GA with a New Multi-Parent Crossover for Solving IEEE-CEC2011 Competition Problems

Saber M. Elsayed, Ruhul A. Sarker and Daryl L. Essam
School of Engineering and Information Technology
University of New South Wales at Australian Defence Force Academy
Canberra 2600, Australia
(E-mails: saber.elsayed@student.adfa.edu.au, r.sarker@adfa.edu.au, d.essam@adfa.edu.au)

***Abstract*— Over the last two decades, many Genetic Algorithms have been introduced for solving optimization problems. Due to the variability of the characteristics in different optimization problems, none of these algorithms performs consistently over a range of problems. In this paper, we introduce a GA with a new multi-parent crossover for solving a variety of optimization problems. The proposed algorithm also uses both a randomized operator as mutation and maintains an archive of good solutions. The algorithm has been applied to solve the set of real world problems proposed for the IEEE-CEC2011 evolutionary algorithm competition.**

***Index Terms*—Numerical optimization, genetic algorithms**

I. Introduction

EVOLUTIONARY algorithms (EAs) have a long history of successfully solving optimization problems. The Evolutionary algorithms (EAs) family contains a wide range of algorithms that have been used to solve optimization problems, such as the genetic algorithm (GA) [1], particle swarm optimization (PSO) [2], differential evolution (DE) [3], evolutionary strategies (ES) [4], and evolutionary programming [5].

The Genetic Algorithm (GA) is widely used for solving many practical optimization problems [1]. GA has the ability to deal with both continuous and discrete variables, is well suited for parallel computing, and can deal with the optimization of extremely complex fitness landscapes. GA has also been successfully applied to noisy problems [6]. However, GA is known to be slow [7], and sometimes GA cannot convergence to the global optimal [8].

Over the last few decades, although many GAs have been introduced for solving optimization problems, both constrained and unconstrained, their performance varies when considered a wide range of problems. In some cases, they do not even perform better than other EAs, such as differential evolution and particle swarm algorithms [6] [9].

In this research, our objective is to improve the performance of genetic algorithms by introducing a new crossover with a randomized operator to replace mutation. The proposed crossover uses three parents to generate three new offspring, two of which are to help exploitation, while the third offspring is for promoting exploration. The randomized operator is to help to escape from local optima and premature convergence.

The idea of multi-parent crossover is not new in the literature. There are different crossovers suitable for continuous problems, such as unimodal distribution crossover (UNDX) [10], simplex crossover (SPX) [11], parent centric crossover (PCX) [12] and triangular crossover (TC) [13]. UNDX uses multiple parents and creates offspring solutions around the center of mass of these parents. A small probability is assigned to solutions away from the center of mass. UNDX has shown excellent performance for highly epistasis problems[10]. On the other hand, there are some areas where the UNDX cannot generate offspring with a given initial population, such as when the population size is small relative to the search space. UNDX also has difficulty to find an optimal solution near the boundaries [10]. Simplex crossover (SPX) is a multi-parent recombination operator for real-coded GAs. The SPX operator assigns a uniform probability distribution for creating offspring in a restricted search space around the region marked by the parents. SPX uses the property of a simplex in the search space. SPX has a balance between exploration and exploitation. The simplex crossover works well on functions having multimodality and/or epistasis with a medium number of parents, such as three parents on a low dimension function and four parents on a high dimension function [11]. However, SPX fails on functions that consist of tightly linked sub-functions [11]. PCX allows a large probability of creating a solution near each parent, rather than near the centre of the parents. PCX is a self adaptive type approach, where a new solution vector keeps moving towards the optimum [12]; however GA with PCX has difficulty in solving separable multimodal problems, whereas DE can solve them successfully [14]. TC uses three parents (where two parents must be feasible and one infeasible) and generates three random numbers to generate a new set of three offspring, where each offspring is generated as a linear combination of those three parents. TC works well where the optimal solution lies on the boundary of the feasible region of problems that have a single bounded feasible region in the continuous domain [13]. It is important to mention here that our proposed crossover neither uses a mean-centric probability distribution, such as UNDX and SPX, nor uses a parent-centric approach, such as PCX.

The proposed genetic algorithm is named as GA-MPC. In GA-MPC, first, an initial population with size *PS* is generated randomly, and the best *m* individuals are stored in an archive

pool, based on the objective function and/or constraint violation. A tournament selection with size *tc* is performed to select good solutions which are stored into the selection pool for performing crossover. In the crossover operations, with crossover rate (*cr*), three individuals are used to generate three new offspring. A randomized operation is applied for each new offspring with a probability (*p*). The generated offspring are merged with the individuals of the archive pool to select the best *PS* individuals, based on the objective function and/or constraint violations, to build the new population for the next generation, and concurrently update the archive pool. The details for our algorithm are described in a later section.

The algorithm was applied to the group of real world optimization problems that have been proposed for the IEEE-CEC2011 evolutionary algorithm competition [15].

This paper is organized as follows: after the introduction, section 2 presents the proposed crossover and the randomized operator. Section 3 describes the design of our algorithm with a description of the constraint handling technique that has been used in this research. The experimental results, and the analysis of those results, are presented in section 4. Finally, the conclusions are given in section 5.

II. THE PROPOSED CROSSOVER AND THE RANDOMIZED OPERATOR

In this section, we describe both the proposed crossover and the randomized operator, with numerical examples to show how they work.

A. The proposed Crossover

In GA, the crossover should have the ability for exploiting information about the search space in generating new offspring. The distribution of offspring should neither be extremely narrow nor extremely wide in comparison to their parents. If the generated offspring are distributed extremely more narrow than the parents, they may lose diversity and converge prematurely. In contrast, if the offspring are distributed extremely wide, they may be too diverse and take too long in converging to optimality [10]. So it should be appropriate to generate offspring that satisfy a balance of exploration and exploitation.

Based on the above principle, we propose the following multi-parent crossover, named here as (MPC). The idea behind MPC comes from the heuristic crossover [16] and the mutation operator in DE [4]. In this research we deal with real-values encoding.

The steps of MPC are:

1- Select three parents based on a selection rule.
2- If one of the selected individuals is the same as another, then replace it with a random individual from the selection pool.
3- Rank these three individuals from the best (x_1) to the worst (x_3), based on their fitness functions and/or constraint violations.
4- Generate a random number β that follows a normal distribution with mean value μ and standard deviation σ.
5- Generate three offspring (o_i):

$$o_1 = x_1 + \beta \times (x_2 - x_3) \quad (1)$$
$$o_2 = x_2 + \beta \times (x_3 - x_1) \quad (2)$$
$$o_3 = x_3 + \beta \times (x_1 - x_2) \quad (3)$$

In equation 1, the worst individual x_3 is subtracted from x_2, then the difference is multiplied by β and then the total is added to the best individual x_1. This allows o_1 to be generated in the direction of a better part of the search space. The same situation applies in equation 3, $f(x_1) \leq f(x_2) \leq f(x_3)$ this allows o_3 to be generated to a better point. In equation 2, we subtract the best point x_1 from an inferior point x_3, this ensures, in almost cases, that $f(o_2) \leq f(x_2)$, and therefore it will extend to a more diverse point. The reason for using $x_3 - x_1$ instead of $x_1 - x_3$ in equation 3 is that moving towards the best individual may let the algorithm to become stuck in a local minimum.

To demonstrate how the proposed crossover works, we use the following simple example.

$$min\ f(x) = x_1^2 + x_2^2$$

subject to:

$$x_1^2 + x_2^2 \leq 15.$$
$$0 \leq x_1, x_2 \leq 100. \quad (4)$$

Imagine that β=0.5, and that the selected three individuals that will follow the crossover are $x_1 = \{3,3\}$, $x_2 = \{4,4\}$ and $x_3 = (6,3)$. $f(x_1) = 18$, $f(x_2) = 32$ and $f(x_3) = 45$.

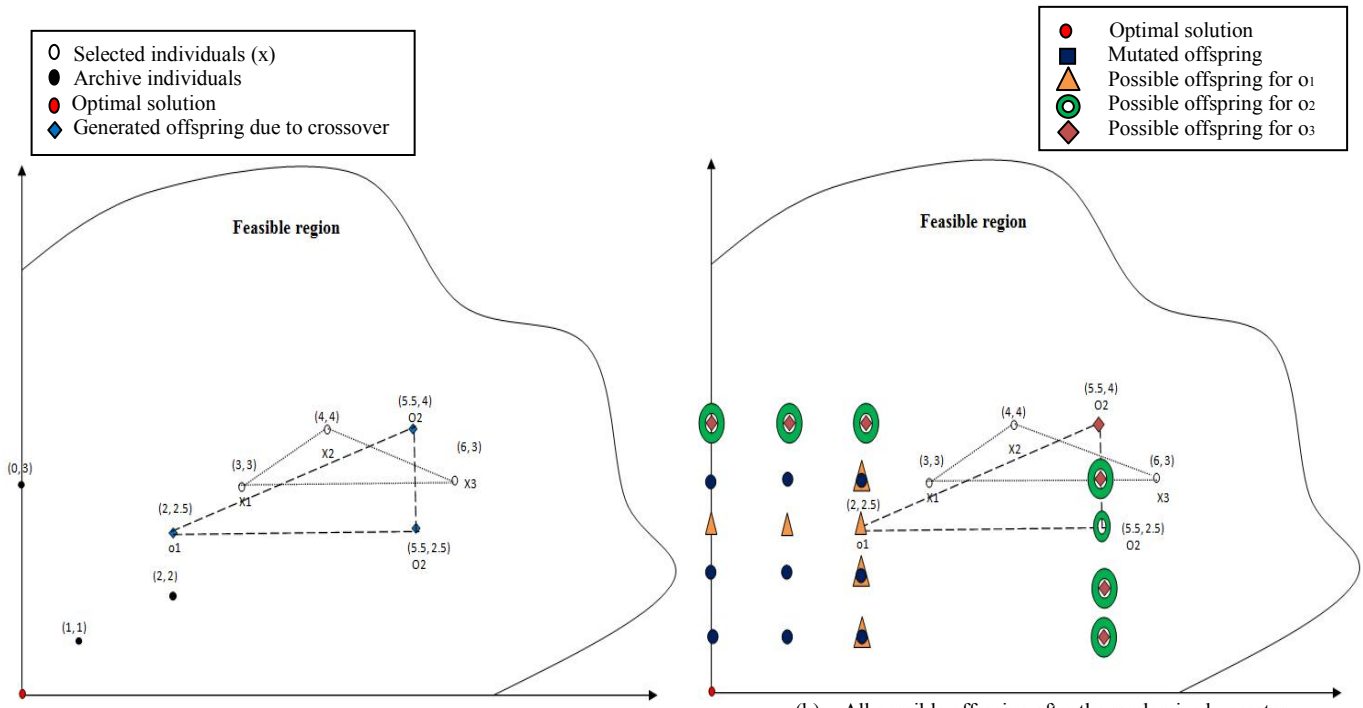
Then the generated offspring will be: $o_1 = \{2, 2.5\}$, $o_2 = \{5.5, 4\}$ and $o_3 = \{5.5, 2.5\}$, $f(o_1) = 10.25$, $f(o_2) = 46.25$ and $f(o_3) = 36.5$. This means that the objective functions of the best and the worst individuals are improved, while the median individual explores new areas. Figure 1.a shows the generated offspring.

B. A Randomized Operator

In this paper, we also propose a randomized operator to diversify the generated offspring. While applying the randomized operator, two points should be considered: the proportion of population undergoing such operation, and the strength of the operator.

In the randomized operator, to generate the final offspring, we use information regarding the generated offspring o_i and information about the archive pool (A) that contains the best m individuals in the current generation. With a probability p, each o'^j_i in the final offspring (i), where $j \in \{1,2,\dots D\}$ and D is the vector dimension, is replaced with x^j_{arch}, where $arch \in \{1,2,\dots,m\}$. Note that to generate the final o'_i, we may use different values for $arch$, this is because we do not want use information from a single vector as that may lead to becoming stuck in a local minima.

To show how the randomized operator works, for the previous example, imagine that we have three individuals in an archive pool: $A_1 = \{1,1\}$, $A_2 = \{2,2\}$ and $A_3 = \{0,3\}$. All



(a) The generated three offspring due to the proposed crossover.

(b) All possible offspring after the randomized operator.

Fig1. A numerical example shows the possible offspring due to the proposed crossover (in left) and the randomized operator (in the right).

possible offspring due to the randomized operator are shown in figure 1.b, in which each offspring o_i can be replaced with one of l possible offspring, where $l = (1 + m) \times D^2$. At the same time, for each three offspring, they may generate mutual offspring. So each three offspring can be replaced by three individuals out of $(3l - 2m^2)$ individuals. In the previous example, m=3, D=2. So for each offspring, it can be replaced by one of a possible 16 offspring, and each consecutive offspring can be replaced by three individuals out of 30 possible offspring. As can be seen from figure 1.b, all points are able to visit good areas, and o'_2cannot be worst than o_2.

III. GA WITH MULTI-PARENT CROSSOVER (GA-MPC)

In this section, we describe the proposed genetic algorithm, and the constraint handling technique.

A. GA-MPC

In this research, our intention is to develop a modified genetic algorithm, consisting of a new crossover operator with a randomized operator with an archive. We named the proposed algorithm as GA-MPC.

In GA-MPC, first an initial population is generated randomly, with size *PS*. Then an archive pool is filled with the best m individuals (based on their constrained violations and/or fitness function). Then a tournament selection procedure with size *tc* takes place, from which the best individual is chosen and saved in the selection pool. For the crossover operation, with a crossover rate (*cr*), for each three consecutive individuals in the selection pool, three offspring are generated as described before. After we generate all offspring, we perform a randomized operator, with a probability *p*, to escape from any local minima. After that we merge the individuals from the archive pool with all of the offspring, and the best *PS* individuals are selected as a new population for the next generation. Also to ensure more diversity, if any individual in the population is the same to another one, then one of them is shifted within the boundary with $N\ (0.5 \times u, 0.25 * u)$, where $u \in [0,1]$. Table I shows the steps of GA-MPC.

B. Constraint handling

The penalty part in the objective functions is handled as follows:

$$\text{PENALTY} = max(0, \text{PENALTY} - tol), \qquad (5)$$

where tol is equal to the penalty value for the top θ-th individual and θ = (0.20 × *PS*). Note that tol is fixed for the first n_1 generations, and then it is adaptively reduced to 0 (using equation 6) during the next n_2 generations.

$$tol = \begin{cases} tol, & if\ t \le n1 \\ tol - tol \times \left(\frac{t-n1}{n2}\right), & if\ t > n1 \end{cases} \qquad (6)$$

where t is the generation number, $n1$ is equal to 15% of the maximum generations (T_{max}), n_2 is equal to 35% of the T_{max}. This means that after 50% of T_{max}, tol is equal to zero. The idea is quite similar to the ε-Constraint [17].

IV. EXPERIMENTAL RESULTS AND ANALYSIS

The performance of GA-MPC has been compared with the state of the art algorithms. The algorithm has been coded using

TABLE I

GA WITH MULTI-PARENT CROSSOVER (GA-MPC)

STEP 1: In generation $t = 0$, generate an initial random population of size PS. The variables in each individual (i) must be within the range as shown below:

$$x_{i,j} = x_{i,j,min} + u \times (x_{ij,max} - x_{ij,,min})$$

where $x_{i,j,min}, x_{i,j,max}$ are the lower and upper bound for decision variable x_j, and u is a random number, $u \in [0,1]$.

STEP 2: Sort all individuals based on their constraint violations and/or objective function, and save the best m individuals in the archive pool (A).

STEP 3: Apply a tournament selection with size TC (randomly 2 or 3), and fill the selection pool.

STEP 4: For each three consecutive individuals, If $u \in [0,1] < cr$

i) Rank these three individuals from $f(x_i) \leq f(x_{i+1}) \leq f(x_{i+2})$

ii) If one of the selected individuals is the same to another, then replace one of them with a random individual from the selection pool.

iii) Calculate $\beta = N(\mu, \sigma)$

iv) Generate three offspring (o_i):

$$o_1 = x_1 + \beta \times (x_2 - x_3)$$
$$o_2 = x_2 + \beta \times (x_3 - x_1)$$
$$o_3 = x_3 + \beta \times (x_1 - x_2)$$

STEP 5: For each o'^{j}_{i}, generate a random number u $\in [0,1]$. If u $\in [0,1] < p$, then $o'^{j}_{i} = x^{j}_{arch}$, where $arch \in [1, m]$.

STEP 6: If there is any duplicate individual, then

$$x_{i,j} = x_{i,j} + N(0.5 \times u, 0.25 * u)$$, where $u \in [0,1]$

STEP 7: Stop if the termination criterion is met; **else** go to **STEP 2**, and set $t = t + 1$.

Matlab, and has been run on a PC with a 3.5 GHz Core 2 Duo processor, 3G RAM, and windows XP. As indicated earlier, we have solved the set of real world test instances introduced in CEC2011. The details of these problems are shown in [15].

We set the parameters: PS = 90, β parameter can affect the performance of our algorithm, we have used different values in our initial experiments, in other research study, and found that have $\beta = N(0.7, 0.1)$ for all test problems except problem 12, where we set $\beta = N(0.5, 0.3)$, The reason to generate β from a normal distribution is normal distribution is very tractable analytically, that is a large number of results involving this distribution can be derived in explicit form, the normal distribution arises as the outcome of the central limit theorem, which states that under mild conditions the sum of a large number of random variables is distributed approximately normally. Finally, the "bell" shape of the normal distribution make it a convenient choice for modeling a large variety of random variables encountered in practice. $cr = 1$, $p = 0.1$, $T_{max} = 1667$, the tournament pool size is selected randomly as 2 or 3. For the number of individuals that are selected to fill the archive pool m =50% of PS. 25 independent runs were performed for each test problem, and the stopping criterion was to run for up to 150,000 fitness evaluations (FEs). The detailed results (best, median, average and standard deviation (*St. d*)), for 50,000, 100,000, and 150,000 FEs, are presented in Appendixes A-D.

V. CONCLUSION

During the last few decades, many evolutionary algorithms have been introduced to solve constrained optimization problems. Genetic algorithm has often been inferior to other EAs, such as DE and PSO. In this paper, we have shown that the efficiency of GA can be improved by adopting a new crossover operator with a randomized operator.

In the proposed algorithm, an initial population was generated randomly and the best m individuals were stored in an archive pool. A tournament selection with size tc = 2 or 3 was initiated to select the best points to act as parents. Then the proposed crossover and its randomized operator were used to generate new offspring. In the proposed crossover, each three consecutive parents were able to generate three offspring. These offspring were then randomized with archive pool individuals. The generated offspring were then merged with the archive pool individual, and the best PS individuals were used as a new population for the next generation.

GA-MPC has been used to solve the set of real world test problems presented in the CEC2011

Elsewhere we have solved another set of constrained problems. The results were found to be both robust and of high quality.

For future work, we need to provide more theoretical analysis of our proposed algorithm. We also need to perform more detailed analysis about the effect of each parameter.

REFERENCES

[1] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," *Addison-Wesley Publishing Corporation*, Inc, Reading, MA, 1989.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimisation," *In Proceedings of the IEEE International Conference on Neural Network*, vol. 4, 1995, pp. 1942–1948.

[3] R. Storn and K. V. Price, "Differential evolution–A simple and efficient adaptive scheme for global optimization over continuous spaces," *Technical Report*, Computer Science Institute, University of California at Berkeley, USA, TR-95-012, 1995.

[4] I. Rechenberg, "Evolutions strategie: Optimierung Technischer Systeme nach Prinzipien der biologischen Evolution," *Fromman-Holzboog, Stuttgart,* 1973.

[5] L. J. Fogel, A. J. Owens and M. J. Walsh, "Artificial Intelligence Through Simulated Evolution," *John Wiley & Sons,* New York, 1966.

[6] V. Jakob and T. René, "A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems," *In Proceedings IEEE Congress on Evolutionary Computation* (CEC2004), vol. 3, 2004. pp. 1980-1987.

[7] D. Abramson and J. Abela, "A parallel genetic algorithm for solving the school timetabling problem*," In Proceedings of the Fifteenth Australian Computer Science Conference (ACSC-15)*, vol. 14, 1992, pp. 1–11

[8] K. Deb, "Multi-objective genetic algorithms: problem difficulties and construction of test problems," *Evol. Comput*., vol. 7, 1999, pp. 205–230.

[9] http://www3.ntu.edu.sg/home/epnsugan/

[10] I. Ono and S. Kobayashi, "A Real-coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover," *In Proc. 7th ICGA*, 1997, pp. 246-253.

[11] S. Tsutsui, M. Yamamura, and T. Higuchi, "Multi-parent Recombination with Simplex Crossover in Real Coded Genetic Algorithms," *In Proc. Genetic Evolutionary Computation Conf. (GECCO'99)*, Jul. 1999, pp. 657–664.

[12] K. Deb, A. Anand and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter evolution", *In IEEE Trans. Evol. Comput.* vol. 10, no. 4, 2002, pp. 371–395.

[13] E. Z. Elfeky, R. Sarker and D. Essam, "Analyzing the simple ranking and selection process for constrained evolutionary optimization," *journal of Computer Science and Technology*, vol. 23, no. 1, 2008, pp. 19-34.

[14] J. Rönkkönen, "Continuous Multimodal Global Optimization with Differential Evolution-Based Methods," *Thesis for the degree of Doctor of Science*, Lappeenranta University of Technology, Lappeenranta, Finland, 2009, ISBN 978-952-214-851-3

[15] S. Das and P. N. Suganthan, "Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems," *Tech. Rep.,* December 2010.

[16] A.H. Wright, "Genetic algorithms for real parameter optimization," *G. J. E. Rawlins (Ed.), Foundations of Genetic Algorithms I*, Morgan Kaufmann, San Mateo, 1991, pp. 205–218.

[17] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites," *in Proc. of the 2006 IEEE Congress on Evolutionary Computation*, 2006, pp. 308–315.

APPENDIX A

THE FUNCTION VALUES ACHIEVED VIA GA-MPC FOR TEST PROBLEMS (T01: T06)

FEs		T01 FM	T02 L-J	T03BCB	T04 STR	T05 Si(B)	T06 Si(C)
5×10^4	Best	0.000000E+00	-2.589442E+01	1.151489E-05	1.3770762E+01	-3.404882E+01	-2.276896E+01
	Median	0.000000E+00	-1.082457E+01	1.151489E-05	1.3855311E+01	-1.999404E+01	-1.418163E+01
	Worst	0.000000E+00	-6.569760E+00	1.151489E-05	1.9530755E+01	-1.741336E+01	-1.054845E+01
	Mean	0.000000E+00	-1.225905E+01	1.151489E-05	1.4369230E+01	-2.133214E+01	-1.434737E+01
	St. d.	0.000000E+00	4.810058E+00	0.000000E+00	1.1815998E+00	4.209240E+00	2.417000E+00
1×10^5	Best	0.000000E+00	-2.842253E+01	1.151489E-05	1.3770762E+01	-3.684504E+01	-2.916501E+01
	Median	0.000000E+00	-2.744630E+01	1.151489E-05	1.3770762E+01	-3.416425E+01	-2.742739E+01
	Worst	0.000000E+00	-1.267310E+01	1.151489E-05	1.4992573E+01	-3.058557E+01	-1.827483E+01
	Mean	0.000000E+00	-2.593014E+01	1.151489E-05	1.3930174E+01	-3.451320E+01	-2.662427E+01
	St. d.	0.000000E+00	4.215754E+00	0.000000E+00	3.3375182E-01	1.261354E+00	2.609085E+00
1.5×10^5	**Best**	**0.000000E+00**	**-2.842253E+01**	**1.151489E-05**	**1.3770762E+01**	**-3.684537E+01**	**-2.916612E+01**
	Median	0.000000E+00	-2.747974E+01	1.151489E-05	1.3770762E+01	-3.500981E+01	-2.742977E+01
	Worst	0.000000E+00	-2.711301E+01	1.151489E-05	1.4329113E+01	-3.410760E+01	-2.125851E+01
	Mean	**0.000000E+00**	**-2.770069E+01**	**1.151489E-05**	**1.3815430E+01**	**-3.503883E+01**	**-2.748811E+01**
	St. d.	0.000000E+00	4.673052E-01	0.000000E+00	1.5460045E-01	8.329248E-01	1.782137E+00

APPENDIX B

THE FUNCTION VALUES ACHIEVED VIA GA-MPC FOR TEST PROBLEMS (T07: T11.2)

FEs		T07 SPRP	T08 TNEP	T09 LSTP	T10 CAAD	T11.1 DED	T11.2 DED
5×10^4	Best	7.751747E-01	2.200000E+02	3.500808E+01	-2.1840989E+01	4.657803E+04	1.026594E+06
	Median	1.741340E+00	2.200000E+02	8.029370E+01	-2.1642572E+01	4.859988E+04	1.037454E+06
	Worst	1.915957E+00	2.200000E+02	1.868963E+02	-2.1473882E+01	5.088145E+04	1.056143E+06
	Mean	1.615097E+00	2.200000E+02	7.777810E+01	-2.1695070E+01	4.882505E+04	1.039756E+06
	St. d.	3.237073E-01	0.000000E+00	3.504446E+01	1.1605510E-01	1.347457E+03	8.970025E+03
1×10^5	Best	5.080677E-01	2.200000E+02	3.054932E+01	-2.1842536E+01	4.794790E+04	1.058026E+06
	Median	7.948024E-01	2.200000E+02	1.298753E+02	-2.1644448E+01	5.075259E+04	1.073148E+06
	Worst	1.682337E+00	2.200000E+02	2.535588E+02	-2.1475681E+01	1.145419E+05	1.275620E+06
	Mean	8.577491E-01	2.200000E+02	1.429435E+02	-2.1702232E+01	5.407931E+04	1.099666E+06
	St. d.	2.726676E-01	0.000000E+00	6.246572E+01	1.1633677E-01	1.292062E+04	5.749759E+04
1.5×10^5	**Best**	**5.000000E-01**	**2.200000E+02**	**7.999523E+01**	**-2.1842539E+01**	**4.730791E+04**	**1.054575E+06**
	Median	7.580619E-01	2.200000E+02	2.026846E+02	-2.1644450E+01	4.939572E+04	1.062514E+06
	Worst	9.334272E-01	2.200000E+02	3.199780E+02	-2.1475684E+01	5.127534E+04	1.064775E+06
	Mean	**7.484090E-01**	**2.200000E+02**	**2.131053E+02**	**-2.1702249E+01**	**4.944331E+04**	**1.061684E+06**
	St. d.	1.249139E-01	0.000000E+00	6.723456E+01	1.1634659E-01	1.165556E+03	2.589416E+03

APPENDIX C

THE FUNCTION VALUES ACHIEVED VIA GA-MPC FOR TEST PROBLEMS (T11.3: T11.8)

FEs		T11.3 ELD	T11.4 ELD	T11.5 ELD	T11.6 ELD	T11.7 ELD	T11.8 HS
5×10^4	Best	1.544419E+04	1.796585E+04	3.175912E+04	1.2049176E+05	1.6914855E+06	9.192127E+05
	Median	1.544419E+04	1.804846E+04	3.176787E+04	1.2066960E+05	1.7123320E+06	9.202711E+05
	Worst	1.544419E+04	1.827838E+04	3.177815E+04	1.2107438E+05	1.7198063E+06	9.224639E+05
	Mean	1.544419E+04	1.805769E+04	3.176757E+04	1.2068623E+05	1.7111924E+06	9.201912E+05
	St. d.	8.171498E-06	6.591602E+01	4.651742E+00	1.3787635E+02	6.0593887E+03	7.881428E+02
1×10^5	Best	1.544419E+04	1.800880E+04	3.257487E+04	1.2145848E+05	1.7061227E+06	9.240235E+05
	Median	1.544419E+04	1.810077E+04	3.258308E+04	1.2158513E+05	2.1732247E+06	9.255485E+05
	Worst	1.544419E+04	1.819443E+04	3.258933E+04	1.2184755E+05	5.3678338E+06	1.153841E+06
	Mean	1.544419E+04	1.811616E+04	3.258289E+04	1.2160090E+05	2.3867958E+06	9.708401E+05
	St. d.	4.974153E-07	5.023044E+01	4.105841E+00	1.0989043E+02	8.1651491E+05	7.696459E+04
1.5×10^5	**Best**	**1.544419E+04**	**1.802651E+04**	**3.269824E+04**	**1.2169315E+05**	**1.6947982E+06**	**9.238452E+05**
	Median	1.544419E+04	1.811060E+04	3.270866E+04	1.2188634E+05	1.7770371E+06	9.251644E+05
	Worst	1.544419E+04	1.820303E+04	3.272170E+04	1.2223143E+05	1.9625046E+06	9.271881E+05
	Mean	**1.544419E+04**	**1.812838E+04**	**3.270875E+04**	**1.2190596E+05**	**1.7813343E+06**	**9.249974E+05**
	St. d.	1.751115E-07	4.871860E+01	6.959762E+00	1.5532313E+02	6.8395166E+04	8.141665E+02

APPENDIX D

THE FUNCTION VALUES ACHIEVED VIA GA-MPC FOR TEST PROBLEMS (T11.9: T13)

FEs		T11.9 HS	T11.10 HS	T12 (Messenger full)	T13 (Cassini 2)
5×10^4	Best	8.991698E+05	9.1068500E+05	7.1285246E+00	8.616461E+00
	Median	9.012621E+05	9.1262534E+05	1.4196530E+01	8.762123E+00
	Worst	9.036658E+05	9.1432215E+05	1.8601084E+01	1.128432E+01
	Mean	9.012870E+05	9.1255892E+05	1.3447781E+01	9.514488E+00
	St. d.	9.679191E+02	9.2249028E+02	3.5831636E+00	1.027311E+00
1×10^5	Best	9.392652E+05	9.2408563E+05	7.0955884E+00	8.610143E+00
	Median	1.042841E+07	9.2448107E+05	1.3745705E+01	8.714559E+00
	Worst	2.560108E+07	9.2487785E+05	1.7375494E+01	1.125212E+01
	Mean	1.142669E+07	9.2448551E+05	1.3006419E+01	9.441149E+00
	St. d.	8.651787E+06	2.1157230E+02	3.3286083E+00	1.014002E+00
1.5×10^5	**Best**	**9.270085E+05**	**9.2369711E+05**	**7.0955595E+00**	**8.398688E+00**
	Median	9.306422E+05	9.2389917E+05	1.3725975E+01	8.620312E+00
	Worst	9.344096E+05	9.2415670E+05	1.6924893E+01	1.081018E+01
	Mean	**9.309053E+05**	**9.2392531E+05**	**1.2818165E+01**	**9.359342E+00**
	St. d.	1.986568E+03	1.2880822E+02	3.2413428E+00	9.454327E-01