Fourth International Workshop on Computational Intelligence & Applications
IEEE SMC Hiroshima Chapter, Hiroshima University, Japan, December 10 & 11, 2008

# Particle Swarm Optimization Combining Diversification and Intensification for Nonlinear Integer Programming Problems

Takeshi Matsui, Masatoshi Sakawa, Kosuke Kato and Koichi Matsumoto
Hiroshima University
1-4-1, Kagamiyama, Higashi-Hiroshima, 739-8527 JAPAN
email:{ tak-matsui, sakawa, kosuke-kato}@hiroshima-u.ac.jp, matsumoto@msl.sys.hiroshima-u.ac.jp

***Abstract*—In this research, focusing on nonlinear integer programming problems, we propose an approximate solution method based on particle swarm optimization proposed by Kennedy et al. And we developed a new particle swarm optimization method which is applicable to discrete optimization problems by incoporating a new method for generating initial search points, the rounding of values obtained by the move scheme and the revision of move methods. Furthermore, we showed the efficiency of the proposed particle swarm optimization method by comparing it with an existing method through the application of them into the numerical examples. Moreover we expanded revised particle swarm optimization method for application to nonlinear integer programming problems and showed more effeciency than genetic algorithm. However, variance of the solutions obtained by the PSO method is large and accuracy is not so high. Thus, we consider improvement of accuracy introducing diversification and intensification.**

## I. Introduction

In general, actual various decision making situations are formulated as large scale mathematical programming problems with many decision variables and constraints.

If a value of the decision variables is integer, the problem is called an integer programming problem. For integer programming problems, we can have optimal solution by application of the dynamic programming fundamentally. However, since optimization problems become larger and more complicated, a high speed and accurate optimization method is expected. In particular, for nonlinear integer programming problems (NLIP), there are not the general strict method or approximation method, such as branch and bound method for linear programming problems. In such a case, a solution method depended on property in problems is proposed. In recent years, a particle swarm optimization (PSO) method was proposed by Kennedy et al. [2] and has attracted considerable attention as one of promising optimization methods with higher speed and higher accuracy than those of existing solution methods. And Kato et al. showed the efficiency of improved PSO method than genetic algorithm for nonlinear programming problems [1].

Moreover we expanded revised particle swarm optimization method for application to NLIP and showed more effeciency than genetic algorithm [5]. However, variance of the solutions obtained by the PSO method is large and accuracy is not so high.

In this research, we focus on NLIP and consider improvement of accuracy combining diversification and intensification.

## II. Nonlinear integer programming problems

In this research, we consider general nonlinear integer programming problem with constraints as follows:

$$\left.\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & g_i(\boldsymbol{x}) \leq 0,\ \ i = 1, 2, \ldots, m \\ & l_j \leq x_j \leq u_j,\ \ j = 1, 2, \ldots, n \\ & \boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T \in Z^n \end{array}\right\} \quad (1)$$

where $f(\cdot)$, $g_i(\cdot)$ are convex or nonconvex real-valued functions, $l_j$ and $u_j$ are the lower bound and the upper bound of each decision variable $x_j$.

## III. Particle swarm optimization

Particle swarm optimization [2] method is based on the social behavior that a population of individuals adapts to its environment by returning to promising regions that were previously discovered [3]. This adaptation to the environment is a stochastic process that depends on both the memory of each individual, called particle, and the knowledge gained by the population, called swarm.

In the numerical implementation of this simplified social model, each particle has three attributes: the position vector in the search space, the current direction vector, the best position in its track and the best position of the swarm.

Step 1: Generate the initial swarm involving $N$ particles at random.

Step 2: Calculate the new direction vector for each particle based on its attributes.

Step 3: Calculate the new search position of each particle from the current search positon and its new direction vector.

Step 4: If the termination condition is satisfied, stop. Otherwise, go to Step 2.

To be more specific, the new direction vector of the $i$-th particle at time $t$, $\boldsymbol{v}_i^{t+1}$, is calculated by the following scheme introduced by Shi and Eberhart [7].

$$\boldsymbol{v}_i^{t+1} := \omega^t \boldsymbol{v}_i^t + c_1 R_1^t (\boldsymbol{p}_i^t - \boldsymbol{x}_i^t) + c_2 R_2^t (\boldsymbol{p}_g^t - \boldsymbol{x}_i^t) \quad (2)$$

In (2), $R_1^t$ and $R_2^t$ are random numbers between 0 and 1, $\boldsymbol{p}_i^t$ is the best position of the $i$-th particle in its track at time $t$ and $\boldsymbol{p}_g^t$ is the best position of the swarm at time $t$. There are three parameters such as the inertia of the particle $\omega^t$, and two parameters $c_1$, $c_2$.

Then, the new position of the $i$-th particle at time $t$, $\boldsymbol{x}_i^{t+1}$, is calculated from (3).

$$\boldsymbol{x}_i^{t+1} := \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} \tag{3}$$

where $\boldsymbol{x}_i^t$ is the current position of the $i$-th particle at time $t$. After the $i$-th particle calculates the next search direction vector $\boldsymbol{v}_i^{t+1}$ by (2) in consideration of the current search direction vector $\boldsymbol{v}_i^t$, the direction vector going from the current search position $\boldsymbol{x}_i^t$ to the best search position in its track $\boldsymbol{p}_i^t$ and the direction vector going from the current search position $\boldsymbol{x}_i^t$ to the best search position of the swarm $\boldsymbol{p}_g^t$, it moves from the current position $\boldsymbol{x}_i^t$ to the next search position $\boldsymbol{x}_i^{t+1}$ calculated by (3). In general, the parameter $\omega^t$ is set to large values in the early stage for global search, while it is set to small values in the late stage for local search. For example, it is determined as:

$$\omega^t := \omega^0 - \frac{t \cdot (\omega^0 - \omega^{T_{\max}})}{0.75 \cdot T_{\max}} \tag{4}$$

where $t$ is the current time, $T_{\max}$ is the maximal value of time, $\omega^0$ is the initial value of $\omega^t$ and $\omega^{T_{\max}}$ is the final value of $\omega^t$.

The search procedure of PSO is shown in Fig. 1. If the next search position of the $i$-th particle at time $t$, $\boldsymbol{x}_i^{t+1}$, is better than the best search position in its track at time $t$, $\boldsymbol{p}_i^t$, i.e., $f(\boldsymbol{x}_i^{t+1}) \leq f(\boldsymbol{p}_i^t)$, the best search position in its track is updated as $\boldsymbol{p}_i^{t+1} := \boldsymbol{x}_i^{t+1}$. Otherwise, it is updated as $\boldsymbol{p}_i^{t+1} := \boldsymbol{p}_i^t$. Similarly, if $\boldsymbol{p}_i^{t+1}$ is better than the best position of the swarm, $\boldsymbol{p}_g^t$, i.e., $f(\boldsymbol{p}_i^{t+1}) \leq f(\boldsymbol{p}_g^t)$, then the best search position of the swarm is updated as $\boldsymbol{p}_g^{t+1} := \boldsymbol{p}_i^{t+1}$. Otherwise, it is updated as $\boldsymbol{p}_g^{t+1} := \boldsymbol{p}_g^t$.

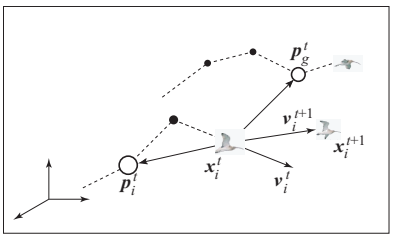


Fig. 1. Movement of an individual.

In the original PSO method, however, there are drawbacks that it is not directly applicable to constrained problems and its liable to stopping around local solutions.

To deal with these drawbacks of the original PSO methods, Kato et al. incorporated the bisection method and a homomorphous mapping to carry out the search considering constraints. In addition, Kato et al. incorporated the multiple stretching technique and modified move schemes of particles to restraining the stopping around local solutions [1].

Moreover we expanded revised particle swarm optimization method for application to NLIP (rPSONLIP) [5]. In rPSONLIP, we enabled application of the revised PSO method to NLIP by incorporating using integer random number in generating initial search position and rounding search direction vector $\boldsymbol{v}_i^{t+1}$ in updating equation (3). However, we simply make the search direction vector integer value and all elements of the search direction vector become 0. Thus, on the element that absolute value is maximum in elements of the search direction vector before rounding, we set the search direction vector 1 or -1 depending on the plus or minus. Therefore, we revised that all of the particles always move. The process of rPSONLIP can be shown in Fig. 2.

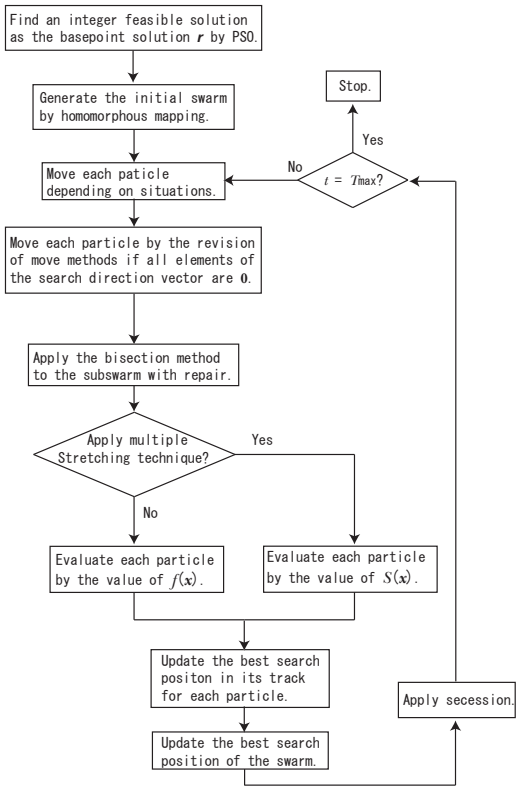


Fig. 2. The algorithm of rPSONLIP.

## IV. rPSO COMBINING DIVERSIFICATION AND INTENSIFICATION

We expanded revised particle swarm optimization method for application to NLIP and showed more effeciency than genetic algorithm [5]. However, variance of the solutions obtained by rPSONLIP is large and accuracy is not so high. In this research, we consider improvement of accuracy of rPSONLIP. At first we introduce new move scheme with loop in order to prevent a particle from stopping around boundary. Next, for diversification (reinforcement of global search), we introduce new move scheme to restrain to stopping around local optimal solutions. Furthermore, for intensification (reinforcement of convergence search in promising region), we introduce local search in the best search position of the swarm.

### A. new move scheme with loop

To deal with drawbacks, we analyzed search process in detail and found that the decision variable of some particles was fixed on the upper or lower bound (boundary) on the way of search. Such a particle is easy to stop at the boundary and causes depression of search efficiency. Moreover, for example in such a situation, in case that the decision variable taking upper (lower) bound value at the current search position takes lower (upper) bound value at the optimal solution, it is not easy for the particle moving in feasible region to move around the optimal solution. Thus, moving with loop from upper to lower or from lower to upper for the decision variable that absolute value is maximum in the element of the search direction vector to boundary outside on the decision variable fixed in the boundary (upper or lower bound), we restrain to the stopping around the boundary for a particle (Fig. 3).

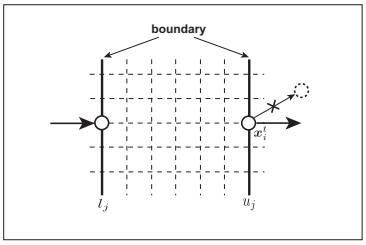


Fig. 3. Move with loop.

We show the results of the application to quadratic integer programming maximizing problem with $n$ = 50 and $m$ = 5 in Table I. In these experiments, we set the maximal search generation number $T_{\max}$ = 5000 and the swarm size $N$ = 100. And the number of trial is 40.

TABLE I
RESULTS OF THE APPLICATION TO THE PROBLEM WITH $n$ = 50 AND $m$ = 5

| method | rPSONLIP with loop | rPSONLIP |
|---|---|---|
| best | 39143.5 | 39204.0 |
| mean | 38077.9 | 37658.5 |
| worst | 36986.5 | 36010.0 |
| time (sec) | 115.677 | 91.648 |

From Table I, incorporating new move scheme with loop proposed in this research, we can improve efficiency of rPSONLIP. On the other hand, the best value of rPSONLIP with loop is inferior to rPSONLIP and it turns worse on accuracy.

### B. diversification

There is often that the best search position of the swarm is not updated for a long generation in PSO. If the best search position of the swarm is not global optimal solution, a particle has to move from this search position. Kato et al. [1] incorporated multiple stretching technique to restrain to stopping around local optimal solutions. and we apply it in rPSONLIP. However, for the problem that the decision variable takes integer, this technique does not work well, and there is the case that a particle stops around local optimal solutions. We propose new move scheme to restrain to stopping around local optimal solutions if the decision variable takes integer. To be more specific, if the best search position of the swarm is not updated for a long generation, we multiply the third term of right-hand side in (2) by -1 and incorporate the direction away from $\boldsymbol{p}_g^t$, then decide next search direction vector $\boldsymbol{p}_i^{t+1}$ following equation (Fig. 4).

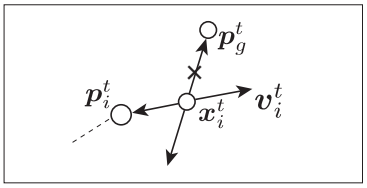


Fig. 4. New move scheme if the best search position of the swarm is not updated for a long generation.

$$\boldsymbol{v}_i^{t+1} = \omega^t \boldsymbol{v}_i^t + c_1 R_1^t (\boldsymbol{p}_i^t - \boldsymbol{x}_i^t) - c_2 R_2^t (\boldsymbol{p}_g^t - \boldsymbol{x}_i^t) \quad (5)$$

In this move scheme we can restrain to stopping around current best search position of the swarm. However, convergence to plural local optimal solutions may occur. Thus, we prevent convergence to plural local optimal solutions by adding the penalty depending on distance from the local optimal solutions. To be more specific, for $q$ local optimal solutions $\bar{\boldsymbol{x}}_k, k = 1, \ldots, q$, we consider the function $S(\boldsymbol{x})$ as follows:

$$d_k(\boldsymbol{x}) = \|\boldsymbol{x} - \bar{\boldsymbol{x}}_k\| \quad (6)$$

$$P_k(\boldsymbol{x}) = \begin{cases} G & d_k(\boldsymbol{x}) \leq 1 \\ 0 & d_k(\boldsymbol{x}) > 1 \end{cases} \quad (7)$$

$$S(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{k=1}^{q} P_k(\boldsymbol{x}) \quad (8)$$

Here, $d_k(\boldsymbol{x})$ is distance between current search position and the local optimal solution $\bar{\boldsymbol{x}}_k$. $P_k(\boldsymbol{x})$ is the function adding the penalty if $d_k(\boldsymbol{x}) \leq 1$, if not, not adding the penalty. $S(\boldsymbol{x})$ is the new evaluation value added summation of the penalty to the objective function value depending on the distance between current search position and $q$ local optimal solutions. Using $S(\boldsymbol{x})$, the evaluation value includes the penalty in a certain region from local optimal solutions and we can prevent convergence to local optimal solutions. We show the results of the application of rPSONLIP with loop and diversification to quadratic integer programming maximizing problem with $n$ = 50 and $m$ = 5 in Table II. In these experiments, we set the maximal search generation number $T_{\max}$ = 5000 and the swarm size $N$ = 100. And the number of trial is 40.

From Table II, incorporating modified new move scheme, efficiency and accuracy of rPSONLIP are improved. In addition, we can reduce computational time more compared that we apply multiple stretching technique.

TABLE II
RESULTS OF THE APPLICATION TO THE PROBLEM WITH $n$ = 50 AND $m$ = 5

| method | rPSONLIP with loop and divesification | rPSONLIP with loop |
|---|---|---|
| best | 39214.5 | 39143.5 |
| mean | 38265.5 | 38077.9 |
| worst | 37193.5 | 36986.5 |
| time (sec) | 78.664 | 115.677 |

### C. intensification

For further improvement of efficiency and accuracy, intensification (reinforcement of convergence search in promising region) is needed. In this research, we introduce local search in the best search position of the swarm since we regard the region around the best search position of the swarm as the promising region. To be more concrete, a particle searches neighborhood in the best search position of the swarm if the best search position of the swarm is not updated for a long generation and moves the best search position besides that of the swarm (Fig. 5). However, visitation occurs if the best search position of the swarm just before is the local optimal solution. Thus, we introduce the method to forbid this situation (tabu).

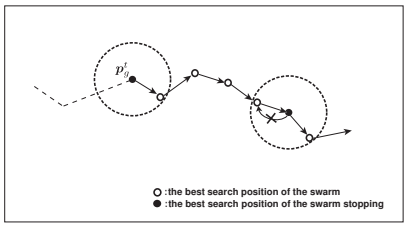


Fig. 5. Local search in the best search position of the swarm.

We show the results of the application of rPSONLIP with loop, diversification and intensificaiton (rPSODINLIP) to quadratic integer programming maximizing problem with $n$ = 50 and $m$ = 5 (same problem as Table I, II) in Table III. In these experiments, we set the maximal search generation number $T_{\max}$ = 5000 and the swarm size $N$ = 100. And the number of trial is 40.

TABLE III
RESULTS OF THE APPLICATION TO THE PROBLEM WITH $n$ = 50 AND $m$ = 5

| method | rPSODINLIP (proposed) | rPSONLIP with loop and diversification |
|---|---|---|
| best | 39258.5 | 39214.5 |
| mean | 38585.3 | 38265.5 |
| worst | 37193.5 | 37193.5 |
| time (sec) | 79.116 | 78.664 |

From Table III, the proposed rPSODINLIP is better than rPSONLIP with loop and diversificiation with respect to the best objective function value, the mean one. From these results, efficiency and accuracy are improved more.

### D. The procedure of rPSO combining diversification and intensification

The procedure of the proposed rPSO combininig diversification and intensification for NLIP (rPSODINLIP) is summarized as follows.

Step 1: Find an integer feasible solution by PSO in consideration of the degree of violation of constraints, and use it as the basepoint of the homomorphous mapping, $\boldsymbol{r}$. Let $t := 0$ and go to Step 2.

Step 2: Generate feasible initial integer search positions based on the homomorphous mapping proposed by Koziel and Michalewicz [4]. To be more specific, map $N$ points generated randomly in the $n$ dimensional hypercube $[-1, 1]^n$ to the feasible region $X$ using the homomorphous mapping, and let these points in $X$ be initial search positions $\boldsymbol{x}_i^0$, $i = 1, \ldots, N$. In addition, let the initial search position of each particle, $\boldsymbol{x}_i^0$, be the initial best position of the particle in its track, $\boldsymbol{p}_i^0$, and let the best position among $\boldsymbol{x}_i^0$, $i = 1, \ldots, N$ be the initial best position of the swarm, $\boldsymbol{p}_g^0$. Go to Step 3.

Step 3: Calculate the value of $\omega^t$ by (4). For each particle, using the information of $\boldsymbol{p}_i^t$ and $\boldsymbol{p}_g^t$, determine the direction vector $\boldsymbol{v}_i^{t+1}$ to the next search position $\boldsymbol{x}_i^{t+1}$ by the modified move schemes proposed by Kato et al. [1]. Next, move it to the next search position by (3) and go to Step 4.

Step 4: If the particle does not move since the current search position and the next search position are the same either, revise $\boldsymbol{v}_i^{t+1}$ to 1 or $-1$ depending on the plus and minus on the element that the absolute value is maximum in the element of $\boldsymbol{v}_i^{t+1}$ before revising an integer value. Go to Step 5.

Step 5: Move with loop from upper to lower or from lower to upper for the decision variable that absolute value is maximum in the element of the search direction vector to boundary outside if the decision variable fixed in the boundary (upper or lower bound). Go to Step 6.

Step 6: Check if the current search position of each paticle in the subswarm with repair based on the bisection method, $\boldsymbol{x}_i^{t+1}$, is feasible. If not, repair it to be feasible using the bisection method, and go to Step 7.

Step 7: Determine whether the new move scheme to restrain to stopping around local optimal solutions is applied or not. If it is applied, go to Step 8. Otherwise, go to Step 9.

Step 8: Determine the direction ector $\boldsymbol{v}_i^{t+1}$ to the next search position $\boldsymbol{x}_i^{t+1}$ by the modified new move scheme explained in section IV-B. Next, move it to the next search positon by (3) and evaluate each particle by the value of $S(\cdot)$ for $\boldsymbol{x}_i^{t+1}$, $i = 1, \ldots, N$. Go to Step 10.

Step 9: Evaluate each particle by the value of $f(\cdot)$ (objective function) for $\boldsymbol{x}_i^{t+1}$, $i = 1, \ldots, N$. Go to Step 10.

Step 10: If the evaluation function value $S(\boldsymbol{x}_i^{t+1})$ or $f(\boldsymbol{x}_i^{t+1})$ is better than the evaluation function value for the best search position of the particle in its track, $\boldsymbol{p}_i^t$, update the best search position of the particle in its track as $\boldsymbol{p}_i^{t+1} := \boldsymbol{x}_i^{t+1}$. If not, let $\boldsymbol{p}_i^{t+1} := \boldsymbol{p}_i^t$ and go to Step 11.

Step 11: If the minimum of $S(\boldsymbol{x}_i^{t+1})$, $i = 1, \ldots, N$ or the minimum of $f(\boldsymbol{x}_i^{t+1})$, $i = 1, \ldots, N$ is better than the

evaluation function value for the current best search position of the swarm, $\boldsymbol{p}_g^t$, update the best search position of the swarm as $\boldsymbol{p}_g^{t+1} := \boldsymbol{x}_{i_{\min}}^{t+1}$. Otherwise, let $\boldsymbol{p}_g^{t+1} := \boldsymbol{p}_g^{t+1}$ and go to Step 12.

Step 12: Determine whether the local search is applied or not. If it is applied, go to Step 13. Otherwise, go to Step 14.

Step 13: Search neighborhood in the best search position of the swarm and update the best search position of the swarm. Go to Step 14.

Step 14: If the condition of the secession is satisfied, apply the secession to every particle according to a given probability, and go to Step 15.

Step 15: Finish if $t = T_{\max}$ (the maximal value of time). Otherwise, let $t := t + 1$ and return to Step 3.

## V. Numerical examples

We apply the proposed PSO (rPSODINLIP) and genetic algorithm for nonlinear integer programming problems (GANLIP) [6] which is one of the exisiting efficient methods, to two nonlinear integer programming maximizing problems with different scale. The number of trial is 40 for rPSODINLIP and GANLIP. Tables IV and V show the results obtained by both methods: the best objective function value of 40 trials, the mean one, the worst one, and the mean computational time. In these experiments, the parameters of GANLIP are set as the population size $N$ = 100. On the other hand, the pareameters of rPSODINLIP are set as the size of the swarm $N$ = 100. And we set the maximal search generation number $T_{\max}$ = 5000 for all problems.

For the problem, as shown in Table IV, the proposed rPSODINLIP can always obtain the optimal value (0.942), while GANLIP cannnot; and the mean computational time of rPSODINLIP is shorter than that of GANLIP. For the problem, as shown in Table V, the proposed rPSODINLIP is better than GANLIP with respect to the best objective function value, the mean one, the worst one, and the mean computational time.

From these results, it is indicated that the proposed rPSODINLIP is superior to GANLIP and rPSODINLIP is promising as an optimization method for nonlinear integer programming problems.

TABLE IV
Results of the application to the problem with $n$ = 20 and $m$ = 3

| method | rPSODINLIP | GANLIP |
|---|---|---|
| best | 0.942 | 0.939 |
| mean | 0.942 | 0.932 |
| worst | 0.942 | 0.923 |
| time (sec) | 39.994 | 76.020 |

TABLE V
Results of the application to the problem with $n$ = 3 and $m$ = 3

| method | rPSODINLIP | GANLIP |
|---|---|---|
| best | 0.915 | 0.906 |
| mean | 0.914 | 0.901 |
| worst | 0.914 | 0.896 |
| time (sec) | 55.748 | 149.026 |

## VI. Conclusion

In this research, focusing on a paritcle swarm optimization, we considered its application to NLIP. In order to improve accuracy of rPSONLIP, we incorporated new move scheme with loop, reinforcement of global search (diversificaiton) and reinforcement of convergence search in promising region (intensification). We showed the efficiency of the proposed rPSODINLIP method by comparing it with an existing method, GANLIP, through their application in some numerical examples.


## Acknowledgment

This research was partially supported by The 21st Century COE Program on “Hyper Human Technology toward the 21st Century Industrial Revolution”.


## References


[1] K. Kato, T. Matsui, M. Sakawa and K. Morihara, An approximate solution method based on particle swarm optimization for nonlinear programming problems, *Journal of Japan Society for Fuzzy Theory and Intelligent Informatics*, Vol.20, No.3, pp.399–409, 2008.

[2] J. Kennedy and R.C. Eberhart, Particle swarm optimization, *Proceedings of IEEE International Conference on Neural Networks*, pp. 1942–1948, 1995.

[3] J. Kennedy and W. M. Spears, Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator, *Proceedings of IEEE International Conference on Evolutionary Computation*, pp.74–77, 1998.

[4] S. Koziel and Z. Michalewicz, Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization, *Evolutionary Computation*, Vol.7, No.1, pp.19–44, 1999.

[5] T. Matsui, K. Kato, M. Sakawa, T. Uno and K. Matsumoto, Particle swarm optimization for nonlinear integer programming problems, *Proceedings of International MultiConference of Engineers and Computer Scientists 2008*, pp. 1874–1877, 2008.

[6] M. Sakawa, K. Kato, M.A.K. Azad and R. Watanabe, A genetic algorithm with double string for nonlinear integer programming problems, *IEEE SMC 2005 Conference Proceedings*, pp. 3281–3286, 2005.

[7] Y.H. Shi and R.C. Eberhart, A modified particle swarm optimizer, *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 69–73, 1998.