DISCRETIZATION OF DECISION VARIABLES IN OPTIMIZATION ALGORITHMS

Martin Marek
Doctoral Degree Programme (2), FEEC BUT
E-mail: xmarek54@stud.feec.vutbr.cz

Supervised by: Petr Kadlec
E-mail: kadlecp@feec.vutbr.cz

Abstract: This paper presents a verification of universal method for discretization of decision space in optimization algorithms. Real-world optimization tasks frequently use discontinuous decision variables and in order to effectively optimize such tasks, it is necessary to exploit an optimization algorithm that meets such requirement. Unfortunately, very few evolutionary algorithms can naturally work with discontinuous decision space. The method that entitles all optimization algorithms to effectively solve problems with discrete variables is here described and experimentally verified.

Keywords: Optimization, evolutionary algorithms, discrete decision space

1 INTRODUCTION

Optimization is a process of finding the best possible solutions. The definition of solution's quality is based on fitness values calculated from fitness functions. Fitness functions describe behaviour of optimized system with properties called decision variables. Optimization process can be thereafter considered as a process of finding minima (or maxima) of fitness values.

If the system is described by one fitness function, the optimization process is called single-objective (SO) optimization. Optimization is called multi-objective (MO) in case of multiple fitness functions. Most real-world optimization problems are multi-objective, where the particular objectives are conflicting. Multi-objective optimization process results in a set of trade-off solutions called Pareto-front.

Parameters of optimized system - decision variables - in common optimization process can be any value from interval limited by prespecified extremes. However, some parameters of the system can be restricted somehow. In real world, there are always some manufacturing precision limits, only limited number of components available in stock, etc. Therefore, a given parameter is discrete and it is not possible to change it continuously. For example, only several dielectric substrates meet a design requirements or capacitors are available in manufacturing series, therefore the permittivity and capacitance used as a decision variable should be discrete.

The naïve approach is to let the optimization run as if all decision variables were continuous and pick final values closest to a set of feasible, discrete values afterwards. However, an optimization algorithm computes with all the possible values - therefore, puts an effort to find best solution with unsuitable values that the user will discard anyway.

A smart approach is to use an optimization method, that can work with discrete decision variables, that, in other words, does not waste computational resources in operating with unfeasible solutions.

Most of the optimization algorithms work with continuous decision variables, but a well known genetic algorithm [1] (and most of its modifications) works with discontinuous decision variables, therefore it uses discrete decision variables by its nature. Actually, the use of continuous decision variables in genetic algorithms is limited by the binary precision of discrete variables.

This suggests that when there is a need to use discrete decision variables, only limited number of optimization algorithms can be exploited. However, this paper presents a simple method, which allows every optimization algorithm to work with discrete decision variables.

Validity of method is verified by comparison of NSGA-II, MOPSO and GDE3 algorithm's performance on several well-known test problems with various discrete decision variables setting. The verification was performed in FOPS (Fast Optimization ProcedureS) optimization toolbox for MATLAB [2] which utilizes appointed discretization method.

All three introduced algorithms are multi-objective optimization algorithms, but the dicretization method can be applied to single-objective optimization methods as well.

Section 2 of this paper briefly describes exploited algorithms - NSGA-II, MOPSO and GDE3. Section 3 discusses principle of a presented method and Section 4 presents testing problems used for experimental verification. Section 5 contains experimental results and Section 6 concludes the paper.

2 OPTIMIZATION ALGORITHMS

2.1 NSGA-II

Elitist Non-dominated Sorting Genetic Algorithm was proposed in 2002 [3]. Principle of genetic algorithm is based on natural genetics and natural selection. Genetic algorithms work with two populations (parent and offspring). Offspring population is derived from parent population in each iteration of genetic algorithm. As the name suggests it is an elitist algorithm and selection operation is substituted by non-dominated sorting.

2.2 MOPSO

Particle Swarm Optimization is evolutionary algorithm which simulates movement of swarm of bees searching for food. The FOPS package implements a version based on [4].

The position of each agent is changed according to its own experience and that of its neighbors. The position $\vec{x}$ of i-th agent in subsequent iteration is changed by adding velocity $\vec{v}$ to a current position:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t), \tag{1}$$

where the velocity vector $\vec{v}$ is defined as follows:

$$\vec{v}_i(t) = W \cdot \vec{v}_i(t-1) + C_1 \cdot r_1 \left[\vec{x}_{pbest} - \vec{x}_i(t)\right] + C_2 \cdot r_2 \left[\vec{x}_{gbest} - \vec{x}_i(t)\right], \tag{2}$$

where r_1, $r_2 \in \langle 0,1 \rangle$ are random values, W is inertia weight, C_1 and C_2 are cognitive and social learning factors, respectively, $\vec{x}_{pbest}$ is personal best position and $\vec{x}_{gbest}$ is global best position.

The main difference from the single-objective version is the selection of global bests. In single-objective version only one solution is said to be the global best, but a multi-objective problem presents multiple trade-off solutions that are said to be the good ones.

2.3 GDE3

Third version of Generalized Differential Evolution is a multi-objective extension of Differential Evolution algorithm proposed in 2005 in [5].

Initially, random population is generated and in each iteration of algorithm the trial vector $\vec{u}$ is created with certain probability from agent's positions $\vec{x}$ using the equation:

$$u_{i,j}(t) = x_{j,r_3} + F \cdot (x_{j,r_2} - x_{j,r_1}), \tag{3}$$

where j denotes j-th decision variable, r_1, r_2, r_3 are randomly picked, mutually different and different from i-th agent's index and parameter F is a scaling factor. The trial vector $\vec{u}_i$ replaces agent's position $\vec{x}_i$ if $\vec{u}_i$ dominates $\vec{x}_i$.

The multi-objective version extends the single-objective one by the selection rule between decision and trial vector if both solutions are non-dominated. In such cases, both solutions are accepted and the extended population has to be pruned with favoring the diversity of non-dominated set.

3 DISCRETIZATION METHOD

Genetic algorithm - an algorithm which works with discrete decision variables by nature - describes the position of an individual in generation by a set of binary values. Density of discrete decision variable samples is given by binary precision, i.e. number of binary values which represents the given decision variable. A position of agent is varied by inverting randomly picked binary places of position vector (crossover and mutation operation).

On the other hand, algorithms with continuous decision variables usually vary the position vector by summing it with another randomly picked position vector (cf. (1) - (3)). Therefore, two real numbered positions values are summed together and create another real numbered position. The presented discretization method lets an algorithm to generate new agent's position on its own and then forces the decision variable to take one value from the discrete decision values set.

The simplest solution to find corresponding discrete value for a given real value would be to find the closest one in an absolute measure. But if discrete samples are non-homogeneously spaced between decision variables limits, then some discrete value would occur more often than the other.

The solution to this difficulty is to divide the range of decision variable by the number of discrete samples N (4). Then the floored difference between real decision variable and the lower limit divided by discrete samples step (δ) gives us the index of discrete sample corresponding to a real value (I).

$$\delta = \frac{x_{\max} - x_{\min}}{N}, \tag{4}$$

where $x_{\max}$ and $x_{\min}$ are the limits of a decision variable and N is the number of discrete samples of decision variable. The index of corresponding discrete value is defined by:

$$I = \left\lfloor \frac{x - x_{\min}}{\delta} \right\rfloor, \tag{5}$$

where $\lfloor y \rfloor$ defines the floor function and x is the real-numbered position.

4 TEST PROBLEMS

The comparative study performed to verify discretization method exploited 7 testing problems: DLTZ4, DTLZ6, DTLZ7 [7], ZDT1, ZDT2, ZDT6 and Poloni's problem [6].

ZDT and DTLZ families of testing problems were picked due to their convenient location of true optimal set. In ZDT problems, the solution is optimal, if all decision variables are zeros except the first variable, which utilizes true Pareto-front. DTLZ problems are similar, except that the first and also the second decision variables utilize the true Pareto-front and in case of DTLZ4, remaining decision variables (except first and second) have to be 0.5. Therefore, such values are obtainable even with roughly discretized decision space. On the other hand, Poloni test problem has more complicated

Algorithm	Samples	ZDT1	ZDT2	ZDT6	DTLZ4	DTLZ6	DTLZ7	MOPOL
NSGA-II	2^{20}	21.45	29.49	1919.04	56.58	4227.23	70.43	54.88
	5001	14.98	20.58	1619.15	96.92	2768.14	48.92	73.72
	501	8.75	9.25	683.47	55.20	879.09	32.60	51.52
	51	0.23	0.05	0.05	12.59	3.93	12.58	57.92
MOPSO	∞	17.13	10.15	5465.71	245.31	5226.11	70.86	12.47
	5001	12.31	5.01	385.78	164.69	2073.57	35.27	19.35
	501	8.93	4.06	100.77	106.91	1632.72	35.84	20.52
	51	13.71	4.98	151.23	17.97	845.12	75.06	26.21
GDE3	∞	2.42	1.56	24.58	62.73	72.83	23.77	42.71
	5001	2.30	2.29	72.55	62.86	177.89	22.58	52.48
	501	0.35	1.32	0.13	63.79	116.69	21.34	51.27
	51	0.03	0.04	0.05	0.69	22.60	11.18	63.79

Table 1: Thousandfold generational distance of several test problems with various discretization.

true optimal set (can be seen in [6, Chapter 8]) and the discretization of decision space might prevent the optimization algorithm from locating it. All ZDT and DTLZ problems had 10 decision variables during tests. Poloni test problem has only two decision variables.

5 RESULTS

To verify functionality of presented method, great number of individual simulations were performed in a comparative study suite of FOPS toolbox. There were 3 optimization algorithms - NSGA-II, MOPSO and GDE3 - exploited on 7 testing problems introduced in Section 4. Also there were four different setting of discrete decision variables. The first setting was continuous decision variables. Remaining settings sampled each decision variable to 5001, 501 and 51 discrete samples, respectively.

Therefore, 3 algorithms, 7 testing problems and 4 discrete variables settings results in 84 different optimization tasks. Moreover, each task was repeated 100 times in order to obtain independent realizations of stochastic processes. All three algorithms processed 40 agents over 100 iterations.

The NSGA-II algorithm was set as follows: *Probability of crossover* - 0.9, *Probability of mutation* - 0.7, *Binary precision* (continuous decision variables) - 20.

The MOPSO algorithm was set as follows: *Inertia weight* - linearly decreased from 0.8 to 0.5, *Cognitive learning factor* - 1.5, *Social learning factor* - 1.0, *Boundary type* - reflecting.

The GDE3 algorithm was set as follows: *Scaling factor* - 0.2, *Probability of crossover* - 0.2.

Table 1 contains average values of generational distances obtained in comparative study multiplied by 1000. Generational distance metric represents average of minimal distances from found non-dominated set to true Pareto-front [6].

Each problem was optimized with four different discretization setting. It is obvious that the lower the density of samples is, the easier is to find the true Pareto-front, i.e. the lower the generational distance value should be. This trend is visible in most cases in Table 1, but there are also several exceptions.

The fact that generational distance values produced by MOPSO and GDE3 algorithms decreases with decreasing density of discrete samples verifies the functionality of used discretization method.

Most of the exceptions occur with the use of MOPSO algorithm and only 50 samples per decision variable. The reason is hidden in MOPSO algorithm behavior rather than in discretization method itself. MOPSO uses external archive of found non-dominated solutions. Majority of continuous non-dominated solutions is dominated due to discretization, therefore an external archive holds only few entries. This paralyzes the exploration capabilities of algorithm, because external archive members are used as x_{gbest} in (2).

Also Poloni's problem (**MOPOL**) shows exceptional trends but in this case it is caused by unsuitably located true Pareto-front (see in [6, Chapter 8]), therefore the discretization of decision space prevents the algorithm from reaching the optimum.

6 CONCLUSION

If a binary precision of decision variables in binary-coded NSGA-II algorithm is decreased, the number of possible positions in decision space is also decreased, therefore it is easier for the algorithm to find the optimum.

Discretization method allows optimization algorithm to work normally with continuous decision variables and the discretization of decision space is forced afterwards. But the results obtained in comparative study proves that such technique has the similar effect on the difficulty of optimization problem, in other words the overall decision space is shrunk.

ACKNOWLEDGEMENT

Research was supported by the grant LO1401 of the National Sustainability Program and Internal Grant Agency of Brno University of Technology project no. FEKT-S-17-4713. Infrastructure of the SIX Center was used.

REFERENCES

[1] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[2] M. Marek, P. Kadlec, and M. Cupal, "FOPS - Fast Optimization Procedures." http://antennatoolbox.com/fops-about.php, 2017. [Online; accessed 31-January-2018].

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[4] M. Reyes-Sierra and C. C. Coello, "Multi-objective particle swarm optimizers: A survey of the state-of-the-art," *International journal of computational intelligence research*, vol. 2, no. 3, pp. 287–308, 2006.

[5] S. Kukkonen and J. Lampinen, "GDE3: The third evolution step of generalized differential evolution," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 1, pp. 443–450, IEEE, 2005.

[6] K. Deb, *Multi-objective optimization using evolutionary algorithms*, vol. 16. John Wiley & Sons, 2001.

[7] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, vol. 1, pp. 825–830, IEEE, 2002.