2F1-02

確率モデル遺伝的アルゴリズムEHBSAにおける戦略、パラメータの調査

Study on the Strategies and Parameters for the Probabilistic Model-Building Genetic Algorithm EHBSA

酒井大輔[*1] 上田和紀[*2]
Daisuke Sakai Kazunori Ueda

[*1]京都大学大学院情報学研究科 Graduate School of Informatics, Kyoto University
[*2]早稲田大学理工学部 Science and Engineering, Waseda University

We studied strategies and parameters for the TSP solutions of EHBSA (edge histogram based sampling algorithm), which is one of the probabilistic-model genetic algorithms, mainly from the perspective of accuracy. As the result of systematic experiments, we found some facts. For example, in the stage of population selection and parallelization, the elite strategy performed best.

For the implementation language, we chose a strongly-typed functional language OCaml, because of richness of libraries and efficiency of programming, and implemented EHBSAWO (EHBSA without template) and ENBSAWT (EHBSA with template) for a single processor and multiple processors,respectively. For multiprocessors, we used ocamlmpi, which is an interface module to the MPI message-passing library. By the parallelization, the accuracy (reciprocal of traversal distance) improved by 1.3 to 1.6 times than that of the single-processor version.

1. 背景と動機

従来の遺伝的アルゴリズム (GA) の欠点を補い、その上で生物のアルゴリズムの高い探索能力をコンピュータ上に実現する新たな手法として確率モデル GA と呼ばれるものが注目されている。確率モデル GA とは、従来の GA のように交叉オペレータにより子個体を生成せず、集団の個体分布を確率モデルで近似し、そのモデルに基づいて子個体を生成する手法である。

確率モデル GA としては、PBIL, compactGA, ECGA (extended compact GA) (参考文献 [5][6]) などがあるが、その適用はほとんど toy problem に限られ、従来の GA の代表的な応用であるスケジューリング問題や巡回セールスマン問題 (TSP) についての応用はほとんど見られない。

そのような中、確率モデル GA に基づいた TSP についての数少ない応用としては EHBSA (参考文献 [4]) があり、従来の GA を超える高いポテンシャルが期待できる。しかし、このアルゴリズムにおける最適な戦略、パラメータはまだ充分には研究されていない。確率モデル GA は従来の GA と同じように、集団数などの多くのパラメータや様々な戦略が存在し、これらの組み合わせによってその性能が大きく変化することを考えると、EHBSA を利用した TSP 解法における効果的な戦略、パラメータを調査することの意義は十分にあると考え、本研究を行なった。

2. EHBSA

EHBSA は確率モデル遺伝的アルゴリズムの一種であり、巡回セールスマン問題などの順序問題に対する解を探索するのに用いる。確率モデル遺伝的アルゴリズムは集団の個体分布を確率モデルで近似するが、EHBSA ではこれを edge histogram matrix (EHM) という行列で行なう。つまり、世代 t の集団があったとき、これから EHM^t をつくりこれを用いて sampling することで、世代 $t+1$ の集団をつくる。この繰り返しにより、より良い解を探索する。

連絡先: 酒井大輔、京都大学大学院情報学研究科知能情報学専攻、sakai@kuicr.kyoto-u.ac.jp

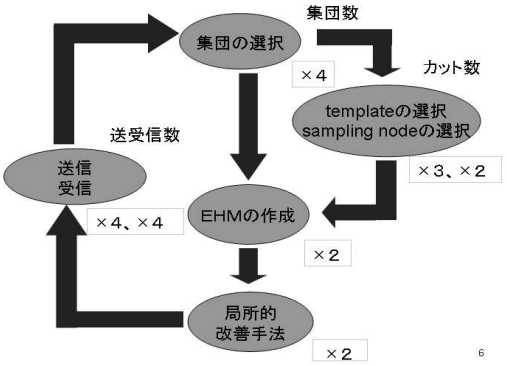

図 1: 全体の流れ

EHBSA のアルゴリズムの簡単な流れは次のようになる。

1. 個体をランダムに生成する
2. 選択オペレータを用い、有望集団をつくる
3. 有望集団から EHM を作成する
4. EHM を基に子個体を sampling する
5. 終了条件が満たされるまで、ステップ２から４をくり返す。

3. EHBSAWO, EHBSAWT

EHBSA には EHBSAWO(EHBSA without template) と EHBSAWT (EHBSA with template) という２種類の手法がある。EHBSAWO は第 2. 節で述べたほぼそのままのアルゴリズムだが、EHBSAWT は集団の中から template としてひとつの要素を取りだしそれらをいくつかの cut によって分け、その中の１つの segment だけを sampling し他の segment はそのままコピーされる。これによって優秀なブロックが次世代にも遺伝すること、世代交代にかかる計算量が短縮されるなどの効果が望める。

4. 実装したシステムと戦略

4.1 実装したシステム

EHBSAWO, EHBSAWT のアルゴリズムを用い、TSP の解法を探索するプログラムをそれぞれ逐次版と並列版の２種類ずつ作成した。その他に上記のアルゴリズムの計測と分析を効率良く行なうために、パラメータファイルを書き、それの直積を計算しそれらすべてを実行するユーティリティと、同様のパラメータファイルを受け取り結果を自動的にグラフにするユーティリティを実装した。

具体的には次のようなパラメータファイルを手作業で用意する。

```
pop_num;20,30,40
city_num;50
max_generation;30,40
population_select;no_population_select
make_ehm_strategy;make_ehm_weight:0.04
use_twoopt;use_twoopt,non_use_twoopt
```

このファイル名を makeparameter とする。これを使い、例えば EHBSAWO で計測したいときには次のように打ち込む。

```
ocaml str.cma unix.cma
 execute_ehbsawo.ml makeparameter
```

このようにすると各行の要素の直積集合の数だけ、自動的に実行し結果をファイルに格納する。上のパラメータファイルでは $3 \times 1 \times 2 \times 1 \times 1 \times 2 = 12$ 回計測する事になる。また、計測が終わっているとしてその結果が見たいときは次のように打つと実験結果を自動的にグラフ化する。

```
ocaml str.cma unix.cma
 read_ehbsawo.ml graph makeparameter
```

本研究では初期集団の選択、template の選択、sampling node の選択、EHM の戦略、局所的改善手法の有無、送受信の戦略に分けて戦略を実装した (この内、template, sampling node の選択は EHBSAWT のみであり、送受信の戦略は並列版のみの実装である)。図 1 にその全体の流れの図がある。丸い枠に囲まれているのが各フェーズを表し、右上に書いてあるものがパラメータ、左下に書いてあるのが実装した戦略の数になる。

1. 初期集団を選択し有望集団を作る。
2. EHBSAWT の場合は template, sampling node を選択する。
3. 有望集団から EHM を作成し、sampling する。
4. もし戦略に組み込まれていれば、2opt 法で局所的改善を行う。
5. 並列版の場合は各 CPU から他の CPU へと集団の要素を送信する。

4.2 実装した戦略

4.2.1 初期集団の選択のフェーズに置いては以下の４つの戦略を実装した。

- no_population_select：全く選択を行わない
- population_select_half：優れた上位半分を使う (つまり同じ内容のものが２つず作られる)
- population_select_roulette：適応度の割合に応じて子孫を残す
- population_select_tournament：ランダムに２つ選び、そのうちの優れた方が受け継がれ、それを集団の数だけ繰り返す

4.2.2 template を選択するフェーズでは以下の３つの戦略を実装した。

- choose_template_randomly：ランダムに template を選ぶ
- choose_template_roulette：適応度に応じて選択する
- choose_template_elite：最も良いものを template として選択する

4.2.3 sampling node を選択するフェーズでは以下の２つの戦略を実装した。

- choose_node_randomly：ランダムに sampling node を選ぶ
- choose_node_roulette：適応度に応じて選択する

4.2.4 EHM を作成するフェーズでは以下の２つの戦略を実装した。実験した中ではこれが最も効果の大きいファクターだった。

- make_ehm_delta：EHBSA の論文に基づいたデルタ関数を使う
- make_ehm_weight：独自の重みづけを行なう EHM を作成する

4.2.5 局所的改善手法として 2opt 法という従来の GA に使われているヒューリスティックスを使うか使わないかの２種類の戦略を実装した。

- use_twoopt：2opt 法を使う
- non_use_twoopt：2opt 法を使わない

4.2.6 送受信の戦略としては送信、受信に以下の４種類ずつ計８個を実装した。

- send_randomly, receive_randomly：ランダムに送受信する
- send_roulette, receive_roulette：適応度に応じて送受信するのを選ぶ
- send_elite：最も適応度の高いもの決められた数だけ送信する
- recive_elite：適応度の最も低いものを受信したものと取り替える
- send_tournament　：ランダムに２つ選びそのうち適応度の高いものを送信する。
- receive_torunament：ランダムに一つ選び、それと受信したものを比較し、優れているときのみ交換する。

4.3 ocamlmpi

本節では ocamlmpi について簡単に紹介する (参考文献 [1][2][3])。ocamlmpi は OCaml の MPI インターフェースであり、OCaml のプログラム上で MPI を利用した並列プログラミングを行える。ocamlmpi を用いることにより従来ほとんど C, C++, Fortran などに限られていた並列プログラムを、OCaml を用いて行えるようになりスクリプト言語の手軽さで並列化された実行コードを得ることが出来る。

ocamlmpi の最も大きな特徴は send と receive で任意の型が送受信出来るという事だ。 例えば、float と int をプロセス０から１まで送りたいとき、Cなどでは面倒なプログラムが必要になるが、ocamlmpi では

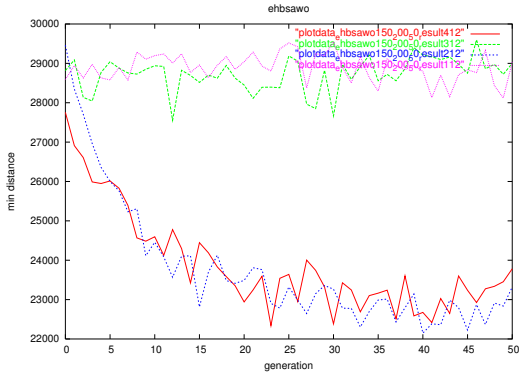

図 2: EHBSAWO における初期集団の選択

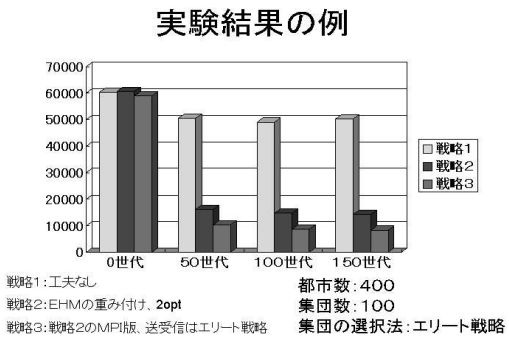

図 3: EHM による比較

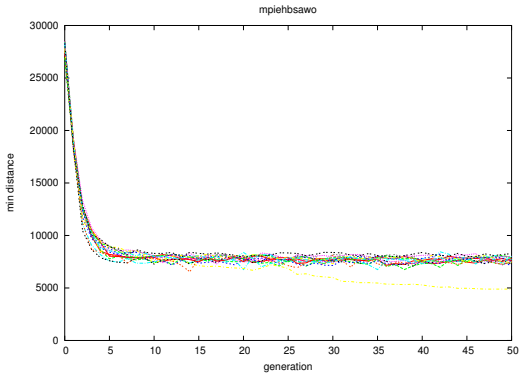

図 4: 並列版 EHBSAWO における送受信戦略

```
if myrank=0 then (
 let a = 1 and b = 2.0 in
  send (a,b) 1 0 comm_world
) else (
 let a,b = (receive 0 0 comm_world) in
  Printf.printf "int:%d, float:%f\n" a b
)
```

などと簡単に出来る。また次のように、受信するときに型を指定してプログラムを読みやすくする事も出来る。

```
 let x : int * float = (receive 0 0 comm_world) in
 let a,b = x in
```

5. 実験結果と考察

実験の結果、実行速度以外は EHBSAWO のほうが EHBSAWT よりも優れていた。これを示しているのが表 1 で、これを見ると解の精度という点では EHBSAWO が EHBSAWT を上回っている事、cut 数を増やすと実行時間は短くなるがその分解の精度が悪化すること。また与えられた都市数に対して集団数が大きすぎると解の精度がかえって悪化する事などが見て取れる。

このうち、cut 数の増加に伴い解の精度が悪化する問題については template をランダムに選ぶせいで、質の良くないものを template にしてしまっている可能性が高いことに原因が有ると思われた。そこで、最も適応度の高いものを template にして実行すれば、解の精度の向上が望めるか、と考え実行したが、ほとんど結果は変わらなかった。結局これは cut することで、EHM の重みづけの効果が少なくなってしまうことに大きな原因があると考えられる。よって EHBSAWT の改善には cut 数を可変にするなど他の手法が必要になる。

今回の研究では、解の精度に焦点を絞っているので以下では EHBSAWO のみを取り上げる。

有望集団を作成する手法としては、エリート戦略、トーナメント戦略が適当であることが分かった (図 2)。また当初の予想を裏切り、これは他の戦略にほとんど依存しなかった。つまり他の戦略、パラメータがどのようなものであれ、初期集団の選択においてはエリート戦略、トーナメント戦略が良い性能を示した。

EHM の作成のフェーズにおいてはデルタ関数を用いたものよりも、都市間の距離を考えた重み付けをおこない EHM を作成する戦略の方が良い性能を示した (図 3)。この EHM の作成における戦略が実装した戦略の中では、解の精度に対して最も大きな影響を持っていた。

MPI を用いた並列化では送受信における戦略には各プロセスが自分の集団の中で最も適応度の高いものを他のプロセスに送り、受け取る側はそれを自分の集団の中で最も適応度の低い要素と入れ替えるというエリート戦略が最も有効だった (図 4)。

また、送受信する数は都市数などに比べ比較的少量の場合のほうが解の精度は良く、逐次版に比べて 1.3 倍から 1.6 倍程度の向上が見られた (表 2)。しかし、本研究で実装した並列 EHBSA のアルゴリズムは安定性に今一つ難が見られた (表 3)。優れた安定性を持つ並列アルゴリズムはこれからの課題である。

なお、計測は SGI Altix 350 (Itanium2 × 8 , memory:40GB) 上で行ない、処理系に OCaml-3.08、拡張モジュールとして ocamlmpi-1.01 を用い、コンパイルは ocamlopt コマンドを使い行なった。また、計測は 1 回のみである。

6. 今後の課題

今後の課題としてはアルゴリズムの改良がある。EHBSA は世代交代にかかる時間は通常の GA よりも短いが、解の精度自体は通常の GA に劣る。これは戦略にエリート戦略を用い EHM を重みづけするというかなり強引に探索領域をせばめる戦略のため、解が局所解に陥る可能性が高いせいである。この点を改善するには探索空間を大きく取れるような戦略、パラメータを増やす必要がある。

また並列化したときに解の精度が大幅に向上することがあ

表 1: cut 数と計測時間、および解の精度

	集団数/最小距離/時間 (s)	集団数/最小距離/時間 (s)
EHBSAWO	150/8823.9/16.6	150/8823.9/16.6
EHBSAWT/1	150/9792.9/16.7	150/9792.9/16.7
EHBSAWT/2	150 /11824.8/7.63	300/12462.9/15.4
EHBSAWT/3	150/13729.6/5.35	450/14720.0/16.0
EHBSAWT/4	150/14436.0/4.43	600/17208.3 /17.9
EHBSAWT/5	150/15950.1 /4.17	750/19280.9/21.3

表 2: 並列版 EHBSAWO の実験結果 1

都市数	ＣＰＵ数	送受信数	並列版の解	逐次版の解	逐次版/並列版
200	7	5	4864.1	7657.1	1.57
400	8	10	8273.5	13525.5	1.63
600	6	10	15428.0	19317.1	1.25
800	8	10	17800.1	24701.0	1.38
1000	6	20	23639.5	30812.7	1.30

表 3: 並列版 EHBSAWO の実験結果 2

都市数	プロセス数	送受信数	並列版の解
200	6	5	8284.6
200	7	5	4864.1
200	8	5	5562.4
400	6	10	14291.7
400	7	10	10221.9
400	8	10	8273.5
600	6	10	15428.0
600	7	10	15757.1
600	8	10	18500.4
800	6	10	18160.2
800	7	10	18956.0
800	8	10	17800.1
1000	6	20	23639.5
1000	7	20	30047.3
1000	8	20	28848.7

るのは確認できたが、これは全く異なるかたちの集団の要素を取り入れることで探索空間が拡大することが理由として考えられる。しかし、本研究で実装した並列 EHBSA のアルゴリズムは最適なパラメータを与えられた条件 (都市数など) から導くのが難しく、精度の良い解を見つけるのに適切な計算ノードの数にも規則性は見つけられなかった。この点の改善は今後の研究課題になる。

参考文献

[1] Xavier Leroy : OCamlMPI: interface with the MPI message-passing interface
http://cristal.inria.fr/ xleroy/software.html#ocamlmpi

[2] 酒井大輔：ocaml 備忘録
http://www.ueda.info.waseda.ac.jp/
~sakai/ocaml/ocamlmemo.html

[3] Xavier Leroy : The Objecttive Caml system release 3.08 Documentation and user's manual, 2004
http://caml.inria.fr/pub/docs/
manual-ocaml/index.html

[4] Shigeyoshi Tsutsui : Probabilistic Model-Building Genetic Algorithms in Permutation Representation Domain Using Edge Histogram, Proc. of the 7th Parallel Problem Solving from Nature - PPSN VII, pp.224-233, 2002.

[5] Georges Harik : Linkage Learning via Probabilistic Modeling in the ECGA, Technical report, IIIiGAL Technical Report, No. 99010, 1999.

[6] Georges R. Harik, G.Lobo, David E. Goldberg : The Compact Genetic Algorithm, Proceeding of the 1998 IEEE Conference on Evolutionary Computation, pp. 523-528, 1998