岡山理科大学紀要 第36号 A pp. 165-174(2000)

グラフ２分割問題における貪欲的交叉法の研究

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

(2000年11月１日　受理)

1　序論

組合せ最適化問題の多くは，NP-困難な問題に属し，組合せで生じる解の数は問題の大規模化に伴い爆発的に増大する．従って，現実的な時間内に最適解を求めることは実用上不可能である．この解決策として，従来から山登り法のようなヒューリスティック解法があり，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．しかしながら，更に上質な近似解を要求される場合が多く，この種のヒューリスティック解法と自然界に存在するアイディアを組合わせたメタ解法が有効とされている．特に，遺伝的アルゴリズム (genetic algorithm, GA) [5] と局所探索法 (local search, LS) を融合した，遺伝的局所探索法 (genetic local search, GLS) [9] は，様々な組合せ最適化問題に対して適用され，優れた近似解を算出可能であることが知られている．GA における本質的操作は交叉にあり，その操作は二つの個体間での染色体を組み換えることによって新しい個体（子）を生成するもので，両親の優れた部分形質 [12] をうまく組合わせて子に継承させることに成功すれば，探索における飛躍をもたらす．そこで，両親の優れた部分形質を子に対して継承させた交叉法に貪欲的な方法を用いれば，さらに効率的な探索をすることができると考えられる．

本研究は，組合せ最適化問題の一つであるグラフ２分割問題 (graph bi-partitioning problem, GBP) に対して GLS の枠組みを利用し，貪欲的操作を含まない古典的な交叉法と提案する貪欲的な操作を含む交叉法の比較を行う．本研究での古典的な交叉法は，多点交叉の変形である一様交叉 (uniform crossover, UX) を用いる．これは，多点交叉が GBP に対して一点交叉，二点交叉よりもよい性能を示す [3] と報告されていることによる．本 GLS では，古典的な交叉法として UX，この UX に貪欲法 [1] を含む交叉法として greedy crossover1 (GX1), greedy crossover2 (GX2) を用いる．この両 GX 法は，GBP を解く上で重要となる点と点との枝のつながりに着目した手法であり，子の生成過程において良い解の近くには更に良い解が存在する可能性がある [7, 2] ことに基づいて実現される．GX, UX をそれぞれ有した GLS の比較から貪欲的な操作を取り入れた GX の効果を検討する．

2　グラフ２分割問題 (GBP)

グラフ２分割問題 (GBP) [6, 11] は NP-困難であり，VLSI 回路設計問題やネットワーク分割問題等に応用し得る工学上重要なものとされている．GBP は，グラフ $G=(V,E)$（ただし，V は点の集合，E は点と点とを結ぶ枝の集合，$n=\mid V \mid$ は偶数とする）が与えられたとき，V の要素を２個の互いに素な部分集合に分割し，各部分集合に含まれる要素の数が均一になるように分割する問題である．さらに，この問題は異なる部分集合間に跨るリンク数（これを cut size という）を最小にするように点を分割するものである（図 1）．

本研究の GBP を次のように定義する．無向グラフ $G=(V,E)$ が与えられたとき，$V=V_0 \cup V_1, V_0 \cap V_1 = \emptyset, V_0, V_1 \subseteq V$ を満たし，$C=\{e_{ij} \in E(v_i \in V_0, v_j \in V_1)\}=\{e_{ij} \mid e_{ij} \in E, v_i \in V_0, v_j \in V_1\}, \mid V_0 \mid = \mid V_1 \mid = n/2$ となる条件下で，$\sum \mid C \mid$ が最小となるような，$\{v_i\}=V_0, \{v_j\}=V_1$ を求める．

3　貪欲的な遺伝的交叉法について

本章では，UX と我々の提案する２つの貪欲的な交叉操作 (GX1, GX2) について説明する．まず，GBP での解を表現するコーディングについて記述する．このコーディング方法による個体は，0,1 のビット列で表される．例えば，点 i が set0 に割り当てられるとき，第 i 遺伝子座の遺伝子は 0 となり，set1 に割り当て

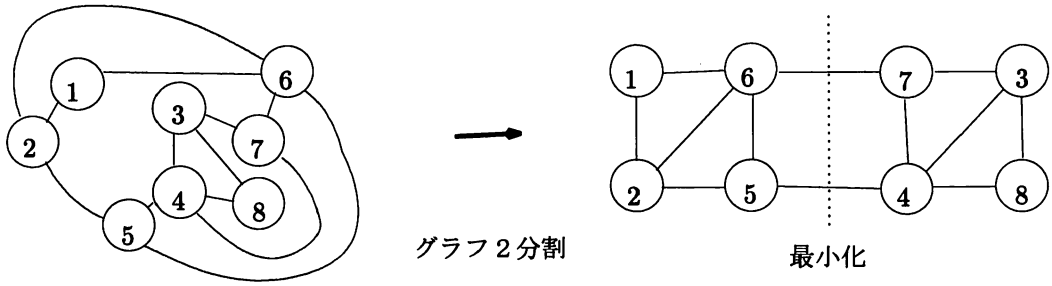

図１ グラフ２分割問題

られるとき，第i遺伝子座の遺伝子は１となる．例えば，図２の分割状態の場合，set0に点1,2,5,6，set1に点3,4,7,8が分割されているので，個体の遺伝子座1,2,5,6の遺伝子は0，遺伝子座3,4,7,8の遺伝子は１となる．また，本GBPでの実行可能解は，それぞれのサブセットに分割された点の数が等しくなければならないという定義に基づいて生成される．よって，各個体の遺伝子0,1の各総数は，'$n/2$'にならなければならない．

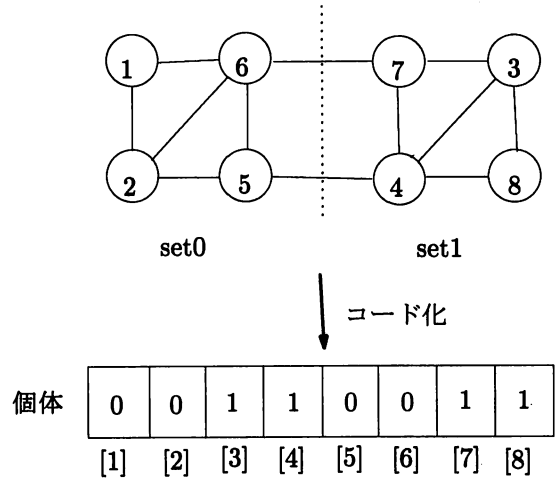

図２ コーディング

GAにおいて，最も重要な役割を果たす操作は交叉であり，選択された個体間での染色体の組換えにより新しい個体を生成するというものである．このような交叉は，個体群の中から任意の２つの個体（親）を選び，さらに，選ばれた１点あるいは多点の交叉点で遺伝子を組み換えることにより，両親の形質を受継ぐ新たな個体（子）を生成する操作である．

本論文では，両親の形質をできるだけ子に継承することと，対象とする問題に対して実行可能性を保持するように子の生成を行うことを'交叉操作の基本的方針'とする．この基本的方針に基づいて，貪欲的な操作を含まない一様交叉(UX)とUXに貪欲的な操作を含ませた交叉(GX1, GX2)について検討する．UXの他にも古典的な交叉法として，１点交叉，２点交叉などが良く知られているが，これらの交叉法よりもUXは，各遺伝子座の情報に着目しているので，GBPに対しては各点の割り当て状況をより適切に子へ継承し得ると考えられる．実行可能性の観点から，UXをそのままGBPに適用できない．なぜなら，GBPの定義である'各部分集合に分割される点の数を均一とする'という解が実行可能解であるため，生成される子は，GBPでの実行可能性を保証することができない．従って，本研究でのUXは，GBPに対して子の実行可能性を保証するように簡単な操作を加えている．GXは，UXに基づいて貪欲的な操作を考慮しつつ子を生成するというものであり，生成される子に対して常に実行可能性を保証する．以下，UXと両GXにつ

いて記述する.

3.1 Uniform Crossover (UX)

UX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し，共通情報をもたない遺伝子座に対しては，実行可能性を考慮しながらランダムに遺伝子0,1を割り当てるというものである（図3）．これは良い点の情報は個体間の共通情報として存在しているとの考えからである．ここで，実行可能性の保持とは，1個体の遺伝子0,1の各総数が'$n/2$'となるように各遺伝子座に対して遺伝子を割り当てていくことを意味する．図4に，UXの処理を記述する.

まず，Step0では，親1，親2の遺伝子座の遺伝子の共通情報を子の遺伝子座に対して割り当てる．このとき，子に割り当てた各共通遺伝子の0,1の各総数 $sub0$, $sub1$ を求める．本研究では，実行可能解である両親を用いているため，この総数は等しい．Step2では，遺伝子が割り当てられていない子の遺伝子座に対して，実行可能解となるように遺伝子0,1をランダムに割り当てる．これは，遺伝子0を割り当てる遺伝子座を $sub0$ 箇所ランダムに選び，残りの遺伝子座に対して遺伝子1を割り当てることを意味する．この操作によって，実行可能性を保証した子を生成することができる.

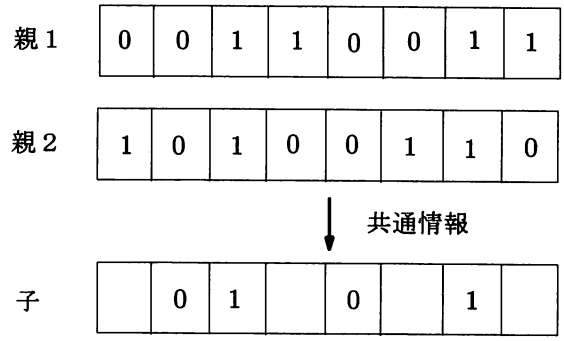

図3 UXの操作

UX アルゴリズム

Step0. 親1，親2の遺伝子座の遺伝子の共通情報を子の遺伝子座に対して割り当てる.

Step1. 共通情報を持たない子の遺伝子座に対して，実行可能解となるように遺伝子0,1をランダムに割り当てる.

図4 UXの流れ

3.2 Greedy Crossover 1 (GX1)

本研究の貪欲的な交叉法 (greedy cossover, GX) はUXを拡張した交叉法で，UXでは両親の各遺伝子座に含まれる0と1の情報が共通ではない場合，ランダムに0または1をランダムに割り当てるが，GXではランダムではなく貪欲的に（cut sizeを小さくするような）遺伝子を割り当てるというものである [10]．このように，両親の共通情報を受継がない部分に貪欲的に遺伝子を割り当てることで，より良いcut sizeを持つ子を得るにはどの点をどちらの部分集合に割り当てれば良いかを考慮した交叉法である．図5に，GX1の流れを示す.

Step1は，両親の間の2つの共通情報は子に継承されるという，UXの最初のプロセスに相当する．したがって，もし両親が共通情報をもつ場合，両親の遺伝子座に対応する子の遺伝子座に対して，両親の遺伝子座の遺伝子が0のとき，0が入れられる．また，1の場合は，1が入れられる．そして，長さ $n/2$ の配列 $Set0(i)$ と $Set1(i)$ には，各部分集合に割り当てられる点が入れられる．これらの配列は，点がどちらの部分集合に割り当てられるかという情報をもつ．もし，両親間の共通情報をもつ第 i 遺伝子座の遺伝子が0

ならば，点 i は，$Set0(i)$ に入れられる．Step2 では，両親の共通情報をもたない点が，Step4 の候補として長さ n の配列 $Keep(j)$ に保持される．$SetA$ と $SetB$ は，どちらかのサブセットを表す変数とする．例えば，$SetA$ が 1 の場合はサブセット 1 を意味し，0 の場合はサブセット 0 を意味する．Step3 は，$SetA$ に対して 0 または 1 をランダムに選ぶ．これは，Step4 のループで最初に割り当てられる点をどちらのサブセットに割り当てるかを決定する．Step4 のループは，$Keep(j)$ の全ての点の候補がなくなるまで繰り返す．Step4.1, 4.2 は，Step4 の繰り返し処理のための $SetA$，$SetB$ の設定操作である．例えば，$SetA$ が 0 とするとき，$SetB$ は 1 となる．長さ n の一時保存配列 $Allocate(k)$ は，$SetA$ に割り当てるためにいくつかの点の候補をもつ．Step4.3 は，$Keep(j)$ のある点とすでに $SetB$ に割り当てた各点との 2 点間の枝の数が最小である $minedges$ を見つける．Step4.4 は，$Allocate(j)$ のある点とすでに $SetA$ に割り当てた各点との 2 点間の枝の数が最大である $maxedges$ を見つける．Step4.5 では，$Allocate(j)$ からランダムに選び，$SetA$ に点を割り当てる．Step4.6 は，$Keep(j)$ から Step4.5 で選ばれた点を消去する．終わりに，Step5 では $Set0(i)$ と $Set1(i)$ に割り当てられた点を長さ n の解に置き換える．

GX1 アルゴリズム

Step1. 両親の共通情報をもつ遺伝子の遺伝子座に対応する点を $Set0(i)$, $Set1(i)$ に対して，割り当てる．(両親の共通情報をもつ遺伝子座の遺伝子は，子の遺伝子座に対してコピーする.)

Step2. 共通情報をもたない点を $Keep(j)$ に保持する．

Step3. ランダムに 0 または 1 を選ぶ．もし，1 ならば $SetA$ は 1 に設定され，0 の場合は 0 に設定される．($Set0(i)$ と $Set1(i)$ のそれぞれの総数は，互いに等しい.)

Step4. $Keep(j)$ が空になるまで次を繰り返す．

Step4.1 $SetA \longleftarrow (1 - SetA)$

Step4.2 $SetB \longleftarrow (1 - SetA)$

Step4.3 $Allocate(k)$ は，$Keep(j)$ と $SetB$ に割り当てられている点*との間の枝数が最小である $minedges$ をもつ点を入れる．(*最初のループではランダムに，2 回目以降は最後に割り当てられた点を 1 つ選ぶ.)

Step4.4 新しい $Allocate(k)$ に，$Allocate(j)$ と $SetA$ に割り当てられている点との間の枝数が最大である $maxedges$ をもつ点を入れる．

Step4.5 $Allocate(j)$ からランダムに点を選び，$SetA$ に割り当てる．

Step4.6 $Keep(j)$ から $SetA$ に割り当てられた点を消去する．

Step5. $Set0(i)$ と $Set1(i)$ から割り当てられた点を長さ n の解に置き換える．

図 5 GX1 の流れ

この交叉法でのポイントは，それぞれのサブセット間での枝のつながりが最小または最大になるという点を繰り返しみつけることである．従って，GX1 は両親の共通情報をもたない部分に対して，遺伝子を貪欲的に割り当てることによって良い cut size をもつ子を得ることができる．

3.3 Greedy Crossover 2 (GX2)

GX2 は，GX1 よりも貪欲的効果をさらに強めた交叉法である．GX2 は，図 5 で示した GX1 の Step4.4 を図 6 のように書き換えたものである．$SetA$ に割り当てられた点を長さ n の $SetA\_vertex(i)$ に入れる．そして，$SetA\_vertex(i)$ の各点に対して Step4.4.1 を実行する．$Allocate2(k)$ は，$SetA$ に割り当てるためにいくつかの候補の点をもつ長さ n の一時保存配列とする．この処理は，生成される子の cut size を減少させることができる 2 点間の枝のつながりを持つ点を繰り返し探すことによって貪欲的な効果を向上させている．

GX2 アルゴリズム

Step4.4 $SetA$ に割り当てられている点を $SetA\_vertex(i)$ として，$SetA\_vertex(i)$ が $SetA\_vertex(0)$ になるまで以下を繰り返す．

Step4.4.1 $Allocate(j)$ の各点と $SetA\_vertex(i)$ の点との間の枝の数が最大である $maxedges$ をもつ点を $Allocate2(l)$ に割り当てる．(もし，$SetA(0)$ の場合，$Allocate2(l)$ の点は $Allocate(j)$ にコピーする．)

図 6 GX2 の流れ (GX1 の書き換え部分)

4 遺伝的局所探索法 (GLS)

遺伝的アルゴリズム (GA) と局所探索法 (LS) を組合せた遺伝的局所探索法 (GLS) について記述する．LS は，反復改善を行う探索手法である．その手法は，現在の解の一部を変化させることにより生成される近傍に含まれる解を調べ，現在の解よりも良好な解があれば，その解に現在の解を置き換えるという操作を，良好な近傍解が算出できなくなるまで繰り返すものである．GA は，生物の進化にならったアルゴリズムで選択，交叉，突然変異の操作から構成される探索手法であり，適用範囲の広い多点探索方法として注目されている．一般に，GA は最適解の周辺には早く近づくが，局所探索能力が弱いという問題が指摘されている．この問題を解決する有効な手法の一つが LS と GA を組合せるというものである [9].

本研究での GLS について記述する．ここでの LS は，1-opt LS (FLIP) に基づいており，0 と 1 のビット列で構成される長さ n の解 x に対して，各部分集合のいずれかから，cut size を最も減少させる一つのビット（つまり，解を最も改善させるビットを意味する）を選び，そのビットを異なる部分集合に移動させる操作を良好な解が算出できなくなるまで繰り返すという操作を行う．まず，ランダムに生成された複数個の初期解に対して LS を実行し局所解に到達させ，これらの局所解の個体群に対して，GA の操作を行う．そして，交叉操作は GA のループの中で，局所解の個体群 PS の中からランダムに 2 つの親 I_a, I_bを選び，子 I_c を生成する．次に，交叉操作によって得られた子に対して LS によって局所解到達させ，新しい I_cを得る．本 GLS では，突然変異操作は加えない．その理由は，もし，突然変異が GLS の枠組みで実行された場合，突然変異操作は子の実行可能性を崩す可能性があるため，GX 本来の探索能力を十分に観測することができなくなると考えるからである．図 7 に，本研究での GLS の流れを示す.

GLS アルゴリズム

Step0. 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.

Step1. ランダムに初期世代の個体群 P^0 である個体 $I_1^0, ..., I_{PS}^0$ を実行可能解となるように生成する.

Step2. 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.

Step3. G が 終了世代 GN に到達するまで以下を繰り返す.

Step3.1 $G = G + 1$ とする.

Step3.2 子が $PS/2$ 個に到達するまで以下を繰り返す.

Step3.2.1 交叉を行う個体 $I_a, I_b \in P^G$ を選ぶ.

Step3.2.2 交叉 (I_a, I_b) によって I_c を得る.

Step3.2.3 I_cに LS を実行し，新しい I_c を得る.

Step3.2.4 I_c は，子の個体 P_c に加える.

Step3.3 (すべての個体 $\in P^G$ と P_c^G のそれぞれの cut size を計算．) 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.

Step4. 最良の個体 $\in P$ を返す.

図 7 GLS の流れ

5 実験結果

本実験では，GBP に対する GLS の枠組みを利用した我々の提案する GX1 と GX2 の貪欲的交叉法の効果を分析するために，ベンチマーク問題を用いて 3 つの交叉法を比較する．使用した問題例は，多くの研究者によって用いられている Johnson ら [6] の 16 個の問題例（点の数 124, 250, 500, 1000) である．これらの問題例に対して，10 回の試行による UX, GX1, GX2 の各交叉法の性能評価および挙動分析を行う．計算機は COMPAQ DESKPRO (Intel Pentium III 600MHz) を使用する．プログラム言語は C である.

表 1 に，本研究で用いた問題例について記述する．$gn.p$ と呼ばれる問題は，点 n のランダムグラフである．任意の 2 点間に枝の存在確率 $p(0 < p < 1)$ 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．表中の Best-known cut-size は，今までに知られている既知の最良解値を示している [4].

表 1 使用した問題例

Graphs ($gn.p$)	Best-known cut-size	Vertex (n)	Edge (E)	$p(n-1)$
g124.02	13	124	298	2.49
g124.04	63	124	636	4.92
g124.08	178	124	1240	9.84
g124.16	449	124	2542	19.68
g250.01	29	250	662	2.49
g250.02	114	250	1244	4.98
g250.04	357	250	1566	9.96
g250.08	828	250	4842	19.92
g500.005	49	500	1250	2.495
g500.01	218	500	2446	4.99
g500.02	626	500	4710	9.98
g500.04	1744	500	10240	19.96
g1000.0025	95	1000	2544	2.4975
g1000.005	445	1000	4992	4.995
g1000.01	1362	1000	10128	9.99
g1000.02	3382	1000	20214	19.98

表 2 から表 5 は，各問題例に対する UX と GX1, GX2 の世代ごとの実験結果を示す．これらの結果は，個体群サイズ : 40，交叉確率 : 1.0，計算打ち切り世代数 : 200 のパラメータを用いたものであり，UX, GX1, GX2 の次に記述する各世代での最良解 (Min)，各世代での最良解の平均値 (Avg)，平均値の解質 (%) を示している．表中の Gens は世代数を示し，Gens 1 は，初期のランダム個体群から 40 回の LS 実行後の結果である．また，Gens 5, 10, 20, 40, 100, 200 の結果は，100, 200, 400, 800, 2000, 4000 回の交叉操作によって生成された子に対して LS を実行して得られたものである．

結果から，g124.02, g124.08, g124.16, g250.08 の 4 つの問題例で，UX と両 GX は共に既知の最良解を算出したことがわかる．そして，p の低い問題例 (g124.02) では，UX, GX1 よりも GX2 が早い世代で既知の最良解を算出し，平均値においても質の高い解を早い世代で得たことがわかる．それに対して，p の高い問題例 (g124.16) では，両 GX よりも UX の方が早い世代で既知の最良解を算出し，平均値においても質の高い解を早い世代で得たことがわかる．また，点の数 250, 500, 1000 の問題例においても，点の数 124 の問題例と同様な p による各操作の傾向を示している．

表 6 に，本 GLS によって得られた UX, GX1, GX2 の操作に対する実験結果を示す．これらの表は，各操作に対する 10 回の試行によって得られた解の最良値である cut size (Min)，最良解の平均値 (Avg)，平均値の解質 (%)，Min が得られたときの平均世代数 (Gens)，各問題例に対する平均計算時間 (Time(s)) を示す．

GX と UX の解質について比較すると，GX は p が低い問題例に対して良い解を算出する傾向があることを観測した．特に，GX2 のその傾向は GX1 と UX よりもより顕著に現れている．これは，p が低い問題例は枝のつながりが密ではないため，貪欲的操作を強めた GX2 は，GX1 と UX よりもより効果的に働いたと考えられる．そして，p の高い問題例では枝のつながりが密であるので，ランダム性の少ない GX2 は UX よりも解質は若干劣る．しかしながら，UX と GX2 の p の低い問題例の解質の差は，p の高い問題例の解質の差よりも大きいことから，これらの問題例に対して GX2 は有効であると考えられる．また，最良解を算出したときの世代数の平均において，多くの問題例で UX，GX1 よりも早い世代で最良解を得ることができた．これは，GX2 が GX1 よりもさらに貪欲的な交叉法であるためと考えられる．

UX と GX の総合的な結果として確率 p が高い問題例ほど，少ない世代で最良解が得られ，p が低い問題例ほど，少ない世代で最良解が得られにくいことがわかる．この理由は，枝の存在確率 p の高い問題例では，点と点を結ぶ枝数が多いため，サブセット間に跨る枝の数 cut size を最小とするような分割のパターンが少なくなり，枝の存在確率 p が低い問題例ほど分割が複雑ではなくなるためと考えられる．

以上のことから，UX と両 GX の各交叉法は問題のタイプに依存することがわかった．また，問題例によっては GX の方が質の高い解を得ることができ，UX のみでも良質な解を生成できることを示した．

6 結論

本研究では，グラフ2分割問題 (GBP) に対する貪欲的な交叉操作として，greedy crossover1 (GX1), greedy crossover2 (GX2) を提案した．この2つの貪欲的な交叉操作 GX1, GX2 は，Uniform Crossover (UX) に基づき実現されている．GX の有効性を観測するために，GBP に対して遺伝的局所探索法 (GLS) の枠組みを利用した UX, GX1, GX2 の3つの交叉法の比較から，貪欲的操作の強い我々の提案した GX2 は GBP に対して質の高い解を算出可能な交叉法であることを示した．

今後は，本論文で検討したランダムグラフ以外の様々なグラフに対して，UX と GX（GX1, GX2）をそれぞれ有した GLS による性能について検討すると共に，Kernighan-Lin アルゴリズム [8] のようなより強力な局所探索法を GLS に取り入れ，GX の有効性を検討する予定である．

参考文献

[1] Battiti, R. and Bertossi, A. (1999), "Greedy, prohibition, and reactive heuristics for graph partitioning," *IEEE Trans. on Computers*, **48(4)**, *361–385.*

[2] Boese, K.D., Kahng, A.B. and Muddu, S. (1994), "A new adaptive multi-start technique for combinatorial global optimizations," *Operations Research Letters*, **16**, *101-113.*

[3] Bui, T.N. and Moon, B.R. (1996), "Genetic Algorithm and Graph Partitioning," *IEEE Trans. on Computers*, **45(7)**, *841–855.*

[4] 藤沢克樹, 久保幹雄, 森戸晋 (1994), "Tabu Search のグラフ分割問題への適用と実験的解析," 電学論, **114-c(4)**, *430–437.*

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

[6] 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.*

[7] 片山謙吾, 成久洋之 (2000), '遺伝的反復局所探索法とその最適化性能,' 電子情報通信学会論文誌, **J83-A(2)**, *179-187.*

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

[9] Kohmoto, K., Katayama, K. and Narihisa, H. (1999), "Performance of a genetic algorithm for the graph partitioning problem," *in proceedings First Western Pacific/Third Australia-Japan Workshop on Stochastic Models*, *315–324.*

[10] 河本敬子, 片山謙吾, 成久洋之 (2000), "貪欲的な操作による遺伝的交叉法の効果," 電子情報通信学会, 信学技報, *COMP2000-13, 57–64.*

[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] 山村雅幸, 小野貴久, 小林重信 (1992), "形質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, **7(6)**, *1049–1059.*

表 2 UX と GX の実験結果（点の数 124）

Graphs	Gens	UX			GX1			GX2		
		Min.	Avg.	(%)	Min.	Avg.	(%)	Min.	Avg.	(%)
g124.02	1	20	21.1	62.3	20	21.1	62.3	20	21.1	62.3
	5	16	17.8	36.9	17	19.3	48.4	14	15.7	20.7
	10	14	16.3	25.3	14	17.7	36.1	**13**	14.5	11.5
	20	**13**	14.5	11.5	14	15.7	20.7	**13**	13.6	4.6
	40	**13**	13.6	4.6	**13**	14.2	9.2	**13**	**13.3**	2.3
	100	**13**	13.5	3.8	**13**	13.7	5.3	**13**	**13.3**	2.3
	200	**13**	13.5	3.8	**13**	13.4	3.0	**13**	**13.3**	2.3
g124.04	1	70	72.3	14.7	70	72.3	14.7	70	72.3	14.7
	5	67	69.5	10.3	65	70.5	11.9	67	69.6	10.4
	10	65	67.7	7.4	65	68.5	8.7	66	68.1	8.0
	20	64	65.7	4.2	65	66.7	5.8	64	65.7	4.2
	40	**63**	64.4	2.2	64	65.5	3.9	64	65.1	3.3
	100	**63**	64.1	1.7	64	64.8	2.8	64	64.6	2.5
	200	**63**	**64.0**	1.5	64	**64.0**	1.5	64	64.5	2.3
g124.08	1	184	191.5	7.5	184	191.5	7.5	184	191.5	7.5
	5	182	185.7	4.3	181	187.1	5.1	181	186.0	4.4
	10	179	183.3	2.9	181	183.3	2.9	181	184.3	3.5
	20	**178**	181.6	2.0	179	182.2	2.3	181	183.3	2.9
	40	**178**	180.7	1.5	**178**	180.5	1.4	180	181.3	1.8
	100	**178**	179.2	0.6	**178**	**178.8**	0.4	**178**	179.9	1.0
	200	**178**	**178.8**	0.4	**178**	**178.8**	0.4	**178**	179.6	0.8
g124.16	1	457	463.3	3.1	457	463.3	3.1	457	463.3	3.1
	5	452	457.8	1.9	454	459.7	2.3	457	460.2	2.4
	10	452	454.4	1.2	452	457.1	1.8	454	457.6	1.9
	20	**449**	452.0	0.6	450	454.3	1.1	452	455.2	1.3
	40	**449**	450.8	0.4	**449**	451.8	0.6	451	452.1	0.6
	100	**449**	450.1	0.2	**449**	451.0	0.4	450	451.6	0.5
	200	**449**	**450.0**	0.2	**449**	450.9	0.4	**449**	451.3	0.5

表 3 UX と GX の実験結果（点の数 250）

Graphs	Gens	UX			GX1			GX2		
		Min.	Avg.	(%)	Min.	Avg.	(%)	Min.	Avg.	(%)
g250.01	1	50	55.7	92.0	50	55.7	92.0	50	55.7	92.0
	5	46	49.1	69.3	47	49.8	71.7	39	42.4	46.2
	10	42	45.4	56.5	44	48.3	66.5	36	38.6	33.1
	20	39	41.5	43.1	43	45.1	55.5	34	36.8	26.8
	40	34	38.3	32.0	38	42.0	44.8	32	35.0	20.6
	100	32	35.3	21.7	35	39.1	34.8	32	33.4	15.1
	200	32	34.8	20.0	34	36.6	26.2	31	**33.1**	14.1
g250.02	1	134	139.7	22.5	134	139.7	22.5	134	139.7	22.5
	5	130	132.3	16.0	130	135.1	18.5	130	133.9	17.4
	10	124	128.9	13.0	130	132.5	16.2	127	130.2	14.2
	20	121	125.7	10.2	124	129.1	13.2	124	127.9	12.1
	40	118	120.7	5.8	118	125.4	10.0	115	121.9	6.9
	100	116	118.1	3.5	116	119.9	5.1	115	119.4	4.7
	200	116	**117.4**	2.9	115	118.6	4.0	115	118.7	4.1
g250.04	1	380	391.9	9.7	380	391.9	9.7	380	391.9	9.7
	5	375	382.7	7.1	378	383.7	7.4	380	384.1	7.5
	10	373	375.9	5.2	373	379.1	6.1	372	380.0	6.4
	20	364	370.4	3.7	369	375.1	5.0	371	376.6	5.4
	40	360	365.6	2.4	359	365.4	2.3	362	370.7	3.8
	100	360	363.6	1.8	359	363.4	1.7	361	366.2	2.5
	200	359	363.1	1.7	359	**362.9**	1.6	361	365.7	2.4
g250.08	1	862	867.8	4.8	862	867.8	4.8	862	867.8	4.8
	5	845	852.1	2.9	854	859.2	3.7	853	857.0	3.5
	10	838	844.1	1.9	843	850.2	2.6	845	850.2	2.6
	20	835	839.7	1.4	837	843.4	1.8	838	844.6	2.0
	40	829	833.6	0.6	829	835.3	0.8	830	836.4	1.0
	100	**828**	831.9	0.4	**828**	831.7	0.4	**828**	833.5	0.6
	200	**828**	**831.6**	0.4	**828**	831.4	0.4	**828**	832.8	0.5

表 4 UX と GX の実験結果（点の数 500）

Graphs	Gens	UX			GX1			GX2		
		Min.	Avg.	(%)	Min.	Avg.	(%)	Min.	Avg.	(%)
g500.005	1	97	100.4	104.8	97	100.4	104.8	97	100.4	104.8
	5	87	90.0	83.6	91	94.9	93.6	73	77.3	57.7
	10	83	88.0	79.5	88	92.4	88.5	70	72.5	47.9
	20	73	82.0	67.3	83	88.2	80.0	65	67.7	38.1
	40	64	71.2	45.3	73	83.6	70.6	62	64.4	31.4
	100	57	64.1	30.8	67	75.2	53.4	58	60.4	23.2
	200	57	62.9	28.3	63	69.9	42.6	58	**59.9**	22.2
g500.01	1	282	289.9	32.9	282	289.9	32.9	282	289.9	32.9
	5	264	272.2	24.8	274	279.9	28.3	258	269.8	23.7
	10	255	266.6	22.2	263	272.4	24.9	252	263.8	21.0
	20	250	256.1	17.4	251	262.9	20.5	249	256.5	17.6
	40	237	241.3	10.6	244	252.7	15.9	234	250.5	14.9
	100	232	235.4	7.9	229	240.4	10.2	230	243.0	11.4
	200	230	**234.4**	7.5	227	237.9	9.1	230	242.2	11.1
g500.02	1	695	708.4	13.1	695	708.4	13.1	695	708.4	13.1
	5	674	686.6	9.6	677	691.0	10.3	687	691.7	10.4
	10	669	676.6	8.0	673	681.6	8.8	675	680.9	8.7
	20	655	662.9	5.8	653	668.1	6.7	665	674.5	7.7
	40	637	647.5	3.4	640	653.0	4.3	653	658.7	5.2
	100	632	641.4	2.4	640	645.3	3.0	642	649.3	3.7
	200	631	**640.6**	2.3	636	642.9	2.6	642	648.7	3.6
g500.04	1	1823	1842.0	5.6	1823	1838.4	5.4	1823	1838.4	5.4
	5	1806	1810.2	3.7	1803	1817.9	4.2	1802	1816.3	4.1
	10	1787	1796.0	2.9	1793	1801.7	3.3	1791	1804.6	3.4
	20	1774	1781.6	2.1	1764	1781.1	2.1	1778	1786.7	2.4
	40	1758	1770.8	1.5	1752	1766.5	1.2	1761	1774.8	1.7
	100	1752	1765.8	1.2	1750	1762.6	1.0	1751	1765.8	1.2
	200	1752	1765.0	1.2	1750	**1760.9**	0.9	1749	1763.4	1.1

表 5 UX と GX の実験結果（点の数 1000）

Graphs	Gens	UX			GX1			GX2		
		Min.	Avg.	(%)	Min.	Avg.	(%)	Min.	Avg.	(%)
g1000.0025	1	196	200.7	109.0	196	200.7	109.0	196	200.7	109.0
	5	183	189.0	96.8	190	196.3	104.4	149	157.3	63.8
	10	177	182.4	90.0	183	192.3	100.3	138	146.0	52.0
	20	160	166.6	73.5	178	186.5	94.2	135	136.2	41.8
	40	135	144.8	50.8	167	177.2	84.5	122	128.2	33.5
	100	120	127.7	33.0	142	154.2	60.6	116	123.2	28.3
	200	115	124.0	29.1	127	140.6	46.4	116	**122.6**	27.7
g1000.005	1	574	592.6	32.8	574	592.6	32.8	574	592.6	32.8
	5	542	562.3	26.0	565	575.1	28.9	558	566.5	27.0
	10	527	542.4	21.6	554	566.9	27.1	536	551.7	23.6
	20	503	518.9	16.3	541	546.8	22.6	524	534.4	19.8
	40	478	494.7	10.9	505	520.2	16.6	502	516.3	15.7
	100	472	484.5	8.6	477	492.0	10.3	490	501.0	12.3
	200	470	**481.5**	7.9	475	485.8	8.9	487	499.9	12.0
g1000.01	1	1535	1550.3	13.8	1535	1550.3	13.8	1535	1550.3	13.8
	5	1499	1506.2	10.5	1499	1519.6	11.5	1500	1513.7	11.1
	10	1464	1477.2	8.4	1469	1495.0	9.7	1477	1498.3	10.0
	20	1412	1432.9	5.2	1437	1455.0	6.8	1446	1465.8	7.6
	40	1392	1407.2	3.3	1407	1420.1	4.2	1419	1434.9	5.3
	100	1379	1398.9	2.7	1393	1404.9	3.1	1406	1416.9	4.0
	200	1379	**1397.2**	2.5	1390	1400.9	2.8	1404	1416.1	3.9
g1000.02	1	3581	3608.8	6.7	3581	3608.8	6.7	3581	3608.8	6.7
	5	3527	3544.4	4.8	3538	3563.4	5.3	3529	3556.0	5.1
	10	3475	3498.2	3.4	3514	3527.7	4.3	3492	3533.0	4.4
	20	3433	3454.6	2.1	3456	3484.8	3.0	3458	3476.2	2.7
	40	3416	3431.5	1.4	3421	3443.9	1.8	3420	3441.6	1.7
	100	3406	3423.6	1.2	3410	3431.0	1.4	3409	3428.6	1.3
	200	3405	**3421.8**	1.1	3403	3427.3	1.3	3408	3426.9	1.3

表 6 UX, GX (GX1, GX2) の実験結果

Graphs	Best	UX				GX1				GX2			
		Min	Avg		Gens	Time	Min	Avg		Gens	Time	Min	Avg		Gens	Time
g124.02	13	**13**	13.5	(3.8%)	29.3	0.6	**13**	13.4	(3.0%)	56.7	2.0	**13**	**13.3**	(2.3%)	15.4	1.0
g124.04	63	**63**	**64.0**	(1.5%)	37.8	0.5	64	**64.0**	(1.5%)	99.3	4.1	64	64.5	(2.3%)	43.0	2.6
g124.08	178	**178**	**178.8**	(0.4%)	76.7	1.3	**178**	**178.8**	(0.4%)	54.1	2.6	**178**	179.6	(0.8%)	69.4	4.5
g124.16	449	**449**	**450.0**	(0.2%)	41.7	1.3	**449**	450.9	(0.4%)	54.8	3.0	**449**	451.3	(0.5%)	62.5	5.5
g250.01	29	32	34.8	(20.0%)	108.5	4.7	34	36.6	(26.2%)	157.6	15.3	31	**33.1**	(14.1%)	82.9	15.5
g250.02	114	116	**117.4**	(2.9%)	98.9	4.4	115	118.6	(4.0%)	123.6	11.8	115	118.7	(4.1%)	92.3	14.9
g250.04	357	359	363.1	(1.7%)	90.0	4.4	359	**362.9**	(1.6%)	87.6	8.3	361	365.7	(2.4%)	87.0	16.4
g250.08	828	**828**	831.6	(0.4%)	70.1	4.3	**828**	**831.4**	(0.4%)	82.0	10.9	**828**	832.8	(0.5%)	82.7	20.3
g500.005	49	57	62.9	(28.3%)	105.9	12.6	63	69.9	(42.6%)	161.7	44.4	58	**59.9**	(22.2%)	93.0	67.1
g500.01	218	230	**234.4**	(7.5%)	114.9	13.0	227	237.9	(9.1%)	148.8	36.3	230	242.2	(11.1%)	81.2	47.2
g500.02	626	631	**640.6**	(2.3%)	112.7	13.4	636	642.9	(2.6%)	148.4	36.5	642	648.7	(3.6%)	97.4	58.4
g500.04	1744	1752	1764.9	(1.1%)	92.1	13.4	1750	**1760.9**	(0.9%)	115.9	34.0	1749	1763.4	(1.1%)	140.0	80.3
g1000.0025	96	115	124.0	(29.1%)	156.3	47.1	127	140.6	(46.4%)	177.1	146.7	116	**122.6**	(27.7%)	77.3	226.7
g1000.005	446	470	**481.5**	(7.9%)	157.5	44.3	475	485.8	(8.9%)	171.2	117.8	487	499.9	(12.0%)	95.4	192.3
g1000.01	1362	1379	**1397.2**	(2.5%)	143.8	40.0	1390	1400.9	(2.8%)	147.9	103.3	1404	1416.1	(3.9%)	99.6	195.2
g1000.02	3382	3405	**3421.8**	(1.1%)	128.6	43.3	3403	3427.3	(1.3%)	143.5	112.7	3408	3426.9	(1.3%)	117.0	236.9

A Study of Greedy Crossovers for the Graph Bi-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 1, 2000)

It is well known that the crossover is an important operator in the genetic algorithm (GA). We select two individuals, i.e., parents, from many individuals of a population, and apply the crossover using the parents to create a new individual (offspring). In the process of the crossover, it is important to create the offspring using the genetic information of the parents. We give the keynote of crossover: 1) it inherits as much good genetic information of parents as possible because they are worth preserving for offspring, and 2) the feasibility of the offspring created by the crossover is guaranteed for a given problem. In this paper, we study the effect of a greedy crossover for the graph bi-partitioning problem (GBP) by comparing greedy crossovers with a classical crossover. For the classical crossover in our study, we use an uniform crossover (UX). We investigate two greedy crossovers: greedy crossover 1 (GX1) and greedy crossover 2 (GX2) that are proposed in this paper. The greedy effect of GX2 is stronger than that of GX1. In order to analyze the effects of the three crossovers (UX, GX1, and GX2), we use a framework of the GA incorporating a local search heuristic. Experimental results show that GX2 is more effective than the others for GBP.