A-034

継続的解探索のための拡散型粒子群最適化
A Diffusion Particle Swarm Optimizer for Continuous Solution Search

渡辺 恭成† 中野 秀洋‡ 宮内 新‡
Yasumasa Watanabe Hidehiro Nakano Arata Miyauchi

1. はじめに

近年、複雑な問題を効率よく解くための手法として粒子群最適化（PSO）が注目されている。PSO では、探索を行う粒子群が互いに情報を共有しながら高速に解へと収束する。しかし、PSO は局所解に陥りやすく、高次元の最適化問題において解探索性能が極端に低下する場合がある。その原因の一つとして、粒子の高い収束性によって解の探索が早期に停止することが挙げられる。そこで本稿では、ある粒子群が解に収束した時点でその粒子群を拡散させ、継続的な解の探索を行うことができる PSO のアルゴリズムを提案する。高次元のベンチマーク関数に対する数値実験を行い、提案アルゴリズムの有効性を検証する。

2. 粒子群最適化（PSO）

PSO は粒子が互いに情報を共有しながら多次元空間内を探索する最適化アルゴリズムである[1]。粒子全体の過去の探索結果の中で最も良い評価値を持つ位置情報を Gbest と呼び、各粒子の過去の探索結果の中で最も良い評価値を持つ位置情報を Pbest と呼ぶ。

各粒子は現在の速度 v_i^t と群れの最良位置である Gbest、各粒子の最良位置である Pbest を用いて以下のように速度を決定し位置を更新する。

$$v_{ij}^{t+1} = W \cdot v_{ij}^t + C_1 \cdot R_1 \cdot \left(pbest_{ij}^t - x_{ij}^t\right) + C_2 \cdot R_2 \cdot \left(gbest_j^t - x_{ij}^t\right) \quad (1)$$

$$x_{ij}^{t+1} = x_{ij}^t + v_{ij}^{t+1} \quad (2)$$

ただし、i は粒子番号、j は次元成分番号、t は探索更新回数、R_1、R_2 は 0 から 1 までの一様乱数、W、C_1、C_2 は各項の重みを示す。

3. 拡散型 PSO

前節で示したように、各粒子は Gbest と Pbest に引き寄せられるように位置を更新していく。しかし、全ての粒子の位置が Gbest に収束した場合、新たに速度が生まれることはなく、その場で探索が終了してしまう。

そこで、Gbest を中心に粒子を拡散させる、拡散型の PSO アルゴリズムを提案する。この手法は、ある程度の数の粒子が Gbest 付近に集まった場合に、Gbest から離れるように粒子の位置を更新していく。

提案アルゴリズムを以下に示す。

Step 1：粒子数 N、収束エリア T_d、濃度閾値 T_s、拡散エリア R、拡散係数 C_{diff}、拡散確率 P_{diff}、拡散強度 K_{diff}、慣性定数 W、学習係数 C_1, C_2 を設定し、全ての粒子の位置ベクトルと速度ベクトルを初期化する。

Step 2：式（3）（4）で濃度を計算し[2]、濃度が T_s 以上、かつ拡散している粒子が一つもなければ、Gbest を中心とする収束エリア T_d 内にある粒子を、確率 P_{diff} で拡散する。

$$density^t = \frac{\sum_i^N \alpha\left(distance_i^t; T_d\right)}{N} \quad (3)$$

$$\alpha\left(z; T_d\right) = \begin{cases} 1 & z \le T_d \\ 0 & else \end{cases} \quad (4)$$

ただし、$distance_i^t$ は時刻 t における粒子 i と Gbest との距離である。

Step 3：拡散する粒子は現在の速度に乱数を加算し、その他の粒子は式（1）で速度を更新する。加算する乱数は探索領域に K_{diff} を乗算した一様乱数である。拡散粒子は速度更新後、現在位置より Gbest から離れるのであれば全次元を確率 1 で、求めた速度によって位置を更新する。現在位置より Gbest へ近づくのであれば式（5）を用いて確率を求める。

$$P_i^t = \frac{C_{diff} \times distance_i^{t^+}}{distance_i^t} \quad (5)$$

$distance_i^t$ は現在の粒子 i の Gbest との距離、$distance_i^{t^+}$ は速度加算後の Gbest との距離を示す。全次元を確率 P_i^t で、求めた速度によって位置を更新し、確率 $\left(1 - P_i^t\right)$ で Step3 の始めに戻りその粒子のみ速度の計算を再度行う。

Step 4：拡散粒子は、拡散エリア R 内にいるか判定を行い、エリアの外に出ている場合は通常の PSO の探索に戻る。

Step 5：Pbest と Gbest を更新する。

Step 6：t が終了探索回数となった場合は探索を終了する。そうでなければ、$t = t + 1$ として Step 2 に行く。

4. 数値実験

数値実験を通して拡散型 PSO の性能を検証する。ベンチマーク関数は、最適解付近に複数の準最適解を持つ多峰性関数である Rastrigin 関数（6）と、設計変数間に依存関係を持つ単峰性関数である Ridge 関数（7）を用いる。次元数はいずれの関数も 100 次元として実験を行う。

$$F_{Rastrigin}(x) = 10n + \sum_{i=1}^{n}\left(x_i^2 - 10\cos\left(2\pi x_i\right)\right) \quad (6)$$

$$Domain \quad \left(-5.12 \le x_i \le 5.12\right)$$

† 東京都市大学　大学院　工学研究科
Graduate School of Engineering, Tokyo City University
‡ 東京都市大学　知識工学部　情報科学科
Department of Computer Science, Faculty of Knowledge Engineering,Tokyo City University

$$F_{Ridge}(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2 \qquad (7)$$

$$Domain \quad (-64\le x_i\le 64)$$

Rastrigin 関数、Ridge 関数ともに全ての次元の位置が 0 のときに最良値 0 をとる。終了探索回数を 100000、実行回数を 50、N を 500、T_d を 0.6、T_s を 0.9、C_{diff} を 0.7、P_{diff} を 0.8、K_{diff} を 0.015、W を 0.9 とし、拡散型 PSO と基本的な PSO の比較を行った。また、R と C_1, C_2 については関数、手法により異なる。拡散型 PSO における Rastrigin 関数の C_1, C_2 は 0.8、Ridge 関数の C_1, C_2 は 0.5、基本的な PSO における C_1, C_2 を 1.0 とし、Rastrigin 関数における R は 5.0、Ridge 関数における R は 60.0 とした。

表 1 は拡散型 PSO と基本的な PSO のそれぞれの関数における実験結果を表している。1.0×10^{-10} 未満の値のとき最適解を発見したとみなし、最適解発見数を求めた。

拡散型 PSO は全ての環境で最適解を求めることが出来た。図 1，2 は 100 次元 Rastrigin 関数における両アルゴリズムの代表的な 3 つの粒子のイタレーション毎の速度を表している。PSO は早期に速度が収束していることに対し、拡散型 PSO は継続的に探索を行っていることが確認できる。継続的に探索を行うことで高次元においても局所解から脱出できることが確認できる。

また、拡散係数 C_{diff} を変更した際の解収束の速度を検証した。図 3，4 は C_{diff} を 0.5、0.7、1.0、1.5 と設定した際の 100 次元 Rastrigin 関数の評価値を示す。C_{diff} は小さいほど粒子は拡散しやすく、大きいほどその場に留まりやすいため、C_{diff} が大きくなるほど収束が遅い。イタレーション 2000 付近では $C_{diff}=0.5,1.0$ でさほど差が見られないが、探索が進むにつれ $C_{diff}=0.5$ の方が収束が早いことが確認できる。$C_{diff}=0.5$ では、早期に拡散が終了してしまうため深い探索ができず、$C_{diff}=1.0,1.5$ では拡散が遅くなる。そのため本実験においては $C_{diff}=0.7$ が最も良い結果となった。

表 1：性能比較シミュレーション

関数	次元		拡散型 PSO	PSO
Rastrigin	100	平均値	1.3×10^{-13}	4.3×10^{2}
		最良値	0.0	2.4×10^{2}
		最適解発見数	50	0
Ridge	100	平均値	1.5×10^{-11}	6.0×10^{4}
		最良値	1.9×10^{-12}	2.4×10^{4}
		最適解発見数	50	0

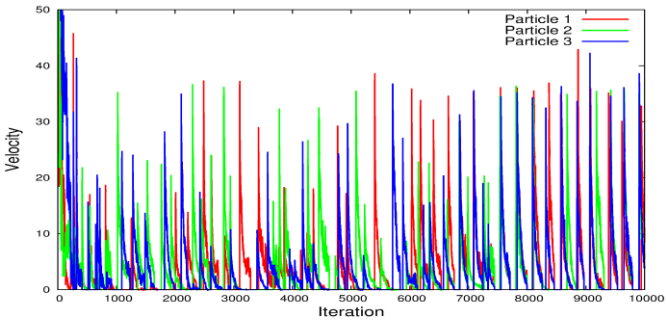

図 1：拡散型 PSO のイタレーション毎の速度

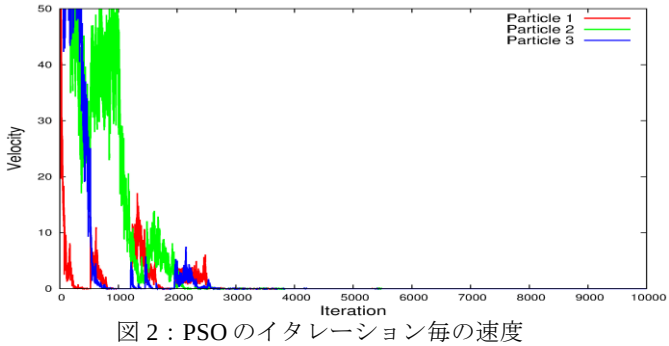

図 2：PSO のイタレーション毎の速度

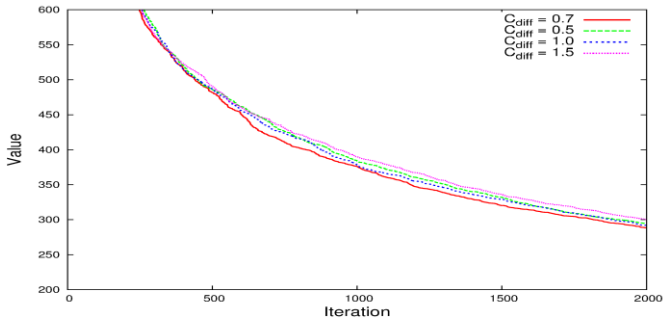

図 3：各 C_{diff} の評価値（$t=0\sim2000$）

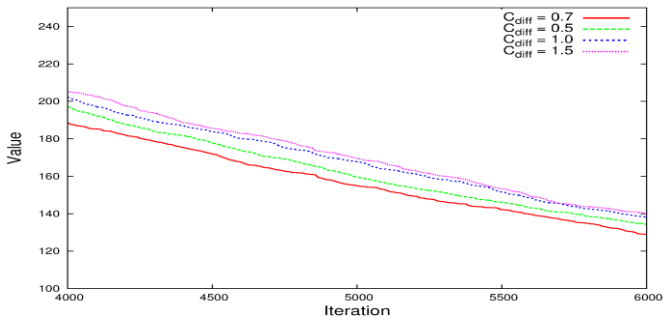

図 4：各 C_{diff} の評価値（$t=4000\sim6000$）

5. おわりに

PSO は粒子の高い収束性により探索が早期に終了してしまうため、局所解からの脱出が困難である。そこで継続的に探索を行う拡散型 PSO を提案した。提案アルゴリズムは継続的探索により局所解から脱出し、高次元においても最適解を求めることが出来ることを実験により確認した。本手法の課題としては、パラメータ数の削減、より複雑な関数での実験、拡散粒子間の相互作用の考慮などが挙げられる。

参考文献

[1]J. Kennedy and R. Eberhart “Particle Swarm Optimization”, Proc. of IEEE International Conference on Neural Network, pp.1942-1948, (1995).

[2]M. Yoshimura, H. Nakano, A. Utani, A. Miyauchi and H. Yamamoto, “An effective sink node allocation scheme for long-term operation of Wireless Sensor Networks using adaptive suppression PSO”, Proc. of IEEE International Conference on Ubiquitous and Future Networks, pp92-97, (2010).