WCCM V
Fifth World Congress on
Computational Mechanics
July 7-12, 2002, Vienna, Austria
Eds.: H.A. Mang, F.G. Rammerstorfer,
J. Eberhardsteiner

Human-like Optimization – A Novel Technique for Computational Design

Tomonari Furukawa*

School of Aerospace, Mechanical and Mechatronic Engineering, J04
University of Sydney, NSW 2006 Australia
e-mail: tomo@acfr.usyd.edu.au

Shinobu Yoshimura and Hiroshi Kawai

Institute of Environmental Studies
University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656 Japan
e-mail: yoshi@q.t.u-tokyo.ac.jp

Key words: Human-like optimization, computational design, multi-objective optimization method, gradient search, center-of-gravity method

Abstract

In the design of many mechanical systems, the human designer knows its topology and shape. The goal for computational design thus results in finding a parameter set that optimizes multiple design goals. This paper presents a novel optimization method for such parametric design, the process of which is based on the human design process, but the search algorithm of which is gradient-based for efficient computation.

In sanction with the human design process, the proposed method first searches for multiple solutions or Pareto-optimal solutions, which correspond to the solution space of the multi-objective optimization problem, directly by minimizing multiple objective functions. The proposed method then finds the final design by the center-of-gravity method proposed by the authors, where the closest to the center-of-gravity of the solutions in parameter space becomes the final design.

There exist several evolutionary algorithms that can find Pareto-optimal solutions. The performance of the proposed optimization was investigated with numerical examples and compared to a multi-objective evolutionary algorithm. The numerical results first demonstrate that the proposed method can search Pareto-optimal solutions only after 20 iterations, which could not be obtained by the evolutionary algorithm even after 500 iterations. The final solution obtained also correlated well with the exact robust solution of the problem.

1 Introduction

Since the topology and the shape are often known, the design of a mechanical system corresponds to the determination of a set of design parameters. The response of a system to a design parameter set is in general given by an implicit non-linear mapping. In the determination, the designer therefore makes full use of his knowledge and experience on the parameter set and its relationship with the design objectives. This human design however fails when his knowledge and experience are not enough. Computational design has received considerable attention with the advance of computer hardware and software accordingly [1,2].

The goal of design is to find a parameter set that optimizes design objectives. In the computational design, an optimization method is hence used to find a design parameter set on behalf of the human designer. The majority of optimization methods developed so far can deal with only a single-objective function [3]. Thus, the design objectives are first scalarized by introducing additional parameters such as weighting factors, which are often unknown [4,5]. The design parameter set can be therefore found with a single-objective optimization method, although the parameter set obtained may not be the one that the designer looks for. This problem is rooted in the fact that the current optimization methods are considerably different from the human design in process.

In this paper, human-like optimization, which adopts the human design process, and subsequent techniques necessary for the optimization, are presented. In the optimization, search for designs that satisfy the design objectives and the determination of the final design are conducted in a stepwise manner, as human designers do. Thus, multi-objective optimization and the Pareto pooling technique are adopted for finding well-distributed possible optimal designs, and the center-of-gravity method is proposed to determine the final design in the optimization. As the forward analysis in the optimization loop often consumes a significant amount of computation time, two multi-objective optimization methods with conventional gradient search algorithms, Multi-objective Steepest Descent (MSD) method and Multi-objective Quasi-Newton (MQN) method, are proposed. Particularly, MQN method is expected as the most efficient multi-objective optimization method.

The next section presents the definition and the detailed algorithms of human-like optimization. MQN and MSD methods are presented in the third section with a standard multi-objective optimization method implementing evolutionary algorithms, Multi-objective Continuous Evolutionary Algorithms (MCEA) [6]. The fourth section describes numerical examples, and conclusions are summarized in the final section.

2 Human-like Design

2.1 Overview

As multiple objectives need to be satisfied in design, the corresponding optimization problem is typically defined to search for parameter set $\mathbf{x}_{\min} \le \mathbf{x} \le \mathbf{x}_{\max}$ in continuous space $\mathbf{x}_{\min}, \mathbf{x}, \mathbf{x}_{\max} \in R^n$, such that objective functions $\mathbf{f}(\mathbf{x}) : R^n \to R^m$ are minimized:

$$\mathbf{f}(\mathbf{x}) \to \min_{\mathbf{x}}, \tag{1}$$

where configuration of the objective functions is often unknown [4]. Figures 1 and 2 compare the human design and the present optimization for computational design. In human design, the designer makes full use of his knowledge and experience on the design parameters and their relationship with

design objectives. The designer first finds several parameter sets by considering their relationship with each objective function. The designer may try to find a single solution at the beginning, but consideration of multiple design objectives inevitably results in several sets because of the trade-off among design objectives. The designer then decides a final design parameter set from the parameter sets using his knowledge on the parameters. Meanwhile, the optimization starts with conversion of design objectives to a single-objective function by introducing weighting factors. A single solution is then found with a single-objective optimizer. The figures clearly show that the processes of the human design and the present optimization are quite different.

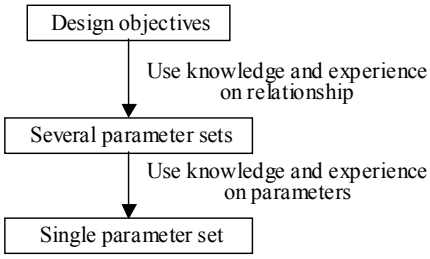


Figure 1: Human design process.

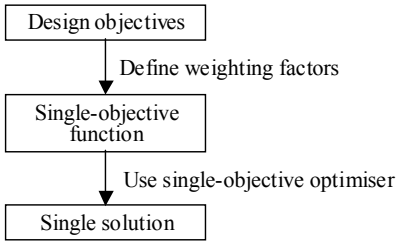


Figure 2: Present optimization process.

Figure 3 shows the proposed optimization for computational design. The multi-objective optimization problem results in a solution space rather than a single solution, so that a multi-objective optimization method first finds multiple solutions, which are equivalent to the solution space. The final solution is then found by considering distribution of the solutions in parameter space. It is easily seen that the process of the proposed optimization is considerably similar to that of human design. Similarities are summarized that

- Multiple design parameter sets are directly obtained by considering multiple design objectives.
- A final design parameter set is chosen by considering design parameter sets in parameter space.

Furthermore, the superiority of the proposed optimization to the human design is the optimality. The optimality is yielded by the facts that

- More optimal multiple design parameter sets are obtained.
- More information is available in the selection of a final design parameter set, as the number of design parameter sets is larger.

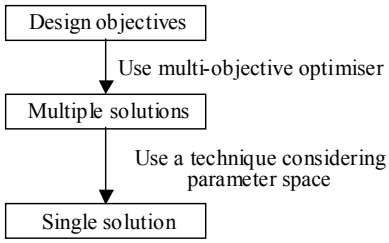


Figure 3: Proposed optimization process.

To achieve the proposed optimization, necessary computational techniques to be developed are

- A multi-objective optimization method that can derive solutions equivalent to the solution space of the multi-objective optimization problem.
- The multi-objective optimization method has to be efficient as the computation of forward analysis, such as finite element analysis, is heavy.

- A technique to determine a robust single solution from solutions by considering their distribution in parameter space.

The next section will describe the computational techniques that satisfy the requirements.

2.2 Search for Multiple Solutions

The discussion of the multi-objective optimization method must start with the definition of solutions to search for. The solutions taking into account the trade-off of multiple design objectives are the so-called Pareto-optimal solutions [7,8]. By definition, parameter set $\mathbf{x}_u \in R^n$ is said to be Pareto-optimal if and only if there is no vector $\mathbf{x}_v \in R^n$ for which $\mathbf{v} = \mathbf{f}(\mathbf{x}_v) = [v_1, ..., v_m]$ dominates $\mathbf{u} = \mathbf{f}(\mathbf{x}_u) = [u_1, ..., u_m]$, i.e., there is no vector $\mathbf{x}_v$ such that

$$v_i \le u_i, \forall i \in \{1, ..., m\} \wedge v_i < u_i, \exists i \in \{1, ..., m\}. \tag{2}$$

Figure 4 shows the flowchart of the proposed framework of the multi-objective optimization method. In order to find multiple solutions, the multi-objective optimization method searches with λ multiple points, i.e., $X(k) \equiv \{\mathbf{x}_k^1, ..., \mathbf{x}_k^\lambda\} \in (R^n)^\lambda$ where $\mathbf{x}_k^i$ is the i th search point at k th iteration. The initial generation of population $X(k)$ is conducted randomly within the range $[\mathbf{x}_{\min}, \mathbf{x}_{\max}]$ unless good search points are known. Each objective function value $f_j(\mathbf{x}_k^i)$ is then calculated with each parameter set $\mathbf{x}_k^i$, finally yielding $F(k) \equiv \left\{\mathbf{f}\left(\mathbf{x}_k^1\right), ..., \mathbf{f}\left(\mathbf{x}_k^\lambda\right)\right\}$. The population of vector functions $F(k)$ is used to further evaluate two independent scalar criteria of each search point. One is the rank in Pareto-optimality $\Theta(k) \equiv \left\{\theta\left(\mathbf{x}_k^1\right), ..., \theta\left(\mathbf{x}_k^\lambda\right)\right\}$ ($\theta : R^n \to N$), and the other is a corresponding scalar objective function $\Phi(k) \equiv \left\{\phi\left(\mathbf{x}_k^1\right), ..., \phi\left(\mathbf{x}_k^\lambda\right)\right\}$ ($\phi : R^n \to R$). The rank is evaluated using the Pareto-optimality rule (2), and search points ranked No. 1 are considered as Pareto-optimal solutions in the current population. On the other hand, the scalar function is used to create the next search point $\mathbf{x}_{k+1}^i$, and the creation depends upon the search algorithms to be used. The next population in canonical form is thus written as

$$X(k+1) = s\left(X(k), \Phi(k), \nabla\Phi(k), \nabla^2\Phi(k)\right) \tag{3}$$

in a general sense where s is the search operator.

Once the iterative computation and Pareto-optimality judgment of $\mathbf{x}_{k+1}^i$ become possible, we want to find effective Pareto-optimal solutions as many as possible such that a solution space can be configured. Another technique proposed here to do so is a Pareto pooling strategy where the Pareto-optimal solutions created historically are pooled besides the iterative search of a new population of search points.

The process of the Pareto pooling technique is as follows. The whole Pareto-optimal solutions obtained in the first generation are saved in this storage. From the second generation, the newly created Pareto-optimal solutions in the optimization loop are compared to the stored Pareto-optimal solutions, and the new set of Pareto-optimal solutions is saved in the storage as illustrated in Figure 5. Some Pareto-optimal solutions may be identical or very close to an existing point. The storage of such solutions is simply a waste of memory, so that they are discarded if they are closer than the resolution we set *a priori*. The creations of the new population and the Pareto-optimal solutions are repeated until a terminal condition is satisfied.

This technique allows the Pareto-optimal solutions created in the past to be kept as solutions and yields a good chance to increase the number of Pareto-optimal solutions, thus making the solution space easier to see. The storage of the solutions independent of the current population also may contribute to the good distribution of the resultant solutions.

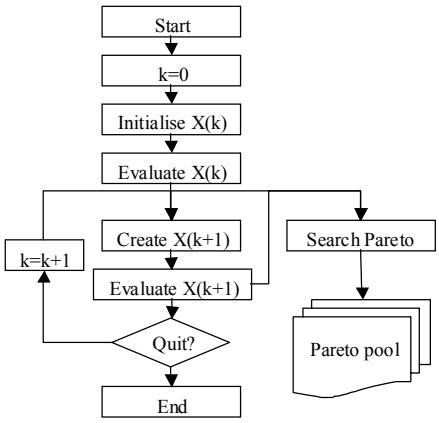


Figure 4: Flowchart of proposed processes.

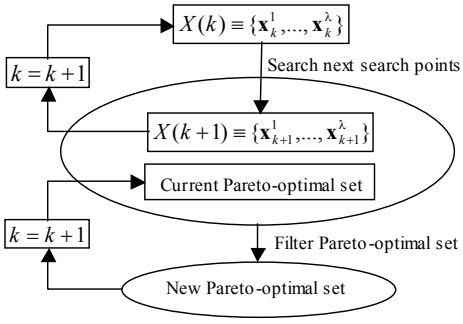


Figure 5: Creation of Pareto-optimal solutions.

2.3 Determination of a Single Solution

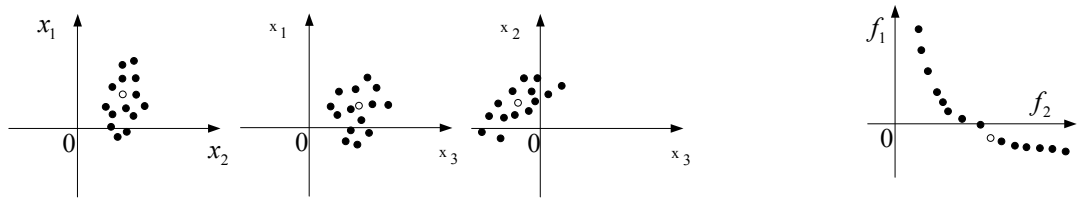
Figure 6 illustrates Pareto-optimal solutions where two objective functions $\mathbf{f} = [f_1, f_2]$ are minimized to identify three parameters $\mathbf{x} = [x_1, x_2, x_3]$. As two-dimensional function space and three-dimensional parameter space are still easy to visualize, one may incorporate human knowledge into computational knowledge-based techniques such as expert systems and fuzzy logic for automatic selection of a single solution. However, if the numbers of objective functions and parameters are considerably large, the knowledge to be constructed to be immense, and such techniques are no longer possible practically. In this case, one prominent way is to select the solution residing in the center of solution space since this solution is robust. The authors here propose a technique where the closest solution to the center-of-gravity is chosen as the solution. Let the Pareto-optimal solutions finally obtained be $\hat{\mathbf{x}}^i$, $i = 1,...,r$. If each solution is evaluated in a scalar manner, i.e., $\varphi(\hat{\mathbf{x}}^i)$, the center-of-gravity is in general given by

$$\hat{\mathbf{x}} = \frac{\sum_{i=1}^{r} \hat{\mathbf{x}}^i \varphi(\hat{\mathbf{x}}^i)}{\sum_{i=1}^{r} \varphi(\hat{\mathbf{x}}^i)}. \tag{4}$$

As the Pareto-optimal solutions must be evaluated equally, we can consider all the Pareto-optimal solutions possess the same scalar value, i.e., $\varphi(\hat{\mathbf{x}}^1) = \varphi(\hat{\mathbf{x}}^2) = ... = \varphi(\hat{\mathbf{x}}^r)$. No matter what the value is, the center-of-gravity results in the form:

$$\hat{\mathbf{x}} = \frac{\sum_{i=1}^{r} \hat{\mathbf{x}}^i}{r}. \tag{5}$$

The effectiveness of the center-of-gravity method cannot be proved theoretically, but it is highly acceptable, as it has been commonly used in fuzzy logic [9] to find a solution from the solution space described by fuzzy sets.



(a) Parameter space (b) Function space

Figure 6: Process of deriving a single solution.

3 Search Algorithms for Multi-objective Optimization Method

3.1 Multi-objective Gradient-based Methods

The greatest advantage of introducing $\Phi(k)$ independent of $\Theta(k)$ is that any conventional search algorithm that is formulated in the form (3) can be implemented [10]. Scalar function $\Phi(k)$ in gradient-based methods proposed by the authors is defined with

$$\phi\left(\mathbf{x}_k^i\right) = \sum_{j=1}^{m} w_j^k f_j\left(\mathbf{x}_k^i\right), \tag{6}$$

where $w_j^k = \mathrm{rand}(0,1)$ contributes to the wide distribution of resultant Pareto-optimal solutions. The notation $\mathrm{rand}(0,1)$ means a uniform random number between 0 and 1. With this $\Phi(k)$, the next state of a search point is given by

$$\mathbf{x}_{k+1}^i = \mathbf{x}_k^i + \Delta\mathbf{x}_k^i, \tag{7}$$

where the step of the search point is determined by

$$\Delta\mathbf{x}_k^i = \alpha\mathbf{d}\left(\mathbf{x}_k^i, \phi(\mathbf{x}_k^i), \nabla\phi(\mathbf{x}_k^i), \nabla^2\phi(\mathbf{x}_k^i)\right). \tag{8}$$

In the equation, α is the search step length iteratively searched as a subproblem by Wolfe's algorithms [11], whereas mapping $\mathbf{d}$ outputs represents the direction of the search step.

In the MSD method, the mapping $\mathbf{d}_{\mathrm{SD}}$ is based on the Steepest Descent (SD) method:

$$\mathbf{d}_{\mathrm{SD}}\left(\mathbf{x}_k^i, \nabla\phi\left(\mathbf{x}_k^i\right)\right) = -\nabla\phi\left(\mathbf{x}_k^i\right). \tag{9}$$

The mapping of MQN method adopts the Quasi-Newton (QN) method, which is most commonly used due to its fast convergence, and is described as

$$\mathbf{d}_{\mathrm{QN}}\left(\mathbf{x}_k^i, \nabla\phi\left(\mathbf{x}_k^i\right)\right) = -\mathbf{A}_k^{-1}\nabla\phi\left(\mathbf{x}_k^i\right), \tag{10}$$

where $\mathbf{A}_k \cong \nabla^2\phi\left(\mathbf{x}_k^i\right)$.

3.2 Multi-Objective Continuous Evolutionary Algorithms

The definition of $\Phi(k)$ in MCEA is given as follows. First, the scalar function value of search point $\mathbf{x}_k^i$ for each objective function is temporarily defined as

$$\phi'_j(\mathbf{x}_k^i) = \frac{f_{\text{worst}j}^k - f_j(\mathbf{x}_k^i)}{f_{\text{worst}j}^k - f_{\text{best}j}^k}, \ \forall j \in \{1,...,m\}, \tag{11}$$

where $f_{\text{best}j}^k$ and $f_{\text{worst}j}^k$ are the best and the worst values of each objective function:

$$\begin{cases} f_{\text{best}j}^k = \min\left\{ f_j(\mathbf{x}_k^i) \mid \forall i \in \{1,...,\lambda\} \right\} \\ f_{\text{worst}j}^k = \max\left\{ f_j(\mathbf{x}_k^i) \mid \forall i \in \{1,...,\lambda\} \right\} \end{cases}. \tag{12}$$

As the scalar function values of search points of the same rank should coincide, the scalar function value of search point $\mathbf{x}_k^i$ for each objective function is calculated as

$$\phi_j(\mathbf{x}_k^i) = \max\left\{ \phi'_j(\mathbf{x}_k^l) \mid \theta(\mathbf{x}_k^l) = \theta(\mathbf{x}_k^i), l \neq i, \forall l \in \{1,...,\lambda\} \right\}. \tag{13}$$

The scalar function value of each search point, $\phi(\mathbf{x}_k^i)$,

$$\phi(\mathbf{x}_k^i) = \sum_{j=1}^{m} \phi_j(\mathbf{x}_k^i), \tag{14}$$

With this $\Phi(k)$, the search algorithms are in canonical form represented as

$$X(k+1) = s_{\text{EA}}\left(X(k), \Phi(k) \right). \tag{15}$$

The first process in the optimization is the recombination where the temporary search point $\mathbf{x}'^i_k$ is created by

$$\mathbf{x}'^i_k = (1-\mu)\mathbf{x}'^i_k + \mu \mathbf{x}'^j_k, \tag{16}$$

In the equation, $j \neq i$, $\forall j \in \{1,...,\lambda\}$, and $\mu = N(0,\sigma^2)$. The mutation is then conducted to create a further temporary search point $\mathbf{x}''^i_k$ with the process

$$\mathbf{x}''^i_k = \text{rand}(\mathbf{x}_{\min}, \mathbf{x}_{\max}). \tag{17}$$

The next search point is copied as $\mathbf{x}_{k+1}^i = \mathbf{x}''^i_k$ with the probability given by

$$P_s(\mathbf{x}''^i_k) = \frac{\phi(\mathbf{x}''^i_k)}{\sum_{j=1}^{\lambda} \phi(\mathbf{x}''^i_k)}. \tag{18}$$

4 Numerical Examples

The optimization problem to find two parameters by minimizing the two objective functions:

$$f_1(\mathbf{x}) = \sum_{i=1}^{2} x_i^{\,2}\,,\ f_2(\mathbf{x}) = \sum_{i=1}^{2} \left(x_i - 1\right)^2 \tag{19}$$

was solved with four cases described in Table 1. The number of search points was 10, and each point was created randomly within the range $\left[\mathbf{x}_{\min}, \mathbf{x}_{\max}\right] = \left[-5, 5\right]$ in all the cases. The solution of the problem is known to be

$$\chi^* = \left\{ r\mathbf{z} \mid 0 \le r \le 1 \right\}, \tag{20}$$

where $\mathbf{z} = \left[1, 1\right]$. The most robust solution is thus $\mathbf{x}^* = \left[0.5, 0.5\right]$, which is located in the center of the solution space. The average distance of Pareto-optimal solutions from the solution space, derived as

$$e_{\text{total}} = \frac{1}{n\sqrt{2}} \sum_{i=1}^{r} \left| \left(\hat{x}_1\right)^i - \left(\hat{x}_2\right)^i \right|, \tag{21}$$

was used to evaluate the Pareto-optimal solutions obtained. Meanwhile, the distance of the computed single solution $\hat{\mathbf{x}}$ from $\mathbf{x}^*$, i.e.,

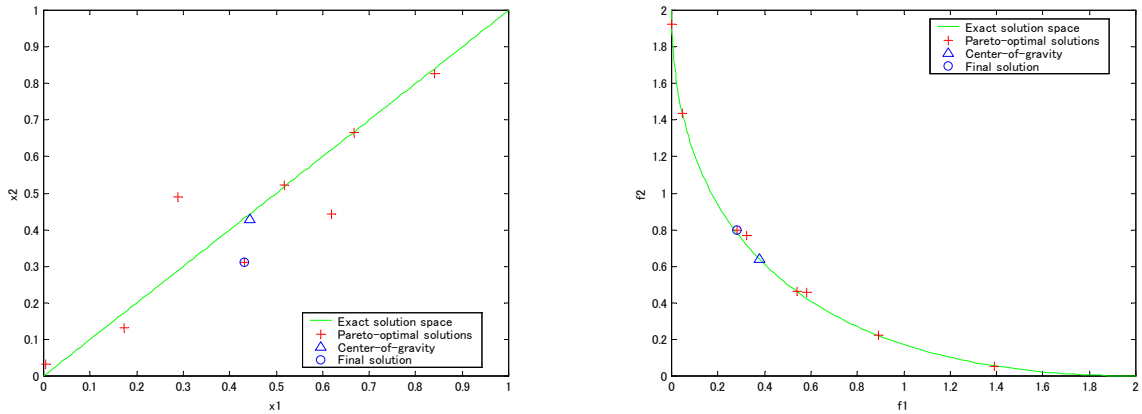
$$e_{\text{single}} = \left\| \hat{\mathbf{x}} - \mathbf{x}^* \right\| \tag{22}$$

was used to evaluate the final computed solution.

Table 2 lists the numerical results, while the final solutions in parameter and function spaces are shown in Figures 7-10. Results of Cases 1-3 first show that MQN method performs best, creating all the Pareto-optimal solutions almost on the solution line. In addition, the final solution by MQN method is closest to the exact robust solution, indicating that MQN method is most effective in search. The computation time taken by MCEA for 20 iterations is however only 11.8 % of that by MQN. Thus, the result by MCEA with 110 iterations, which consumed more time than the computation time of MQN, is also provided. The Pareto-optimal solutions of this case are still not as accurate as those by MQN, although improvement can be seen from the result after 20 iterations. The bottleneck of evolutionary algorithms is however the premature convergence. Search points are hard to get closer to the solution space when they are close enough. The result equivalent to the performance of MQN method was not achieved even after 500 iterations.

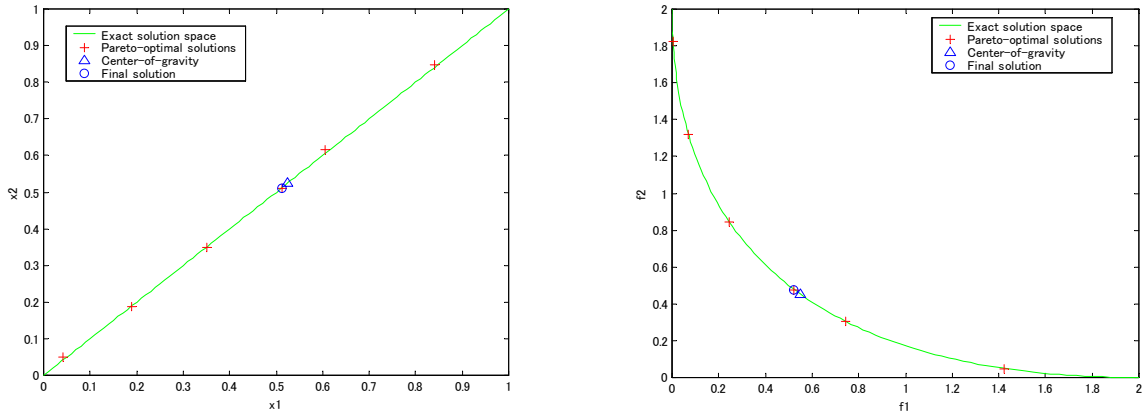
5 Conclusions

Human-like optimization has been presented as a novel technique for parametric computational design. MQN method has then been proposed as the most efficient multi-objective optimization method. Numerical results first indicate that MQN method is superior to MSD method and a conventional multi-objective optimization method implementing evolutionary algorithms (MCEA) by observing the distances of Pareto-optimal solutions from the exact solution space. Good approximate Pareto-optimal solutions, which could not be found by the conventional method even after 500 iterations was obtained only after 20 iterations. The robustness of the final solution derived by MQN method has been also confirmed through its comparison to the exact robust solution of the problem. As the proposed method is probabilistic, the results described in this paper may not be obtained consistently. However, the results at least indicate that the proposed method is a breakthrough to the multi-objective optimization and computational design. Further studies include more investigations of the proposed method as well as the application of the proposed optimization to the design of practical mechanical systems.



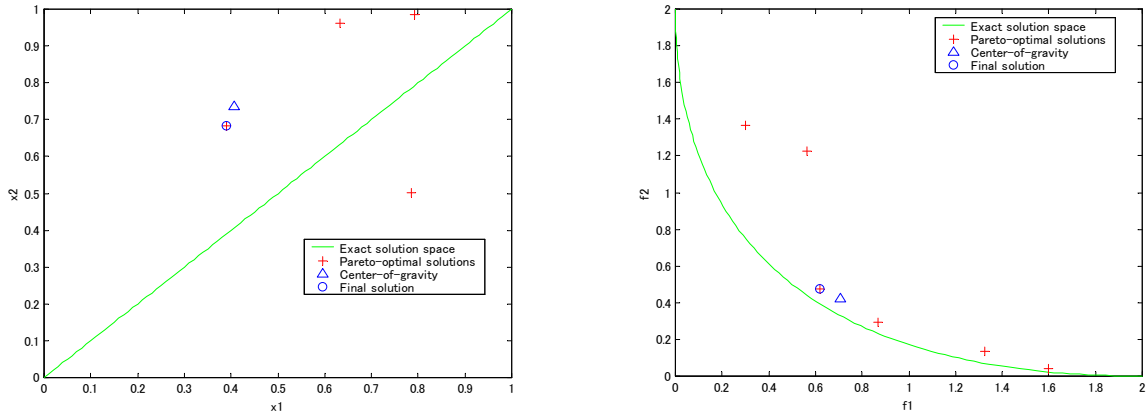
(a) Parameter space (b) Function space

Figure 7: MSD after 20 iterations



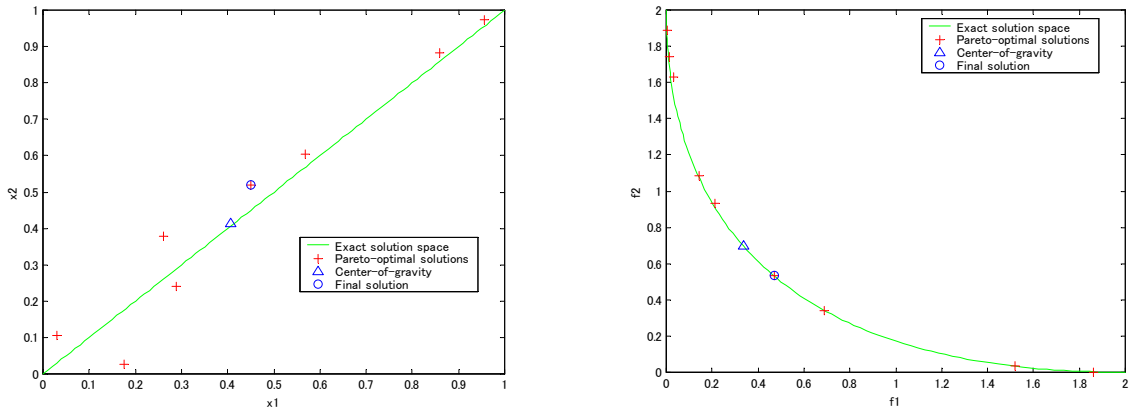
(a) Parameter space (b) Function space

Figure 8: MQN after 20 iterations



(a) Parameter space (b) Function space

Figure 9: MCEA after 20 iterations



(a) Parameter space (b) Function space

Figure 10: MCEA after 110 iterations

Table 1: Input parameters.

Case	Method	Iterations
1	MSD	20
2	MQN	20
3	MCEA	20
4	MCEA	20

Table 2: Numerical results.

Case	Comp. time [sec]	r	e_{total}	e_{single}
1	79.93	9	0.0381	0.1125
2	82.70	10	0.0046	0.0108
3	9.73	10	0.1796	0.3067
4	86.83	12	0.0428	0.0737

References

[1] S. Yoshimura, T. Kowalcyzk, T. Furukawa, G. Yagawa, An Automated CAE System for Multidisciplinary Structural Design and Its Application to Micromachines, Advances in Comp. Eng. Sci., (1997), 520-525.
[2] J. Oda, S. Kundu, Study of Structural Optimization Technique Using Evolutionary Cellular Automata, JSME Int. J., Vol. 42, (1999), 348-354.
[3] G.L. Lemhauser, A.H.G. Rinnooy Kan, M.J. Todd, Handbooks in Operations Research and Management Science Vol. 1: Optimization, Elsevier Science (1989).
[4] Y. Bard, Nonlinear Parameter Estimation, Academic Press (1974).
[5] C. Dixon, Nonlinear Optimisation, The English Universities Press (1972).
[6] T. Furukawa, Parameter Identification with Weightless Regularization, Int. J. Num. Meth. Eng., 52, (2001), 219-238.
[7] C.A. Coello, A Comprehensive Servey of Evolutionary-based Multi-objective Optimization Techniques, I. J. Knowl. Inf. Sys. 1(3), (1999), 269-308.
[8] C.M. Fonseca, P.J. Fleming, An Overview of Evolutionary Algorithms in Multi-objective Optimization, I. J. Evol. Comp., 3(1), (1995), 1-16.
[9] L.A. Zadeh, Fuzzy Sets, Inform. Cont., 8, (1965), 338-353.
[10] T. Furukawa, S. Yoshimura, G. Dissanayake, General Optimiser for Continuous Inverse Analysis, Inv. Prob. Eng. Mech. III (Eds: Tanaka and Dulikravich), Elsevier Science (2001), 281-290.
[11] P. Wolfe, The Symplex Method for Quadratic Programming, Econometrica, 27, (1959), 382-398.