Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 Vol II
IMECS 2008, 19-21 March, 2008, Hong Kong

Particle Swarm Optimization for Nonlinear Integer Programming Problems

Takeshi Matsui, Kosuke Kato, Masatoshi Sakawa, Takeshi Uno, Koichi Matsumoto *

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. To be more specific, we develop 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 show 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.**

Keywords: ***nonlinear integer programming problem, particle swarm optimization, rounding***

1 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, 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. [1] and has attracted considerable attention as one of promising optimization methods with higher speed and higher accuracy than those of existing solution methods. And we showed the efficiency of improved PSO method than genetic algorithm for nonlinear programming problems [3].

However there are almost no application of the PSO method for integer programming problem.

*Graduate School of Engineering, Hiroshima University, Email: matsui@msl.sys.hiroshima-u.ac.jp

In this research, we focus on nonlinear integer programming problems and consider an application of the PSO method.

2 Nonlinear integer programming problems

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

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

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.

3 Particle swarm optimization

Particle swarm optimization [1] 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 [2]. 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. The process can be outlined as follows and shown in Fig. 1.
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

ISBN: 978-988-17012-1-3

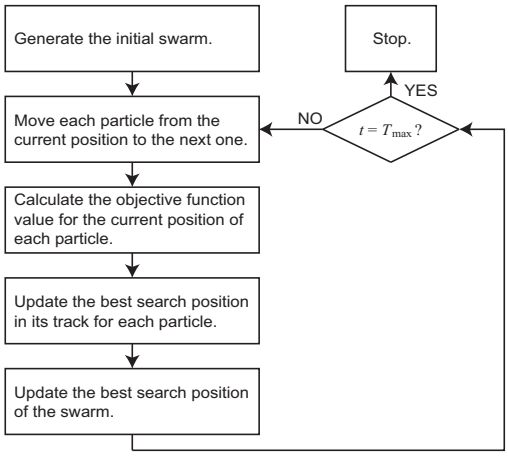

Figure 1: The basic algorithm of PSO.

scheme introduced by Shi and Eberhart [4].

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

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 ω^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).

$$\boldsymbol{x}_i^{t+1} := \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} \qquad (3)$$

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 ω^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:

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

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

The search procedure of PSO is shown in Fig. 2. 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 $\boldsymbol{p}_g^{t+1} := \boldsymbol{p}_g^t$.

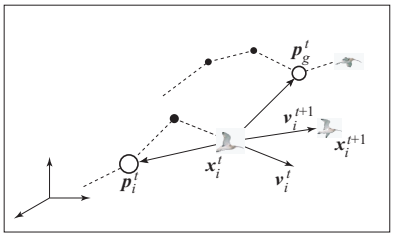

Figure 2: Movement of an individual.

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, we incorporated the bisection method and a homomorphous mapping to carry out the search considering constraints. In addition, we incorporated the multiple stretching technique and modified move schemes of particles to restraining the stopping around local solutions [3].

4 Improvement of the PSO method for nonlinear integer programming problems

In this research, to apply the PSO method to nonlinear integer programming problems, we devise generating initial search points, rounding of values obtained by the move scheme and the revision of move methods.

4.1 Generating initial search points

In the original PSO method, we generate real value at random in the search space. Since the problem focusing in this research is for decision variables to be an integer value, we modify to generate integer random value in the search space.

4.2 Rounding of values obtained by the move scheme

In general, since the value obtained by the move scheme of the PSO method is noninteger value, we cannot apply the PSO method to nonlinear integer programming problems. Thus, we revise decision variables to apply to nonlinear integer programming problems by the rounding of values obatained by the move scheme.

Table 1: Results of application of the PSO method (10 times)

best	-196.656
average	-189.033
worst	-162.004
time (sec)	6.504

4.3 Revision of move methods

Table 1 shows the result of the application in an nonlinear integer programming problem has 30 decision variables. The optimal solution of this problem is $(7, 7, \ldots, 7)^T$ and the optimal value is -196.656. From Table 1, on efficiency of evaluating a solution, the precision of solutions is shown since an optimal value is obtained as the best value. On the other hand, we cannot regard the PSO method adequately on accuracy of solutions. Thus, we analyzed the search process of the PSO method in detail, we could find out that a part of the swarm did not move on the search process. This is attributed to all elements becoming 0 when we revise search direction vector element into an integer value. Therefore we revise move methods to make certain that all particle move when all elements of the search direction vector $\boldsymbol{v}_i^{t+1}$. To be concrete, we revise $\boldsymbol{v}_i^{t+1}$ into 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. We show the result of the application of the revised PSO (rPSONLIP) method to the same problem in Table 2.

Table 2: Results of application of the rPSONLIP method (10 times)

best	-196.656
average	-196.220
worst	-192.937
time (sec)	6.394

From Table 2, introducing the revision of move methods, the improvement of the PSO method is shown on accuracy.

4.4 The procedure of revised PSO for nonlinear integer programming problems

The procedure of the revised PSO proposed in this paper summarized as follows and is shown in Fig. 3.
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. 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 ω^t by eq.(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 explained in section 4.2. Next, move it to the next search position by eq.(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: 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 6.
Step 6: Determine whether the multiple stretching technique is applied or not. If it is applied, go to Step 7. Otherwise, go to Step 8.
Step 7: Apply the multiple stretching technique, i.e., each particle is evaluated by the value of $S(\cdot)$ for $\boldsymbol{x}_i^{t+1}$, $i = 1, \ldots, N$.
Step 8: Evaluate each particle by the value of $f(\cdot)$ (objective function) for $\boldsymbol{x}_i^{t+1}$, $i = 1, \ldots, N$. Go to Step 9.
Step 9: 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 10.
Step 10: 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 11.
Step 11: If the condition of the secession is satisfied, apply the secession to every particle according to a given probability, and go to Step 12.
Step 12: Finish if $t = T_{\max}$ (the maximal value of time). Otherwise, let $t := t + 1$ and return to Step 3.

5 Numerical example

In order to show the efficiency of the propsed PSO (rPSONLIP), we apply the original rPSO [3] and the pro-

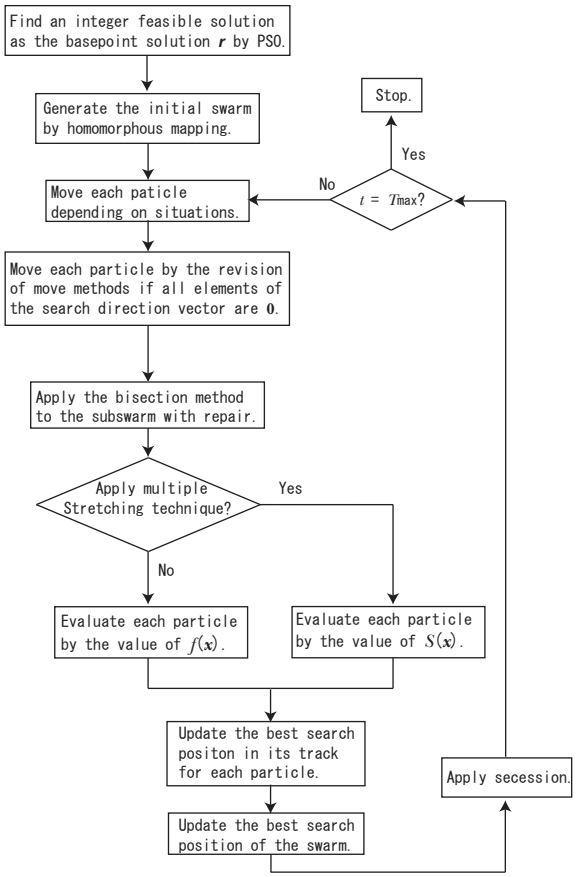

Figure 3: The algorithm of the proposed PSO method.

posed PSO (rPSONLIP) to nonlinear integer programming problems. The results obtained by these two methods are shown in table 3. In these experiments, we set the swarm size $N = 70$, the maximal search generation number $T_{\max} = 5000$, the inertia weight initial value $\omega^0 = 1.2$, the inertia weight last one $\omega^{T_{\max}} = 0.1$, weight parameters $c_1 = c_2 = 2$, R_1^t, R_2^t are uniform random number in the interval [0, 1].

Table 3: Results for a problem with $n = 40$ and $m = 22$

method	rPSONLIP	rPSO [3]
best	−4360.209	−4340.597
average	−3892.419	−3787.965
worst	−3363.007	−3084.424
time (sec)	21.148	19.749

From Table 3, in the application of rPSONLIP, we can get better solutions in the sense of best, average, worst and the difference between best and worst than those obtained by rPSO [3]. Therefore, it is indicated that the proposed PSO (rPSONLIP) is superior to rPSO for nonlinear integer programming problems.

6 Conclusions

In this research, focusing on a particle swarm optimization (PSO) method, we considered its application to constrained nonlinear integer programming problems. In order to deal with an integer value, we incorporated a new method for generating initial search points, the rounding of values obtained by the move scheme and the revision of move methods. And 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.

Acknowledgment

This research was partially supported by The 21st Century COE Program on “Hyper Human Technology toward the 21st Century Industrial Revolution”. This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (C), 18510127, 2006.

References

[1] J. Kennedy, R.C. Eberhart, “Particle swarm optimization,” *IEEE International Conference on Neural Networks*, Piscataway, NJ, pp. 1942–1948, 11/95.

[2] J. Kennedy, W. M. Spears, “Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator,” *IEEE Int. Conf. Evolutionary Computation*, Anchorage, Alaska, pp. 74–77, 5/98.

[3] T. Matsui, K. Kato, M. Sakawa, T. Uno, K. Morihara, “Particle swarm optimization based heuristics for nonlinear programming problems,” *International MultiConference of Engineers and Computer Scientists 2007*, Hong Kong, China, pp. 2312–2317, 3/07.

[4] Y.H. Shi, R.C. Eberhart, “A modified particle swarm optimizer,” *IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, pp. 69–73, 5/98.