岡山理科大学紀要 第35号 A pp. 113-122(1999)

グラフ分割問題に対する遺伝的アルゴリズムの性能

河本　敬子・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

*岡山理科大学工学部情報工学科

(1999年11月4日　受理)

1　序論

組合せ最適化問題の多くは，分岐限定法のような厳密解法により最適解が求められる．しかしながら，大規模な問題に対しては組合せの数が飛躍的に増大するので，現実的な時間内に厳密な最適解を求めることは実用上不可能であることが知られている．この解決策として，ヒューリスティック法（heuristics method）などが提案されている．ヒューリスティックアルゴリズムは，NP困難問題のグラフ分割問題 (Graph Partition Problem:GPP) などに適用され，実用的な時間内に最適解に近い解（近似解）を算出可能である．たいていのヒューリスティックアルゴリズムは，伝統的なヒューリスティックスとメタヒューリスティックスの2つのタイプに分けることができる．前者の代表的なアルゴリズムとしては，Kernighan と Lin[8] によって提案された，Kernighan-Lin アルゴリズムである．これは，GPP に対する最も強力なローカルサーチアルゴリズムの一つとしてよく知られている．その他の強力なローカルサーチ (Local Search:LS) も [4] などで提案されている．このような，ヒューリスティックアルゴリズムでは短時間に良好な近似解を算出可能であるが，更に多くの計算時間を許容し，より優れた近似解が要求される場合がある．その場合に利用されるアルゴリズムとして，メタヒューリスティックスが多く提案されている．その代表的なアルゴリズムとして，*simulated annealing*, *tabu search*, *genetic algorithm* に基づく，強力な手法 [1, 3, 5, 10, 11] がある．

本論文では，GPP に対する遺伝的アルゴリズム (Genetic Algorithm:GA) について記述する．本論文の目的は，Johnson らのベンチマークグラフを使用し，MSLS(Multi Start Local Search)，シミュレーテッド・アニーリング法 (Simulated Annealing:SA) に対して，単純 LS を組合わせた GA の性能を比較することである．MSLS と SA は単純 LS と同じ近傍構造を含んでいるが，あらかじめ定められた計算時間内または評価回数において，GA がより良い性能を有することを示す．

2　グラフ分割問題

グラフ分割問題 (GPP) とは，幾つかのノードとノード間を結ぶ何本かのリンクで構成されたグラフが与えられた時，それを幾つかのグループに分割し，グループ間に跨るリンクが最小になるような分割を求める問題である（図 1）．本論文では，GPP を次のように定義する．n ノードからなる無向グラフ $G = (V, E)$ (V はノード，E はノード間の枝の集合である.）を与え，$V = \{v_1, v_2, ..., v_n\}$ である時，GPP は V_1 と V_2 の2つのサブセットにノードが均一になるようなグラフに分割する．

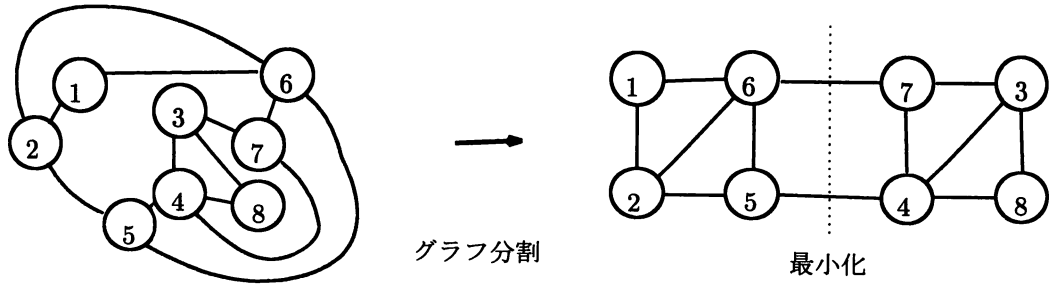

図 1　グラフ分割問題

3 遺伝的アルゴリズム と ローカルサーチ

遺伝的アルゴリズム (GA) とローカルサーチ (LS) の概要を述べる.

3.1 遺伝的アルゴリズム

遺伝的アルゴリズム (GA) は, 1970 年に, J. Holland によって提唱された手法である. 一般に GA は, 選択, 交叉, 突然変異のオペレータから構成される. 選択 (selection) オペレータは, 集団における適応度の分布に従って, 次世代に生存する個体群を確率的に決定する. 選択は, 集団の中から個体を取捨選択するだけで, 新しい個体を生成するものではない. 交叉 (crossover) オペレータは, 両親の染色体を部分的に組み替えることにより新しい子の個体を生成するもので, GA による探索の主推進力である. 探索空間を探索することで, 得られた個体群から個体のペアを選ぶことによって生成された子が, 親の有効な特質を受け継ぎ, 探索空間の新しい領域に移動することができるというメカニズムを用いている. 突然変異 (mutation) オペレータは, 通常, 第 2 の探索オペレータであると考えられている. このオペレータは, 染色体上のある遺伝子座の遺伝子を対立遺伝子に置き換えることにより, 交叉だけでは生成できない子を生成して個体群の多様性を維持する働きをする. GA は, 選択により望ましい解を重点的に得ると同時に, 交叉と突然変異によって解の探索範囲を広げているので, それらの両方のオペレータが有効に作用すれば, その威力を十分に発揮することになる. GA がこれまでの古典的な探索法と大きく異なる点は, [12] によれば, 次の 4 つの特徴にあるといえる.

(1) 問題をコーディングしたものを直接利用する.
(2) 1 点探索ではなくて, 多点探索である.
(3) サンプリングによる探索で, ブラインドサーチ (blind search) である.
(4) 決定論的規則ではなく, 確率的オペレータを用いる探索である.

このような 4 つの特徴により, GA がこれまでの古典的手法にない特徴を備えた新たな探索手法になりうることが期待されている. このような GA を取り入れることで, Holland は再組合わせと遺伝子の突然変異によって, 要素の個体の適応度を最適化されるまで探索するというアルゴリズムを構成するという, 遺伝的進化論の概念を用いた. しかしながら, それらの概念に従う GA は, 組合せ最適化問題のための良質な解を見つけることができないと考えられている. それは, GA が局所探索についてのメカニズムをもっていないからである. この欠点を改善するために, GA とローカルサーチを組合せたヒューリスティックアルゴリズムは, 短い計算時間内に局所探索で良い解に到達させることに役立つ. この組合せアプローチは, 一般に "*genetic local search*" (GLS) アルゴリズムと呼ばれる.

3.2 ローカルサーチとその近傍

組合せ最適化問題におけるローカルサーチは, 局所最適解まで, 与えられた解をより良い近傍まで動かすことに基づいている. ローカルサーチの方法を以下に述べる.

1 つの解 x, 近傍解 $x' \in N(x)$ を与え, 可能な "近傍"解を, 規則正しく又はランダムに生成させる. もし, x'が $f(x') < f(x)$ のとき, 新しい現在の解 x へ移動する. 再び, 近傍解 $x' \in N(x)$ は, 近傍 N で見つけられるが, 解を減少させる可能性がなくなるまで, 生成され評価される. 従って, 最終解 x は定義した近傍 N において, "局所最適解"に到達することができる.

GPP のための我々のローカルサーチアルゴリズムは, 分割 (解) x の近傍 x'を, set0 と set1 の 1 つのノードを交換することによって定義される. ローカルサーチによって得られた最終解は, この近傍で実行可能解である局所最適解でなければならない. [8] で提案されている Kernighan-Lin アルゴリズムは, k ($k \geq 2$) ノードで実行され, 単純近傍構造によって実行する我々のローカルサーチアルゴリズムよりも良い解を得ることができる.

遺伝的アルゴリズム

0. 個体群サイズ PS と世代数 GN を定義する．初期世代 $G=0$ とする．
1. ランダムに初期世代の個体群 P^0 である個体 $I_1^0,...,I_{PS}^0$ を生成する．
2. それぞれの個体 $I^0 \in P^0$ は，新しい P^0 を得るために既存の LS によって局所解にする．
3. G が GN に到達するまで以下を繰り返す．
   - 3.1 $G=G+1$ ととする．
   - 3.2 すべての子が $PS/2$ に到達するまで以下を繰り返す．
     - 3.2.1 新しい I_c を得て，突然変異 (I_c) を実行する．
     - 3.2.2 交叉を行う個体 $I_a, I_b \in P^G$ は，ランダムに選ぶ．
     - 3.2.3 I_c を得るまで，交叉 (I_a, I_b) を実行する．
     - 3.2.4 I_cに LS を実行し，新しい I_cを得る．
     - 3.2.5 I_c は，子の個体 P_c に加える．
   - 3.3 (すべての個体 $\in P^G$ と P_c^G のそれぞれの cut size は，計算される．) 選択したより良い個体 PS は，現在の P^G と P_c^G から次の新個体群 P^{G+1} となる．
4. 最良の個体 $\in P$ を返す．

図 2 GA の流れ

4 実験に基づく詳細

4.1 GA の手法

組合せ最適化問題を解くためのいくつかのアルゴリズムは，実行不可能解の探索空間を探索することがある．この場合，実行可能解に修正するためにペナルティ関数などが必要となる．我々の GA によって実行された個体は探索中，常に実行可能解が保証され，適応度は，cut size によって計算される．我々の単純 GA の手法を図 1 に示す．

すべての初期個体は，Step 1 でランダムに生成される．個体は，それぞれのノード位置で set 0 と set 1 の分割を区別するために 0 と 1 の値を持つ．実行可能解を保持する目的で，set 0 と set 1 のそれぞれの数は，$n/2$ に等しい．Step 2 で，それらの個体は，3 節で述べられたローカルサーチヒューリスティックによって局所的に改善される．従って，交叉操作後以外の個体はすべて局所最適解である．Step 3 で実行される GA のメインループは，あらかじめ設定した世代数 GN まで実行される．メインループの処理は，突然変異，交叉，選択オペレータから成る．それらのオペレータの詳細は次で述べる．

4.1.1 突然変異オペレータ

個体群の多様化は，遺伝的探索のために重要である．選択オペレータは集団の多様性を決定するために実行されるが，これを与える他の一因は突然変異オペレータであるかもしれない．しかしながら，ローカルサーチを組合せた GA は，一般に 10 から 40 までの個体 [11] のサイズを含んでいる．この場合，そのような小さい個体サイズは “初期収束”に導かれやすい．従って，この欠点を避けるために突然変異オペレータによって実行された個体は，交叉オペレータでは探索できない探索空間の新しい領域に移動することができる．本研究における突然変異は，突然変異確率によって，set 0 からランダムに選んだノードと set 1 からランダムに選んだノードを入れ替えることを意味する．

4.1.2 交叉オペレータ

交叉操作を適用する前に2つの親を選ぶカップリング方法を，図2のStep3.2.2に示す．

GPPにおける1点，または多点交叉のような交叉オペレータは，実行可能解ではない子孫を作る可能性がある．これらの交叉方法は，オペレータが単純であるということが知られているが，最終解の修正なしに，実行可能解は保証されない．我々のGAで，Step3.2.3の交叉操作は，親1，親2の遺伝子座の遺伝子の共通情報を子に対して保存するというものである．親1，親2の各遺伝子座が0の場合，子に0をコピーし，1の場合は子に1をコピーする．実行可能解を維持する目的のために，残りの子の遺伝子座に，遺伝子0，1の数が$n/2$となるように，ランダムに0，1を割り当てる[11]．この後，Step3.2.4の交叉によって，生成した子孫は，2節で述べられた単純ローカルサーチアルゴリズムによって，最適化される．従って，2つの親から1つの子を生成することから子孫の数は$PS/2$個となる．

4.1.3 選択オペレータ

親または生成した子孫に対する選択オペレータは，GAでの進化に基づく探索において重要である．選択方法によって，より良い個体は次世代に生き残る．本研究での選択方法を，図2のStep3.3に示す．この方法は，[7]で述べられたような比較的高い選択圧力をもっている．その方法の戦略を以下に述べる．

子孫と親の良いまたは最良の個体は，遺伝的探索の間，次の親のために個体群の中で常に保持される．もし，それらが親と子孫の個体間で同じ適応度（cut size）が存在するのならば，親は淘汰され，子孫が新しい個体となる[7]．

4.2 比較されるアルゴリズム

本実験では，MSLS，SAに対して上で述べたGAを比較する．以下，MSLS，SAについて記述する．

4.2.1 マルチ・スタート・ローカルサーチアルゴリズム

MSLSは，ランダムに作られた初期解から，複数回のLSの実行をする．このアルゴリズムは単純であり，比較的良い解を得ることができると知られている．その操作は，GA，SAよりも長い実行時間または，設定したローカルサーチ(LS)の実行回数まで繰り返される．本実験での終了判定は，LSによって実行される回数であり，我々はこれを5000回に設定する．この回数は，GAの最終世代GNにおいて，LSを実行した回数よりも多い．

4.2.2 シミュレーッテッド・アニーリング法

SAは，LS法と標準メトロポリス[9]からなる．SAは，1つの解でのみ実行され，LSの近傍探索によって，得られた現在の解のその近傍が，現在の温度による基準に応じて受理される．もし，解の目的関数が改善している状態（減少）に向かっているならば，状態遷移を行ない，そうでなければ，受容確率$e^{-\delta/T}$（評価値の改善値の改悪量$\delta = f(x') - f(x)$）で状態遷移を行なうかを決める．ここで，Tは温度と呼ばれるパラメータである．この遷移ルールでは温度Tが高ければエネルギーを増加させる状態遷移も高い確率で受理され，温度が低ければこのような状態遷移は行なわれにくくなる．上記の操作で，SAは温度を徐々に低下させながら繰り返すこと（アニーリング）により，探索の初期ではエネルギーの多少の壁を乗り越えて大域的な探索を，終盤では現在の状態の近傍での局所的な探索を行ない（近似）最適解を得ようとするものである．従って，たとえ局所解に落ちても，この受容確率のために脱出することが可能である．SAの流れを，図3に示す．初期温度Tと最終温度FTのパラメータ，低減率αの0.98は，本研究での実験要領に適した設定を行った．

Simulated Annealing Algorithm

0 初期温度 T, 最終温度 FT, 低減率 $0 < \alpha < 1$ を定義する.

1 ランダムに初期解 x を生成する.

2 T が FT に到達するまで，次の操作を行なう.

2.1 現在の T が到達するまで次を繰り返す.

2.1.1 ランダム近傍 $x' \in N(x)$ を選ぶ.

2.1.2 $\delta = f(x') - f(x)$ を設定する.

2.1.3 もし，$\delta \leq 0$ ならば,$x = x'$とする.

2.1.4 そうでなければ，受容確率 $e^{-\delta/T}$ で，$x = x'$ とする.

2.2 温度 $T = \alpha \times T$ で，温度を下げる.

3 探索中に最良解が見つかれば戻る.

図 3 SA アルゴリズムの流れ

5 実験結果

本実験では，多くの研究者によって用いられている Johnson ら [5] の 8 個の問題例，ノード数 124，250 に対して GA，MSLS，SA のアルゴリズムの性能評価および挙動分析を行い，GA の有効性を検討する．$Gn.p$ と呼ばれる問題は，ノード n のランダムグラフである．任意の 2 点間に確率 p で枝を与えることによって無向グラフを構成する．一つの点から出ている枝の平均本数（平均次数）は，約 $p(n-1)$ である．8 個の問題例の最良解を，表 1 に示す．全ての実行は，COMPAC DESKPRO（CPU : 450Hz，Memory : 196MB）上で行い，プログラム言語は C である．試行回数は，各問題例に対して，5 回毎行なう.

ここで，本実験での GA のパラメータについて記述する.

個体群サイズ PS として，世代数 GN まで計算を行う．本研究の交叉確率 P_c と突然変異確率 P_m を以下に記述する．LS を組合わせた GA では、多くの研究者は，一般に個体サイズ PS が 10 から 40 の範囲 [7, 11] で設定している．本研究では，2 つの個体群サイズ PS=40，20 で計算する．GN は，両方の PS の場合について，200 世代とする．P_c はすべて 1.0 として，突然変異確率は PS=20 に対して，P_m=0.0005，PS=40 に対して P_m=0.0001 に設定する.

表 1 MSLS と SA アルゴリズムの実験結果と各グラフの Best-known cut-size

	best-known	MSLS		Time(s)	SA		Time(s)	Time(s)
instance	cut-size	best	avg.	avg.	best	avg.	best	avg.
G124.02	13	13	13.6	29.0	13	13.2	5.0	14.8
G124.04	63	63	64.0	58.0	63	64.6	4.0	9.8
G124.08	178	178	178.4	61.6	178	178.8	3.0	10.2
G124.16	449	449	450.6	67.8	449	451.4	4.0	19.2
G250.01	29	35	38.8	138.4	29	30.4	6.0	58.0
G250.02	114	119	121.4	278.4	114	115.2	22.0	115.4
G250.04	357	360	362.0	611.6	357	361.0	12.0	191.0
G250.08	828	830	833.6	678.2	828	828.0	48.6	156.8

表 2 個体数 40 での LS を組合わせた GA の実験結果

G124.02	GA		Time(s)
gens	best	avg.	avg.
1	19	20.4	1.0
5	15	16.8	1.8
10	14	15.2	2.4
20	13	14.0	3.8
40	13	13.0	6.6
100	13	13.0	14.0
200	13	13.0	25.8

G124.04	GA		Time(s)
gens	best	avg.	avg.
1	64	68.4	1.2
5	63	64.2	2.8
10	63	64.0	4.0
20	63	63.2	6.8
40	63	63.0	11.6
100	63	63.0	24.0
200	63	63.0	44.8

G124.08	GA		Time(s)
gens	best	avg.	avg.
1	179	181.2	1.4
5	178	179.2	4.0
10	178	179.0	6.8
20	178	178.0	11.8
40	178	178.0	21.2
100	178	178.0	44.4
200	178	178.0	81.0

G124.16	GA		Time(s)
gens	best	avg.	avg.
1	453	457.4	1.4
5	449	449.0	7.6
10	449	449.0	13.2
20	449	449.0	23.8
40	449	449.0	42.8
100	449	449.0	88.6
200	449	449.0	160.8

G250.01	GA		Time(s)
gens	best	avg.	avg.
1	48	49.6	2.0
5	39	42.2	5.0
10	39	40.8	7.8
20	33	36.0	13.4
40	31	31.4	24.8
100	29	30.0	55.4
200	29	30.0	105.6

G250.02	GA		Time(s)
gens	best	avg.	avg.
1	121	129.2	2.8
5	117	122.4	8.0
10	117	119.8	13.4
20	114	116.6	23.8
40	114	114.2	42.8
100	114	114.2	92.6
200	114	114.2	172.8

G250.04	GA		Time(s)
gens	best	avg.	avg.
1	367	372.8	5.4
5	361	364.8	17.6
10	357	359.8	28.8
20	357	357.4	50.4
40	357	357.0	90.0
100	357	357.0	191.8
200	357	357.0	350.0

G250.08	GA		Time(s)
gens	best	avg.	avg.
1	842	845.6	3.4
5	832	833.8	27.6
10	828	829.3	51.0
20	828	828.0	95.4
40	828	828.0	173.0
100	828	828.0	373.6
200	828	828.0	675.2

表 3 個体数 20 での LS を組合わせた GA の実験結果

G124.02	GA		Time(s)
gens	best	avg.	avg.
1	19	20.8	0.8
5	16	16.8	1.0
10	15	16.0	1.6
20	13	13.8	2.0
40	13	13.8	3.6
100	13	13.0	7.4
200	13	13.0	13.6

G124.04	GA		Time(s)
gens	best	avg.	avg.
1	64	70.8	0.8
5	64	65.8	1.8
10	64	65.0	2.6
20	63	63.4	3.8
40	63	63.2	5.8
100	63	63.2	12.6
200	63	63.0	22.8

G124.08	GA		Time(s)
gens	best	avg.	avg.
1	179	180.8	1.0
5	179	179.8	2.2
10	178	179.0	4.0
20	178	178.4	6.4
40	178	178.2	11.2
100	178	178.0	24.6
200	178	178.0	46.8

G124.16	GA		Time(s)
gens	best	avg.	avg.
1	467	473.4	1.0
5	449	449.0	4.0
10	449	449.0	6.8
20	449	449.0	11.8
40	449	449.0	20.4
100	449	449.0	43.4
200	449	449.0	82.0

G250.01	GA		Time(s)
gens	best	avg.	avg.
1	48	50.2	1.0
5	38	42.8	2.8
10	37	38.4	4.0
20	33	32.2	6.8
40	31	33.4	11.8
100	30	31.2	27.0
200	29	30.2	52.4

G250.02	GA		Time(s)
gens	best	avg.	avg.
1	124	130.8	1.8
5	120	124.2	3.6
10	120	121.0	7.0
20	116	117.2	12.0
40	115	115.4	20.8
100	114	114.4	47.6
200	114	114.4	91.6

G250.04	GA		Time(s)
gens	best	avg.	avg.
1	375	384.8	2.6
5	369	370.6	8.6
10	360	363.6	14.6
20	359	360.2	24.2
40	359	358.8	43.0
100	359	358.6	94.8
200	357	358.0	180.6

G250.08	GA		Time(s)
gens	best	avg.	avg.
1	849	850.0	3.4
5	831	835.2	15.2
10	828	830.0	27.4
20	828	828.8	47.0
40	828	828.0	81.6
100	828	828.0	183.8
200	828	828.0	350.0

表 1 は，MSLS と SA アルゴリズムの実験結果を示す．表中の bestknown は，Johnson らが，Simulated Annealing，Kernighan と Lin の近似解法，およびその改良法を用いて得た最良解の値を示している．本研究での MSLS の 1 回の実行は，ランダムに生成した実行可能解からローカルサーチを用いて，5000 回繰り返す．従って，5 回の試行結果である表 1 は，MSLS が，左から最良値，解の平均値，各問題例で要した平均実行時間（秒）を示す．また，SA は，左から最良値，解の平均値，最良解が得られた時間，各問題に要した平均実行時間を示している．Best の時間は，各問題例で SA の実行中に最良解が得られたときの実行時間を表す．

表 2，表 3 に，個体数 20，40 を用いた単純ローカルサーチを組合わせた GA の実験結果を示す．ここで，表中の gens は世代数を示す．gens 1 は，初期ランダム個体群から 40 回のローカルサーチ実行後の値を示す．また，gens 5, 10, 20, 40, 100, 200 の結果は，120, 220, 820, 2020, 4020 回のローカルサーチ実行後，交叉によって生成された解を表す．

ここで，MSLS と GA の実行結果，実行時間について GA と SA の実行結果を比較する．もし，SA の初期温度の設定が，本実験よりも長い実行時間が可能であるならば，SA は表 1 で示した結果よりも良い平均 cut size を得られると考えられる．表 1 での SA の結果は，試行した全ての問題について，最良解が比較的短い実行時間で少なくとも 1 つは得られることから，良い結果を得ることができたといえる．また，個体 40 と 20 を用いた GA の平均結果は，SA の平均結果より良い．SA で必要とした最終実行時間の比較において，我々の GA は，SA 以上の平均 cut size を算出することができた．たとえば，表 2 と表 3 で示した g250.08 の問題例の平均結果は，GA が個体数 20，40 で，それぞれ，828.0，SA もまた 156s 以内の平均で 828.0 の最良解を得ることができた．しかしながら，これらの GA の平均実行時間は，それぞれ 95.4(s) と 81.6(s) である．GA の実行時間は，表 1 で示された G250.08 の問題例の SA よりも，平均実行時間が明らかに速い．これらの結果から，GA の性能はほとんどの場合，SA と MSLS のアプローチ以上であるということを観測された．

6 結論

本論文は，MSLS と SA アルゴリズムのようなヒューリスティックスに対して，単純ローカルサーチを組合わせた遺伝的アルゴリズムの性能を示した．GA がほとんどの場合に SA，MSLS よりも良い探索能力をもっていることを述べた．もし，本論文で記述した我々のローカルサーチよりも，Kernighan-Lin アルゴリズムのような，より強力なローカルサーチと組合せた GA ならば，さらに良い性能を示すことができたと思われる．

今後の研究では，Merz [11] らによって行われたような GPP の解空間を分析し，より大きいまたは他のタイプの問題例を用いることで，我々の GA の性能を示す予定である．

参考文献

[1] Battiti R. and Bertossi A. (1999), Greedy, prohibition, and reactive heuristics for graph partitioning, *IEEE Trans. on Computers.*

[2] Brunetta L., Conforti M., and Rinaldi G. (1997), A branch and cut algorithm for the equicut problem, *Mathematical Programming*, **78**, 243–263.

[3] Bui T.N. and Moon B.R. (1996), Genetic algorithm and graph partitioning, *IEEE Trans. on Computers*, **45(7)**, 841–855.

[4] Fiduccia, C.M. and Mattheyses, R.M. (1982), A linear-time heuristic for improving network partitions, in *Proc. the 19th ACM/IEEE Design Automation Conference*, 175–181.

[5] Johnson D.S., Aragon C.R., McGeoch L.A., and Schevon C. (1989), Optimization by simulated annealing: An experimental evaluation; Part I, graph partitioning, *Operations Research*, **38**, 865–892.

[6] Goldberg D.E. (1989), *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley.

[7] Katayama K. and Narihisa H. (1999), Performance of genetic approach using only two individuals, in *Proc. 1999 IEEE Systems, Man, and Cybernetics Conference.*

[8] Kernighan B.W. and Lin S. (1970), An efficient heuristic procedure for partitioning graphs, *Bell Systems Technical Journal*, **49**, 291–307.

[9] Kirkpatrik S., Gelatt C.D., and Vecchi M.P. (1983) Optimization by simulated annealing, *Science*, **220**, 671–680.

[10] Martin O.C. and Otto S.W. (1995), Partitioning of unstructured meshes for load balancing, *Concurrency: Practice and Experience*, **7(4)**, 303–314.

[11] Merz P. and Freisleben B. (1998), Fitness landscapes, memetic algorithms and greedy operators for graph bi-partitioning, *University of Siegen, Technical Report*, No. TR-98-01.

[12] De Jong K.A. (1975), An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Doctoral dissertation, *University of Michigan* .

Performance of a Genetic Algorithm for the Graph Partitioning Problem

Keiko KOHMOTO, Kengo KATAYAMA* & Hiroyuki NARIHISA*

Graduate School of Engineering
**Department of Information & Computer Engineering*
Faculty of Engineering
Okayama University of Science
Ridaicho 1-1, Okayama 700-0005, Japan

(Received November 4, 1999)

A genetic algorithm (GA) for the graph partitioning problem (GPP) is investigated by comparison with standard heuristics on well-known benchmark graphs of Johnson et al. In general, the practical performance of a conventional genetic approach, which performs only simple operators without a local search strategy, is not sufficient. However, it is known that a combination of a GA and a local search can produce good solutions. In this paper, we incorporate a simple local search algorithm into a GA, and compare the search abilities of the GA for solving the GPP with standard heuristics such as a multi-start local search and a simulated annealing, which executes with the same neighborhood structure of the simple local search. In our investigations, we demonstrate that the GA is better than the other competitors in times of computation times and evaluation limits.