark functions used in this paper have earlier been used by many researchers, we compare our results with the three papers (Runarsson and Yao, 2000, Mezure-Montes and coello, 2005, Li *et al*, 2005) that represented the best results with evolutionary algorithms we found.

These test problems are introduced in the paper by Runarsson and Yao (2000), where each function formula are given and the characteristics of the corresponding problem are described.

We found several other papers, suggesting several other methods for optimizing these functions, but many of these papers represented their results with only 4 to 7 problems of the 13 benchmark problems. We feel that leads to the biased and weak conclusions. We reach the optimum with 5 of the 13 benchmark problems. If we would have chosen to report our results only with those five, we would have misrepresented the potentials of our method. We think it is necessary to also represent how it works with the other 8 benchmark problems.

There is a paper by Morovic and Wang (2003) that clearly demonstrates the importance of not using a limited benchmark set when trying to represent the results with some algorithm. They showed that by selecting convenient five-image subset from the set of 15 test images, they could get any of the six Gamut Mapping Algorithms they tested to appear like it is the best when compared to the others. They looked 21 research papers on GMAs and find out that all these papers have represented results by using only 1 to 7 test images from the 15 image test set.

1.4 Some discussion about the method

In order to have co-evolution the species should have some interaction. In this case one might suspect that there is no beneficial interaction, since the result of crossover between “bad” and “good” population members are unlikely to result any good solutions neither to the maximization or minimization of the problem. However, in this case we do have two races of the same species in the same population. They are of the same species, since they can mate with each others, and their genetic structure is the same. They are of different race, since they have one profound difference; the other end of the population is usually occupied by feasible solutions (race1), and another with unfeasible solutions (race 2).

Flores and Smith (2003) conclude that the presence of infeasible solutions speeds up the search. So, the interaction between feasible and infeasible solutions should be beneficial.

The main reason that the presence of infeasible solutions in the population is beneficial is that most of the test problems are such that the optimum is located in the border of infeasible area.

When we minimize the sum of constraint violations we are moving towards the feasible area. However, we should reach the feasible area from the different directions, since both the search for the feasible area and the search of optimum can get stuck to the local optimum. Therefore using different or alternating strategies and target functions becomes reasonable.

2 The proposed method

In the constrained optimization heuristic method such as a genetic algorithm is likely to generate a lot of infeasible solutions that violate the constraints. The usual way to handle constraints is to add penalty term to the fitness function, so that it penalizes the constraint violation. It is difficult to define a proper penalty function and thus one often either over penalize or under penalize. In over penalized situation all feasible solutions are considered better than infeasible. This usually means that the possibly beneficial genetic information that infeasible solutions may possess will quickly disappear from the population and we will get stuck to the local optima. In the case of under penalizing the penalty is not hard enough and all our solutions may violate constraints.

In this paper we do not want to use any penalty functions. Instead we use multiple sorting rules in order to sort the population, so that the feasible and infeasible solutions will remain in the same population. One may argue that using the sum of

constraint violations is the same thing as using the penalty function. We think that it should rather be seen as using the minimization of constraint values also as objectives.

When we do changes to the sorting rules it means that different infeasible solutions become the strong individuals among the population. It also means we are moving towards the feasible area from the different direction.

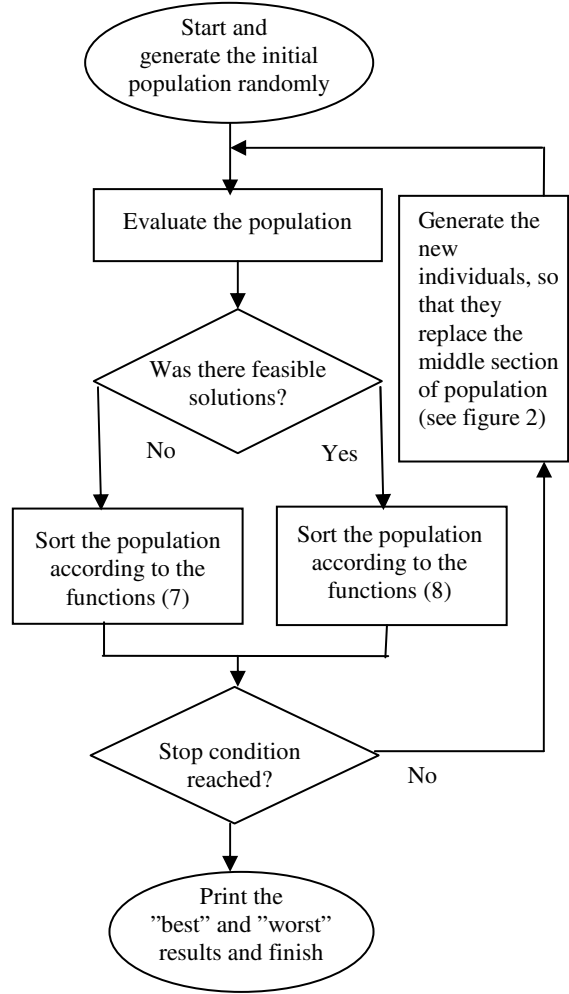


Figure 1. The flow chart of the proposed method, MMGA with multiple sorting.

We have the constrained problem:

$$\text{Maximize } f(x), \quad x = (x_1, \dots, x_n) \in R^n \quad (1)$$

with box coefficients that limit the search space:

$$x_{i_{\min}} \leq x_i \leq x_{i_{\max}}, \quad i = \{1, \dots, n\} \quad (2)$$

and with the inequality coefficients:

$$g_j(x) \leq 0, \quad j = \{1, \dots, m\} \quad (3)$$

The equality constraints:

$$h_k(x) = 0, \quad k = \{1, \dots, q\} \quad (4)$$

are transformed into an inequality constraints by defining that:

$$g_j(x) = |h_k(x)| - \delta \leq 0 \quad (5)$$

That way all the constraint violations will be positive. If the target function is minimization problem, we change the sign and get maximization, so we are minimizing the constraints violations and maximizing the function value.

In the proposed method we use steady-state genetic algorithm (fig. 1) with some amount of elitism N_e , in such a way that $N_e/2$ of the best individuals and $N_e/2$ of the worst individuals will survive to the next generation (figure 2). In practice this means that the population should evolve towards the both directions, the global maxima and minima.

The parent selection scheme is such that each of the individuals in the population has equal opportunity to become parents, $1/N$. This was chosen because if we use some kind of emphasis on good individuals, like ranking order or fitness value, as a base for unequal mating opportunity, only the "bad" and "good" individuals would mate, and the middle ones would get a very little chance to mate. We also compare every new individual against the members of old population and the other new individuals, so that no identical individuals exist in the population.

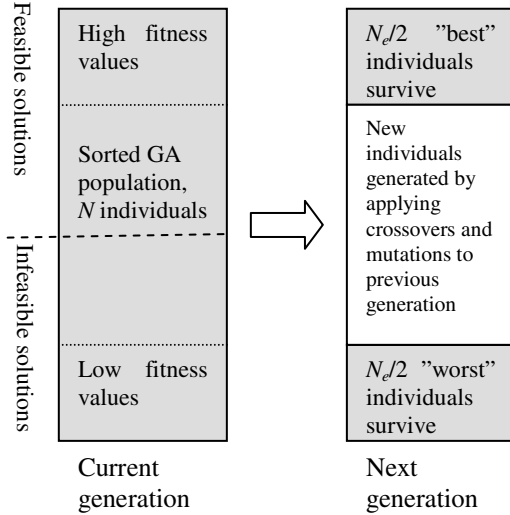


Figure 2. The elitism scene of the proposed method, both a number of individuals with highest fitness values and lowest fitness values will survive to the next generation.

We use multiple sorting with two different settings. In the beginning of the optimization, before

our algorithm reaches the feasible area (one or more individuals of the population are feasible solutions for that problem), we sort the whole population according to the first function of the function pair (7), and after that the bottom half of the population $[N/2, N]$ by the second function of pair (7). The function pair is selected randomly in every generation; so that each function pair has 25% chance to be the pair that biases the optimization direction.

When the feasible area is reached, all the feasible solutions are first sorted according to the function value $f(x)$, the rest of the population or at least the bottom 25% $[3N/4, N]$ is sorted according to one of the four alternating functions seen in (8). The other function that minimizes the other end is again randomly chosen in every generation, and each of the four functions has the 25% chance to be selected. Two of these functions use only the coefficient violations for biasing our search, and the other two use different combinations of function value and coefficient violations.

We decided to use target function that returns us three values:

$$\begin{cases} f(x) \\ \sum_{j=1}^m g_j(x) \\ \text{Max} \{g_j(x)\} \end{cases} \quad (6)$$

They represent the target function value, the sum of all constraints violations, and the highest constraint violation accordingly.

These values are used in different functions pairs that we decided to use for sorting the population before reaching the feasible area, they are:

$$\begin{cases} \text{Maximize: } f(x), \text{ Minimize: } \sum g_j(x) \\ \text{Maximize: } -\text{Max}\{g_j(x)\}, \text{ Minimize: } \sum g_j(x) \\ \text{Maximize: } f(x) - \sum g_j(x), \text{ Minimize: } \text{Max}\{g_j(x)\} \\ \text{Maximize: } f(x) - \sum g_j(x), \text{ Minimize: } \sum g_j(x) \end{cases} \quad (7)$$

Different sorting function biases the search direction differently. Each function pair has at least one function that leads the search towards feasible area. The other function may be just lead towards the higher function $f(x)$ values, but it may also lead toward the feasible area from the different direction. The reason for using the multiple sorting is to force the population ends differ from each other; so that the other end contains the infeasible solutions and hopefully the other end will contain feasible ones.

Table 1. The amount of parameters, n , and the amount of inequality constraints, m , of each test problem g01 to g13. Also the number of generations (Best, Median, and Worst) that MMGA needed to reach the feasible area of each function, and also the descriptive statistics of 30 MMGA test runs with each problem. * Marks the average of 21 out of the 30 test runs that reached feasible area.

Test problem			Reach feasible			Optimal solution	Descriptive statistics of the MMGA results				
g	n	m	B	M	W		Best	Median	Average	Worst	St.Dev.
01	13	9	42	61	329	-15.000	-15.000	-15.000	-15.0000	-15.0000	0
02	20	2	1	1	1	0.803619	0.80355	0.793961	0.79185	0.76144	0.01100
03	10	1	13	33	87	1.000	1.0000	0.99997	0.99980	0.99823	0.00044
04	5	6	1	1	1	-30665.539	-30665.487	-30665.243	-30665.043	-30664.060	0.43048
05	4	5	423	3375	Infea	5126.498	5126.503	5290,247	5378.786 *	Infeasible	275.349
06	2	2	5	21	50	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814	0
07	10	8	19	33	60	24.306	24.448	25.986	25.589	29.509	1.237
08	2	2	1	1	6	-0.095825	-0.0958	-0.0958	-0.0958	-0.0958	0
09	7	4	1	2	6	680.630	680.643	680.684	680.691	680.791	0.03422
10	8	6	25	97	460	7049.331	7054.002	7607.787	8034.110	10346.330	1477.12
11	2	1	14	62	194	0.750	0.750	0.750	0.750	0.750	0
12	3	729	1	1	1	-1.000	-1.000	-1.000	-1.000	-1.000	0
13	5	3	66	146	56	0.05395	0.06629	0.86610	0.77731	0.99041	0.23066

When the feasible area is reached our function pairs changes to:

$$\left\{ \begin{array}{l} \text{Minimize} : \sum g_j(x) \\ \text{Maximize} : f(x), \text{ Minimize} : \text{Max}\{g_j(x)\} \\ \text{Minimize} : \sum g_j(x) - f(x) \\ \text{Minimize} : \text{Max}\{g_j(x)\} - f(x) \end{array} \right. \quad (8)$$

The reason for changing the sorting functions randomly from generation to generation is to make more unlikely that our optimization would get stuck to the local minimum. When the sorting function changes among the infeasible solutions, the different members of the population becomes more fit, and that biases the search direction differently, so we are entering the feasible area from the different direction. This procedure also seems to reduce the time needed before the algorithm reaches the feasible area.

3 Experimental results

In order to test the proposed method we decided to use a real-value coded GA with uniform and arithmetic crossovers and Gaussian mutation. The size of the GA chromosome depends on target function, *i.e.* how many parameters (n) it has (table 1).

First, we run some preliminary tests in order to find a set of good GA parameters for these problems. We did not found any universal parameter set that would work with all of the problems, so we

selected the following: population size $N=100$, elitism $N_e=50$, mutation percentage $50/n$ (table 1) with Gaussian mutation and crossover ratio 50 (both uniform and arithmetic crossovers with 50/50 chance), execute 6999 generations. This means total 350 000 function evaluations ($100+6998*50$). The high elitism was used because it forces the two races in the population interact, having interracial crossovers, more efficiently. High mutation percentage, since the amount of average Gaussian mutation was small.

Our test function set in these experiments was taken from study of Runarsson and Yao (2000). The limits of optimization space (box coefficients for X_i) were also adopted from them, as well as the degree of violation $\delta=0.0001$ for the equality constraints $|h(x)| - \delta \leq 0$. They report that they used $200*1750$ ($=350 000$) function evaluations, and they used 30 test runs with each problem, and so did we. Our results in this paper are compared to their results with all of the 13 problems, and also against the results represented by Li *et al* (2005) and by Mezures-montes and Coello (2005).

Table 1 shows how quickly our MMGA with multiple alternating sorting methods reached the feasible area with each of these problems (Best, Median and Worst case of the 30 test runs). With the problem g05 we reached the feasible area only 21 times out of 30 test runs, so the worst result was infeasible. With the all other problems our method reached the feasible area between 1 to 460 generations (that corresponds from 1 to 23000 trials).

Table 2. The best, average and worst results with our method (MMGA), and with methods represented by Runarsson and Yao (RY, 2000) Li *et al* (LJW, 2005), and Mezura-Montes and Coello (MC, 2005) with different test functions.

g	Best results				Average result				Worst result			
	MMGA	RY	MC	LJW	MMGA	RY	MC	LJW	MMGA	RY	MC	LJW
01	-15.000	-15.000	-15.000	-15.000	-15.000	-15.000	-15.000	-15.000	-15.000	-15.000	-15.000	-15.000
02	0.80355	0.80352	0.80360	0.80360	0.79185	0.78198	0.78523	0.78571	0.76144	0.72629	0.751322	0.74448
03	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.998	1.000	1.000	1.000
04	-30665.49	-30665.54	-30665.54	-30665.54	-30665.04	-30665.54	-30665.54	-30665.54	-30664.06	-30665.54	-30665.54	-30665.54
05	5126.503	5126.497	5126.599	5126.498	5378.786	5128.881	5174.492	5133.910	Infeas.	5142.472	5304.168	5165.659
06	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814	-6875.940	-6961.284	-6961.810	-6961.814	-6350.262	-6952.482	-6961.780
07	24.448	24.307	24.327	24.312	25.589	24.374	24.475	24.498	29.509	24.642	24.843	24.825
08	-0.0958	-0.0958	-0.0958	-0.0958	-0.0958	-0.0958	-0.0958	-0.0958	-0.0958	-0.0958	-0.0958	-0.0958
09	680.643	680.630	680.632	680.630	680.691	680.656	680.643	680.650	680.791	680.763	680.719	680.696
10	7054.00	7054.316	7051.903	7049.499	8034.110	7559.192	7253.047	7168.782	10346.33	8835.655	7638.366	7474.948
11	0.750	0.750	0.750	0.750	0.750	0.750	0.750	0.750	0.750	0.750	0.750	0.750
12	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
13	0.06629	0.053957	0.053986	0.053986	0.77731	0.05701	0.166385	0.055683	0.99041	0.21692	0.468294	0.06171

The table also contains the descriptive statistics of our test runs. They show that our method reached the optimal solution in every test run with the problems g01, g06, g08, g11, and g12.

Table 2 shows the comparison of our results: best, average, and worst values to the results represented by Runarsson and Yao (RY, 2000) with their evolutionary strategy with stochastic ranking, and by Li *et al* (LJW, 2005) with GA hybrid with interpolation, and by Mezura-Montes and Coello (MC, 2005) with multimembered evolution strategy.

The comparison shows that all of the compared methods reached the optimum every time with the problems g01, g08, g11 and g12. With the problem g06 our method was the only one that reached the optimal solution every time. With the problem g04 all the other methods except ours reach the optimum every time.

Our method works well with the problem g02, where our average and worst results were better than those got by others. With the problem g03 our worst result was worse than those got by others, but the best and average was as good. With g09 our method did not reach optimum and performed slightly worse than other methods.

The most difficult problems for our method seem to be the test problems g05, g07, g10 and g13. With these problems the best result with our method was close to the results got by others, but the average and worst results of 30 test runs were much worse.

With g05 our method also failed to reach feasible area 9 times out of 30.

4 Conclusions and future

In this paper we studied if MMGA (min-max genetic algorithm) is worth of considering in special problems where we need to find both the function minima and maxima. In the case of constrained problem we can define the function so that the same GA maximizes values of the feasible solutions and the other end minimizes the constraint violations of infeasible solutions. In this paper that was achieved by multiple sorting, where sorting rules alternate randomly.

We used conditional fitness function that changes when feasible area was reached, *e.g.* in the beginning, the both ends of population are used for speed up the search of feasible area. When feasible area is reached the other end will start to maximize the feasible solutions, and the other end is dedicated for minimizing either the sum or maximum constraint violations or their sum with the target function value.

The results show that the MMGA method with multiple sorting works relatively well with these constrained test problems. It reaches the feasible region relatively fast and consistently. Unfortunately with one of the benchmark problems we had some difficulties to reach feasible area. We need to study the characteristics of that problem more precisely in order to find out the reason.

Our results were relatively good when compared with the other methods, but our method still needs some fine tuning, since with four of the benchmark problems, our average and worst results were not good. We need to do further analyze of what characteristics of these problems causes the obstacles

for our method, and improve our method accordingly.

Acknowledgements

The research work of the author of this paper was funded by the research grant from the TietoEnator fund of the Finnish Cultural Foundation.

References

- Alander, J.T. *Personal comments*. 10.6. 2004.
- Bäck, T., Fogel, D.B., and Michalewicz, T. (eds.). *Evolutionary Computation 2 Advanced Algorithms and operators*. Institute of Physics Publishing, Bristol and Philadelphia, 2000.
- Bellomo, D. Naso, D., and Turchiano, B. Improving genetic algorithms: an approach based on multi-elitism and Lamarckian mutation. In *IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, Hammamet, Tunisia, 6-9, Oct. 2002: 89-94, 2002.
- Brodie E. III, and Brodie E. Jr. Predator-prey arms races. *BioScience* **49**: 557-568, 1999.
- Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* **6**(2): 182-197, 2002.
- Duarte Flores, S., Smith, J. Study of Fitness Landscapes for the HP model of Protein Structure Prediction. In *The 2003 Congress on Evolutionary Computation CEC '03 Volume 4*, 8-12 Dec., IEEE Service Center, Piscataway, NJ: 2338-2345, 2003.
- Holland, J. *Adaptation in Natural and Artificial Systems*. The MIT Press, Cambridge, MA, 1992.
- Li, H., Jiao, Y.-C., and Wang, Y.: Integrating the simplified interpolation into the genetic algorithm for constrained optimization problems. In Hao et al. (eds) *CIS 2005, Part I, Lecture Notes on Artificial Intelligence 3801*, Springer-Verlag, Berlin Heidelberg: 247-254, 2005.
- Mantere T. Reciprocal elitism in GA optimization. In J. Alander, P. Ala-Siuru, and H. Hyötyniemi (eds.), *Step 2004 – The 11th Finnish Artificial Intelligence Conference, Vol. 3, Origin of Life and Genetic Algorithms*, Heureka, Vantaa, 1-3 September 2004, Finnish Artificial Intelligence Society, Helsinki: 151-160, 2004.
- Mantere, T., Alander, J.T. Developing and testing structural light vision software by co-evolutionary genetic algorithm. In *QSSE 2002 The Proceedings of the Second ASERC Workshop on Quantative and Soft Computing based Software Engineering*, Feb 18-20 2002, Banff, Alberta, Canada. Alberta Software Engineering Research Consortium (ASERC) and the Department of Electrical and Computer Engineering, University of Alberta: 31-37, 2002.
- Mantere, T., Alander, J.T. Testing a structural light vision software by genetic algorithms – estimating the worst case behavior of volume measurement. In D. P. Casasent and E. L. Hall, eds., *Intelligent Systems and Advanced Manufacturing: Intelligent Robots and Computer Vision XX: Algorithms, Techniques, and Active Vision*, volume SPIE-4572, Newton, MA, October 29-31 2001, SPIE, Bellingham, Washington, USA: 466-475, 2001.
- Mezure-montes, E., Coello, C.A.C. A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Transactions on Evolutionary computation* **9**: 1-17, 2005.
- Morovic, J., Wang, Y. Influence of test image choice on experimental results. In *Proceedings of 11th Color Imaging Conference*, Scottsdale, AR, Nov. 3-7: 143-148, 2003.
- Runarsson, T.P., and Yao, X.. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation* **4**(3): 284-294, 2000.