Journal of Information & Computational Science 10:9 (2013) 2633–2640 June 10, 2013
Available at http://www.joics.com

Constrained Problem Optimization Based on Self-adaptive Particle Swarm Optimization with Novel Penalty Mechanisms ⋆

Jinyin Chen*, Dongyong Yang

College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China

Abstract

Penalty mechanism is widely used for dealing with constrained optimization problems, in this paper novelty mechanism combined with self-adaptive particle swarm optimization is brought up. Based on traditional H, J and P strategy, improved approaches are adopted for coping with constrained problems. On the other hand, in order to improve PSO's exploratory capacity, turbulence operations are used. Compared with traditional PSO, self-adaptive parameters are applied to improve PSO's expanding capacity. Finally several benchmark functions testify the high performance of the brought up algorithm complexities with GA, and their algorithm complexity and time complexities are analyzed in detail.

Keywords: PSO; Improved H, J, P Strategy; Exploratory Capacity; Penalty Mechanism; Turbulence Operations

1 Introduction

Particle Swarm Optimization (PSO) was firstly brought up in 1995 by Kennedy and Eberhart, which has attracted lots of concerns in last decades. PSO has been widely applied for engineering optimization problems, especially numerical optimization problems [1]. Constrained Optimization (CO) problems are the mainstream research area for wider practical applications, so PSO has developed as an efficient tool for constrained optimization recently [2]. Penalty mechanism has been proved as an efficient strategy for dealing with constrained problem, especially combined with GA, Ant clone algorithm, etc. [3]. For the reason that GA has more complex operations compared with other optimization algorithm, so we follow the idea that combine penalty mechanism with improved PSO to unconstrained problems by penalty functions in optimizing Co.

In this paper, in order to improve the exploratory capacity of PSO, turbulence operation is adopted to help particles explore in a wider area during the late stage of evolution. Based on traditional penalty functions, three improved penalty function mechanisms are brought up to

⋆Project supported by the Nature Science Foundation of Zhejiang Province (No. Y1100378).
*Corresponding author.
Email address: chenjinyin@163.com (Jinyin Chen).

1548–7741 / Copyright © 2013 Binary Information Press
DOI: 10.12733/jics20101816

implement PSO for constrained optimization. According to traditional H strategy, self-adaptive parameter adjustment operation is adopted in this paper to improve algorithm performance dealing with different optimization problems. Simplified P strategy is implemented for practical applications. In the last part of this paper, different benchmark functions are optimized based on self-adaptive PSO with novel penalty mechanisms to testify its high performance compared with GA.

2 Turbulence Operation in PSO

Since standard PSO was put forward by Kennedy and Eberhart in 1995 for the first time [6]. However premature convergence is a vital problem for PSO. According to experiments of optimizing benchmark functions, it is found that standard PSO is almost get convergence much earlier than maximum iteration, reflecting that solution diversity totally disappears during the optimizing process and PSO itself has no capability to maintain certain diversity of solution for better optimum. The main principle of turbulence probability is that it decreases accompanied with time and defined as follows.

$$temp = current_{generation}/total_{generation} \tag{1}$$

$$prob_{turbulence} = temp^{1.7} - 2.0 \times temp + 1.0 \tag{2}$$

While at the beginning of optimization, particles fly in a rather broad area to get a better position, and with time goes by the majority of particles get static and their velocities are zero. Turbulence operation will assist particles get random velocity to keep on searching for better optimum. Turbulence is adopted to overcome the premature and the steps of PSO with turbulence operations are as bellow.

Step 1. Initialize population and calculate fitness as F=$objevtFunctionValue$+$penaltyValue$. Fitness calculation of particles includes object function value and penalty function value.

Step 2. Turbulence depends on dynamic probability calculated by Eq. (1) and Eq. (2). Turbulence operations are adopted to prevent PSO from premature convergence by pouring new randomly generated solutions into solution space to maintain diversity. The disturb values are in range of $[-1, 1]$. Change each dimension of particle's position x to x' according to Eq. (3), Eq. (4), Eq. (5). If particle's fitness is better than the old one, then replace the old one.

$$x_i^{'} = (x_i - \triangle\gamma) + \mu_i \times \triangle\gamma\beta^{(k)}(1 - \beta^{(k)}) \tag{3}$$

$$\beta^{(k)} = \mu_i\beta^{(k-1)}(1 - \beta^{(k-1)}) \tag{4}$$

$$\beta^{(0)} = \frac{(x_i^{'} + \triangle\gamma_i - (x_i - \triangle\gamma))}{\triangle\gamma} \tag{5}$$

Step 3. Particle fly based on formula (6-7) to update velocity and position as standard PSO.

$$V_i^{k+1} = \omega \times V_i^k + c_1 \times rand_1^k \times (pbest_i^k - X_i^k) + c_2 \times rand_2^k \times (gbest_i^k - X_i^k) \tag{6}$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \tag{7}$$

Step 4. If the global best particle is the solution for the problem or the maximum iteration is reached, then end the algorithm, else transfer to Step 2.

3 Penalty Function Strategies

The CO problem can be represented as the following nonlinear programming problem: Minimize $f(\overrightarrow{x}), \overrightarrow{x} \in S^n$, where S is the solution space, including feasible region and infeasible region. Subject to equalities $E_i(\overrightarrow{x}) = 0, i = 1, \cdots, m$, and inequalities constraints $I_i(\overrightarrow{x}) \leq 0, i = m+1, \cdots, p$. In some particular optimizing situations, inequalities constraints can be represented as $E_i(\overrightarrow{x}) \leq 0$ instead of $E_i(\overrightarrow{x}) = 0$, where E is tolerance allowed (a very small number) and decided by the demanded precision.

Three classic strategies here selected as the basic penalty function for optimization. In this approach, a constrained problem is transformed into a non-constrained one. The function under consideration is transformed as follows:

$$F(\overrightarrow{x}) = \begin{cases} f(\overrightarrow{x}) & \text{when } \overrightarrow{x} \geqslant 0x \in feasibleRegion, \\ f(\overrightarrow{x}) + penalty(\overrightarrow{x}) & \text{otherwise,} \end{cases} \tag{8}$$

and the optimizing object is transformed to minimum problem as described as minimize $f(\overrightarrow{x})$, $\overrightarrow{x} \in S^n$. Following is the analysis of H strategy [6], J strategy [7] and P strategy [8], and respective improvements on them.

3.1 H Strategy for PSO

Penalty functions can be classified into two main categories: stationary and non-stationary. Stationary penalty functions adopts fixed value of penalty during the process of minimization, while in contrast, the non-stationary penalty functions, penalty values are modified for different cases during optimization. H strategy penalty is a kind of stationary mechanism for its fixed value during optimization as follows:

$$F(\overrightarrow{x}) = f(\overrightarrow{x}) = P(\overrightarrow{x}) \tag{9}$$

$$P(\overrightarrow{x}) = \begin{cases} 0 & \text{when } \overrightarrow{x} \geqslant 0x \in feasibleRegion, \\ \Sigma_{i=1}^{F} R_i H_i^2(\overrightarrow{x}) & \text{otherwise,} \end{cases} \tag{10}$$

where H_i is defined as weighted value for penalty for $H_i = max0, g_i(\overrightarrow{x})$, for $i = 1, 2, ..., p$. $g_i(\overrightarrow{x})$ is the distance that $\overrightarrow{x}$ from the certain constrained equalities or inequalities. We suppose equalities are presented as $E_i(x) = 0$, and inequalities are denoated as $I_i(x) \leq 0$, then $g_i(\overrightarrow{x})$ is defined as Eq. (11). Balance influence from different constraints on penalty function by adopting $g_i(\overrightarrow{x})$.

$$g_i(\overrightarrow{x}) = \begin{cases} \|E_i(\overrightarrow{x})\| & \text{if } E_i(\overrightarrow{x}) \neq 0, \\ I_i(\overrightarrow{x}) & \text{if } I_i(x) > m, \end{cases} \tag{11}$$

where m is the summary number of constrained equations and inequalities. Constant R_i is a multi-stage assignment function which means different penalty function uses different weight for several levels. In all cases, value of R has to be fixed before penalty function defined for certain applications while it is difficult to choose the optima value. The value of R_i is decided by the counter of object function, in other word, value of R_i should be decided by the ratio of object function and penalty, which can be represented as Eq. (12).

$$Scale = \frac{f(\overrightarrow{x})}{\Sigma_{i=1}^{p} H_i(\overrightarrow{x})} \tag{12}$$

where f is the objection function value. $\Sigma_{i=1}^{p} H_i(\overrightarrow{x})$ represents constraints violates sum. Because we need to guarantee in this way the penalty function will work efficiently as expected after testing. With the larger of object function, the larger penalty coefficient R_i will be adopted. Self-adaptive H strategy is proposed based on the value of object function and constraints violates. Benchmark functions will be testified in part 4.

3.2 J Strategy for PSO

Traditional J strategy is described in paper [5], based on the conception of J strategy, we adopt a self-adaptive penalty function which is proposed as Eq. (13).

$$\begin{aligned} p(\alpha,\beta) &= \rho_k^{\alpha} \times svc(\beta, \overrightarrow{x})\rho_k = C \times k, k = \#generation\ svc(\beta, \overrightarrow{x}) \\ &= \sum_{i=1}^{m} D_i^{\beta}(\overrightarrow{x}) + \sum_{j=1}^{\beta}(\overrightarrow{x}), \beta = 1, 2, ... \end{aligned} \tag{13}$$

where α, β, and C are parameters of the method and k is the number of generation under consideration. The values used in this paper are α=1, β=1, C=0.5.

3.3 P strategy for PSO

P strategy holds the main principle that "Any feasible solution is better than a non-feasible one". P strategy is a non-stationary penalty function. Different from above two functions, P strategy function adopts penalty value only if constraint violates are above zero, so the penalty function could be defined as follows:

$$p(\overrightarrow{x}) = r\sum_{j=1}^{p} f_i(\overrightarrow{x}) + \rho(\overrightarrow{x}, t)\rho(\overrightarrow{x}, t) = max0, maxf(\overrightarrow{x}) - minf(\overrightarrow{x}) + \sum_{j=1}^{p} f_j(\overrightarrow{x}) \tag{14}$$

$\rho(\overrightarrow{x}, t)$ is a time-varying penalty, which means value defined in the range of $maxf(\overrightarrow{x}) - minf(\overrightarrow{x})$ and constraints violates. Three kinds of situations are taken into account.

Case 1. $maxf(\overrightarrow{x}) - minf(\overrightarrow{x}) + \sum_{j=1}^{p} f_i(\overrightarrow{x}) < 0$. In this case, the distance of $maxf(\overrightarrow{x})$ and $minf(\overrightarrow{x})$ is larger than x's constrained violate value, so the extra penalty function value would be zero which means it's total penalty value would be less.

Case 2. $maxf(\overrightarrow{x}) - minf(\overrightarrow{x}) + \sum_{j=1}^{p} f_i(\overrightarrow{x}) > 0$. In this case, extra penalty value would be decided by x'value, so the constraints value would be more than $maxf(\overrightarrow{x}) - minf(\overrightarrow{x})$.

Case 3. $maxf(\overrightarrow{x}) - minf(\overrightarrow{x}) + \sum_{j=1}^{p} f_i(\overrightarrow{x}) = 0$. This would be a critical case of Case 1 and Case 2.

Based on the analysis of $maxf(\overrightarrow{x}) - minf(\overrightarrow{x})$ and constraint violate of x, self-adaptive penalty function is proposed by calculate constraints violate and other solutions. Because in most cases, penalty function need to be carefully chosen for best unavailable solution and worst available solution, a self-adaptive P strategy is come up in this paper. Firstly, the trend of $\rho(\overrightarrow{x}, t)$ can be divided into three cases.

Case 1. If $\sum_{j=1}^{p} f_i(\overrightarrow{x})$ is constant while $maxf(\overrightarrow{x}) - minf(\overrightarrow{x})$ decreases with time, then x in latest iteration is the best one because $\rho(\overrightarrow{x}, t)$ decreases.

Case 2. If $maxf(\overrightarrow{x}) - minf(\overrightarrow{x})$ doesn't change while $\sum_{j=1}^{p} f_i(\overrightarrow{x})$ decreases with time, conclusion in Case 1. still holds true because $\rho(\overrightarrow{x}, t)$ decreases.

Case 3. If both $maxf(\overrightarrow{x}) - minf(\overrightarrow{x})$ and $\sum_{j=1}^{p} f_i(\overrightarrow{x})$ decrease with time, then penalty value x, which is $p(\overrightarrow{x}) = r\sum_{j=1}^{p} f_j(\overrightarrow{x}) + \rho(\overrightarrow{x}, t)$ is less than before and conclusion in Case 1. is still true.

Based on analysis of above cases, we come to the conclusion that in the same iteration, their $\rho(\overrightarrow{x}, t)$ is only related with violates. If solution compared with itself in different iterations, it's $\rho(\overrightarrow{x}, t)$ decreases or at least the same. So $\rho(\overrightarrow{x}, t)$ would be denoted as the latest constraint violates only, and penalty value could be calculated by equ

$$p(\overrightarrow{x}) = r\sum_{j=1}^{p} f_i(\overrightarrow{x}) + \rho(\overrightarrow{x}, t)\rho(\overrightarrow{x}, t) = max0, \sum_{j=1}^{p} f_i(\overrightarrow{x}) - minf(\overrightarrow{x}) + \sum_{j=1}^{p} f_j(\overrightarrow{x}) \tag{15}$$

where r is a constant. The value set for r in this paper is 2.

4 Constrained Problem Optimization Based on Different Strategies

4.1 Parameters Settings and Benchmark Functions

Parameters settings in PSO and GA are as follows.

Table 1: Parameters settings in PSO and GA

Parameters	*PopSize*	*MaxIter*	*VelRange*	*CPro*	*MPro*	$C1, C2$	ω
PSO	20	2000	$[-1, 1]$	NULL	NULL	2, 2	[0.4, 0.9]
GA	200	2000	NULL	0.5	0.03	NULL	NULL

And $VelRange$ has to be adjusted and is usually set in dependency on the scale of the respective optimization problem. In many cases it is set to the distance between lower and upper limit or to half this range for each dimension. Different benchmark functions are chosen shown in Table 2, where n is the number of decision variables, LI is the number of linear inequalities, NI is the number of nonlinear inequalities, LE is the number of linear equalities and NE is the number of nonlinear equalities.

4.2 Result Analysis

We testify the proposed algorithm compared with GA on benchmark function shown in Table 2 and Table 3 by showing optima value, average value and worst value. And self-adaptive penalty mechanisms are adopted including H strategy, J strategy and P strategy combined with PSO. Each algorithm runs for 20 times and average values are recorded because both GA and PSO are random searching optimization algorithm.

Based on the following tables, we can conclude that optima value of GA and self-adaptive PSO have relationship with features of benchmark functions. We can conclude as follows:

Table 2: Comparison of improve H, J and improved P strategy optimizing on benchmark functions

ALgorithm	G01	*G*05	G06	*G*07	*G*09	*G*10	*G*11	*G*13
GA	−13.60	5231.37	−6340.27	25.68	685.36	7122.36	0.76	0.096
HPSO	−15.0	5126.51	−6961.81	24.36	680.63	7131.39	0.75	0.069
JPSO	−15	5126.59	−6961.81	24.69	680.63	7049.70	0.75	0.056
PPSO	−15	5226.23	−6961.81	24.32	680.63	7049.69	0.75	0.098

Table 3: Comparison of improve H, J and improved P strategy optimizing on benchmark functions

ALgorithm	*F*01	F02	*F*03	F04	F05	F06
GA	1.37	−6952.33	677.50	−30672.00	−31443.20	−213
HPSO	1.39	−6961.81	680.63	−30672.00	−31028.9	−213
JPSO	1.39	−6961.81	680.69	−30672.00	−31523.86	−312
PPSO	1.39	−6961.81	680.68	−30672.00	−31512.85	−312

(1) Variable numbers D. By optimizing g-series benchmark functions, their dimensions are $2 < D < 7$, both GA and PSO could get optima values. However when the dimension of function is $D > 7$, such as $g01$ and $g07$, compared with PSO, GA cannot achieve optima value only get to -13.60 for g01, and 25.68 for $g07$. So we come to the conclusion that dealing with high-dimension optimization problems such as dimension up to seven, GA may not get the optima. In Table 2, we can see that PSO could get much better optima value when coping with constrained problem $g01$ and $g07$, especially PSO with self-adaptive H penalty function, which improves that our proposed algorithm could make PSO has better optimization performance compared with GA when dealing with high-dimension problems.

(2) Function types includes quadratic, nonlinear and linear. From Table 2 and Table 3, there are three types of functions, both GA and proposed PSO with H, J and P strategy could achieve optima or near optima values independent on types of functions.

(3) Constrained types includes equalities and inequalities. When dealing with constraints as equalities, such as $g05$ and $g03$, ε is the controller of optimization precision. The time complexity of optimizing depends on constraints violates, for PSO it costs less time with improved H strategy and more time with improved P strategy. Contrast to PSO, GA has more complicated operations so it costs more time than PSO when combined with the same penalty functions. When constraints are inequalities, PSO with self-adaptive penalty function may have more challenge than the former one.

PSO with self-adaptive H penalty function yields to better performance compared with others by running 20 times to get the mean value. PSO with P penalty function may get best mean value which means it has more stable performance. And PSO with P penalty function could get better global solutions by adding minimum penalty from neighborhood solutions and proved to be a stable optimizer.

4.3 Algorithm Complexity

Several parameters are set as M for population size, N for variable number, and S for constraints number, and nth calculation time for improved H strategy is defined as P_n:

$$P_n \leq M + M(N+N+N) + \frac{1}{2}M(M-1) + S \times N = M + 3MN + \frac{1}{2}M(M-1) + S \times N \quad (16)$$

Because parameter M is much larger than N, the algorithm complexity for H strategy is $o(M^2)$. And for J strategy, penalty function is omorphic with H strategy, so the algorithm complexity is $o(M^2)$. Improved P strategy has complicated operation in calculating penalty function, and P_n is defined as:

$$\begin{aligned} P_n &\leq M + M(N+N+N) + \frac{1}{2}M(M-1) + S \times N \times \left[\frac{1}{2}M(M-1)\right] \\ &= M + 3MN + \frac{1}{2}(M+SN)(M-1) \end{aligned} \quad (17)$$

5 Conclusions

In this paper particle swarm optimization with turbulence operations is used for the optimization of constrained problem with novel penalty mechanisms, which are improved H, J, and improved P penalty functions with PSO. Self-adaptive parameters are adopted in improved H strategy to satisfy different constrained problems, while unified P strategy is brought up which is easier in algorithm complexity. All proposed algorithms are testified by several benchmark functions compared with GA. And optimization results prove that PSO with novel penalty functions have better performance than GA.

References

[1] G. T. Pulido, C. A. C. Coello, A constraint-handling mechanism for particle swarm optimization, Evolutionary Computation, CEC2004, June 2004, 19-23

[2] Kaiyou Lei, Yuhui Qiu, A study of constrained layout optimization using adaptive particle swarm optimizer, Journal of Computer Research and Development, Vol. 10, 2006

[3] Karin Zielinski, Rainer Laur, Constrained single-objective optimization using particle swarm optimization, IEEE Congress on Evolutionary Computation, July 16-21, 2006

[4] K. E. Parsopoulos, M. N. Vrahatis, Particle swarm optimization method for constrained optimization problems, Proceedings of the 2002 Euro-International Symposium on Computation Intelligence, 2002, 214-220

[5] Graham Kendall, Yan Su, A particle swarm optimization approach in the construction of optimal risky portfolios, Proceedings of the 23rd IASTED International Multi-conference Artificial Intelligence and Applications, Innsbruck, Austria, Feb. 2005, 14-16

[6] Yong Wang, Zixing Cai, Wei Zhen, Hui Liu, A new evolutionary algorithm for solving constrained optimization problems, Journal of Central South University (Science and Technology), Vol. 37, No. 1, 2006, 119-123

[7] Tetsuyuki Takahama, Setsuko Sakai, Constrained optimization by a constrained particle swarm optimizer, Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol. 9, No. 3, 2005, 282-289

[8] J. Kennedy, R. C. Eberhart, Particle swarm optimization, Proc. IEEE International Conf. on Neural Networks, Perth: IEEE Piscataway, 1995, 1942-1948