FACTA UNIVERSITATIS (NIŠ)
Series: Electronics and Energetics vol. 14, No. 1 April 2001, 19-32

MULTI-CRITERION OPTIMIZATION OF ROBOT TRAJECTORIES WITH EVOLUTIONARY STRATEGIES

Matthias Ortmann

Abstract: The optimization of robot trajectories with genetic algorithms represents a multi-criterion problem because each trajectory consists of many points to be reached with different positions of each joint. The performance of the optimization of a multi-objective problem depends on the coding of the problem and the used algorithm. In this paper first the used coding of a robot arm is presented. Then the algorithm and three possibilities of weighting of the various genes with three different transition functions for each possibility are introduced. In contrast a special technique of selection is presented which performs a manipulation in the case of the selection leading to a speeding up of the whole algorithm. Each of the presented algorithms converge to the Pareto-optimal set by using a special criterion for the end of the optimization.

Key words: Multi-criterion optimization, robot trajectories, Pareto-optimal solution.

1. Introduction

Real world problems mostly consist of multiple objectives which should be optimized simultaneously. Evolutionary algorithms [1], [2] are best suited for the optimization of problems that are too complex to be solved by exact methods like linear programming and gradient search [3], [4]. If the algorithm should find a solution for a problem with one objective, it mostly is easy to find a solution with a good fitness in a short time. But if the algorithm should find a solution for a multi-objective problem, the time needed for solving increases dramatically because the sum of the various objectives mostly is used for the selection. On the other hand optimal solutions according to one objective, if such an optimum exists, often imply unacceptably low

Manuscript received December 12, 2000.

The author is with department of data processing at the Ruhr-University Bochum, 44780 Bochum, Germany (e-mail: ortmann@etdv.ruhr-uni-bochum.de).

performance in one or more of the other objective dimensions [5]. A compromise with acceptable results mostly sub-optimal in the single-objective sense should be reached. In other words, the algorithm should find the *Pareto-optimal solution* [6], [7]. Here no criterion is dominated by any other [8]. The aim of an optimization should be to satisfy all criterions and to find a solution as fast as possible.

Robot trajectories are mostly generated by teach-in methods. Here the various points of the trajectory have to be programmed with the real robot, which is a difficult and time-consuming business. The results mostly are not optimal with regard to the stress in the joints. Therefore it is much better to program the robot off-line by first making a simulation. To get comparable or better results than in the case of teach-in, the robot and the robot cell should be described carefully and in detail. With this model the algorithm should work optimizing the trajectory while considering the detection of collisions of the robot arms with obstacles [9] and other robot arms [10].

In this paper the multi-criterion problem of finding a trajectory of a robot arm should be investigated. The problem should be reduced to the calculation of the trajectory of one robot, disregarding the case of collision. First, stress should lie on the special way of coding of the robot arm. Then the optimization criterion for the trajectory should be introduced followed by three techniques of weighting. In contrast a special technique of manipulation in the case of the selection should be introduced.

2. Description of the Problem to be Solved

First the robot arm should be described. Without loss of generality a robot with five degrees of freedom plus opening and closing of the fingers is used. These kinds of robot arms are called globally degenerated [11]. An abstract example of the robot used can be seen in Fig. 1. The upper arm has the same length as the lower arm. The robot only has joints for rotational use.

To describe the robot in a mathematical way, to each joint is assigned a variable which is, with regard to the evolutionary algorithm, called $GeneX$. The X represents the number of the corresponding joint. Each gene needs a minimum and maximum value to avoid collisions because of the three-dimensional shape of the links. For symmetrical reasons, the offsetvalue which is added to the minimum and subtracted from the maximum, is the same called $GeneX_{min}$. The described circumstances are shown in Fig. 2. To get a higher position accuracy, the angle of 360° is divided into multiple discrete steps that a variable $GeneX \in \mathbb{N}$ is justifiable. The number of steps

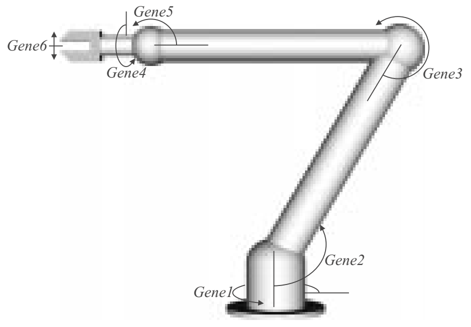

Fig. 1. Coordinates of the robot arm.

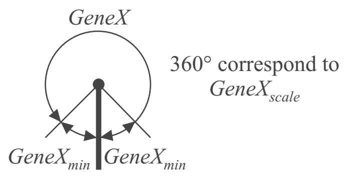

Fig. 2. Coordinates of a joint.

is given for each joint individually by the value of the variable $GeneX_{scale}$. As conclusion each position i of the robot can be written as follows:

$Gene1$:	$Gene1_{min} \leq Gene1 \leq Gene1_{max}, \in \mathbb{N}$
$Gene2$:	$Gene2_{min} \leq Gene2 \leq Gene2_{max}, \in \mathbb{N}$
$Gene3$:	$Gene3_{min} \leq Gene3 \leq Gene3_{max}, \in \mathbb{N}$
$Gene4$:	$Gene4_{min} \leq Gene4 \leq Gene4_{max}, \in \mathbb{N}$
$Gene5$:	$Gene5_{min} \leq Gene5 \leq Gene5_{max}, \in \mathbb{N}$
$Gene6$:	$Gene6_{min} \leq Gene6 \leq Gene6_{max}, \in \mathbb{N}$
$Time1$:	point of time in ms, $\in \mathbb{N}$

Obviously an evolutionary strategy is performed. In this representation the point of time is added to the position of the robot. The whole vector should be called *chromosome*. To get an individual, many chromosomes are lined up at a distance of 40 ms, which is the normal time between two positions when moving a robot arm.

start				end
$Chromosome1$	$Chromosome2$	$Chromosome3$	$\cdots$	$ChromosomeN$
$Gene1_1$	$Gene1_2$	$Gene1_3$		$Gene1_1$
$Gene2_1$	$Gene2_2$	$Gene2_3$		$Gene2_1$
$Gene3_1$	$Gene3_2$	$Gene3_3$		$Gene3_1$
$Gene4_1$	$Gene4_2$	$Gene4_3$	$\cdots$	$Gene4_1$
$Gene5_1$	$Gene5_2$	$Gene5_3$		$Gene5_1$
$Gene6_1$	$Gene6_2$	$Gene6_3$		$Gene6_1$
$Time_1$	$Time_2$	$Time_3$		$Time_1$

This individual describes the whole trajectory of a transportation problem. To specify points between the beginning and the end of the trajectory that should be reached during the movement, further points can be added to the trajectory. These points subdivide the path of the robot arm into many parts called *regions*. These regions are smaller than the whole trajectory and should therefore be optimized easier because of the smaller search space. Another advantage is that, with the knowledge about the previous and the following region, especially the gradient should be mentioned here, the problem can be parallelised by distributing the regions to various computers in a network or to the processors in one computer. Here the second possibility is used. Therefore the program has a structure shown in Fig. 3.

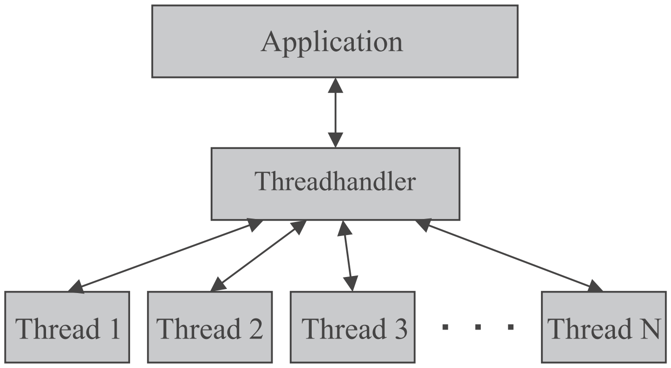

Fig. 3. The structure of the program.

The whole trajectory is given from the application to the *threadhandler*. The threadhandler extracts the regions and distributes them to the working *threads*. Here the regions are optimized by an evolutionary strategy described below. The regions in the threads are called individuals, too. The optimized parts are given back to the threadhandler. Here they are composed to the trajectory and are given back to the application.

3. Optimization Criterion for the Genes

According to the last section the whole trajectory consists of many user defined points with an individual distance of time. Depending on the distance of time further points are generated between these given points. The values of the genes of these points are initialized randomly. Now the aim of the optimization should be a trajectory with a smooth trend especially at the transition from one region to the other.

In principle the genes are independent from each other if only the case without collisions of the robot with obstacles is considered. Therefore it is possible to calculate the fitness values for each gene independently and sum them up to the fitness value of the individual. The fitness value F_{GeneX} of one gene is calculated as follows

$$F_{GeneX} = \sum_{i=1}^{chom} \left[w(i) \cdot curvature_{GeneX}(i)\right]^2 . \tag{1}$$

Here *chrom* is the number of chromosomes of the individual. For the first and the last value of *curvature* in a point, the gradient of the previous and the following region or otherwise the difference of the gradients to the previous and the following point are used. $w(i)$ is a weighting function to get a trajectory with less stress and wear in the joints of the robot. Fig. 4 illustrates the weighting function used for eleven chromosomes.

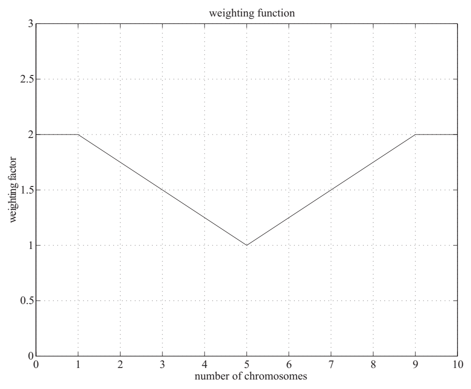

Fig. 4. Weighting of the chromosomes.

$w(i)$ can be expressed in a general mathematical way as follows

$$w(i) = \begin{cases} 2, & i \leq 1 \\ \frac{9}{4} - \frac{1}{4} \cdot i, & 2 \leq i < chrom/2 \\ -\frac{1}{4} + \frac{1}{4} \cdot i, & chrom/2 \leq i < chrom - 2 \\ 2, & chrom - 1 \leq i < chrom \end{cases} \tag{2}$$

The fitness values F_{GeneX} of the genes are summed up to the fitness value of the whole individual.

$$F_{Individual} = \sum_{X=1}^{6} F_{GeneX} . \tag{3}$$

The fittest individual is the one with the smallest fitness value. If the evolutionary strategy should select by the fitness $F_{Individual}$, there are the typical difficulties of multi-criterion optimizations. In the next section three different weighting methods with three different transition functions for each method should be introduced.

4. Possibilities of Weighting the Fitness of the Genes

There surely are many possibilities for the superposition of the fitness values of the genes to one fitness value which is meaningful for the selection. In this paper three different possibilities should be introduced. All of them extend equation (3) to the weighted fitness value $F^w_{Individual}$

$$F^w_{Individual} = \sum_{X=1}^{6} t(F_{GeneX}) \cdot F_{GeneX}, \tag{4}$$

$t(F_{GeneX})$ is a transition function which is dependant on the fitness value of each gene. These fitness functions and special weighting techniques should be introduced now.

4.1 Fixed weighting

In the case of fixed weighting one of the three transition functions in Fig. 5 is used for all genes. Dependant on the fitness value F_{GeneX} the

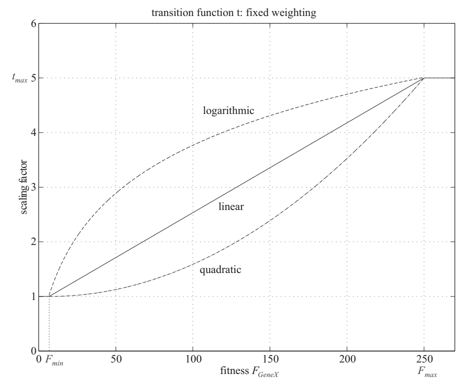

Fig. 5. Fixed weighting

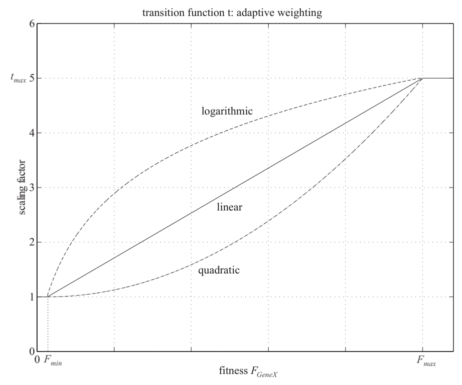

Fig. 6. Adaptive weighting

fitness value of each gene is multiplied by the corresponding value of the transition function. The values F_{max} and t_{max} are user defined but fixed from the beginning of the optimization. The value of F_{min} is calculated as follows: to get a defined criterion for the end of the optimization, a value $F_{Individual,min}$ is defined. A solution of the problem is found if an individual has a fitness value that is smaller than the given minimum. Since the value of $F^{w}_{Individual}$ is the sum of the six F_{GeneX}, the value of F_{min} should be $F_{Individual,min}$ divided by six.

The logarithmic transition function performs a stretching especially of the minor fitness values, which is most important for the speed of the optimization as described in section 6.2. The linear function only performs a linear scaling for all fitness values. The quadratic function corresponds to an upsetting of the minor fitness values.

4.2 Adaptive weighting depending on one gene

The disadvantage of the fixed weighting method is that function $t(F_{GeneX})$ is static. The function is given at the beginning without regarding the range of the fitness values of the genes. It must be better to use a dynamic ranging which calculates the F_{max} value with regard to the maximum of the fitness values of all genes X of the first population. This fact is illustrated in Fig. 6. The value of t_{max} is still user defined as in the case of the fixed weighting. The value of F_{min} is calculated as described above.

4.3 Adaptive weighting depending on all genes

In the last section the transition function is dependant on only one gene. This should be a correct method if the fitness values F_{GeneX} are within the same range. It must be better to calculate the transition function for each gene individually. This possibility is performed in the case of the adaptive weighting depending on all genes. Here, in contrast to the other two methods, not only one transition function for all genes is used. Here each gene X has its own weighting function. The difference between the functions is the value of F_{max}. It is calculated individually for all genes depending on the value of F_{GeneX} of the first population.

5. Manipulation in the Case of the Selection

The last section presents three methods for calculating the fitness value $F^{w}_{Individual}$ by summing up the fitness values F_{GeneX} of each gene with weighting. In this case the following selection only takes the fitness value $F^{w}_{Individual}$ of the individual into account. But here is a special situation of a multi-criterion problem. The six fitness values of the genes are nearly independent from each other. They all describe a special path of the corresponding joint of the robot and for the global problem they only describe an optimal trajectory if all paths of the genes are optimal. But there still is the fact that each gene has its own start and end point of the regions and of the whole trajectory. This means that the fitness values of the genes, if no collision is regarded, are independent from each other. Theoretically the multi-criterion optimization can be subdivided into six optimizations each for one gene, which can be performed one after the other. This serial optimization results in a great amount of time for the optimization. To avoid this, a special kind of selection should be introduced now.

As stated the fitness values of the genes are independent from each other. This offers the possibility of performing the three serial optimizations in one parallel. This means that the recombination, the mutation and the calculation of the fitness are performed as usual for all genes. Only the selection is slightly different. Not $F_{Individual}$ but the six F_{GeneX} are relevant for the selection. The individuals of generation $n+1$ are not the best individuals of generation n as usual. The individuals of generation $n+1$ are built by first copying the best paths of $Gene1$ of all individuals of generation n then the best paths of $Gene2$ and so on. The used individuals of generation n for $Gene1$ must not be the same as for $Gene2$ and so on. This is not a selection with biological background but a kind of manipulation in the case of the selection for multi-criterion optimizations.

6. Results

6.1 Test conditions

For the simulations a computer with two 733 MHz PIII processors with 256 MB RAM was used. The hard disk is out of interest because the whole optimization is performed in the RAM. The operating system is Windows NT 4.0.

The optimization takes place in the thread structure described in Fig. 3. As described, the threadhandler distributes the regions of the trajectory to the threads. Here the regions are optimized by performing a simple evolutionary strategy in the form of

```
1. initialisation of the start population
}
   3. recombination:          a $(\mu, \lambda)$-strategy is performed
   3. mutation:               to/from each gene is added/subtracted a random
                              number
   4. fitness calculation:    calculation of the fitness according to the equations
                              (1) and (3)
   5. selection:              the best individuals are copied to the next parent
                              population
}while (best fitness value $< F_{Individual,min}$)
```

For the sake of simplicity a trajectory with six regions is used. Each region has a length of 360 ms which corresponds to a number of chromosomes of ten for each region. The first three genes should be optimized from coordinate 100 to 200 or the other way round. The genes of the wrist and the hand are not taken into consideration for the calculation. Fig. 8 shows a part of the whole trajectory of $Gene1$. The values between the user defined points at 360 ms and 720 ms are generated randomly.

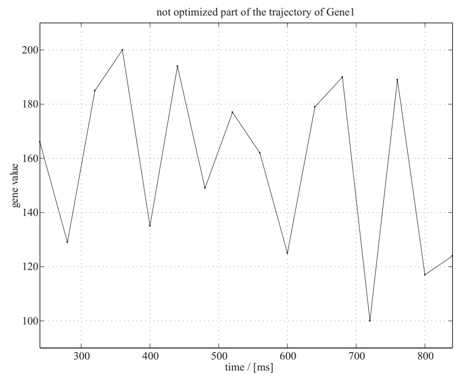

Fig. 7. Part of the trajectory

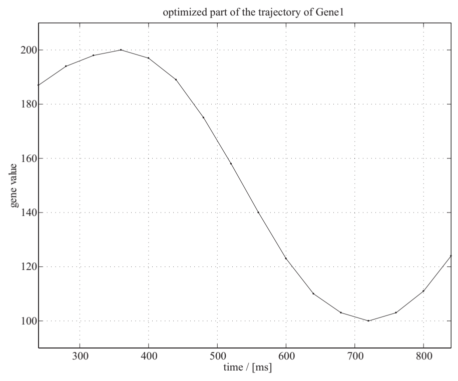

Fig. 8. Optimized part of the trajectory

Fig. 8 shows a part of the way of $Gene1$ of the optimized trajectory. The other two genes $Gene2$ and $Gene3$ have nearly the same way. This Pareto-optimal result can only be found because the optimization is performed as long as the fitness value $F^{w}_{Individual}$ of the best individual is smaller than the user defined value $F_{Individual,min}$ and $F_{Individual,min}$ is nearly three times the value of the best achievable fitness of one gene X. Here the value of

$F_{Individual,min}$ is 8. Only these settings will have Pareto-optimal results. The disadvantage of searching an individual with a fitness value smaller than $F_{Individual,min}$ is that, in case $F_{Individual,min}$ is chosen too large, one gene is dominating the other genes. If $F_{Individual,min}$ is chosen too small, no result can be found.

All genes are mutated by fifty percent. The *mutation value* that is added or subtracted is generated by

$$mutationvalue = signature \cdot percent \cdot mutval \tag{5}$$

$$signature = \begin{cases} 1, & 50\% \\ -1, & 50\% \end{cases} \tag{6}$$

$$percent = \begin{cases} 1, & percentage \\ 0, & 100\% - percentage \end{cases} \tag{7}$$

percentage is user defined. *mutval* is equally distributed and dependant on the fitness value of the individual [12]. The time needed for an optimization is dependant on this value. Fig. 9 shows the number of populations needed for an optimization dependant on the mutation value[1]. Here a single-objective optimization is compared to a multi-objective optimization without weighting with regard to the needed populations dependant on the mutation

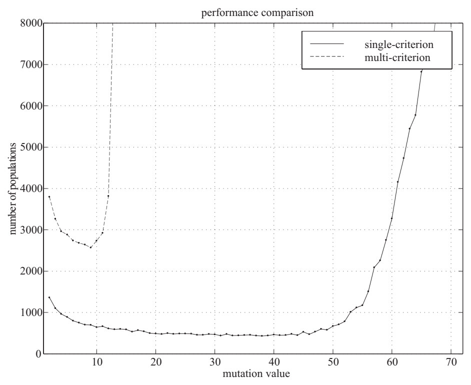

Fig. 9. Performance comparison.

[1]Since an evolutionary strategy is a stochastic process, the measured numbers of populations and the times given here and in the next sections are the mean values of 30 simulations using exactly the same set of parameters. If the mutation values are too great, a result will not be found at all or only sometimes. If no result can be found, a maximum of 5000 populations in each region is calculated.

value. It is obvious that possible amounts of the mutation value are much greater for a single-objective optimization than for a multi-objective optimization. The optimal value in the first case is around 38 with 430 needed populations and in the second one around 9 with 2578 needed populations.

6.2 Results of weighting

Table 1 shows the results of the three weighting techniques with the three different transition functions. The values represent the mean values of the needed numbers of populations calculated as described above. The values in brackets behind the population numbers are the mutation values that are used to calculate these results with.

Table 1. Performance comparison

	fixed weighting	adaptive weighting depending on one gene	adaptive weighting depending on all genes
linear	2573 (8)	2601 (7)	2630 (7)
logarithmic	2469 (8)	**2390** (7)	2622 (7)
squared	2602 (8)	2653 (8)	2632 (7)

With regard to Fig. 9 the amount of populations needed for a multi-objective optimization generally is much larger than in the case of a single-objective one. Additionally the mutation values can be chosen much greater in the single-objective case. For each method the squared transition function is the worst one. Because this function jolts small fitness values, the use of this function is leading to an ineffective superposition if one fitness value is much smaller than the others. The logarithmic transition function is the best solution because of the stretching effect on small values. In combination with the adaptive weighting method depending on one gene, this function is the best solution for weighting. It is eight percent faster than without weighting. The fixed weighting and the adaptive weighting method depending on one gene are better than the adaptive weighting method depending on all genes because in the last case the superposition is not effective enough. Here small fitness values of one gene are dominated by large fitness values of the other genes. The adaptive weighting method depending on one gene is more effective because the transition function is calculated adaptively with regard to the fitness value of the first population and not fixed.

In Table 1 only the performance with regard to the number of populations is compared. Another aspect is a comparison of the time needed for the calculation of one population. All three weighting techniques calculate the arrays needed for the fitness calculation in the first run of the optimization. This time can be neglected in comparison to the whole time needed for the optimization. The time for multiplying the weighting array by the fitness values of each gene can be neglected, too. The time needed for one population with or without weighting using a (5,25)-strategy is 0.23 ms.

6.3 Results of manipulation

Fig. 10 shows the performance of the manipulation in the case of the selection. In contrast to the weighting methods, the mutation value can be chosen greater but not as large as in the case of a single-objective optimization. The optimum can be found at a mutation value of 16 with 553 needed populations. The optimization is therefore more than four times faster than the best weighting method because on the one hand the search space is one third smaller and on the other hand it can be walked through much faster because the mutation value is greater.

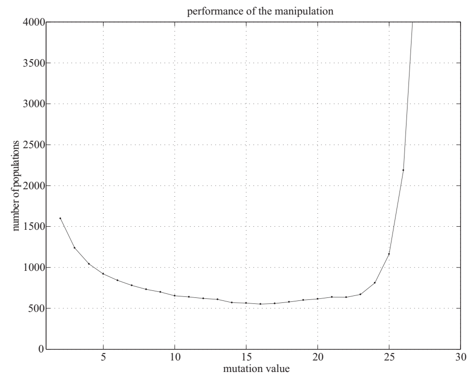

Fig. 10. Performance of manipulation

Section 5 suggests the possibility of performing three optimizations serially one after the other. According to section 6.1 around 1290 populations would be needed for three optimizations. If this number of populations is compared to the number of the multi-objective optimization with or without weighting, the last one is clearly worse because nearly twice the number of populations is needed. But in the case of the manipulation only 553 populations are needed for a whole multi-objective optimization, which is twice as fast as three optimizations serially one after the other.

In contrast to the algorithm with or without weighting, the time needed for the optimization with manipulation in the case of the selection is not the same because with or without weighting, the population is sorted only once in order to be able to copy the best individuals of the children population n into the next parent population $n + 1$. In the case of mutation each gene has to be sorted individually. This means that the time-consuming sorting algorithm should be calculated three times. Here the algorithm ShellSort [13] is used, which is a complex but fast sorting algorithm. The time needed for the calculation of one population increases to 0.24 ms. That is around four percent more than in the case of weighting.

7. Conclusion

This paper presents three methods of weighting with three different transition functions and one method with a manipulation in the case of the selection. According to section 6.2 the way of adaptively weighting the fitness values of each gene depending on one gene by using a logarithmic transition function is the fastest possibility of finding a solution with weighting measures and nearly eight percent faster than without weighting but two times slower than three serial single-objective optimizations. The technique of manipulation in the case of selection, which corresponds to a parallel optimization of the genes, is according to section 6.3 around four times faster than the weighting methods because the selection is more effective and the mutation value can be chosen around two times greater so that the search space is walked through much faster. This algorithm is two times faster than three serial single-objective optimizations. The fact that the manipulation can only be performed in the case of independancy of the three genes should be taken into consideration.

Acknowledgements

Thanks are due to Professor Wolfgang Weber for his interest in evolutionary algorithms, especially the field of multi-criterion optimization.

REFERENCES

1. Bäck, T.: *Evolutionary Algorithms in Theory and Practice - Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford University Press, 1996.
2. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, 1996.
3. Zitzler, E., Thiele, L.: *Comparison of Multiobjective Evolutionary Algorithms. Empirical Results.* TIK-Report No. 70, 1999.

4. Schwefel, H.-P.: *Numerische Optimierung von Computer-Medellen mittels der Evolutionsstrategie - mit einer vergleichenden Einführung in die Hill-Climbing- und Zufallsstrategie.* Birkhäuser Verlag, 1977.

5. Fonseca, C. M., Fleming P. J.: *An overview of evolutionary algorithms in multiobjective optimization.* Evolutionary Computation 3(1), 1995, pp. 1-16.

6. Rudolph, G.: *On a multi-objective evolutionary algorithm and its convergence to the Pareto set..* In Proc. of the 5th IEEE Conference on Evolutionary Computation, IEEE Press, Piscataway (NJ), 1998, pp. 511-516.

7. Laumanns, M., Rudolph, G., Schwefel H.-P.: *Approximating the Pareto Set: Concepts, Diversity Issues and Performance Assessment.* Technical Report CI-72/99, University of Dortmund. 1999.

8. Zitzler, E., Thiele, L.: *Multyobjective evolutionary algorithms: a comparatative case study and the strength Pareto approach.* IEEE Transactions on Evolutionary Computation, Vol. 3, No. 4, 1999.

9. Lin, M. C.: *Efficient Collision Detection for Animation and Robotics.* Department of Electrical Engineering and Computer Science, University of California, 1993.

10. Borgolte, U.: *Flexible, realzeitfähige Kollisionserkennung in Mehrroboter-Systemen.* Springer-Verlag, 1991.

11. Freund, E., Weber, H.: *Robotertechnologie I. Vorlesungsskript.*

12. Ortmann, M.: *Die Mutation bei Genetischen Algorithmen.* Frühjahrstagung der Studiengruppe für Elektronische Instrumentierung, 2000.

13. Shell, D. L.: *A high-speed sorting procedure.* Communications of the ACM, 1959.