G-06

関数最適化問題を解く ACO for Continuous Domains の有効性の検証
Introduction to Ant Colony Optimization to Solve Function Optimization Problem

刀根　拓也†　李　朝盛†　徐　暢†　髙橋　良英†
Tone Takuya　Li Chaosheng　Xu Chang　Takahashi Ryouei

1. はじめに

アントコロニー最適化法 Ant Colony Optimization (ACO)[1]は「フェロモンを媒介にした間接的コミュニケーションによる蟻の群知能の発現」という考え方で組合せ最適化問題の解を確率的に探索する最適化法である。関数 $y=f(x_1, \cdots, x_i, \cdots, x_n)$の最適化問題(FOP)とは、関数 f の最小値 f_{min} を実数空間 R のある定義域 D[i]で求める問題である[2]。本実験では実数連続空間の中で解を探索できるように ACO フェロモンコミュニケーションの考え方を拡張した方法である Ant Colony Optimization for Continuous Domains (ACO_R)[3]の有効性を、FOP を解く手法として着目されている Covariance Matrix Adaptation-Evolution Strategy (CMA-ES) [4]との比較により検証する。

2. 連続空間内で解を探索するためのアントコロニー最適化法（ACO_R）

2.1 背景

ACO は、組合せ最適化問題(COP)の解を確率的に探索する汎用的最適化手法として開発された。COP は N^M 個の離散個の解候補の（N と M は整数）中から最適解を探索する問題である。この考え方に基づき FOP の解を探索する手法として段階的探索空間局所化機能[5]が提案された。ACO_R は探索域を連続空間に拡張する別の方法である。

2.2 ACO_R のアルゴリズム[3]

a) 初期個体生成処理

k 個の個体から成る初期個体群 $S=\{S_1, S_2, \cdots, S_k\}$ を以下の手順で生成する。個体は解候補のことである。

(1)以下に示すように、第 i 番目の蟻 Ant(i)は各独立変数 x_j の値 S_{ij} を定義域 D[j]を閉区間(min_j, max_j)としてランダムに一様乱数 random を用いて生成する。(i=1,2,$\cdots$,k), (j=1,2,…,n)

$$S_{ij}=\text{random}\ (min_j, max_j)$$

上式にて、$min_j=-a_j+x_{j0}$, $max_j=a_j+x_{j0}$ であり、a_j と x_{j0} は問題対応に初期値が与えられる。

(2) (1)の S_{ij} 値をその要素に持つ個体 S_i の関数値 f_i の値を計算する。　$f_i=f(S_{i1}, S_{i2}, \cdots, S_{in})$

b) ソート

(1) S の要素（個体）を関数値の昇順に並べ、Sort 結果の集合を改めて S とする。

S:=Sorting（S), $S=\{S_1, S_2, \cdots, S_k\}$

(2) 関数値の小ささ $f(S_i)$に応じたフェロモン量 ω_i を探索した個体 S_i 毎に以下のように計算する。(i=1,2,・・・,k)

$$\omega_i=\frac{1}{qk\sqrt{2\pi}}e^{-\frac{(i-1)^2}{2(qk)^2}}$$

q: フェロモンの重み調整パラメータ。

この値が小さければ小さい程上記正規分布 $N(1,(qk)^2)$の分散は小さく，最小値を探索したランク 1 の蟻 S_1 の近傍を次世代の蟻が探索する確率が高くなる。実験では q=1.0.

i: 解 S_i のランク（順位）

c) 世代更新処理

(1)終了条件チェック・・終了条件（収束条件を満たすまたは観測最大世代数を超える）を満たすまで以下の処理(2)から(7)を繰り返す。終了条件を満たせば、最適解が探索できたとしてジョブを終了する。

(2)次世代の m 個の個体集合 $S'=\{S'_1, S'_2, \ldots, S'_l, \cdots, S'_m\}$を以下の手順で生成する。

(a) 個体 S'_1 毎に、前世代の k 個の探索解の中からどの解 S_j の近傍を探索するかをフェロモンの重み ω_j の大きさに合わせ以下の確率 p_j で確率的に選択する（ルーレット方式選択）

$$p_j=\frac{\omega_j}{\sum_{\gamma=1}^{k}\omega_\gamma}$$

(b)(a)で選択した個体 $S_j=(x_{j1}, \ldots, x_{ji}, \ldots, x_{jn})$の独立変数 x_{ji} の値を S_{ji} とする。この時、平均 $\mu_{ji}=S_{ji}$、標準偏差を σ_{ji} とする正規分布 $N(\mu_{ji},\sigma_{ji}^2)$に従う乱数 S'_{li} を次式に示すように生成する。これを個体 $S'_l=(x'_{l1}, x'_{l2}, \cdots, x'_{ln})$の独立変数 x'_{li} の値とする。

$$S'_{li}=N(\mu_{ji},\sigma_{ji}^2)$$

(c) (b）の標準偏差 σ_{ji} は以下の式で求める。

$$\sigma_{ji}=\xi\sum_{\gamma=1}^{k}\frac{|S_{ji}-S_{\gamma i}|}{k-1}$$

上式にてξは収束調整パラメータであり、その値が小さければ小さい程収束は速くなる。しかし多様性[5]は失われる。実験ではξ=0.85。

(3) 現世代の個体集合 S と新世代向けに(2)で追加した個体集合 S'を併せた集合を S'' とする。

S'':=S+ S'

ここで，　$|S''|=k+m$,　$|S|=k$,　$|S'|=m$.

(4) S'' を関数の値の昇順でソートした結果を改めて S'' とする。

S'' :=Sorting（S''）

(5)S'' から探索した関数値の大きな適応度の劣る m個の個体集合 Sm を削除し、その集合を改めて S とする。

S:= S'' −Sm

$S=\{S_1, S_2, \cdots, S_k\}$ としよう。

これまで見つかった最良解 $X_{best\_so\_far}$ はここで更新される。

(6) 関数値の小ささ $f(S_i)$に応じたフェロモン量 ω_i を再計算する。(i=1,2,・・・,k)

ω_i の計算式はｂ)(2)で示した式と同じである。

2.3 本実験での工夫点

a) NO_TRIALS

次世代の各個体 $X(=X_j)$の生成にあたって以下の手続きを最大 NO_TRIALS 回繰り返す。この繰り返し回数を越えれば、以下の(1)の手続きで生成した個体を次世代の個体として選択する。本実験では NO_TRIALS=100 で実験した。

(1)個体 $X=(x_1, x_2, \ldots, x_n)$にて変数 x_i 値を、正規分布 $N(\mu_i,\sigma_i^2)$に

†京都情報大学院大学 , The Kyoto College of Graduate Studies for Informatics, 京都市

表 1 ACO_R と CMA-ES の機能と性能の評価

テスト関数	変数数 n	最適化法	①最適解探索回数	②平均最小値	③最良最小値	④標準偏差	⑤平均個体生成数	⑥平均コンピュータ時間(秒)
Griewank's function	10	CMA-ES	0/15	9.80E-11	3.24E-11	4.44E-11	610	1
		ACO_R	15/15	0	0	0	359780	68
Perm function (D,BETA)	5	CMA-ES	0/15	0.0145436	0.00384	0.00857	47500	3
		ACO_R	0/15	0.0257126	0.0045198	0.0205	77810	28

従う乱数 r とする。その際、定義域内 $D[i]=(-a_i+x_{j0},a_i+x_{j0})$ でその解 r が生成されるまで、無限回試行する。

(2)(1)で生成した解が、これまでに探索できた最良解 $X_{best_so_far}=(x_{1b},x_{2b},\ldots,x_{nb})$に比較して解の改良がなされているかを判断する。すなわちこれまでに探索できた関数最小値以下の関数値が探索できたか($f(X)<=f(X_{best\_so\_far})$)を判断する。判断結果が TRUE（真）の場合は解が見つかったとして繰り返しから抜ける。FALSE（偽）なら処理(1)に戻る。

b）正規分布に従う乱数の乱数系列

各独立変数 x_i が正規分布 $N(\mu_i,\sigma_i^2)$に従う乱数 r_i の時、各独立変数 x_i 毎に乱数列を生成せず、全ての独立変数 x_i について唯一つの乱数系列で乱数を生成するようにした。

c) DIVERSITY_TYPE

世代 g において独立変数 x_i が正規分布 N（μ_i, σ_i^2）に従って分布する時、標準偏差 σ_i が 10^{-10} 以下となったなら,解探索の多様性を確保するため、σ_i を以下のように拡大させた。

σ_i = RANGE[i]= a_i。

d）生成した k 匹の個体の内、どの個体を中心に次世代の個体を生成するのか否かの選択の契機について：論文[3]では、上記の選択を個体生成の度に行っているが、選択効率を向上させるため、本論文では、世代毎（入れ替える m個の個体毎）に上記の選択を行っている。

3. CMA-ES

CMA-ES は突然変異操作系列によって最適解を探索する手法である（μ, λ）-ES による最適化法の 1 つである。CMA-ES は、ある世代(g + 1)において λ 個の個体群を生成する。その個体群は、多変量正規分布 $N(m(g), \sigma(g)^2\times\Sigma'(g))$に従って最適解が確率的に存在するという仮定のもとで、ランダムに生成される。ここで $m(g)$は,前世代 g において,ランダムに生成された λ 個の個体群の中から選択された関数値の小さな μ 個の優良な個体群の重心ベクトルである。この重心ベクトルは μ 個の個体を関数値の小さな順に重みづけした平均値ベクトルである。上記重心ベクトル $m(g)$の世代による変化量を考慮して、μ 個の個体群の共分散行列の値 $\Sigma(g)$を調整し共分散行列 $\Sigma'(g)$としている。 $\sigma(g)$はステップサイズと呼ばれる解収束を調整するパラメータであり時間と共に 0 に減少する。上記重心ベクトル $m(g)$と調整共分散行列 $\sigma(g)\times\Sigma'(g)$をその平均値と共分散分析行列に持つ多変量正規分布に従う乱数を発生させ、その中から次世代 g+1 の個体を λ 個ランダムに選択する方法が CMA-ES である。

4. テスト関数と本実験での探索値

最適化手法を評価する標準関数として良く知られた以下の標準関数を利用した。本実験で探索した最小値も示す。

(1) Griewank's function

<定義式> $z=f(x_1,x_2,\bullet\bullet\bullet,x_{10})=\frac{1}{4000}\sum_{i=1}^{10}x_i^{\,2}-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$

<定義域> $-600\leqq x_i\leqq 600$ (i=1, 2, 3..., 10)

<最適最小値> 全ての x_i (i=1, 2,..., 10)について $x_i=0$ の時, 最小値 0.0 を探索した。

(2) Perm function type (D,BETA)

<定義式> $z=f(x_1,x_2,\cdots,x_n)=\sum_{j=1}^{n}(\sum_{i=1}^{n}(i^j+\beta)\left[(\frac{x_i}{i})^j-1\right])^2$

<定義域>$-n\leqq x_i\leqq n$ (i=1,2,・・・,n), $\beta=0.5$

<最適最小値>n=5 の時 x_1~x_5=1.0362317595821954 8853～4.99189831589622468044（各変数値差は約１）の時、 f＝0.0025887790451236 6374 の最小値を探索している。

5. 実験結果

4 節で述べた 2 つの標準テスト関数を用いて ACO_R と CMA-ES の機能と性能の C プログラムによる評価実験を行った。実験結果を表 1 に整理する。表では、15 回の独立な試験を行なった結果を（A）機能（①最適解探索回数②平均探索最小値③最良最小値④標準偏差）と（B）性能（⑤最小値を見つけるのに要した平均個体生成数⑥平均コンピュータ時間(秒）)の観点から整理した。

6. 結論

実験の結果、（１）機能の観点からは、総合的に ACO_R が CMA-ES より優れていること（２）性能の観点からは、CMA-ES が ACO_R より優れていることがわかった。しかし、CMA-ES が探索した解と ACO_R が探索した解の誤差は 10^{-9}～10^{-11} であり微小である。今後実験空間を拡張して ACO_R の有効性を検証する。

参照文献

[1] M. Dorigo and T. Stützle: “Ant Colony Optimization”, The MIT Press, 2004.

[2] X.-S. Yang, “Test Problems in Optimization, in: Engineering Optimization: An Introduction with Metaheuristic Applications (Eds. X.-S. Yang)”, John Wiley & Suns, 2010.

[3] K. Socha, M. Dorigo, “Ant Colony Optimization for Continuous Domains,” European Journal of Operational Research, Vol. 185, pp. 1155–1173, 2008.

[4] N. Hansen, “The CMA Evolutionary strategy, A comparing review,” StudFuzz 192, pp. 75–102, Springer-Verlag Berlin Heidelberg, 2006.

[5] R. Takahashi and Y. Nakamura, “Ant Colony Optimization with Stepwise Localization of the Discrete Search Space to Solve Function Optimization Problems,” Proceedings of 16th IEEE International Conference on Machine Learning and Applications (ICMLA17), pp.701-706, 2017.