並列 GA による TSP の効率的解法について

平林　永行・片山　謙吾*・成久　洋之**

岡山理科大学大学院工学研究科修士課程情報工学専攻

*岡山理科大学大学院工学研究科博士課程システム科学専攻

**岡山理科大学工学部情報工学科

(1997年10月6日　受理)

あらまし

本論文は，巡回セールスマン問題（TSP）に対する並列ハイブリッド遺伝的アルゴリズム（Parallel Hybrid Genetic Algorithms, PHGA）を用いて計算時間の短縮による加速率および効率性について記述する。具体的な効率性評価のために3つの優れた交叉法を取り上げる。比較に用いた3つの交叉法は，TSP に対して現在最も有望とされる Mühlenbein らの交叉（Müh-X）と，Starkweather らの分析によって優れた交叉法とされる，改良された Edge Recombination Crossover (IER-X)，最近提案された交叉法，完全サブツアー交換交叉（Complete Subtour Exchange Crossover, CSE-X）である。本実験では，ある定められた枠組みの下で各交叉法を用いて並列化による効率性を示す。

1．まえがき

組合せ最適化問題の多くは NP-完全なクラスに属し，大規模な問題に対して多項式時間で最適解を得ることは不可能であるとされている。しかしながら，理論的に最適解を多項式時間で求める解法が存在しなくても，経験的に比較的良好な解を求めるアルゴリズムは数多く提案されている。その一つとして，生物の進化過程にヒントを得た遺伝的アルゴリズム（Genetic Algorithm, GA）[1,2,3]が近年注目を集めている。GA は確率的探索原理により，さまざまな組合せ最適化問題や応用問題[4]に適用され，その成果を上げつつある探索アルゴリズムの一つである。一般的に GA は，遺伝的操作（選択，交叉，突然変異）を持ち，解空間に対して大域的な探索が非常に優れ，良好な解を算出できる解法である。

強力なロバスト性を誇る GA は，扱う問題に対して合致した構造を取りやすく，アルゴリズムの詳細な部分にわたり，いろいろな工夫を施すことで更に高性能な GA を開発することが可能である。そのような GA アプローチの一つとして，ハイブリッド GA（Hybrid GA, HGA）の研究が数多く行われている[7,8,18]。HGA は探索効率の向上を図り，高精度な解を得る目的で，局所的な探索に優れる Local Search（局所探索法）を組み込んだ，GA の拡張版である。

並列処理は逐次処理の操作手順を分割することによって効率よく処理を行い，逐次処理で得られる解を劣化させることなく計算時間の短縮を計るものである。一般的にGAは複数の候補解を常に保持しているので，逐次処理においてはかなりの計算時間を必要とし，更にLocal Searchをハイブリッドすることにより膨大な計算時間を必要とする場合が少なくない。よって本論文の検討事項として，HGAで保持される候補解の集団を並列化することにより，どの程度の計算時間の短縮が期待できるか，並列計算機を使用して，その加速率および効率性を示す。

その効率性評価の具体例として，巡回セールスマン問題(Traveling Salesman Problem, TSP) に対して現在有望とされるMühlenbein ら交叉と最近提案された新しい交叉法，完全サブツアー交換交叉をHGAに施した場合を検討する。片山らは，交叉性能を評価するHGAの枠組みを提案している[20]。従って本論文ではその枠組みを利用し，HGAで保持される候補解の集団をサブ集団に分割した並列化による，各交叉法のアプローチの違いによって生じる効率性の相違を検討する。

2．巡回セールスマン問題

組合せ最適化問題を代表する巡回セールスマン問題 (TSP)[5,15]とは，n 個の点から構成される無向完全グラフ $G=(V, E)$，枝上の距離関数 $d: E \rightarrow Z^+$ が与えられたとき，すべての点をちょうど1度ずつ経由する巡回路 (Hamilton 閉路) で，枝上の総距離を最短とする巡回路を求める最小化問題である。本論文で扱う TSP は，2点 i, j 間の双方向の距離が等しい($d_{ij}=d_{ji}$)対称巡回セールスマン問題 (symmetric TSP) である。実行可能解の数は，$(n-1)!/2$ に上る。

```
procedure genetic algorithms
01 : begin
02 :  k := 0;
03 :  P(k) := a set of initial feasible solutions;
04 :  evaluate structures in P(k);
05 :  while stopping-criterion ≠ yes do
06 :    begin
07 :      k := k+1;
08 :      P(k) := select from P(k-1);
09 :      alter structures in P(k) by crossover;
10 :      alter structures in P(k) by mutation;
11 :      evaluate structures in P(k);
12 :    end
13 :  return the best solution;
14 : end
```

図1　一般的な遺伝的アルゴリズム

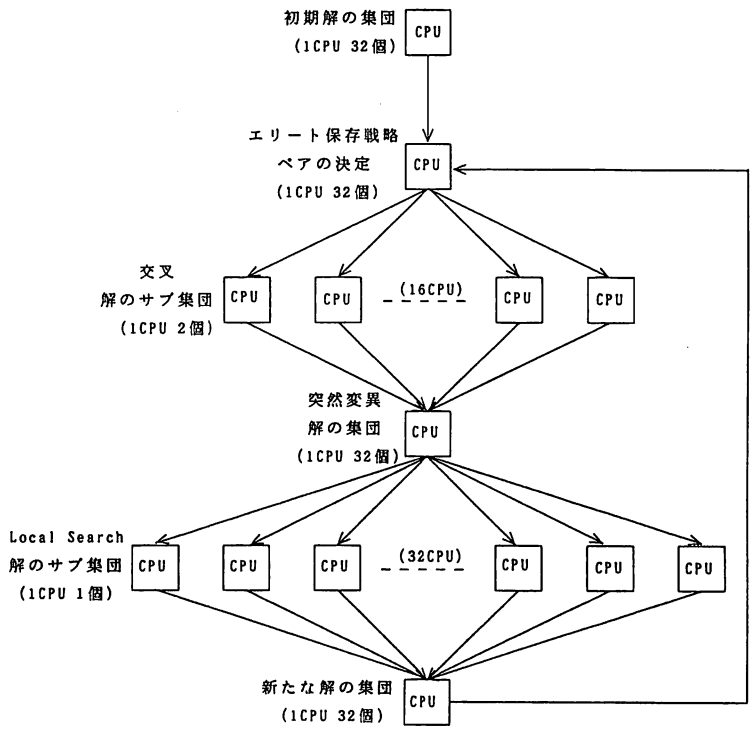

図2　32 CPU の処理手順

3．遺伝的アルゴリズム

遺伝的アルゴリズム（GA）は，問題に対する候補解を記号列からなる個体（individual）に対応させるためのコード化を必要とする。すなわち，問題空間と GA 空間での対応づけを意味する。そこでは，問題空間での実際の解を表現型，GA 空間での記号列の個体を遺伝子型とよぶ。GA の基本的な流れを図1に示す。GA では，世代 k において，複数個の遺伝子型から構成される個体 I を保持し，それを集団（population）$P(k)=\{I_1^k, \cdots, I_p^k\}$ とよぶ。更に各個体は一つまたは複数の染色体(chromosome)を持ち，複数の遺伝子(gene)の並びによって特徴づけられる。あらかじめ定められた環境に適応した個体が，子孫(offspring)を残す確率が高くなるように世代交代を繰り返し，遺伝子およびその集団を進化させる過程をとる。なお，各個体がその環境に適応する度合を適応度(fitness)，染色体と適応度を対応させる関数を適応度関数(fitness function)とよぶ。基本的に GA は，選択(selection)，交叉（crossover），突然変異（mutation）とよばれる遺伝的操作（genetic operations）を，生物進化における淘汰，繁殖などの原理として持っている。

4．並列ハイブリッド遺伝的アルゴリズム

並列ハイブリッド遺伝的アルゴリズム（Parallel Hybrid GA, PHGA）について記述す

る。並列化に際しては，GA で用いるすべての候補解（個体）の集団をいくつかのサブ集団で構成する方法などさまざま考えられる[2,3,7,8]。本実験において，PHGA で並列化の対象とするのは，主として交叉とハイブリッドされた Local Search の処理部分であり，分割されたサブ集団に対して各世代ごと行われる。つまり，対象となる個体集団に対して，複数個の CPU で並列化する場合，各 CPU は，$\frac{\text{個体集団}}{\text{CPU 数}}$ 個の個体で基本的に構成される。

以下，並列化に関する実験の詳細について記述する。実験では並列計算機を使用し，CPU 数が最大32までの場合を検討する（なお，処理中に保持される GA の全個体数は32）。PHGA で並列化の対象となるのは，交叉とハイブリッドされた Local Search の処理部分であるので，1 CPU に割り当てられる個体数は，CPU数 1，2，4，8，16，32に対して，それぞれ32(16)，16(8)，8 (4)，4 (2)，2 (1)，1 (1)個の個体が Local Search に適用される。( )内の数値は交叉を適用する際に 1 CPU に割り当てられるペアー数を示す。つまり，交叉を適用する際には最低でも二つの個体が必要となるので，GA で保持される全個体数に対しては，$\frac{\text{個体数}}{2}$ 個のペアーで構成される。よって，CPU 数16，32で，そのペアー数は 1 になることに注意されたい。なお，図 2 は，32個の CPU を用いて並列化を行った場合の処理手順を表す。本 PHGA では，交叉の部分，Local Search の部分を並列化しており，それぞれで最後に処理が終了した CPU に合わせて同期を取っている。

5．比較する交叉法について

TSP に対する交叉にはさまざまな方法が提案されている。その多くは交叉により実行不可能解になることを回避する工夫が施される。以下に取り上げる 3 つの交叉は常に実行可能解を保証した方法である。特に Mühlenbein らの交叉は，現在 TSP で最も有望とされる方法として多くの紹介がある[12,13,15]。

5．1 Mühlenbeinらの交叉

図 3 に Mühlenbein らの交叉（Müh-X）[7]の簡単な例を示し説明する。長さ n の 2 つの親（順列）に対して一つの順列に基準を置き，交叉が行われる。図 3 では，親 1 に基準を置く。親 1 の順列に対して長さ10から $n/2$ のサブツアーをランダムに決定し切り取る（S_1, S_2, S_3が切り取られた親 1 のサブツアー）。そのサブツアーに含まれない点を，親 2 で構成された順番（a, b, c, d, e）を壊さないように組み込むことにより子孫を生成する。また，Müh-X は，Local Search とのハイブリッド化を強く要求する交叉法である。

5．2 Improved Edge Recombination Crossover

Improved ER-X（IER-X）[16,17]は，二つの順列に含まれる点と点の間のリンク状態を示す Edge Table を作成し，それに含まれる Edge の少ない点，または強くリンクされている点を次の候補点として子孫を生成する。図 4 に IER-X の簡単な例を示し説明する。

図 4 では，親 1（a, b, c, d, e, f, g, h）と親 2（g, c, b, a, d, h, f, e）に対して，Edge Table を作成する。親 1 では，City a のリンクは，b と h である。また親 2 では，b と d

親 1	c	a	b	S_1	S_2	S_3	e	d
親 2	a	S_1	b	c	S_3	d	S_2	e
子孫	a	S_1	S_2	S_3	b	c	d	e

図 3　Mühlenbein らの交叉の一例

親 1	a	b	c	d	e	f	g	h
親 2	g	c	b	a	d	h	f	e

City	Edge Table			
a	h	$-b$	d	
b	$-a$	$-c$		
c	$-b$	d	g	
d	c	e	a	h
e	d	$-f$	g	
f	$-e$	g	h	
g	f	h	e	c
h	g	a	d	f

図 4　改良 edge recombination crossover の一例

である。その状況から，親 1 と親 2 には互いに b が含まれているので，この場合に Edge Table ではマイナスの記号をつけることで，強くリンクされていること示す。このような処理を行い Edge Table を作成し，子孫を生成する。子孫生成に際しては，ランダムを使用する。仮に City f がランダムに選ばれた場合には，Edge Table の City f の欄を見る。そこでは，City f の次の候補都市-e, g, h が含まれている。次の都市を選ぶ場合には，その候補都市からマイナス記号のついた都市を優先する。また，複数ついている場合や全くついていない場合には，その候補都市のそれぞれの欄をチェックし，候補都市の数の少ない都市を選び，子孫を完成させる処理過程をとる。使用された都市はその都度，この Edge Table から削除されていく。よって，そのような条件から，City f の次の都市は City e となり，次々に都市が決定されていく。なお，上述した条件が満足されないことがまれに現れる。この例では City d と h が City a の次の候補都市として現れる。その場合には，ランダムに City d と h の都市が選ばれる。よって，子孫は（f, e, g, c, b, a, d, h）もしくは，（f, e, g, c, b, a, h, d）の場合があり得る。

5．3 完全サブツアー交換交叉

完全サブツアー交換交叉（complete subtour exchange crossover, CSE-X）[19]は，2 つの親に共通する全く同じ方向性を有するサブツアー（逆方向のサブツアーも含む）を共有サブツアーと考慮して子孫を生成する。以下，簡単にその特徴を挙げる。

- 共有サブツアーの列挙が極めて高速な $O(n)$の時間で可能
- 交叉で生成される最良の子孫を次世代に残すが，子孫の生成法は順列に対する TSP の 2-Opt 操作[13,15]に相当，またはそれが可能

図 5 は，対象となる共有サブツアーが両親に 2 つ存在する場合の CSE-X の交叉例を示す。図 5 で対象となる共有サブツアーは，親 A の $\boxed{b,c}$ と $\boxed{f, g, h}$，親 B の $\boxed{h, g, f}$ と $\boxed{c, b}$ である。CSE-X で対象となる共有サブツアーは，その部分集合の方向性が互いに同順及び逆

親A(a, [b, c], d, e, [f, g, h], i, j)
親B(e, j, [h, g, f], a, i, [c, b], d)
⇩
子1(a, [c, b], d, e, [f, g, h], i, j)
子2(a, [b, c], d, e, [h, g, f], i, j)
子3(a, [c, b], d, e, [h, g, f], i, j)
子4(e, j, [f, g, h], a, i, [c, b], d)
子5(e, j, [h, g, f], a, i, [b, c], d)
子6(e, j, [f, g, h], a, i, [b, c], d)

図5 完全サブツアー交換交叉の一例

順になる場合のみが考慮される。例えば，子1では親Bの[c, b]を親Aの[b, c]の部分へ組み込む操作を施している。また，子4では親Aの[f, g, h]を親Bの[h, g, f]の部分へ組み込んでいる。残りの子孫も，同様な操作により生成する。つまり，山村らによるサブツアー交換交叉[6]で対象とされた共有サブツアーの選び方を一部簡略化したものと考えられる。そうすることによって，さまざまなメリットが文献[19]で報告されている。

6. 実　　験

6.1 実行環境

本実験では並列ハイブリッド遺伝的アルゴリズム（PHGA）を検討している。並列計算機は，インテル社の MIMD 処理方式の超並列計算機 Paragon XP/S15（プロセッサ数：296，各プロセッサ演算性能：75 MFlops，各プロセッサメモリー：32 MB，通信速度：250 MB/S）を使用した。使用言語はすべてC言語である。

6.2 パラメータ設定と実験の詳細

本実験では，並列化による3つの交叉法により PHGA の効率性を検討するので，交叉法以外の PHGA 操作は同一にしておく必要がある。そこで，文献[20]で示された交叉性能を評価する HGA の枠組みを利用する。その枠組みの概要を以下に示す。

- 制御パラメータなしの基本的な近傍操作を有した Local Search をハイブリッドする
- 極端に多様性をもたらす突然変異ではなく，一般的に使用されるものを採用
- エリート保存戦略を採用

GA はいくつかのパラメータを必要とする。基本的には個体数 PS，交叉確率 p_c，突然変異確率 p_m である。このようなパラメータはユーザー自身が計算機環境や必要とする解の精度，問題規模などのさまざまな要因に応じて決定しなければならない。また最適なパラメータ設定は，最適な探索を行う上で必要不可欠である。しかしながら，本実験で利用する枠組みにおいては，パラメータの違いによる正確な評価の困難を避けるために，PS：

32, p_c：1.0, p_m：0.005とすべて一定とした。またその枠組みから，GA における選択にはエリート保存戦略[1,2]を採用する。突然変異については，TSP に対して一般的な一回の 2-Opt 近傍の交換をランダムに施す。対象とした問題は，TSP のベンチマーク問題集 TSPLIB[11] から最適解既知の問題を31問選び，各問題，各アルゴリズムに対して20回の試行を行う。PHGA 処理の終了条件は，200世代で計算を打ち切る。

6.3 並列計算の評価

本節では，各交叉法を有する PHGA に対して CPU 数が 1 から32までの場合を検討する。(詳細は 4 に記述)

表 1，2，3 は，kroA100問題に対する，各 PHGA が保持する CPU 数別の解の質(最良，最悪，平均値)，平均の計算時間，加速率および最適解算出回数を示し，図 6，7 は，kroA100問題を用いた各 PHGA の CPU 数の変化による加速率と計算時間の変化につい

表 1　CPU 数別の PHGA (M̈uh-X) の実験結果（試行回数：各20回）

CPU 数	PHGA (M̈uh-X) (%)			Cpu Time (sec)	加速率	最適解
	Min.	Max.	Avg.	Avg.	Avg.	算出回数
1 CPU	0.000	0.348	0.108	1974.3	-	9
2 CPU	0.000	0.710	0.074	1050.5	1.9	14
4 CPU	0.000	0.334	0.049	580.2	3.4	13
8 CPU	0.000	0.348	0.096	338.6	5.8	9
16CPU	0.000	0.287	0.037	215.2	9.2	12
32CPU	0.000	0.301	0.090	154.0	12.8	9

表 2　CPU 数別の PHGA (IER-X) の実験結果（試行回数：各20回）

CPU 数	PHGA (IER-X) (%)			Cpu Time (sec)	加速率	最適解
	Min.	Max.	Avg.	Avg.	Avg.	算出回数
1 CPU	0.000	0.428	0.117	1845.7	-	9
2 CPU	0.000	0.301	0.027	990.3	1.9	15
4 CPU	0.000	0.348	0.025	568.2	3.2	17
8 CPU	0.000	0.409	0.073	347.0	5.3	10
16CPU	0.000	0.108	0.035	231.8	8.0	9
32CPU	0.000	0.428	0.066	173.3	10.7	12

表 3　CPU 数別の PHGA (CSE-X) の実験結果（試行回数：各20回）

CPU 数	PHGA (CSE-X) (%)			Cpu Time (sec)	加速率	最適解
	Min.	Max.	Avg.	Avg.	Avg.	算出回数
1 CPU	0.000	0.301	0.030	1065.6	-	18
2 CPU	0.000	0.301	0.050	572.9	1.9	16
4 CPU	0.000	0.301	0.044	317.0	3.4	17
8 CPU	0.000	0.301	0.059	195.5	5.5	16
16CPU	0.000	0.301	0.015	136.0	7.8	19
32CPU	0.000	0.301	0.060	107.8	9.9	16

て表す。なお，初期解は TSP 専用アルゴリズムで貪欲性に基づく Nearest Neighbor (NN) 法により生成されている。まず解の質については，3つの PHGA で算出される解の質と最適解算出回数に若干の違いは見られるものの，各 PHGA については複数に並列化を行おうが得られる解に違いはほとんど見られず，並列化によって解の劣化は確認されない。次に計算時間および加速率では，CPU 数増加に伴う並列化によって計算時間の短縮に成功している。しかし図6から加速率では，理想値から徐々に離れる傾向があり，1CPU に対して32CPU の場合では，約1/13，1/11，1/10の時間をそれぞれの PHGA は要することを確認した。また PHGA (Müh-X) が加速率は優れているが，図7から全体の計算時間では，PHGA (CSE-X) が他の交叉法より約1/2～2/3の計算時間で終了し，更に各 PHGA もほぼ同程度の解を算出している。しかし，最適解算出回数では PHGA(CSE-X) が，頻繁に最適解を算出していることが確認できる。なお，解の質は $\frac{\text{得られた解}-\text{最適解}}{\text{最適解}} \times 100\%$ で算出し，N 個の CPU に対するスピードアップ (加速率) $S(N)$は，$\frac{\text{1 CPU での計算時間}}{N \text{ 個の CPU での計算時間}}$ で算出した。

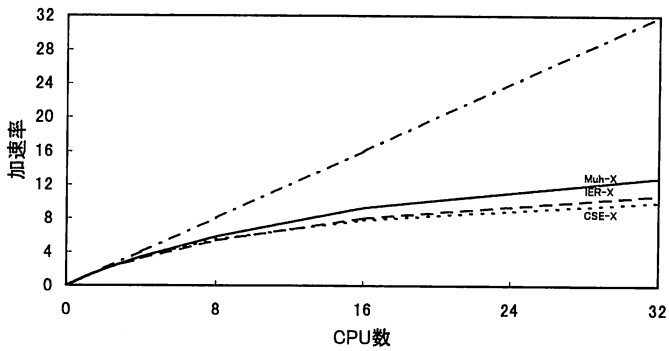

図6　加速率の変化 (kroA100)

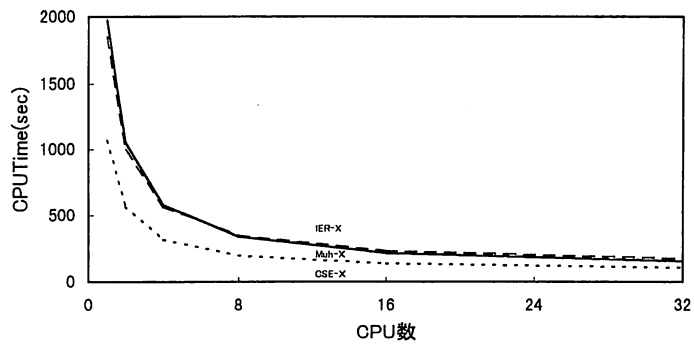

図7　計算時間の変化 (kroA100)

6.4 TSPLIB による評価

表4に TSPLIBから都市サイズ48～2392都市までの30問を取り上げ，NN 法で得られれた初期解に対して，32個の CPU を使用し，PHGA(Müh-X)，PHGA(IER-X)と PHGA (CSE-X)が算出した，解の質(最良，最悪，平均値)を示し，図8は，都市サイズ48～2392都市の計算時間の変化を表す。表4から，Müh-X，IER-X と CSE-X を比較した場合，問題サイズが比較的小さい場合では解の質にあまり大きな違いは見られないが，問題サイズが大きくなるに従い徐々に CSE-X が良好な解を算出していることがわかる。さらに図8から，比較的都市サイズが小さいときに各 PHGA は視覚的に見るとあまり計算時間に差はないが，都市サイズが大きくなるにつれて PHGA (Müh-X) は PHGA (CSE-X) の約2倍，PHGA (IER-X) は PHGA (CSE-X) の約3倍もの計算時間が必要になるということがわかる。

表4　各 PHGA の実験結果（試行回数：各20回，32CPU 使用）

TSPLIB	Optimal	PHGA (Müh-X) (%)			PHGA (IER-X) (%)			PHGA (CSE-X) (%)		
		Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.
att48	10628	0.000	0.000	0.0000	0.000	0.094	0.014	0.000	0.000	0.000
eil51	426	0.000	0.469	0.235	0.000	0.704	0.211	0.000	0.704	0.223
pr76	108159	0.000	0.237	0.065	0.000	0.250	0.102	0.000	0.672	0.120
kroA100	21282	0.000	0.301	0.090	0.000	0.428	0.066	0.000	0.301	0.060
kroB100	22141	0.000	0.673	0.235	0.000	1.369	0.430	0.000	0.696	0.335
rd100	7910	0.000	0.834	0.281	0.000	0.923	0.338	0.000	1.732	0.331
eil101	629	0.159	1.749	1.153	0.159	2.226	0.906	0.000	1.431	0.350
lin105	14379	0.000	0.327	0.043	0.000	0.536	0.081	0.000	0.153	0.008
pr107	44303	0.000	0.697	0.202	0.000	0.799	0.185	0.000	0.305	0.063
pr124	59030	0.000	0.219	0.027	0.000	0.366	0.072	0.000	0.078	0.004
pr136	96772	0.245	2.412	0.925	0.240	1.351	0.665	0.013	1.583	0.412
pr144	58537	0.000	0.000	0.000	0.000	0.085	0.028	0.000	0.000	0.000
kroA150	26524	0.385	1.267	0.917	0.015	1.312	0.774	0.000	1.817	0.572
kroB150	26130	0.134	1.359	0.789	0.199	1.546	0.789	0.000	1.286	0.414
pr152	73682	0.000	0.782	0.217	0.000	1.566	0.431	0.000	0.185	0.102
kroA200	29368	0.439	1.423	0.909	0.163	1.171	0.746	0.136	0.804	0.334
kroB200	29437	0.866	2.524	1.635	0.900	2.813	1.864	0.071	2.307	1.187
pr226	80369	0.005	0.317	0.099	0.020	0.439	0.094	0.000	0.302	0.020
pr264	49135	0.041	1.398	0.548	0.000	1.626	0.506	0.000	1.089	0.202
pr299	48191	1.260	3.156	1.989	1.251	2.986	2.124	0.268	1.089	0.689
lin318	42029	1.190	3.117	2.276	1.030	2.877	1.950	0.742	2.476	1.626
pr439	107217	1.166	2.331	1.751	0.869	2.074	1.441	0.242	2.538	1.171
d493	35002	1.926	3.928	2.856	2.626	4.108	3.261	0.920	2.274	1.587
att532	27686	3.319	4.490	3.887	3.236	4.677	4.049	0.842	2.879	2.136
u574	36905	3.346	5.926	4.763	3.943	5.856	4.908	2.314	4.257	3.101
rat575	6673	3.794	5.286	4.775	4.208	6.275	5.362	1.919	3.219	2.583
rat783	8806	4.826	6.359	5.618	5.087	6.700	5.963	2.941	4.020	3.488
dsj1000	18659688	4.521	6.154	5.436	6.033	7.115	6.541	2.876	4.802	3.659
pr1002	259045	4.377	6.041	5.085	5.026	6.593	5.852	2.183	3.456	2.907
u1060	224094	4.233	6.125	5.399	5.510	6.714	6.139	2.906	4.676	3.781
pr2392	378032	6.192	8.029	7.121	7.049	8.891	8.124	4.301	5.352	5.002

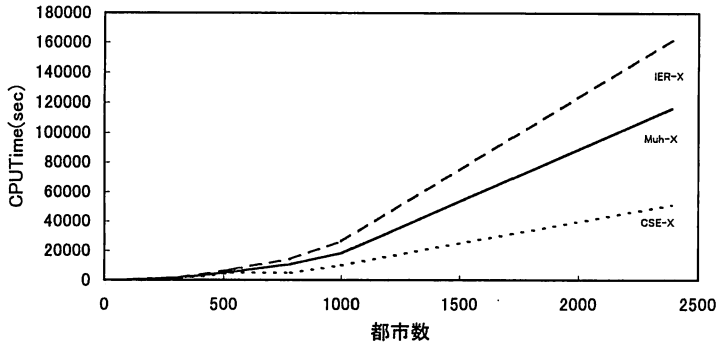

図8 計算時間の変化

6.5 考　　察

表1，2，3，図6，7から，加速率と計算時間に対して考察する。3つの交叉法について，1CPU での計算時間を比較すると，PHGA (Müh-X) と PHGA (IER-X) は，PHGA (CSE-X) の約2倍もの計算時間が必要となってくる。これは，PHGA (Müh-X) の場合ハイブリッドした Local Search の処理部分でかなりの計算時間が必要になり，PHGA (IER-X) の場合は，交叉の処理部分でかなりの計算時間が必要になるからだと思われる。また CPU 数を増やし，32CPU を用いた並列化の場合では，PHGA (Müh-X) は PHGA (CSE-X) の約1.5倍，PHGA (IER-X) はPHGA (CSE-X) の約1.7倍の計算時間を要している。PHGA (Müh-X) は，並列処理部分が高い割合で並列化され，1CPU のときに比べ効率的に PHGA (Müh-X) に働いたものと考えられる。次に加速率については，32CPU のとき，Müh-X で12.8倍，IER-X で10.7倍，CSE-X で9.9倍であった。このことより，PHGA (CSE-X) と PHGA (IER-X) より PHGA (Müh-X) が，並列処理することにより高い割合で並列化が行われていることがわかる。これは，Local Search の処理部分において CPU の数を増やすことにより計算時間が効率よく分散されていくからだと考えられる。PHGA (CSE-X) と PHGA (IER-X) が PHGA (Müh-X) より加速率が劣るのは，PHGA (CSE-X) と PHGA (IER-X) においてそれほど Local Search の処理部分で計算時間を費やしていないことが考えられる。しかし，図8では，PHGA(Müh-X) よりPHGA (IER-X) の方が計算時間を必要としているのは，PHGA (IER-X) は交叉の処理部分 (Edge Table の作成，子孫の生成) でかなりの計算時間を費やしていると考えられる。また，CSE-X と IER-X は交叉時にあまり親から離れていない子孫を生成し，Müh-X は，CSE-X と IER-X より比較的離れた子孫を生成しているために，Local Search で局所最適解を算出する処理に時間が必要なのである。表4に示した比較的大きな問題群にも同様なことが考察できる。

次に，これらの加速率を更に向上させる方法について考察する。本 PHGA では，最後

に処理が終了した CPU に合わせて同期を取っている。よって，CPU 数を増加しても加速率は上がらないと考えられる。交叉と Local Search の処理部分では，Local Search の方が処理を終える時間がまちまちなので，Local Search の処理部分において計算時間のかかる CPU を無視し，いくつかの CPU が処理を終了した時点で次の処理に移ることによって，ある程度のスピードアップが可能であると考えられる。

更に GA の並列化に関しては，さまざまな場合が考えられる。Local Search (LS) を複数回繰り返す方法として，Multiple Start Local Search (MSLS) がある。これは比較的簡単に実装でき(並列処理も可能)，その効果については極めて期待できる[13,15]。従って，今回取り上げた HGA を，Multiple Start HGA (MSHGA) として，並列処理することなどが考えられる。つまり，HGA 処理を各プロセッサに割り当て実行し，最良の結果を算出する方法である。

7. むすび

本論文は，巡回セールスマン問題 (TSP) に対する並列ハイブリッド遺伝的アルゴリズム (PHGA) について検討した。PHGA の実験では，逐次処理で算出された解の精度を劣化することなく並列化したことを確認し，並列化による計算時間の短縮が逐次処理に比べ，32個の CPU を使用した我々の並列化実装形態においては，Mühlenbeinらの交叉 (Müh-X) で約 1/13，改良された Edge Recombination Crossover (IER-X) で約 1/11，完全サブツアー交換交叉 (CSE-X) で約 1/10の時間で算出可能であることを示した。更に，基本的な枠組みを用いて，TSP に対して有望とされる Mühlenbein らの交叉 (Müh-X) と，Starkweather らの分析によって優れた交叉法とされる，改良された Edge Recombination Crossover (IER-X) そして最近提案された完全サブツアー交換交叉 (CSE-X) の並列化における効率性の相違を比較した結果そこでは，各交叉法のアプローチの相違によって，並列化に対して加速率や計算時間に大きな違いが表れることなどを示した。

謝　辞

超並列計算機 Paragon XP/S15の使用にあたり御配慮頂きました岡山理科大学情報処理センターの方々に感謝いたします。

参考文献

1) D. E. Goldberg : "Genetic algorithms in search Optimization and machine learning," Addison-Wesley Publishing Company Inc., 1989.
2) 北野宏明編：“遺伝的アルゴリズム，”産業図書，pp. 3 -60，1993.
3) 樋口哲也，北野宏明：“遺伝的アルゴリズムとその応用，”情処学論，Vol.**34**，No. **7**，pp.871-883，1993.
4) L. Chambers : "Practical handbook of genetic algorithms applications," CRC Press Inc., Vol. **1**,

pp.143-172, 1995.

5) M. O. Bell, T. L. Magnanti, G. L. Nemhauser and C. L. Monma : "Handbooks in operations research and management science network models, "North Holland, Vol. **7**, pp.225-330, 1995.

6) 山村雅幸, 小野貴久, 小林重信 : "形質の遺伝を重視した遺伝的アルゴリズムに基づく巡回セールスマン問題の解法," 人工知能誌, Vol. **7**, No. **6**, pp.1049-1059, 1992.

7) H. Mühlenbein, M. Gorges-Schleuter and O. Krämer : "Evolution algorithms in combinatorial optimization," Parallel Computing 7, North-Holland, pp.65-85, 1988.

8) H. Mühlenbein : "Parallel genetic algorithms in combinatorial optimization," Computer Science and Operations Research, Pergamon Press, pp.441-453, 1992.

9) S. Lin and B.W. Kernighan : "An effective heuristic algorithm for the traveling salesman problem," Operations Research, Vol.**21**, pp.498-516, 1973.

10) K.-T. Mak and A. J. Morton : "A modified Lin-Kernighan traveling-salesman heuristic," Operations Research Letters, Vol.**13**, pp.127-132, 1993.

11) G. Reinelt : "TSPLIB," ftp ://softlib. rice. edu/pub/tsplib/tsplib. tar.

12) D. S. Johnson : "Local optimization and the traveling salesman problem," Proc. 17th. Colloquium on Automata, Lang., and Prog., pp.446-461, Springer-Verlag, 1990.

13) D. S. Johnson, L.A. McGeoch : "The traveling salesman problem : a case study," Local Search in Combinatorial Optimization, Wiley-Interscience Series in Discrete Mathematics and Optimization, pp.215-310, 1997.

14) 小西健三, 屋鋪正史, 瀧　和男 : "温度並列シミュレーテッドアニーリング法の巡回セールスマン問題への適用と実験的解析," 信学論, Vol.**J80**-D-I, No. **2**, pp.127-136, 1997.

15) 久保幹雄 : "メタヒューリスティックス, "離散構造とアルゴリズムIV, 近代科学社, pp.171-230, 1995.

16) T. Starkweather, S. McDaniel, K. Mathias, D. Whitley and C. Whitley : "A comparison of genetic sequencing operators," Proc. 4 th ICGA, pp.69-76, 1991.

17) A. Y. C. Tang and K. S. Leung : "A modified edge recombination operator for the traveling salesman problem," Parallel Problem Solving for Nature, (H.-P. Schwefel and R. Männer, eds.), pp.180-188, 1994.

18) K. Katayama, H. Sakamoto and H. Narihisa : "An efficiency of hybrid mutation genetic algorithm for traveling salesman problem," Proc. 2 nd Australia-Japan Workshop on Stochastic Models, Gold Coast, Australia, pp.294-301, 1996.

19) 片山謙吾, 成久洋之 : "完全サブツアー交換交叉に対する共有サブツアーの高速列挙アルゴリズム," 信学技報, COMP 97-30, pp. 9 -16, July, 1997.

20) 片山謙吾, 平林永行, 成久洋之 : "TSPに対する並列ハイブリッド遺伝的アルゴリズムの一検討," 信学技報, COMP 97-31, pp.17-24, July, 1997.

A Study on Efficient Parallel Genetic Algorithms for the Traveling Salesman Problem

Hisayuki HIRABAYASHI, Kengo KATAYAMA and Hiroyuki NARIHISA*
Graduate School of Engineering
**Department of Information & Computer Engineering*
Faculty of Engineering
Okayama University of Science
Ridai-cho 1-1, Okayama 700-0005, Japan
(Received October 6, 1997)

In this paper, we investigate parallel efficiency of the computation time and the acceleration percentage for the crossover operator of genetic algorithm (GA). There have been proposed various crossovers operator for the traveling salesman problem (TSP). Among these crossover operators, following three crossover operators are considered to be most competent ones : a crossover of Mühlenbein et al. (Müh-X), an improved edge recombination crossover (IER-X), and a complete subtour exchange crossover (CSE-X).

We present the computation results of efficiency of these three crossover operators by using a parallel supercomputer, Paragon, Intel Corporation and compare them.

As a consequence of the comparison for the above mentioned promising crossover operators, our proposed CSE-X is concluded as most efficient one.