探索領域の分割を導入した挿入操作PSO戦略の提案

猪股 能成[1,a)]　高見 利也[2,b)]

概要：巡回セールスマン問題に対して短時間で良い解を目的として，粒子群最適化を基にしたアルゴリズムである挿入操作 PSO 戦略は，問題点として高速化が挙げられている．我々は，解が収束するまでに要する時間を短縮することを目指し，挿入操作 PSO 戦略に，探索領域の分割を導入した方法を提案する．提案手法では，分割したそれぞれの領域で粒子に解の探索を行わせ，ある程度収束したら，分割していた領域を結合し，探索を継続させる．本研究では，従来手法と提案手法に対し，探索性能と処理時間について比較を行った．その結果，分割して探索を行う時間が試行によって分散したが，提案手法よりも処理時間を短縮することができた．

A proposal of Insertion-based PSO strategy introducing search-area partitioning

Inomata Yoshinari[1,a)]　Takami Toshiya[2,b)]

Abstract: Insertion-based PSO strategy (IPSO) is a method to solve Travelling Salesman Problem (TSP) based on Particle Swarm Optimization Algorithm. This method has a problem in its calculation time. Our purpose is to reduce the time until solving TSP by IPSO, and we propose a method to improve the IPSO by search-area partitioning. In our method, each particle searches the divided areas, before searching the overall area. We compare the performance and calculation time between IPSO and our method. As a result, our method reduces the calculation time without performance deterioration.

1. はじめに

群知能とは，単純な機能を持つエージェントが集団運動を発現させるアルゴリズムの総称である [1]．群知能では，鳥や魚などといった社会性生物がモデリングされたアルゴリズムが多く存在しており，一般的には，モデル内のエージェントに相互作用を持たせている．この群知能の代表的な応用に粒子群最適化（Particle Swarm Optimization，以下 PSO）がある．PSO は，探索の対象となる目的関数が与えられたとき，複数の粒子が互いに情報を共有しながら最適解を求めて探索空間内を動き回るアルゴリズムである [2]．また，従来のヒューリスティック解法と比較して，PSO は高速に解を探索することができ，目的関数が連続系であるような問題に広く応用できると言われている [3]．一方で，巡回セールスマン問題（Traveling Salesman Problem，以下 TSP）のような組合せ最適化問題には不向きとされてきた．TSP とは，訪問する都市群とその都市間の距離が与えられ，それらの都市を 1 度だけ通るような巡回路のうち，距離の総和を最小とするものを求める問題である [4]．

本庄氏らは，TSP を PSO で実用的に解けるようにした手法として，挿入操作 PSO 戦略 （Insertion-based PSO strategy，以下 IPSO）を提案した [5]．この手法は，粒子の位置を巡回路に置き換え，部分経路挿入法と呼ばれる手法を用いて解の更新を行うものである．しかし，IPSO では探索したい都市数やパラメータによっては計算量が増大し，探索時間が増えてしまう問題点がある．

そこで我々は，IPSO の高速化を実現するために，探索領域の分割に着目した．探索する領域を分割することによって，解が収束するまでに要する時間を短縮することが目的である．

[1] 大分大学大学院 工学研究科
[2] 大分大学 理工学部
Oita University, Oita-shi, Oita 870–1123, Japan
[a)] v19e3002@oita-u.ac.jp
[b)] takami-toshiya@oita-u.ac.jp

本稿では，まず提案手法のベースとなる PSO と IPSO について説明する．その後に我々が提案する手法について説明し，IPSO との比較実験の結果について述べる．

2. 粒子群最適化

PSO は，Kennedy と Eberhart によって提案された手法で，非常にシンプルなアルゴリズムながら，広い範囲の機能を最適化させるのに適しているとされている [6]．この手法では，解空間に粒子を複数配置し，個々の粒子が周囲の粒子と相互作用しながら粒子の位置および速度を更新していく．PSO における，一般的な更新式を以下に示す．

$$\boldsymbol{v}_i(t+1) = w\boldsymbol{v}_i(t) + c_1 r_1(\boldsymbol{pb}_i - \boldsymbol{x}_i(t)) + c_2 r_2(\boldsymbol{gb} - \boldsymbol{x}_i(t)) \quad (1)$$

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t) \quad (2)$$

ここでの w, c_1, c_2 は定数，r_1, r_2 は 0 以上 1 未満の一様乱数とする．粒子 i の速度 $\boldsymbol{v}_i$ の更新では，そのほかに，粒子自身が持つ最良解 $\boldsymbol{pb}_i$（Personal Best）と，粒子全体で共有する最良解 $\boldsymbol{gb}$（Global Best）が使われる．

以下に PSO のアルゴリズムを示す．

(1) 乱数により粒子の位置と速度を初期化
(2) 目的関数で解を計算
(3) それまでの $\boldsymbol{pb}_i$ と解を比較し，改善されていれば更新
(4) 粒子全体における最良解を $\boldsymbol{gb}$ に保存
(5) 上記の式に基づき，各粒子ごとに速度と位置を更新
(6) 終了条件に達したかを判定し，達していない場合は（2）から（5）を再度実行

終了条件は，更新計算の実行回数，もしくは，最良解の収束具合によって決められることが多く，このようにシンプルなアルゴリズムによって，解の探索を進めることが出来る．ただし，粒子の初期配置に偏りがあると，粒子がローカルな局所解に収束してしまい，厳密な最適解に行きつかない可能性があったり，終了条件によっては，最適解を発見する前に探索が終了してしまう可能性があるため，注意が必要である．

ローカルな局所解に収束するのを避けるために，周囲にいる粒子の最良解 $\boldsymbol{lb}_i$（Local Best）を使う方法がある．これは，上記の更新式において，$\boldsymbol{gb}$ の代わりに $\boldsymbol{lb}_i$ を使う方法で，後述する IPSO でも，この方法が使われている．

3. 挿入操作 PSO 戦略

TSP を PSO のアルゴリズムで解けるように本庄氏らが提案した手法が，挿入操作 PSO 戦略（IPSO）である [5]．この手法では，探索する都市数を N としたとき，粒子の位置を N 次元のベクトルで表現する．この位置座標が TSP における巡回路にあたり，目的関数はその巡回路での総移動距離にあたる．また，一般的な PSO と異なる点は速度の更新である．速度の更新には，部分経路の挿入が行われており，以下で説明する．

まず，部分経路そのものについて説明する．部分経路とは，ある巡回路の一部分を抽出したものである．例えば，1 から 8 でナンバリングされている都市を $[1,4,7,8,5,3,6,2]$ の順で訪れる巡回路があったとする．（番号 2 の都市を訪れたら，番号 1 の都市に戻る．）この巡回路から 4 都市の部分経路を抽出する場合，$[1,4,7,8]$ や $[8,5,3,6]$，$[6,2,1,4]$ などを部分経路の候補として考えることができる．

IPSO では，$\boldsymbol{pb}_i$ と $\boldsymbol{lb}_i$ のそれぞれから部分経路を抽出し，それらを各粒子の解 $\boldsymbol{x}_i(t)$ へ挿入することで，次の時刻での解 $\boldsymbol{x}_i(t+1)$ を求める．部分経路を挿入するアルゴリズムを以下に示す．また，アルゴリズムのイメージを図 1 に示す．

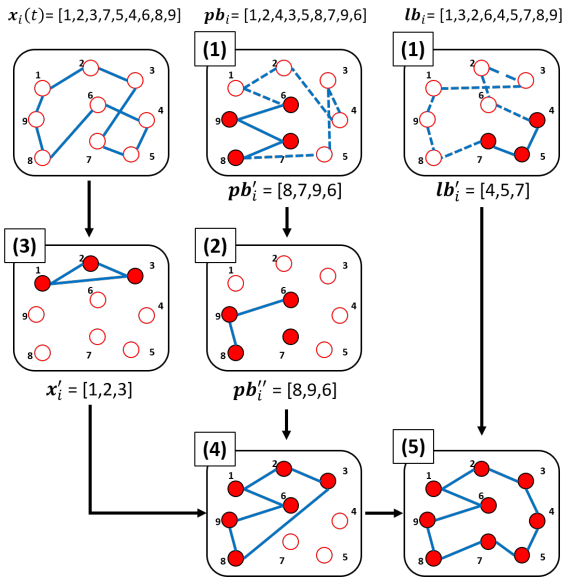

図 1　部分経路の挿入アルゴリズム
Fig. 1　Algorithm of Insetrion method

(1) 部分経路 $\boldsymbol{pb}'_i, \boldsymbol{lb}'_i$ の生成
粒子 i の $\boldsymbol{pb}_i$ から，長さ $[c_1 r_1(N+1)]$ の連結成分をランダムに抽出し，部分経路 $\boldsymbol{pb}'_i$ とする．同様に，$\boldsymbol{lb}_i$ から，長さ $[c_2 r_2(N+1)]$ の連結成分をランダムに抽出し，部分経路 $\boldsymbol{lb}'_i$ とする．ここでの c_1, c_2 はそれぞれ $0 \leq c \leq 1$ を満たす定数，r_1, r_2 は $[0,1]$ の一様乱数である．

(2) $\boldsymbol{pb}'_i$ の再構成
$\boldsymbol{pb}'_i$ と $\boldsymbol{lb}'_i$ の両方に含まれる都市が存在する場合，$\boldsymbol{lb}'_i$ に含まれる都市を $\boldsymbol{pb}'_i$ から除去し，部分経路 $\boldsymbol{pb}''_i$ とする．

(3) $\boldsymbol{x}'_i$ の生成
$\boldsymbol{x}_i$ から，$\boldsymbol{lb}'_i$ または $\boldsymbol{pb}''_i$ に含まれる都市を除いた閉路 $\boldsymbol{x}'_i$ を生成する．

(4) $\boldsymbol{pb}''_i$ の挿入
$\boldsymbol{x}'_i$ のすべての辺（図 1 だと，$(1,2),(2,3),(3,1)$）の中

で，その辺の両端の都市間に $\boldsymbol{pb}_i''$ を挿入したときに最も総距離が短くなるような辺を選択し，その辺に対して $\boldsymbol{pb}_i''$ を挿入する．そのときに生成された閉路を $\boldsymbol{x}_i''$ とする．

(5) $\boldsymbol{lb}_i'$ の挿入

$\boldsymbol{x}_i''$ の辺の中で $\boldsymbol{pb}_i''$ に含まれていない辺に対し，その辺の両端の都市間に $\boldsymbol{lb}_i'$ を挿入したとき，最も総距離が短くなるような辺を選択する．そして，その辺に対して $\boldsymbol{lb}_i'$ を挿入し，次の時刻での $\boldsymbol{x}_i$ とする．

IPSO では，PSO のアルゴリズムにおける位置と速度の計算（前節の（5））の部分を部分経路の挿入アルゴリズムに置き換えて探索を行う．

4. 提案手法

IPSO の問題点は，この手法の核となる挿入操作に時間を要することである．部分経路を挿入する際，条件にあった辺を選択するため，解空間がある程度，限定されている状況においては有効な手法といえる．しかし，粒子が広範囲に位置している状況では，$\boldsymbol{pb}_i$ や $\boldsymbol{lb}_i$ が不安定であるため，探索したい都市数が多いほど解空間が厳密解付近に限定されるまでに時間がかかる．

この問題を解消するため，我々は探索領域の分割を提案する．既存の手法では，図 2 （a）のように 1 つの問題に対して，1 つの巡回路で解を探索していたが，これを図 2（b）のように，2 つの巡回路に分割し，解の探索を始める．そして，それぞれの解が収束したところで 2 つの巡回路を 1 つにし，従来手法通りに探索する．

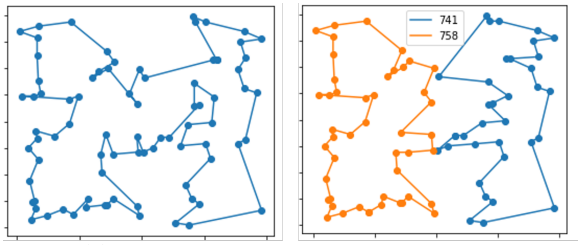

図 2　提案手法のイメージ
Fig. 2　Illustration of proposal method

提案手法の詳細なアルゴリズムを以下に示す．

(1) 探索領域の分割

探索したい都市群を任意の分割数 p で分割する．説明のために，$p = 2$ の場合における分割後の都市群を A, B とする．

(2) 粒子の初期化

各粒子 i が A と B のどちらを探索するかを割り振り，$\boldsymbol{x}_i$ に適当な巡回路を設定し，$\boldsymbol{pb}_i = \boldsymbol{x}_i$ とする．

(3) $\boldsymbol{pb}_i$ と $\boldsymbol{lb}_i$ の更新

$C(\boldsymbol{x}_i) < C(\boldsymbol{pb}_i)$ ならば，$\boldsymbol{pb}_i = \boldsymbol{x}_i$ とする．

また，以下の式に従い，$\boldsymbol{lb}_i$ を更新する．

$$\boldsymbol{lb}_i = \arg\min_{\boldsymbol{x} \in L_i^p} C(\boldsymbol{x}) \tag{3}$$

$C(\boldsymbol{x})$ は，ある巡回路 $\boldsymbol{x}$ の総距離を計算する関数，L_i^p は，粒子 i の近傍 L_i における $\boldsymbol{pb}$ の集合である．

(4) 粒子の位置を更新

従来手法と同様に，部分経路の挿入を用いて次の時刻での粒子の位置を求める．

(5) 終了条件の判定

分割中に終了条件を満たした場合は （6）へ進み，分割をしておらず終了条件を満たした場合は探索を終える．それら以外は （3）へ戻る．

(6) 探索領域の結合

A を探索していた粒子の解と B を探索していた粒子の解を結合し，新たな巡回路を生成する．この際，ある粒子 i の解をどの粒子の解と結合させるかは，乱数により決定し，結合後に $\boldsymbol{pb}_i = \boldsymbol{x}_i$ とする．全ての粒子が結合を終えたら，（3）へ戻る．

（1）での領域の分割方法は，都市群の座標の中央値を用いる．また，（2）での割り振りは，分割後の都市数の比に応じて決定する．具体的には，粒子数 $m = 100$，都市数 $N = 20$ で，分割後の都市数の比が $A : B = 12 : 8$ だったとき，A を 60 個，B を 40 個の粒子で探索させる．

5. 実験

5.1 実験条件

今回提案する手法が，既存手法と同精度の解を求めることができ，かつ計算時間を短縮できているか確認するために次のような実験を行う．

まず，本研究で扱う TSP の問題は，TSP の問題集として知られている TSPLIB[7] に記載されている，kroA100 と lin105（それぞれ都市数が 100，105 の問題）である．kroA100 は従来手法が提案されたときの実験に使用された問題で，lin105 は kroA100 と同程度の都市数で，都市の配置が大きく異なる問題として我々が選んだ．これらの問題では，都市群が位置座標によって与えられており，図 3 と図 4 のような配置となっている．それぞれの図にある赤線は，x 座標の中央値を表している．今回の実験では分割数を 2 とするため，この直線によって探索領域が分割される．このとき，分割後の都市数は，kroA100 が 49 : 51，lin105 が 54 : 51 となり，極端な偏りはない．

終了条件は，$\boldsymbol{pb}_i$ と $\boldsymbol{lb}_i$ の更新と粒子の位置を更新する操作を 1 ステップとして，5000 ステップに達した時とする．また，分割時の終了条件は，いずれかの分割領域において $\boldsymbol{gb}$ の総距離が 10 回，変わらなかった時とする．$\boldsymbol{gb}$ の定義は以下に示す．M^p は，ある領域を探索している全粒子の $\boldsymbol{pb}_i$ の集合である．

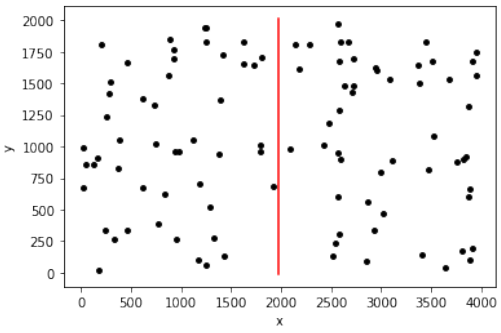

図 3　都市の配置（kroA100）
Fig. 3　Cities' layout for kroA100

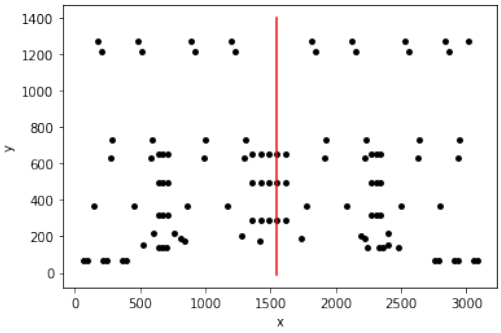

図 4　都市の配置（lin105）
Fig. 4　Cities' layout for lin105

$$\boldsymbol{gb} = \arg\min_{\boldsymbol{x} \in M^p} C(\boldsymbol{x}) \tag{4}$$

探索に用いる粒子数 m は 32 個，領域の分割数は先述の通り $p = 2$ とする．部分経路の抽出で用いるパラメータ c_1, c_2 は，本来，調整が必要であるが，従来手法の論文内では，同じ条件下で kroA100 を解いた際，$c_1 = 0.7, c_2 = 0.1$ の時に最も厳密解に近づいたとされていたため，本研究でも同じ値を用いる．

今回は，厳密解との誤差と処理時間を各ステップごとに計算する．各々の手法で 10 回試行し，平均した結果を示す．実験環境の CPU は，Intel Core i7-8700 (3.20GHz)，Python のバージョンは 3.7.6 を使用する．

5.2　実験結果（kroA100）

5.2.1　探索性能の比較

図 5 は，kroA100 を各手法で解いたときの最良解 $C(\boldsymbol{gb})$ と厳密解の誤差を 100 ステップごとに計算した結果の平均値をグラフ化した．横軸がステップ数，縦軸が誤差で，実線が従来手法，破線が提案手法となっている．提案手法の方の線が途中からになっているのは，領域を分割していたため厳密解との誤差を計算できないからである．実験の結果，どちらの手法においても，0.1%未満の誤差になるまで探索できており，分割したことによる探索性能の悪化はないといえる．

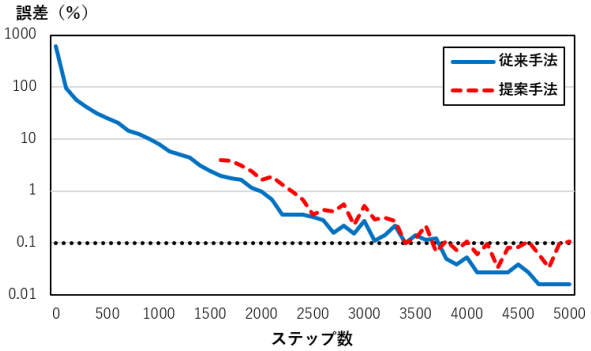

図 5　平均誤差の比較（kroA100）
Fig. 5　Comparison of average error (kroA100)

5.2.2　処理時間の比較

各手法における 1 ステップごとの処理時間の推移を図 6 と図 7 に示す．図 6 は，提案手法で行った 10 回の試行のうちの 2 回分について，図 7 は，従来手法と提案手法のそれぞれにおける 10 回分の平均について，100 ステップごとに処理時間を出力させたものである．どちらの図も横軸がステップ数，縦軸が処理時間となっており，図 7 の菱形は従来手法，バツ印は提案手法の結果となっている．

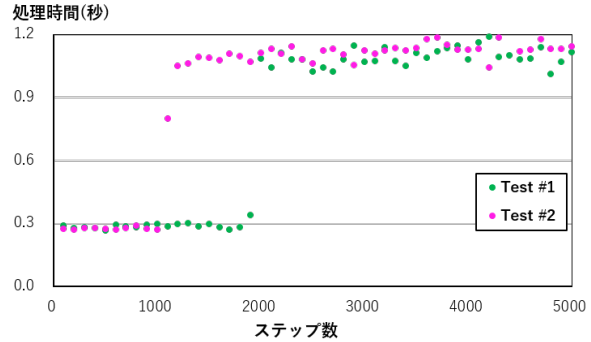

図 6　提案手法における異なる 2 試行での処理時間
Fig. 6　Calculation time for two attempts in proposal method

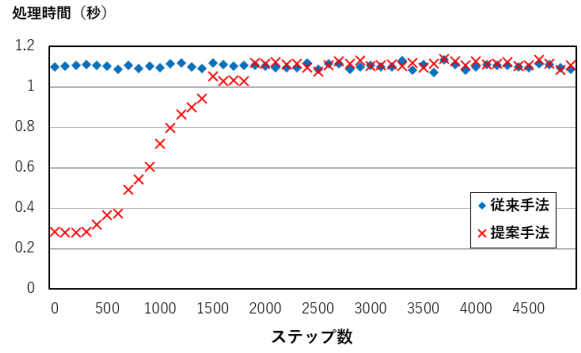

図 7　各ステップの平均処理時間
Fig. 7　Average calculation time at each step

提案手法の試行では，図 6 のように試行ごとに処理時間

が急に遅くなるタイミングが存在する．これは領域を分割した状態での探索が終わるタイミングと一致しているが，試行ごとにこのタイミングは大きく異なっている．また，図 7 において，およそ 500 から 1500 ステップの部分で提案手法の処理時間が徐々に増加しているように見える．これは 10 回分の試行を平均したことによるもので，それぞれの試行で領域分割を終了するタイミングが異なるからである．

表 1　平均処理時間
Table 1　Average calculation time

	分割中	分割なし	合計処理時間
従来手法	None	1.103 秒	5515.984 秒
提案手法	0.292 秒	1.114 秒	4668.681 秒

表 1 は，各手法における処理時間の平均と，合計処理時間の平均である．提案手法ではさらに分割をしていた時とそうでないときで分けて表示させている．この表と図 7 から，分割しているときの処理時間は，分割していないときに比べて高速化できていることが分かる．定性的に見ると，5000 ステップを終えるのにかかった時間が，従来手法よりも約 1.2 倍速くなっている．

5.3　実験結果（lin105）
5.3.1　探索性能の比較

lin105 を各手法で解いたときの各ステップにおける平均誤差を図 8 に示す．これを見ると，提案手法の誤差が従来手法よりも悪化しているように見える．しかし，10 回の試行のうちで最も良い結果を得られた試行における誤差の推移（図 9）を見ると，どちらも 0.1%未満の誤差になるまで探索できていることが分かる．

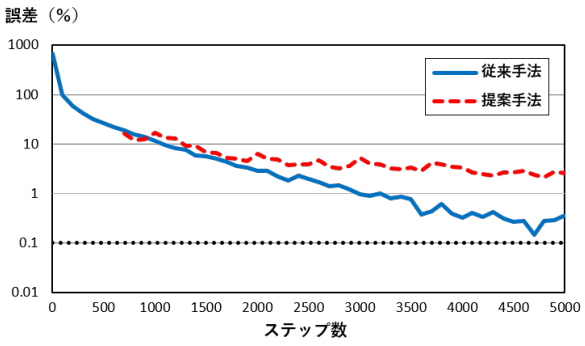

図 8　平均誤差の比較（lin105）
Fig. 8　Comparison of average error (lin105)

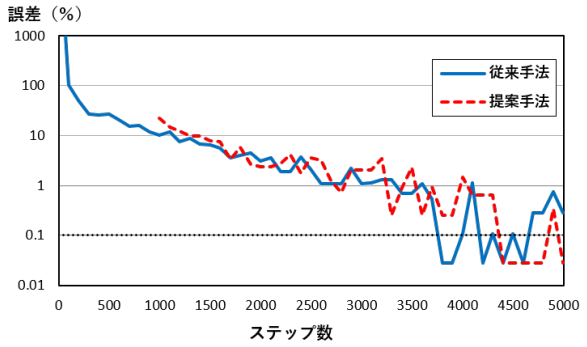

図 9　最良解を得られた試行の誤差推移
Fig. 9　Error of best attempt

5.3.2　処理時間の比較

各手法における 1 ステップごとの処理時間の推移を図 10 と図 11 に示す．kroA100 の時と同様に，図 10 は，提案手法で行った 10 回の試行のうちの 2 回分について，図 11 は，従来手法と提案手法のそれぞれにおける 10 回分の平均についてのグラフである．

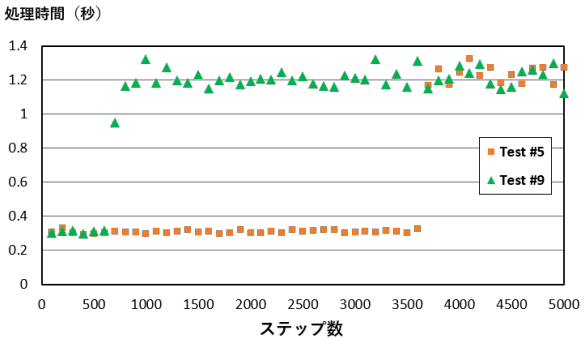

図 10　提案手法における異なる 2 試行での処理時間
Fig. 10　Calculation time for two attempts in proposal method

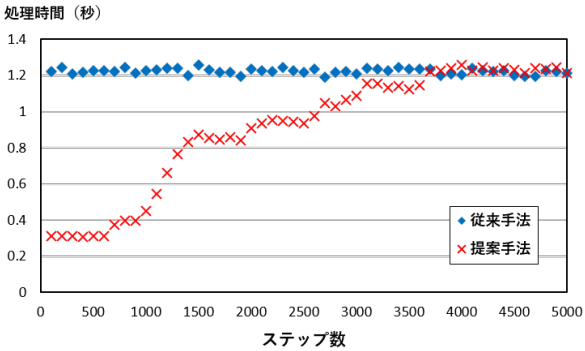

図 11　各ステップの平均処理時間（lin105）
Fig. 11　Average calculation time at each step (lin105)

図 10 では，10 回の試行のうち，最も早く分割探索が終了したものと，最も遅く分割探索が終了したものを選択して表示させている．この図より，分割探索が終了するタイ

ミングが最大でおよそ 3000 ステップの開きがあることが分かる．また，図 11 についても，kroA100 で計測した図 7 と比べると，kroA100 の時よりも試行ごとに分割探索が終わるタイミングが分散していることが分かる．

5.4 考察

今回の実験では，分割中の処理時間が，分割していない時と比べておよそ 1/4 になった．これは，実験に使用した TSP の問題が，分割した際に都市数がほぼ半々になり，偏りが生じなかったため，各粒子の計算量が，$O(N^2)$ から $O((N/2)^2)$ に減ったと考えられる．

しかし，どちらの問題においても，領域を分割した状態での探索を終えるタイミングにはバラツキがあった．特に lin105 の問題では，分割探索の終了が遅すぎて，そのあとの探索を十分に行えてないと思われる試行が見られた．これは乱数によって抽出する部分経路の長さにバラツキが生じ，各粒子の解が一時的に悪化することを考慮した終了条件になっていない，または，分割後の都市の配置によるものだと考えており，追加実験を含め，検討が必要である．

6. まとめ

本研究では，IPSO に対し，探索領域の分割を導入し，解が収束するまでに要する時間の短縮を目指した．

TSP の問題である kroA100 を解いたときの探索性能の比較では，探索領域を分割した場合でも 0.1%未満の誤差まで探索できており，探索性能は悪化しないことが分かった．また，合計の処理時間が従来手法よりも約 1.2 倍速くなるなど，高速化を達成できた．しかし，別の問題では分割探索を終えるタイミングが試行によって分散してしまい，10 回分の試行を平均すると，従来手法と同程度の誤差にはならなかった．

今後の課題としては，分割探索を終了する条件の再検討と，分割探索終了後の高速化が挙げられる．分割探索を終了する条件については，同じ問題，もしくは同程度の都市数をもつ問題では分割探索を終了するタイミングをなるべく揃えることによって，試行ごとで結果に差が出ることを防ぎたいので，探索の終了条件および領域の分割方法を再検討する必要がある．また，分割探索終了後の高速化については，今回，高速化を達成できたのが 5000 ステップのうち 1000 ステップ前後だったため，残りの数千ステップについて高速化の検討をする必要がある．例えば，分割探索が終わった後について，$\boldsymbol{pb}_i$ と $\boldsymbol{lb}_i$ から抽出する部分経路の長さを出来るだけ長くするなどして計算量を削減できないかを検討したい．

参考文献

[1] D. Martens, B. Baesens, T. Fawcett: Editorial survey: swarm intelligence for data mining. Machine Learning, Vol. 82, pp.1-42 (2011).
[2] 斎藤利通: 粒子群最適化と非線形システム. IEICE Fundamentals Review, Vol.5, No.2, pp.155-161 (2011).
[3] 進藤卓也: 粒子群最適化法の動特性解析と応用. 日本工業大学 博士論文 (2014).
[4] 山本大輔, 三木彰馬, 榎原博之: 強化学習を用いた巡回セールスマン問題の解法. 情報処理学会第 81 回全国大会講演論文集, No.1, pp.357-358 (2019).
[5] 本庄将也, 飯塚博幸, 山本雅人, 古川正志: 巡回セールスマン問題に対する粒子群最適化の提案と性能評価. 知能と情報 (日本知能情報ファジィ学会誌), Vol. 28, No. 4, pp.744-755 (2016).
[6] J. Kennedy and R. Eberhart: Particle Swarm Optimization, IEEE International Conference on Neural Networks, Vol. 4, pp.1942-1948 (1995).
[7] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/