社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－MPS－67 (40)
2007－BIO－11 (40)
2007／12／21

# リンケージ同定とコンテクスト依存交叉を用いた遺伝的アルゴリズムの並列化

辻 美和子† 棟朝 雅晴† 赤間 清†

事前知識によらずに自動的に問題構造を検出するコンペテント遺伝的アルゴリズム (cGA)[2] の並列計算機上での実行は，幅広い問題に対して問題解決環境を提供できる可能性を持っている．代表的な並列 cGA としては，BOA の並列化である DBOA，PBOA[7]，並列 BOA[6]，LINC の並列化である pLINC[5] などが存在する．しかし，BOA の並列化はモデルに関する制約やバックトラックを要する．LINC は単純な並列化が可能なものの，大規模な問題に対してもともとの計算量の大きさから，並列計算機上でも大きな計算コストを要する．本論文では，単純に並列化でき比較的少ない計算量で問題構造の検出が可能な $D^5$ [12] と得られた情報に基づいて重複するビルディングブロックを組み合わせることができる交叉手法 CDC[11] を用いた $D^5$-GA+CDC の並列化に取り組む．また，並列 $D^5$-GA+CDC や他の並列 cGA の性質を明らかにするために比較実験を行なう．

# Parallelization of a genetic algorithm using linkage identification and context dependent crossover

Miwako Tsuji ,† Masaharu Munetomo † and Kiyoshi Akama†

Parallelized competent genetic algorithms (cGAs) which can detect problem structures automatically can give us problem solving environments for a wide spectrum of real-world problems. As such algorithms, the DBOA, PBOA[7] and parallel BOA[6] which are based on BOA and pLINC[5] which is based on LINC had been proposed. However, model buildings in the parallelized BOAs are performed under some restrictions or using backtracking to obtain feasible models. While pLINC can be parallelized in a simpler way, the original LINC spends huge number of fitness evaluations. In this paper, we try to parallelize $D^5$ [12] which can also be parallelized in a simple way and requires relatively small number of fitness evaluations. We also try to parallelize population evolution with context dependent crossover (CDC)[11] which can combine overlapping building blocks effectively. Moreover, we perform some experiments to compare the paralleled $D^5$-GA + CDC and other cGAs to reveal their properties.

## 1. はじめに

GRID などの並列計算機上で遺伝的アルゴリズム (GA) を用いて幅広い問題に対して問題解決環境を提供する試みが提案されている[3]．単純 GA (sGA) は，解くべき問題に関する事前知識がない場合に，十分な性能を発揮することができないため，これらの問題解決環境においては，事前知識によらずに問題構造を自動的に解析するコンペテント GA (cGA)[2] の利用が検討されている．本論文では，$D^5$[12] と CDC (Context Dependent Crossover) を組み込んだ GA の並列化に取り組む．さらに，並列 $D^5$-GA+CDC と他の並列 cGA の並列化効果について実験的に比較・検証し，位置付けする．

## 2. 並列コンペテント遺伝的アルゴリズム

sGA の並列化に関しては多くの試みが存在する[1]．マスタースレーブ・モデルは，適応度評価を複数のプロセッサで行ない，選択などのオペレータをマスターで実行する．島モデルは各プロセッサが小さい個体群を進化させ，数世代ごとに一部の個体を通信して入れ換える．だが，並列 sGA は交叉によるビルディングブロック (BB) 破壊に起因する探索効率の減少などを克服できなかった．適切に BB を組み合わせるためには互いに関連する変数を密に符号化する必要があるが，そのような符号化は，(並列) sGA では人間の専門知識に頼るため困難である．

一方で，専門知識によらない問題構造解析を目的とした cGA[2] の並列化も進められている．cGA は，リンケージと呼ばれる同一の BB を構成する変数の間の依存関係を検出し，適切に符号化が行なわれているかのように探索を行なう．しかし，分布の推定や余分な適応度評価などを実行するため，大きな計算機コストを要する．cGA の 1 カテゴリである分布推定アルゴリズム (EDA) は，有望な個体群の分布を推定して確率モデルを構築し，モデルから子個体を生成する．適切なモデルを構築するためには，個体群とモデルとの適合度を繰り返し計算する必要があるため，適応度評価に加えてモデル構築も並列化することが提案されている．モデルとしてベイジアンネットワークを用いる BOA の並列化には，DBOA，PBOA[7] や並列 BOA[6] が存在する．これらは，可能なモデルの構築のために，モデルへの制約やバックトラックを必要とする．また，一般的な EDA は，選択圧のかからない適応度への寄与の小さな BB のモデル化が困難である．他のアプローチとして，値の摂動による適応度の変化量から変数間のリンケージを検出する手法である LINC を並列化した pLINC[5] がある．pLINC は LINC における個体長 $l$ に対して準二次の適応度計算部分を並列に実行し，モデルに制約を課さない．

本論文では，$D^5$ [12] と CDC[11] を組み込んだ GA の並列化を行なう．$D^5$ は，適応度差と分布推定を組み合わせた手法で，計算量は $l$ の増加に対して準線形かつ寄与の小さな BB を検出が可能である．CDC は，得られた依存関係の情報に基づく交叉手法であり，重複する BB を効果的に交換することができる．

## 3. リンケージ同定手法 $D^5$ と交叉手法 CDC

本論文で並列化に取り組む GA の性能促進のための 2 つの重要な技術，リンケージ同定手法 $D^5$ と交叉手法 CDC について述べる．$D^5$-GA+CDC は以下のように実行される：

- $D^5$ によりリンケージ同定を行なう
- 得られたリンケージ情報を利用した交叉手法・CDC を用い個体群を進化させる

$D^5$ は，EDA と LINC などの摂動法を組み合わせた手法で，EDA 困難な問題に摂動法より少ない適応度評価回数で適用可能である．$D^5$ は以下のように実行される：

† 北海道大学情報基盤センター 大規模計算システム研究部門

サイズ $n$ の個体群 $P$ を初期化
for $i = 1$ to $l$
  for each $\boldsymbol{s} \in P$
    $\boldsymbol{s}' = s_1 \cdots \overline{s}_i \cdots s_l$ /* 変数 $i$ を摂動 */
    $\Delta f_i(\boldsymbol{s}) = f(\boldsymbol{s}') - f(\boldsymbol{s})$ /* 適応度差を計算 */
  end
  /* 適応度差 $\Delta f_i$ に基づいて個体を分類 */
  classify $\boldsymbol{s} \in P$ into $C_1, C_2, \cdots$
            according to $\Delta f_i(\boldsymbol{s})$
  for each $C_i$ /* 各クラスタの分布を推定 */
    $C_i$ において最小のエントロピーを与える変数の集合を探索しリンケージ集合 $V_i$ とする
  end
end

すべての変数に関するリンケージ集合を集めた後，冗長な集合を除いて，交叉に用いる最終的なリンケージ集合を構成する．

同一のリンケージ集合に属する変数からなる部分列は BB 候補とみなすことができる．BB どうしが重複する場合，一方の BB を交換することがもう一方を破壊する可能性あるが，それらを適切に組み合わせる手法として CDC が提案されている．CDC は Yu らによる交叉手法[13] の拡張で，親個体組の具体的な値を用いて BB 破壊の可能性を詳細に調査することにより，重複する BB を破壊せずに組み合わせる．このアルゴリズムは以下である：

ノードがリンケージ集合 $V_i$，枝が対応するノードの間の重複を示すグラフ $G = (N, E)$ を構築する
for 親個体組ごと
  · $G$ から親個体組上で同じ BB を示すノードを除去する
  · 重複部分の親個体の値が同じなど BB 破壊につながらない重複に対応する枝を除去する
  · ノード $n_1, n_2$ を無作為に選択する
  · $n_1 \in N_1, n_2 \in N_2$ かつ $|E| - |E_1| - |E_2|$ が最小になるように $G$ を $G_1 = (N_1, E_1)$，$G_2 = (N_2, E_2)$ に分割
  · $\mathcal{V} = \bigcup_{V_j \in N_1} V_j$ とし，$\mathcal{V}$ に属する変数を交換
end

## 4. D$^5$-GA+CDC の並列化

本章ではリンケージ同定手法 D$^5$ と交叉手法 CDC を用いた GA の並列化について述べる．いずれの手法も独立に実行できる繰りかえし部分が多いことから，単純な並列化が可能である．

D$^5$ の並列化を**図 1** に示す．まず，マスターは各プロセッサに個体群を送信する．スレーブは $l/n_p$ 個 ($n_p$ はプロセッサ数) の変数に関して，前章で示したリンケージ同定を行ない，結果をマスターに送信する．その後，マスターは得られた依存関係の情報をまとめて重複を許したリンケージ集合を構成する．以下では並列 D$^5$ を pD$^5$ と呼ぶ．

CDC を用いた個体群進化の並列化を**図 2** に示す．マスターはスレーブに個体群と担当すべき交叉ペアのリストを送信する．スレーブは CDC による交叉を実行した後，得られた子個体の評価を行なう．その後，子個体とその適応度をマスターに送信する．前述のように CDC はグラフの再構築や最小カットの探索を含むが，それらも子個体の適応度評価とともに並列で実行される．マスターは得られた子個体を親個体と置き換える．置き換えには，並列でない CDC を用いた探索と同様にニッチングのための置き換え

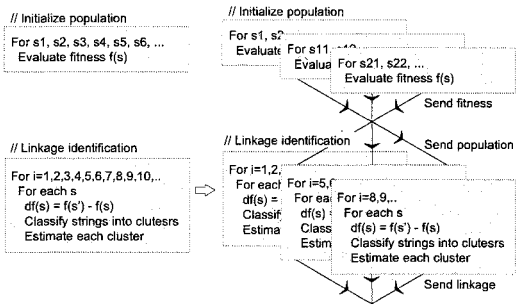


**図 1** D$^5$ の並列化．左が元のアルゴリズム，右が並列

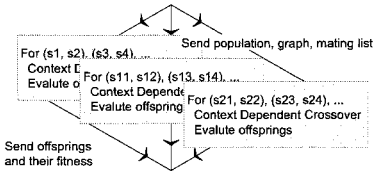


**図 2** CDC を用いた進化の並列化

手法である限定トーナメント選択 (RTR)[8] が用いられた．

## 5. 実　　験

### pD$^5$ と pLINC

pLINC と pD$^5$ を比較し，それぞれのリンケージ同定における台数効果や実行時間について検証する．テスト関数に 5bit トラップ関数の和を用いた：

$$f(\boldsymbol{s}) = \sum_{j=1}^{m} \mathrm{trap5}(\boldsymbol{s}_{v_j}), \tag{1}$$

$$\boldsymbol{s}_{v_1} = s_1 s_2 s_3 s_4 s_5, \boldsymbol{s}_{v_2} = s_6 s_7 s_8 s_9 s_{10}, \cdots$$

$$\mathrm{trap5}(\boldsymbol{s}_{v_j}) = \begin{cases} \frac{4-u}{5}, & u \leq 4 \\ 1, & u = 5 \end{cases} \tag{2}$$

$\boldsymbol{s}$ は個体，$m$ は部分関数の数，$f_j$ は $j$ 番目の部分関数，$\boldsymbol{s}_{v_j}$ はその部分解，$u$ は部分列に存在する 1 の数である．問題サイズは，$l = 100, 200, 300$ と変化させた．また，評価に時間がかかる情況を調査するために，適応度評価時間に $0.00, 0.02, 0.04$ 秒の待ち時間をつけた．個体群サイズは，それぞれの手法で最小のコストでリンケージ同定が可能な大きさとし，交叉確率 1.0，突然変異確率 0.0 とした．RTR のウィンドウサイズは $l/8$ とした．実験には，IBM x3455 (デュアルコア AMD プロセッサ 1.8GHz，1GB メモリ) 16 台を用いた．

**図 3** に，適応度評価待ち時間が $0.00, 0.02, 0.04$ 秒のときの pD$^5$ と pLINC のプロセッサ数毎の総実行時間と台数効果を示す．評価待ち時間が 0 のとき，同じ問題サイズに対する pD$^5$ と pLINC は，ほぼ同じ時間で探索を終えているが，$n_p \leq 2$ では pD$^5$ のほうが pLINC よりも多くの時間を要する．D$^5$ の適応度評価以外の処理が LINC のそれよりも複雑なためである．これについては次の実験で調査する．台数効果に関しては，評価時間が短いので通信のオーバヘッドの比率が大きくなり，どちらの手法でも低い

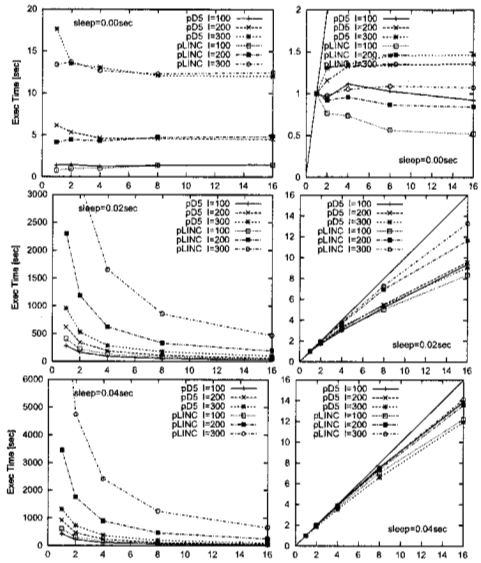


**図 3** 待ち時間が 0.00, 0.02, 0.04 秒 (上，中，下) のときの $pD^5$ と pLINC のプロセッサ数毎の総実行時間 (左) と台数効果 (右)

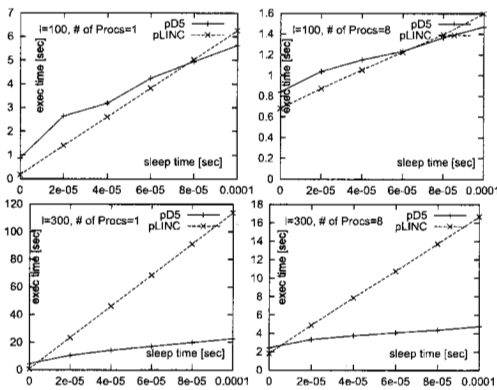


**図 4** 評価時間待ち時間と総リンケージ同定実行時間 $l = 100, 300$

値にとどまっている．

図 3 から，待ち時間が大きくなり，また問題サイズが大きくなるにつれ pLINC の実行時間が大きくなっていることがわかる．これは LINC の適応度評価回数が，問題サイズの準二次で増加するためである．pLINC はそのぶん大きな台数効果を得ており，待ち時間が 0.02 秒のときは特にその差が大きい．待ち時間が 0.04 秒のときは，どちらの手法に関しても適応度評価がより支配的になるため台数効果が大きくなる．

続いて，評価時間が短いときの pLINC と $pD^5$ におけるリンケージ同定時間をより詳しく検証する．短い待ち時間に関して sleep 関数の精度がよくないため，各プロセッサによる適応度評価を数えて，1 評価あたりの待ち時間に掛けた時間分をまとめて停止させるシミュレーション実験とした．**図 4** にプロセッサ数が 1 および 8 のときの評価待ち時間とリンケージ同定時間の関係を示す．pLINC は，リンケージを得るためにより多くの適応度評価を実行するが，$pD^5$ は pLINC よりも複雑な処理を行なっているため，すべての $l$ に関して待ち時間が小さいときには $pD^5$ が pLINC よりも多くの実行時間を要し，待ち時間が大きくなると pLINC のほうが多くの実行時間を要している．この逆転は，$l$ が大きいとき，より少ない待ち時間で見られる．

$pD^5$ と pLINC において，受信〜送信の間に各プロセッサが行なう処理について考える．1 回の評価時間を $t_f$ 秒，pLINC の個体数を $n_l$ とすると，各プロセッサの評価時間は $\left(\frac{l(l-1)}{2}+l\right) n_l \frac{t_f}{n_p}$ である[4]．$pD^5$ の個体数を $n_d$，各変数ごとのクラスタリング時間を $t_c$，分布推定時間を $t_e$ とすれば，プロセッサ毎の処理時間は，$n_d \frac{t_f}{n_p} + \frac{(t_c+t_e)l}{n_p}$ である．$pD^5$ が pLINC よりも実行時間が短いのは，

$$\left(\frac{l(l-1)}{2}+l\right) n_l \frac{t_f}{n_p} > l n_d \frac{t_f}{n_p} + \frac{(t_c+t_e)l}{n_p}$$

より $t_f > \frac{(t_c+t_e)}{\left(\frac{l-1}{2}+1\right)n_l - n_d}$ のときである．$n_l, n_d$ は $O(\log l)$ 以下のゆるやかさで増加するため[4),10)]，$l$ が増加すると $t_f$ の閾値は小さくなる．また，閾値の大きさは $n_p$ に影響されない．

**$pD^5$ と並列 BOA 〜BB 重複のある場合**

次に，$pD^5$ と並列 BOA[6] に関して，BB が重複する関数を用いて実験を行なった．並列 BOA は，確率モデル構築および適応度評価を並列に実行する．モデル構築では，各プロセッサが独立に依存関係を調査し，一定間隔ですべてのプロセッサのモデルをマスターにまとめ，可能なモデルになるようにバックトラックを行なう．

本実験では，確率的環状重複関数を用いた：

$$f(\boldsymbol{s}) = \sum_{j=1}^{m} f_j(\boldsymbol{s}_{\boldsymbol{v}_j}), \tag{3}$$

上式の部分列 $\boldsymbol{s}_{\boldsymbol{v}_j}$ は互いに重複する．$j$ 番目の部分関数を構成する変数は次式で決定される：

$$\boldsymbol{v}_j = (N(3j, \sigma^2) \bmod l, N(3j, \sigma^2) \bmod l, \cdots).$$

$N(\mu, \sigma^2)$ は平均 $\mu$，分散 $\sigma^2$ の正規分布である．$\sigma^2$ が大きくなれば，各部分関数を構成する変数は無作為に選択される．本実験では $\sigma^2 = 1^2$ および $\sigma^2 = 5^2$ の場合を考えた．正規分布の平均 $\mu$ の間隔は 3 とした．部分関数は 5bit トラップ関数とした．個体長は $l = 60$ および $l = 90$ とした．個体群サイズは各関数で十分に最適解が得られる値とした．適応度評価に 0.00, 0.04, 0.08 秒の待ち時間をつけた場合についてそれぞれ実験を行ない，最適解が得られるまでの実行時間を調査した．

**図 5** にそれぞれの関数に対する実行時間と台数効果を示す．待ち時間 0.00 秒のとき，BOA が毎世代実行するモデル構築に計算コストがかかるため，並列 BOA は特に大きな実行時間を要している．待ち時間が大きくプロセッサ数が少ないとき，$pD^5$ と並列 BOA はほぼ同じ実行時間で解を得ている．これは，本実験で用いた関数 (3) のような各部分関数の適応度寄与に差がない場合においては，オリジナルの $D^5$ および BOA の適応度評価回数がそれぞれ $O(l \log l)$[10] および $O(l^{1.55})$[9] で増加し，本実験の $l = 60, 90$ の段階では両者の差が小さいためである．台数

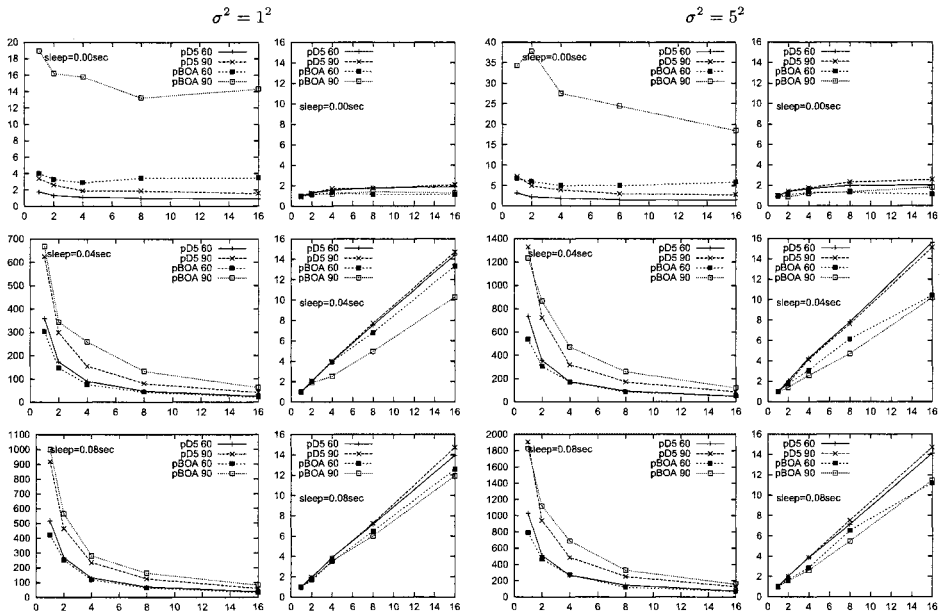


**図 5** 式 (3) に対する pD$^5$ と並列 BOA の実行時間と台数効果．横軸はいずれもプロセッサ数．
$\sigma^2 = 1^2, 5^2$．待ち時間は上から 0.00, 0.04, 0.08s

効果を見ると，待ち時間が 0.04 秒のとき pD$^5$ のほうが並列 BOA よりも大きな効果を得ている．並列 BOA では，バックトラックによるモデル構築が 1 プロセッサによる実行のときと同等の性能を得られないため，台数効果が減少していると考えられる．また，pD$^5$ は進化の前処理としてすべてのリンケージ同定を行なうため，1 プロセッサでまとめて処理する量が大きいが，並列 BOA の場合は各世代ごとのベイジアンネットワーク構築の一定期間ごとに通信を行なうため，このオーバヘッドが大きく共有メモリなどの環境にない場合には並列化の効果が小さくなる．待ち時間が 0.08 秒のとき，より適応度評価が支配的になることから，並列 BOA の台数効果が増加し，pD$^5$ に近くなる．

## 6. おわりに

本論文では，D$^5$ と CDC を組み込んだ GA の並列化を行ない，他の cGA との比較実験を行なった．pD$^5$ は，pLINC と比べて，評価時間が比較的大きく問題サイズが大きいときに有効だった．評価時間が非常に短いとき，適応度評価回数が pD$^5$ のほうが少なくても，実行時間においては pLINC のほうが高速である．評価時間が増加するにつれて，pD$^5$ のほうが高速に解を得られるようになる．D$^5$ は，BOA と異なり単純なマスタースレーブによるモデル化が可能であり，高い並列化の効果が得られる．以上から，特に規模が大きく適応度評価に多くの時間を要する問題に対して，並列 D$^5$-GA + CDC は強力な問題解決環境を提供することが可能であると考えられる．

## 参考文献


1) Cantú-Paz, E.: Designing Efficient and Accurate Parallel Genetic Algorithms, PhD Thesis, Univ. of Illinois at Urbana-Champaign (1999).
2) Goldberg, D.E.: *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, Kluwer Academic Publishers (2002).
3) Munawar, A., Munetomo, M. and Kiyoshi, A.: Optimization problem solving framework employing GA with linkage identification over a GRID environment, *Proc. the CEC*, p. 1191–1198 (2007).
4) Munetomo, M. and Goldberg, D.E.: Identifying Linkage Groups by Nonlinearity/Non-monotonicity Detection, *Proc. GECCO*, Morgan Kaufmann Publishers, p.433–440 (1999).
5) Munetomo, M., Murao, N. and Akama, K.: Empirical Investigations on Parallelized Linkage Identification, *Proc. the PPSN VIII, LNCS 3242*, p.322–311 (2004).
6) Munetomo, M., Murao, N. and Akama, K.: Empirical Studies on Parallel Network Construction of Bayesian Optimization Algorithms, *Proc. CEC , Vol.2*, p. 1524–1531 (2005).
7) Očenášek, J.: Parallel estimation of distribution algorithms, Doctroral dessertation, Brno Univ. of Technology (2002).
8) Pelikan, M. and Goldberg, D.E.: Hierarchical Problem Solving and the Bayesian optimization algorithm, *Proc. GECCO* (2000).
9) Pelikan, M., Sastry, K. and Goldberg, D.E.: Scalability of the Bayesian optimization algorithm, *International Journal of Approximate Reasoning*, Vol.31, No.3, p.221–258 (2002).
10) Tsuji, M., Munetomo, M. and Akama, K.: Population Sizing of Dependency Detection by Fitness Difference Classification, *FOGA, LNCS 3469*, p.282–299 (2005).
11) Tsuji, M., Munetomo, M. and Akama, K.: A crossover for complex building blocks overlapping, *Proc. GECCO*, p. 1337–1344 (2006).
12) Tsuji, M., Munetomo, M. and Akama, K.: Linkage Identification by Fitness Difference Clustering, *Evolutionary Computation*, Vol.14, No.4, p.383–409 (2006).
13) Yu, T.-L., Sastry, K. and Goldberg, D.E.: Linkage learning, overlapping building blocks, and systematic strategy for scalable recombination, *Proc. GECCO*, p.1217–1224 (2005).