The Fourteenth International Symposium on Artificial Life and Robotics 2009 (AROB 14th '09),
B-Con Plaza, Beppu, Oita, Japan, February 5 - 7, 2009

Robust Optimization Using Multi-Objective Particle Swarm Optimization

Satoshi Ono Yohei Yoshitake Shigeru Nakayama
Department of Information and Computer Science,
Faculty of Engineering, Kagoshima University
1-21-40, Korimoto, Kagoshima, 890-0065 Japan
{ono, sc104059, shignaka}@ics.kagoshima-u.ac.jp

Abstract

This paper proposes an algorithm searching for solutions which are robust against small perturbations of design variables. The proposed algorithm formulates robust optimization as a bi-objecitve optimization problem, and finds solutions by Multi-Objective Particle Swarm Optimization (MOPSO). Experimental results have shown that MOPSO has better search performance to find multiple robust solutions than a previous method using multi-objective genetic algorithm.

1 Introduction

Particle Swarm Optimization (PSO)[1] is one of stochastic, population-based optimization algorithms inspired by swarm intelligence of insects which form a group and move such as bird, fish, bee and so on. PSO has recently been investigated and applied to many real-world problems because of its simplicity and good search performance. PSO is effective in problems whose design values are represented by real values in particular. Multi-Objective Particle Swarm Optimization (MOPSO)[2, 3, 4] is also proposed to solve multi-objective optimization problems involving more than one objective function.

In recent years there has been renewal of interest in robust optimization techniques as a practical optimization methodology considering margins of errors, noises, aged deterioration, and other uncertainties on design, production, observation and so on [5, 6, 7, 8, 9]. General optimization algorithms evaluate solution candidates with focusing only on optimality of an objective function. If a solution obtained by the algorithms is sensitive to small perturbations of variables, it may not be appropriate or risky for practical use. Such small variation may cause undesired deviations of engine performance in automobile valuvetrain control, or collisions or interference in controlling machines. Robust optimization[7, 8, 10] finds solutions which are moderately good in terms of optimality and also good in germs of robustness against small perturbations of values, as shown in Figure 1. In many practical optimization tasks, there is a need to search for robust solutions whose value of optimization function is su ciently high and will not change due to the small variation of parameter values.

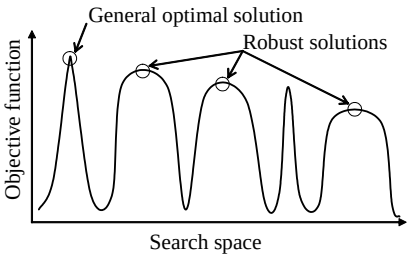

Figure 1: The difference between general optimization and robust optimization.

Design For Six Sigma (DFSS)[11, 12] is a methodology for designing new products or processes and can be considered as a robust optimization algorithm. But in DFSS, an optimization function equation involves weight parameters which must be adjusted manually, and sigma level must be specified before starting a search.

Aiming to resolve the above drawbacks of DFSS, Design For Multi-Objective Six Sigma (DFMOSS) has been proposed[6, 8]. DFMOSS performs Monte Carlo simulation and evaluates solution candidates with two objective functions: mean value of given objective function and its deviation. DFMOSS therefore does not need to adjust the weight parameters in objective function of DFSS, and to specify sigma level in advance. Although DFMOSS can find multiple robust optimal solutions simultaneously, DFMOSS requires high computational cost.

In this paper, we propose a robust optimization method by using MOPSO in order to verify the effectiveness of MOPSO against robust optimization. The

Step 1: Initialize all particles; place them at random positions with random velocities.
Step 2: Evaluate all particles.
Step 3: Store Pareto solutions chosen randomly into the archive.
Step 4: Divide objective function space onto hypercube.
Step 5: Determine personal bests of all solutions.
Step 6: *Until* evaluation time reaches the limit, *repeat* **step 7**:
Step 7: *For each* particle i, *repeat* **step 8** through **12**.
Step 8: Update i's velocity and position.
Step 9: *If* i moves too slowly, reset its position and velocity.
Step 10: Evaluate i.
Step 11: Update personal best and the archive.
Step 12: Redivide the hypercube if necessary.

Figure 2: The outline of the proposed algorithm.

proposed algorithm formulates a robust optimization problem as a multi-objective optimization problem by following the idea of DFMOSS, and utilizes MOPSO to search for robust solutions instead of Multi-Objective Genetic Algorithm in DFMOSS. The proposed algorithm also utilizes hypercube method in order to maintain its archive to be diverse. Experimental results have shown that the proposed algorithm could find robust optimal solutions with higher discovery rate than DFMOSS in a benchmark function.

2 The proposed algorithm

2.1 Overview

The algorithm proposed in this paper finds multiple robust solutions simultaneously based on the following basic ideas:

1. **Formulating robust solution search as a bi-objective optimization problem.** The proposed algorithm replaces single-objective robust solution search to bi-objective optimization problem by using Monte Carlo Simulation (MCS). The two objective functions are mean of the given objective function values at sampled points nearby a particle to be evaluated and their standard deviation.

2. **Using Multi-objective Particle Swarm Optimization (MOPSO).** The proposed method is based on MOPSO[2] whereas the previous work uses MOGA[8]. PSO is promising for problems involving continuous design variables. In MOPSO, a particle moves toward one of known Pareto solutions and searches around the solution exploitatively. Although MOGA uses Pareto ranking scheme to handle multi-objective optimization problem, MOPSO manages Pareto optimal solutions by storing grid-structured archive [13, 2, 4]. Dividing an objective space into hypercubes allows to maintain the diversity of Pareto solutions.

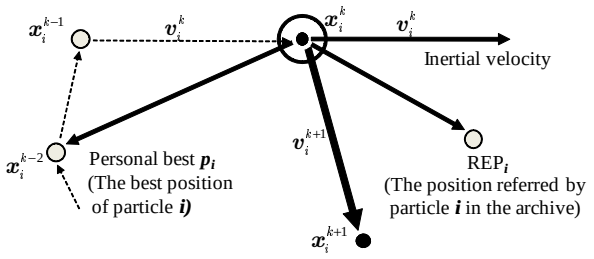

Figure 3: Velocity and position update.

2.2 Search by MOPSO

First, each particle i is initialized; its position $\boldsymbol{x}_i$ and its velocity $\boldsymbol{v}_i$ are defined by random. And then, the archive is initialized by storing Pareto solutions derived from a set of positions chosen randomly from search space. Each position in the archive is assigned to a particle as REP_i by random. Objective space is divided onto $d \times d$ hypercubes by dividing each objective into d equal divisions. Personal best $\boldsymbol{p}_i$ is initialized by $\boldsymbol{x}_i$.

After initialization, the proposed algorithm iterates particles' position and velocity update. As shown in Figure 3, particle velocity is updated considering its personal best position and referring solution's position in archive by following equation:

$$\boldsymbol{v}_i^{k+1} = w\boldsymbol{v}_i^k + c_1 r_1(\boldsymbol{p}_i - \boldsymbol{x}_i^k) + c_2 r_2(REP_h - x_i^k), \quad (1)$$

$$\boldsymbol{x}_i^{k+1} = \boldsymbol{x}_i^k + \boldsymbol{v}_i^{k+1}, \quad (2)$$

where $\boldsymbol{x}_i^k$ and $\boldsymbol{v}_i^k$ indicate position and velocity of particle i at step k, $\boldsymbol{p}_i$ indicates personal best of particle i which is the best position of all positions the particle passed so far, REP_h is a solution in archive which is referred by particle i, c_1 and c_2 are two positive constants called cognitive and social parameter, r_1 and r_2 are random numbers uniformly distributed within [0, 1].

Particle i is evaluated in its current position $\boldsymbol{x}_i^{k+1}$. If the current position dominates personal best $\boldsymbol{p}_i$ of particle i, then the position replaces $\boldsymbol{p}_i$. If neither of

the current position and $\boldsymbol{p}_i$ is dominated by the other, $\boldsymbol{p}_i$ is selected from them randomly.

Particle i is reset when it violates the constraint, i.e., it stikcs out of defined domain. Particle i is also reinitialized when satisfying following two conditions:

- Particle i moves too slowly, i.e., its speed $|v_i|$ goes down under a threshold T_v, and
- There is no improvement on $\boldsymbol{p}_i$ for more than Tr steps.

2.3 Particle evaluation

Two objective functions, mean $\mu_f(\boldsymbol{x})$ and standard deviation $\sigma_f(\boldsymbol{x})$ of given objective function values, are statistical values calculated based on Monte Carlo Simulation (MCS). Namely, probability distribution which imitates unevenness of design variables is assumed as Gaussian distribution, mean of the distribution is set to be the value of variables $\boldsymbol{x}^{(i)}$ of individual i and standard deviation of the distribution to be a specified value, and random points are sampled nearby $\boldsymbol{x}^{(i)}$.

The proposed algorithm uses Pareto ranking scheme as DFMOSS, and Pareto solutions are stored in the archive which are divided onto d^2 hypercubes. Fitness sharing is therefore conducted by this archive structure.

Solutions are stored in archive by following two rules:

1. If a particle finds a new good position which dominates the solution the particle is referring, then the solution in the archive is replaced by the new position.
2. If both a solution found by a particle and the solution referred by the particle are non-dominant each other, then the new position is stored in archive without eliminating the referred solution. If the number of solutions in the archive exceeds the limit N_p, a solution is selected randomly from the hypercube which has the most solutions.

Archive is re-partitioned in the case that the maximum or minimum value of a design variable in the archive is updated.

3 Evaluation

A benchmark function $f_t(\boldsymbol{x})$ is defined to experimentally validate search performance of the proposed algorithm.

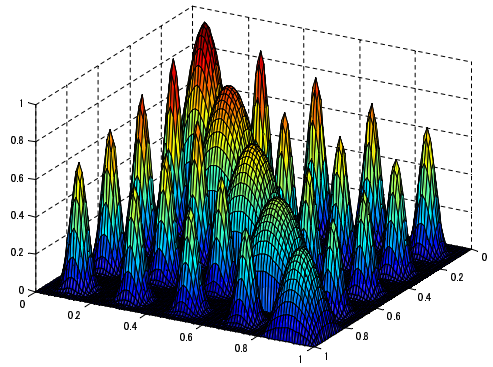

Figure 4: Tested function f_t $(n = 2)$.

$$f_t(\boldsymbol{x}) = \prod_{k=1}^{n} e^{-\frac{x_k - 0.1}{n}} \left|\sin^{m(\boldsymbol{x})}(5\pi x_k)\right| \quad (0 \le x_k \le 1) \tag{3}$$

$$m(\boldsymbol{x}) = \begin{cases} 6 & if \bigvee\limits_{a \in \{0, 0.2, \ldots, 0.8\}} \left(\bigwedge\limits_{k=1, \ldots, n} a < x_k < a + 0.2 \right) \\ 1 & Otherwise \end{cases} \tag{4}$$

The tested function f_t has five robust solutions, and f_t at $n = 2$ is shown in Fig. 4. We used the function whose dimension n was from 2 to 5. Upper and lower specification limits (USL and LSL) are parameters which should be specified for each target problem[11]; in this experiment, LSL was set to 0.1 and USL was not used.

A robust solution whose mean value $\mu_f(\boldsymbol{x}^{(i)})$ and standard deviation $\sigma_f(\boldsymbol{x}^{(i)})$ of an objective function satisfy the following equation is regarded as sigma level $l\sigma$:

$$\mu_f(\boldsymbol{x}^{(i)}) - l\sigma_f(\boldsymbol{x}^{(i)}) \ge \text{LSL}. \tag{5}$$

The higher l is, the more robust $\boldsymbol{X}^{(i)}$ is against small perturbations of $\boldsymbol{x}^{(i)}$. Sigma levels of solutions found by a tested algorithm were calculated after the search.

Parameters of MOPSO were configured as follows: number of particles was set to 1,000, and w, c_1, c_2, T_v, T_{rs}, N_p, T_r were set to 0.9, 1.2, 1.2, 10^{-3}, 100, 1,000, and 100, respectively. Sampling was performed by using Descriptive Sampling (DS)[14], and sampling number and range of DS were 1,000 and 0.02. Parameters of DFMOSS were configured as follows: population size, crossover method, crossover rate, mutation rate, and a parameter of Pareto ranking c were 100,

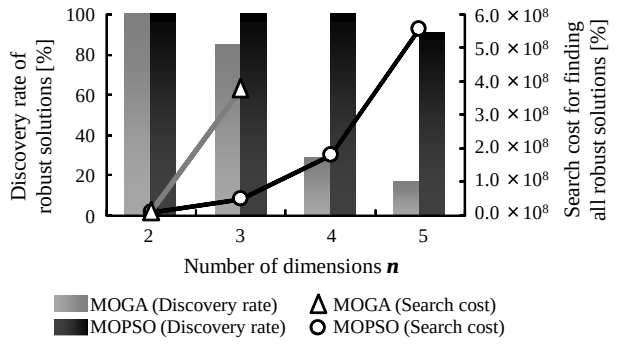

Figure 5: Experimental results.

BLX-α ($\alpha = 0.5$), 1.0, 0.2 and 0.1, respectively. The maximum number of function calls was set to 1.0×10^9.

Figure 5 shows the discovery rate of robust solutions over 30 runs, and search cost that is a function evaluation time for finding all robust solutions averaged over the runs succeeded in finding all robust solutions. The proposed algorithm could find all robust solutions simultaneously even when $n = 4$ and find almost all of solutions when $n = 5$, whereas DFMOSS could find all solutions only when $n = 2$ and found no solutions when $n = 5$.

4 Conclusions

In this paper, we propose an algorithm for robust solution search using multi-objective particle swarm optimization. The proposed algorithm formulates robust optimization as a bi-objective optimization problem which involves two objective functions of mean and standard deviation on sampled values of an objective function.

Experimental results have shown that the proposed MOPSO could find robust solutions better than DFMOSS using MOGA.

In the future, we plan to examine in higher dimensional problems and to adopt multi-objective memetic particle swarm optimization.

Acknowledgements

This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientist (B), No.20700211, 2008–2010.

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Procedings of the IEEE International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[2] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[3] J. E. Fieldsend, "Multi-objective particle swarm optimization methods," Technical Report #419, Department of Computer Science, University of Exeter, 2004.

[4] L. Cagnina, S. Esquivel, and C. A. C. Coello, "A particle swarm optimizer for multi-objective optimization," *Journal of Computer Science & Technology*, vol. 5, no. 4, pp. 204–210, 2005.

[5] E. Kazancioglu, G. Wu, J. Ko, S. Bohac, Z. Filipi, S. J. Hu, D. Assanis, and K. Saitou, "Robust optimization of an automobile valvetrain using a multiobjective genetic algorithm," in *Proceedings of ASME 2003 Design Engineering Technical Conferences (DETC'03)*, 2003.

[6] K. Shimoyama, K. Fujii, and H. Kobayashi, "Development of realistic optimization method of tsto spaceplane — multi-objective and robust optimization," in *Proceedings of 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2004.

[7] S. L. Padula, C. R. Gumbert, and W. Li, "Aerospace applications of optimization under uncertainty," *Optimization and Engineering*, vol. 7, no. 3, pp. 317–328, 2006.

[8] K. Shimoyama, "Robust aerodynamic design of mars exploratory airplane wing with a new optimization method," Ph.D. dissertation, University of Tokyo, 2006.

[9] D. Huang, F. Fabozzi, and M. Fukushima, "Robust portfolio selection with uncertain exit time using worst-case var strategy," *Operations Research Letters*, vol. 35, pp. 627–635, 2007.

[10] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments — a survey," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, 2005.

[11] G. Brue and R. G. Launsby, *Design for Six Sigma*. Mcgraw-Hill, 2003.

[12] M. Sokovic, D. Pavletic, and S. Fakin, "Application of six sigma methodology for process design," *Journal of Materials Processing Technology*, vol. 162-163, pp. 777–783, 2005.

[13] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. JohnWiley & Sons, Ltd., 2001.

[14] E. Saliby, "Descriptive sampling: A better approach to monte carlo simulation," *Journal of the Operational Research Society*, vol. 41, no. 12, pp. 1133–1142, 1990.