PARALLEL CUDA IMPLEMENTATION OF THE DESIRABILITY-BASED SCALARIZATION APPROACH FOR MULTI-OBJECTIVE OPTIMIZATION PROBLEMS

Eren Akça
HAVELSAN A.Ş., Ankara, Turkey
erenakca88@gmail.com

Ökkes Tolga Altınöz
Department of Electrical and Electronics Engineering
Faculty of Engineering and Architecture, TED University, Ankara, Turkey
tolga.altinoz@tedu.edu.tr

Sadi Uçkun Emel
HAVELSAN A.Ş., Ankara, Turkey
semel@havelsan.com.tr

Asım Egemen Yilmaz
Electrical and Electronics Engineering Department
Ankara University, Ankara, Turkey
aeyilmaz@eng.ankara.edu.tr

Murat Efe
Electrical and Electronics Engineering Department
Ankara University, Ankara, Turkey
efe@eng.ankara.edu.tr

Tayfur Yaylagul
HAVELSAN A.Ş., Ankara, Turkey
tyaylagul@havelsan.com.tr

Abstract In this study, we present the results obtained for the parallel CUDA implementation of the previously proposed desirability-based scalarization approach for the solution of the multi-objective optimization problems. Our simulations show that compared to the sequential Java implementation, it is possible to find the same solutions (up to 16-time faster manner) by parallel CUDA implementation. We also try to outline our experiences of troubleshooting throughout the implementation as guidelines for upcoming researchers working in the same field.

Keywords: Aggregation, CUDA, Desirability functions, Genetic algorithm, GPGPU programming, Multi-objective optimization, Parallelization, Scalarization.

1. Introduction

The problem for determining the best possible solution set with respect to multiple objectives is referred to as a multi-objective (MO) optimization problem. There are many approaches for the solution of these kinds of problems. The most straightforward approach, the so-called "scalarization" or "aggregation" is nothing but to combine the objectives in order to obtain a single-objective [6]; and the most common method of this sort is the weighted sum [5, 7, 8].

The solution of the MO optimization problem is a set that contains trade-off solutions [11]. On the other hand, the problem instance defined via the scalarization technique yields a single solution. In order to find another trade-off solution, the parameters throughout the scalarization process shall be varied, and the resulting problem instance (which is different than the previous one) shall be solved. For the particular case in which the problem is bi-objective (with the objective functions f_1 and f_2) and the weighted-sum method is applied, the aggregated objective function is $w_1 f_1 + (1 - w_1) f_2$, where the weight w_1 is a real number between 0 and 1. By varying this weight, a new single-objective problem instance (and a new solution) is obtained. Obviously, for this simple case, the parameter setup of the "scalarization scheme" consists of only the weight w_1. For higher number of objectives and different scalarization techniques, the scalarization scheme might address a parameter set with multiple elements.

Scalarization techniques were popular in 1980s and early 1990s, prior to development of powerful multi-objective optimization algorithms such as the Non-Dominated Sorting Genetic Algorithm (NSGA) [10], NSGA-II [3] or Vector Evaluated Genetic Algorithm (VEGA) [9], etc. After the development of these powerful and successful multi-objective optimization algorithms, scalarization techniques were considered to be old-fashioned, and they were abandoned. By the time, especially af-

ter the evolution and rapid development of multi-core architectures in 2000s, researchers have started to reconsider and revisit the scalarization techniques since these techniques are usually suitable for parallelization when carefully implemented. In this study, with a similar motivation, we demonstrate how one of these techniques can be parallelized and implemented on the General Purpose Graphic Processing Units (GPGPUs) via the Compute Unified Device Architecture (CUDA) framework.

2. The Main Idea Beneath the Scalarization and the Weighted Sum Approach

As stated in the previous section, the main aim in a multi-objective optimization problem is to find a set of trade-off solutions. The optimality of a particular solution is determined via the definition of "domination" in the Pareto space. In Figure 1, the pictorial descriptions of the domination and the set of non-dominated solutions (i.e. the Pareto front) are given for a simple bi-objective problem.

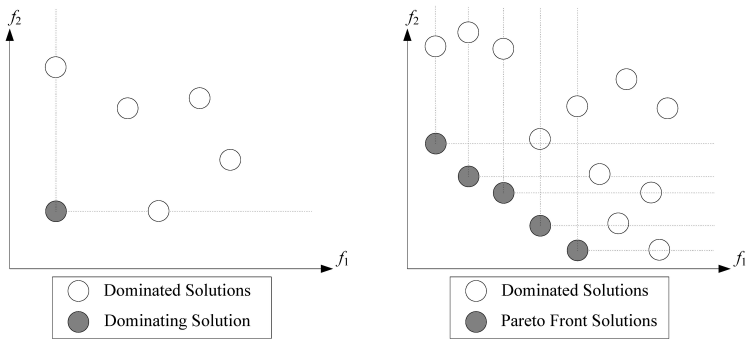

Figure 1. Pictorial descriptions of domination and the Pareto front.

In order to obtain a set of solutions for a bi-objective optimization problem via the weighted-sum method (for which the aggregated objective function $w_1 f_1 + (1 - w_1) f_2$ is constructed), the weight w_1 shall be varied between 0 and 1 in a systematical manner; and a new solution shall be found for each value of w_1. Even though this approach seems to be simple and well-working, it fails especially when the Pareto front is concave (totally or between two particular points in the Pareto space). In order to illustrate this, let us try to understand the main idea beneath the weighted-sum approach. As seen in Figure 2, the aggregated objective function $w_1 f_1 + (1 - w_1) f_2$ corresponds to a line in the Pareto space (particularly the $f_1 f_2$ plane). Throughout the optimization process, in which $c = w_1 f_1 + (1 - w_1) f_2$ is minimized, the corresponding

line is shifted by preserving its slope. Eventually, the solution obtained is nothing but the intersection of the corresponding line with the Pareto front curve (which is considered to be convex, for the time being) as seen in Figure 3.

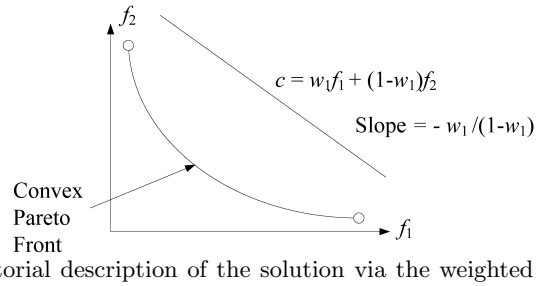

Figure 2. Pictorial description of the solution via the weighted sum approach for convex Pareto front

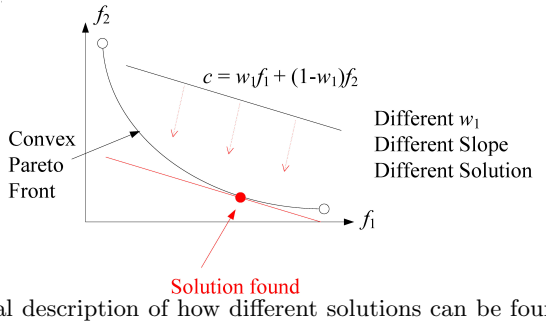

Figure 3. Pictorial description of how different solutions can be found by altering the weight in the weighted sum approach for convex Pareto front.

As seen in Figure 2, the slope of the line is determined by the weight, and altering this parameter will yield a different line. As seen in Figure 3, altering the weight would yield a different solution (in case the Pareto front is convex).

On the other hand, let us consider the case for which the Pareto front is concave between two points A and B as seen in Figure 4. In this case, the point B is found as a solution. In case the weight is altered as seen in Figure 5, the point A is found as an alternative solution. Unfortunately, in such a case, even if the weight is altered in its whole range, no points on the Pareto front other than A and B can be found by this approach [2]. Without a topological proof, this can be observed pictorially by intersecting different-slope lines with this concave Pareto front in Figures 4 and 5.

Since it is impossible to know whether the Pareto front is convex or concave in real life problems, the weighted-sum approach cannot be applied confidently. Hence, in this study we propose to apply the desirability-functions (by altering their shapes systematically) for scalarization as defined in [1]. The next section is devoted to description of this method.

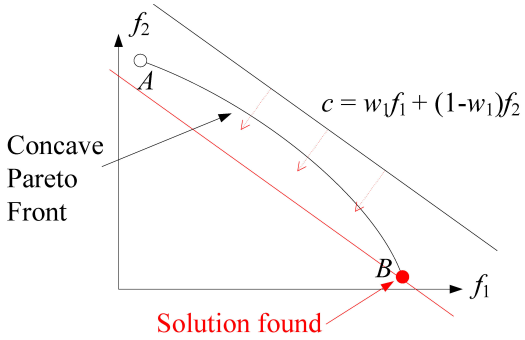

Figure 4. Pictorial description of the solution via the weighted sum approach for concave Pareto front (where the point B is found as the solution).

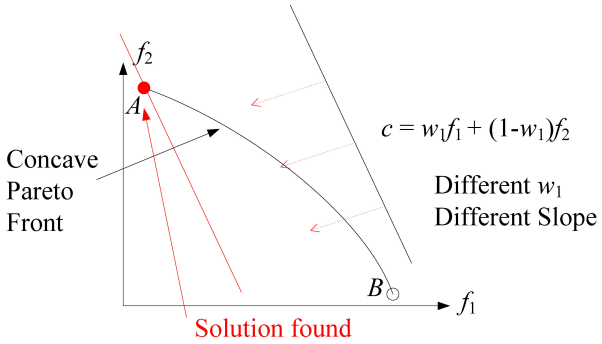

Figure 5. Pictorial description of the solution by altering the weight in the weighted sum approach for concave Pareto front (where the point A is found as the alternative solution).

3. Desirability Function-Based Scalarization

Previously in [1], an alternative scalarization method utilizing the so-called desirability functions was proposed. The concept of the desirability functions was first introduced by Harrington in 1965 for multi-objective industry quality control. After the proposition of the desirability function concept, Deringer and Suich [4] introduced different desirability function formulations. The main idea beneath the desirability functions is as follows:

- The desirability function is a mapping from the domain of real numbers to the range set $[0, 1]$.
- The domain of each desirability function is one of the objective functions; and it maps the values of the relevant objective function to the interval $[0, 1]$.

- Depending on the desire about minimization of each objective function (i.e. the minimum/maximum tolerable values), the relevant desirability function is constructed.
- The overall desirability value is defined as the geometric mean of all desirability functions; this value is to be maximized.

Particularly, for a bi-objective optimization problem in which the functions f_1 and f_2 are to be minimized, the relevant desirability functions $d_1(f_1)$ and $d_2(f_2)$ can be defined as in Figure 6. The desirability functions are not necessarily defined to be linear; certainly, non-linear definitions shall also be made as described in [1]. Throughout this study, we prefer the linear desirability functions.

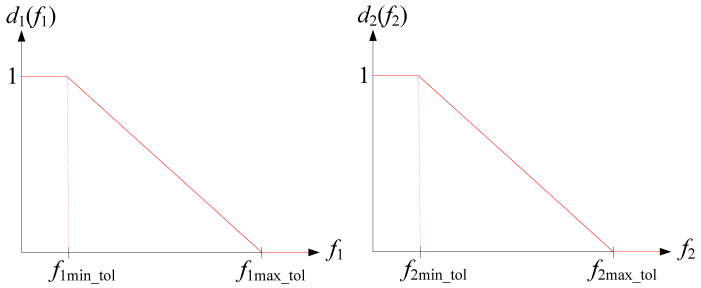

Figure 6. The linear desirability functions constructed for the bi-objective optimization problem.

In [1], a method for extraction of the Pareto front was proposed by altering the shapes of the desirability functions in a systematical manner. Particularly, by:

- Fixing the parameters $f_{1_{\max\_tol}}$ and $f_{2_{\max\_tol}}$ seen in Figure 6 at infinity, and
- Varying the parameters $f_{1_{\min\_tol}}$ and $f_{2_{\min\_tol}}$ systematically,

it is possible to find the Pareto front regardless of its convexity or concavity. This claim can be illustrated for the bi-objective case as follows: as seen in Figure 7, the parameters $f_{1_{\min\_tol}}$ and $f_{2_{\min\_tol}}$ determine the sector which is traced throughout the solution. The obtained solution corresponds to a point for which the geometric mean of the two desirability values. As seen in Figure 8, even in the case of concave Pareto front, the solution can be found without loss of generality. In other words, unlike the weighted-sum approach, the method proposed in [4] does not suffer from the concave Pareto fronts.

In [1], the applicability and the efficiency of the proposed scalarization approach was demonstrated via some multi-objective benchmark functions. Each single-objective problem (i.e. the scalarization scheme) was

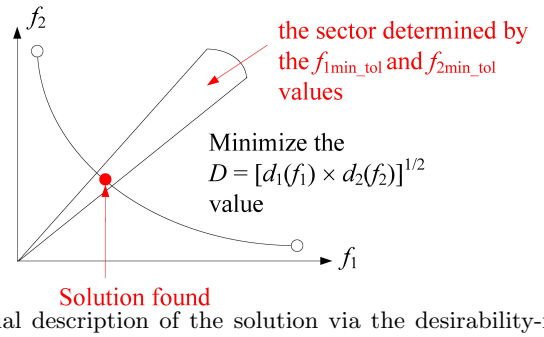

Figure 7. Pictorial description of the solution via the desirability-function based approach for convex Pareto front.

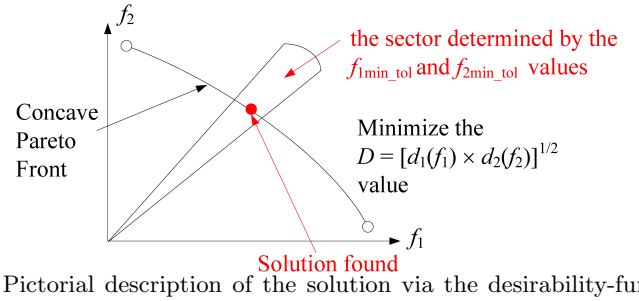

Figure 8. Pictorial description of the solution via the desirability-function based approach for concave Pareto front.

solved with Particle Swarm Optimization. Despite no explicit demonstration or proof, it was claimed that:

- There were no limitations about the usage of Particle Swarm Optimization; i.e. any other heuristic algorithm could be incorporated and implemented.
- The proposed method can be easily parallelizable.

In this study, we demonstrate the validity of these claims by incorporating the Genetic Algorithm in the proposed method, and performing a parallel implementation on GPGPUs via the CUDA framework. The next section is devoted to the implementation details.

4. Parallel CUDA Implementation of the Desirability Function-Based Scalarization

The main idea of our parallel implementation throughout this study is illustrated in Figure 9. Each scalarization scheme is handled in a separate thread; after the relevant solutions are obtained, they are gathered in a centralized manner to constitute the Pareto front from which the human decision maker picks a solution according to his/her needs. This

approach ensures that the number of solutions found that can be found in parallel is limited by the capability of the GPGPU card used. If the problem is a mission-planning problem as in our case, the two objectives might be "minimizing the overall mission completion time" and "minimizing the overall mission risk"; and the Pareto front is a plethora of various alternative mission plans with different overall completion time and risk values. In this case, the planner (or the commander) will pick up a solution (i.e. a mission plan to be executed) by using his/her own initiative. Regardless of the choice, it will be certain that the particular picked solution will be non-dominated.

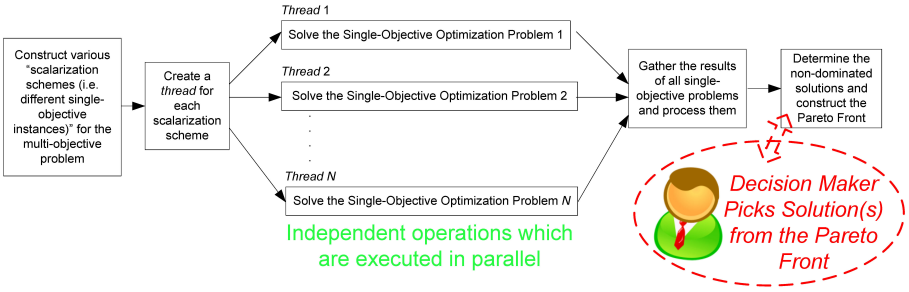

Figure 9. Pictorial description of the parallel CUDA implementation of the desirability-function based approach.

As stated before, we implemented an elitist Genetic Algorithm for verification of the aforementioned claims. The parallel CUDA implementation was compared to the sequential Java implementation in the environment seen in Table 1. The Genetic Algorithm parameters used throughout the simulations are given in Table 2.

Table 1. The environment in which the simulations are run.

Device and Global Memory	Quadro K5000 and 4GB
CUDA Cores	1536
GPU and Memory Clock Rate	706 MHz and 2700 MHz
Memory Bus Width	256-bit
Maximum Number of Threads per MP and Block	2048 and 1024
CUDA Driver and Capability	5.5 and 3.0

Table 2. Genetic Algorithm parameters throughout the simulations.

Chromosome size	16-bit
Population size	100
Number of generations (iterations)	100
Elitism Rate	0.2
Mutation Rate	0.1
Crossover Rate	0.9

It was seen that both implementations (sequential Java and parallel CUDA) were able to find the same solutions but in different elapsed times. As seen in Figure 10, if the number of Pareto front solutions increase, the advantage of the parallel CUDA appears dramatically.

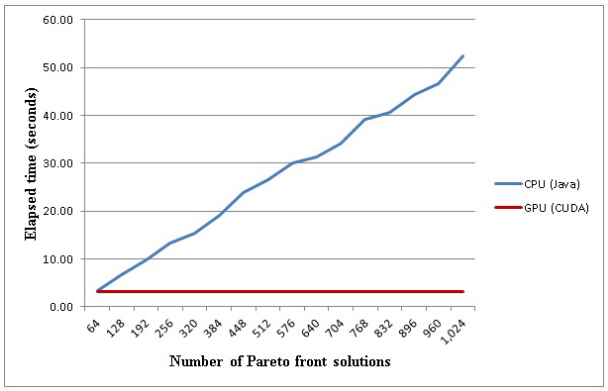

Figure 10. Comparison of the sequential Java and the parallel CUDA implementations.

These results show how efficient the parallel CUDA implementation will be in case numerous Pareto solutions are required by the decision maker in a multi-objective optimization problem.

5. Lessons Learned and Future Work

Throughout the implementation, we have experienced and observed the following:

- Memory allocations shall be made in advance in a bulk manner, since such operations deteriorate parallelism in case they are made

one by one. Bulk memory allocation and pointer assignment for usage of each thread is possible in our case, since it is quite straightforward to determine how much memory will be required by the Genetic Algorithm (since the parameters listed in Table 2 are definite).

- Random number generation is another issue. Various alternative approaches can be preferred:
  - To generate each random number at the CPU when required and to transfer it to the GPU: This is the least efficient and the slowest approach.
  - To generate all random numbers at the CPU prior to the solution and to transfer them to the GPU: This is better than the previous one.
  - To generate each random number directly at the GPU with no CPU-GPU communication need: This is the most efficient and the fastest approach. GPU generated random numbers demonstrate sufficient randomness features.
- Windows operating systems assigns a default 2-second time-out for the processes initiated at the peripheral devices. As seen in Figure 10, for our problem, the duration of the execution of each CUDA thread was about 3 seconds (which exceeds the 2-second time-out), causing the relevant process to be killed by the Windows operating system prior to completion. This problem was resolved by brute-force editing (and setting it to higher values) of the relevant key value in the registry.

The results shown in Figure 10 were obtained without any parallel/concurrent implementation of the single-objective optimization problem. As seen in Figure 11, some steps in the Genetic Algorithm can be executed in a parallel or concurrent manner.

More specifically, the computation of the objective and the fitness functions for the individuals in the Genetic Algorithm population can be performed concurrently on GPGPUs, in case the stream mechanism in CUDA is utilized as seen in Figure 12. The impact of such a modification in the implementation might not be drastic for the benchmark problems since they usually consist of computationally cheap functions. But in case of computationally expensive functions as in our mission-planning problem, the advantage of utilization of the streams is expected to be quite dramatic. As a future work, our aim is to perform the utilization of the streams as well as the shared memories in the GPGPUs, which are not being used in the current implementation.

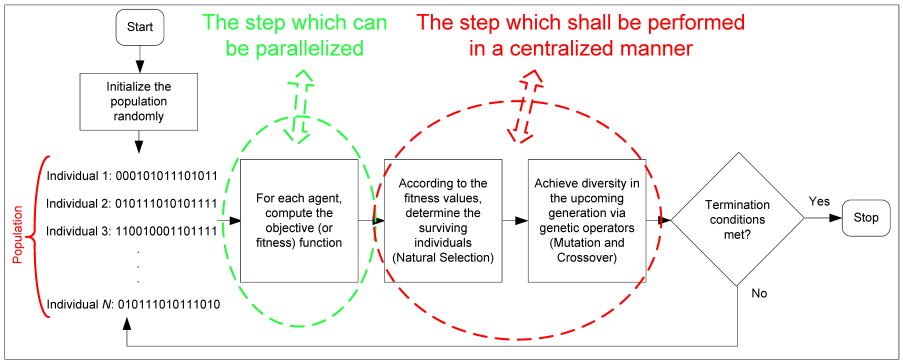

Figure 11. The steps of the Genetic Algorithm with indications of parallelization.

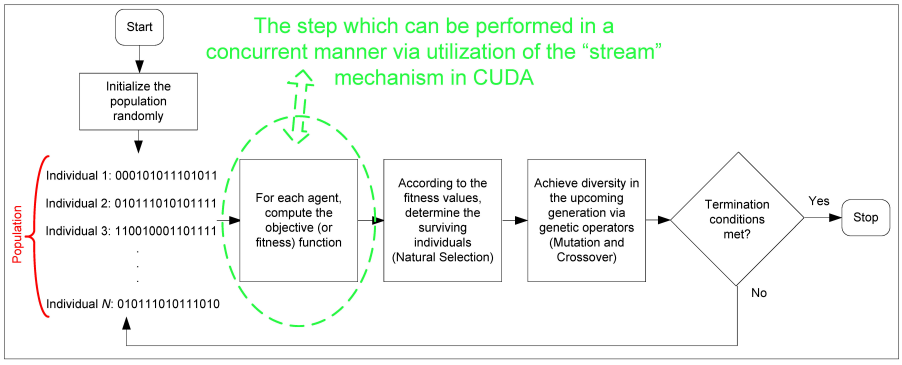

Figure 12. The step of the Genetic Algorithm which can be executed concurrently via utilization of the stream mechanism in CUDA.

In conclusion, in this study we have demonstrated that it is possible to achieve up to 16 times faster GPU implementations for the multi-objective problems via a careful and intelligent CUDA implementation. Further improvements in the implementation shall be made in the near future. Moreover, we have verified the claims in [4] and demonstrated the efficiency of the proposed approach.

Acknowledgement

This study was made possible by grants from the Turkish Ministry of Science, Industry and Technology (Industrial Thesis - San-Tez Programme and HAVELSAN; with Grant Nr. 01568.STZ.2012-2) and the Scientific and Technological Research Council of Turkey – TUBITAK (with Grant Nr. 112E168). The authors would like to express their gratitude to these institutions for their support.

References

[1] O. T. Altinoz, A. E. Yilmaz, and G. Ciuprina. A Multiobjective Optimization Approach via Systematical Modification of the Desirability Function Shapes. In *Proc. 8th International Symposium on Advanced Topics in Electrical Engineering*, pages 23–25, 2013.

[2] R. S. Burachik, C. Y. Kaya, and M. M. Rizvi. A new scalarization technique to approximate Pareto fronts of problems with disconnected feasible sets. *J. Optimiz. Theory App.*, appeared online, June 2013, DOI 10.1007/s10957-013-0346-0.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE T. Evolut. Comput.*, 2:182-197, 2002.

[4] G. Derringer and R. Suich. Simultaneous optimization of several response variables. *J. Qual. Technol.*, 12:214–219, 1980.

[5] J. Keski and R. Silvenneinen. Norm methods and partial weighting in multicriteria optimization of structures. *Int. J. Num. Meth. Eng.*, 24:1101–1121, 1987.

[6] R. Marler and S. Arora. Transformation methods for multiobjective optimization. *Eng. Optimiz.*, 37:551–569, 2009.

[7] R. Marler and S. Arora. The weighted sum method for multi-objective optimization: new insights. *Struct. Optimization*, 41:853–862, 2010.

[8] S. F. P. Saramago and V. Steffen, Jr. Optimization of trajectory planning of robot manipulators taking into account the dynamic of the system. *Mech. Mach. Theory*, 33:883–894, 1998.

[9] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proc. International Conference on Genetic Algorithm and their Applications*, 1985.

[10] N. Srinivas and K. Deb. Multi-Objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation*, 2:221-248, 1995.

[11] H. Trautmann and J. Mehmen. Preference-based pareto optimization in certain and noisy environments. *Eng. Optimiz.*, 41:23–38, 2009.