パターンに基づく遺伝的アルゴリズムの部分解保護

Pattern-based Preservation of Building Blocks in Genetic Algorithms

亀谷由隆 [1*]　　Chativit Prayoonsri[1]
Yoshitaka Kameya[1]

[1] 東京工業大学大学院情報理工学研究科
[1] Graduate School of Information Science and Engineering

Abstract: As stated in the building block hypothesis, we expect genetic algorithms (GAs) to create building blocks (BBs) and combine them appropriately in the evolutionary process. However, such BBs are often destroyed by unwanted crossovers, soon after they are created. Also, we may suffer from a "loose" encoding of chromosomes since BBs are in general unknown. In this paper, we propose a framework named GAP (GA with patterns), in which key patterns are extracted from significantly "good" chromosomes and protect such key patterns against unwanted crossover. GAP is applicable to optimization problems with fixed-point encoding and permutation encoding, and unlike perturbation-based linkage learning methods, GAP does not require extra fitness evaluations. Experimental results with the royal road problems and traveling salesman problems show the performance improvement of GAP over standard GAs.

1　はじめに

積み木仮説 (building block hypothesis) [1, 2] に述べられているように，遺伝的アルゴリズム（genetic algorithm, 以下 GA）の進化過程においては部分解が生成され，またそれらの部分解が適切に組み合わされることが期待される．しかしその一方で，生成された部分解が早い段階で交叉によって破壊されてしまうという現象が見られる．また，対象とする最適化問題にどのような部分解が存在するか不明な場合も一般にあり得るため，部分解が散在するような染色体表現（符号化）がなされてしまう可能性がある．

上の問題に対処するため，これまで様々なリンケージ学習手法が開発されてきた．リンケージ学習手法は主に 3 種類に分類される [3, 4]：

- messy GA [5], LLGA [6] 等のリンケージ適応手法
- LINC [7], LIMD [3], D^5 [8], ILI [9] 等の摂動 (perturbation) に基づくリンケージ同定手法
- BOA [10], DSMGA [4] 等のモデル構築手法

これらのリンケージ学習手法の多くは遺伝子の意味が座位 (locus) に依存する表現（多くの場合はビット表現であるので，以下では単にビット表現と書くことにする）を対象としており，部分解は座位の単位で同定・保護される．我々の知る範囲では，順列表現を扱える手法は少なく，マルコフ連鎖に基づくモデル構築手法である EHBSA [11] で扱われるのみである．

本論文では GAP (GA with patterns) と呼ばれる枠組みを提案する．GAP は Gero と Kazakov の遺伝子工学 (genetic engineering) アプローチ [12] の拡張と見なすことができる．すなわち GAP では，非常に優良な染色体からギャップを含む部分列を重要なパターンとして複数抽出し，これらのパターンの出現が交叉によって破壊されることを防ぐ．GAP は，系列パターン発見手法を用いて重要なパターンを抽出するため，ビット表現だけでなく順列表現も扱える点が Gero と Kazakov の方法と異なる．更に GAP では，抽出されたパターンに基づき，(LEGO [13] のように）細粒度（遺伝子単位）で，(CDC [14] のように）各交叉における親染色体に依存して部分解の保護を行う．従って座位レベルで重複する部分解も GAP では自然に扱うことができる．更に，摂動に基づく方法と異なり，GAP では適合度関数の評価を追加的に行う必要がなく，むしろ逆に進化を加速させることで，適合度関数の評価回数を減らすことができる．また，GAP の別の利点として，理解可能性が高いことも挙げられる．GAP は標準的な GA や Gero と Kazakov の方法と同様に進化論・遺伝学の観

*連絡先：東京工業大学大学院情報理工学研究科計算工学専攻
〒 152-8552 東京都目黒区大岡山 2-12-1
E-mail: kameya@mi.cs.titech.ac.jp

1. 集団を初期化し，$\Delta^{(0)}$ とする．$t := 0$ とおく．
2. $\Delta^{(t)}$ から切捨て選択により，2 種類の染色体集合を求める：
   (a) Δ_{sel}（切捨て率 r_{sel}），
   (b) Δ_{mine}（抽出切捨て率 r_{mine}，$r_{\mathrm{mine}} \leq r_{\mathrm{sel}}$）
3. Δ_{mine} からパターン集合 Π を抽出する．
4. Δ_{sel} に対し，Π に基づいて交叉を適用する．突然変異を適用し，新たな集団 $\Delta^{(t+1)}$ を得る．
5. $t := t + 1$ と更新する．
6. 停止条件を満たしたら終了する; そうでなければステップ 2 に進む．

図 1: GAP の概要．

点から理解でき，また抽出されたパターンを観測することで，進化過程で何が起こったかを事後的に解析することも可能である．

以下では次の構成をとる．2 節では GAP について記述する．2.1 節で系列パターン発見手法が GAP でどのように導入されるかについて述べる．また，2.2 節では部分解の保護方法について述べる．3 節で実験結果を示し，4 節で関連研究を述べる．5 節で結論を述べる．

2 提案方法

2.1 飽和系列パターンの発見

図 1 に GAP の概要を示す．GAP は基本的に通常の GA に従うが，ステップ 2b, 3 において良好な染色体に頻出するパターン Π を抽出し，ステップ 4 において Π に基づき交叉を行う点が異なる．Π は非常に高い適合度を持つ集団 Δ_{mine} から抽出される．一方，Δ_{sel} は通常の選択操作によって選ばれる集団である ($\Delta_{\mathrm{mine}} \subseteq \Delta_{\mathrm{sel}}$)．ここで Π 中の各パターンは推定部分解 (induced building block) と見なすことができる．現在は，切捨て選択 (truncation selection) を用いた完全な世代交代を考えており，突然変異操作は従来通り行う．また，ステップ 3 を抽出ステップと呼び，一方ステップ 4 中で推定部分解を保護する操作を保護ステップと呼ぶ．次の 2 つの節ではそれぞれ抽出ステップと保護ステップについて述べる．

2.1.1 頻出パターンと部分解

高速な抽出ステップを実現するため，BIDE [15] と呼ばれる系列パターン発見アルゴリズムを用いる．BIDE は深さ優先型の系列パターン発見アルゴリズムである PrefixSpan [16] を拡張したものであり，系列集合 Δ に頻出する飽和 (closed) 部分列を全て列挙する．GAP においては Δ が集団に対応し，Δ 内の系列データが染色体に対応する．ここで幾つかの用語と記法を導入する．系列 $s = \langle \alpha_1, \alpha_2, \ldots, \alpha_n \rangle$ が与えられたとき，$1 \leq i_1 < i_2 < \cdots < i_m \leq n$ かつ $\beta_1 = \alpha_{i_1}$, $\beta_2 = \alpha_{i_2}$, $\ldots$, $\beta_m = \alpha_{i_m}$ $(m > 0)$ を満たす $i_1, i_2, \ldots, i_m$ が存在するような $s' = \langle \beta_1, \beta_2, \ldots, \beta_m \rangle$ を「s の部分系列」と呼ぶ[1]．例えば $\langle a \rangle$, $\langle b \rangle$, $\langle c \rangle$, $\langle a, b \rangle$, $\langle a, c \rangle$, $\langle c, b \rangle$ $\langle a, c, b \rangle$ は $\langle a, c, b \rangle$ の部分系列である．また，s' が s の部分系列であるとき,「s' は s に出現する」といい，$s' \subseteq s$ と表す．$s' \subseteq s$ かつ $s' \neq s$ であるとき，$s' \subset s$ と表す．

順列表現においては，染色体は単純に遺伝子の系列データとして表すことができる．一方，ビット表現においては，messy GA や LLGA と同様に，染色体を座位-遺伝子（ビット）のペアから成る系列データと見なす．ただし GAP においては，$l < l'$ のとき，座位-遺伝子のペア (l, a) が他のペア (l', a') の前方に出現するとする．例えば，順列表現の染色体 $\langle a, c, d, e, b \rangle$ はそのまま系列データとして扱い，ビット表現の染色体 $\langle 1, 0, 1, 1, 0 \rangle$ は系列データ $\langle (1, 1), (2, 0), (3, 1), (4, 1), (5, 0) \rangle$ に翻訳して考える．

更に，パターンと呼ばれる（部分）系列として s を考える．そして $\sigma(s, \Delta)$ を集団 Δ 中で s が出現する染色体の数とする．このとき $\sigma(s, \Delta)$ は Δ における s のサポートと呼ばれ，与えられた集団 Δ と閾値 $\sigma_{\min}(> 0)$ に対して $\sigma(s, \Delta) \geq \sigma_{\min}$ であるとき，パターン s は頻出であるという．閾値 $\sigma_{\min}$ は最小サポートと呼ばれる．文脈より明らかな場合，本論文では $\sigma(s, \Delta)$ を $\sigma(s)$ と略記する．$\Delta (= \Delta_{\mathrm{mine}})$ が非常に適合度の高い染色体の集合であり，かつ $\sigma_{\min}$ が充分に大きいとき，頻出パターンの各々を部分解であると考えることにする．加えて上の定義から，パターンにはギャップが含まれてよいことに注意する．例えば，順列表現 $\langle a, c, d, e, b \rangle$ には $\langle a, c \rangle$ や $\langle a, d \rangle$ などのパターンが出現する．ビット表現を翻訳した系列 $\langle (1, 1), (2, 0), (3, 1), (4, 1), (5, 0) \rangle$ には $\langle (2, 0), (5, 0) \rangle$（スキーマ記法では $\langle *, 0, *, *, 0 \rangle$ に対応する）などのパターンが出現する．以上より，GAP ではビット表現や順列表現において柔軟なパターンを取り扱う．

[1] 元々の BIDE の記述では，α_j と β_j はアイテム（記号）の集合であるが，ここではより簡単な場合を考えている．

2.1.2 飽和パターン

先に述べたように，BIDE は頻出する飽和パターンを列挙することができる．パターン s に対して，$s \subset s'$ であるような s' がいずれも $\sigma(s) > \sigma(s')$ であるとき，「s は飽和している」という．言い換えれば，$s \subset s'$ かつ $\sigma(s) = \sigma(s')$ であるような s' が存在する場合，s は飽和パターンではない．

飽和パターンは同じ出現回数を持つパターン同士の中で最も情報量が多いという意味で好ましい．加えて，飽和パターン発見手法を用いると，類似した長い染色体が多く含まれる集団から効率的にパターンを見つけることが可能になる．例えば極端な場合として，染色体 $\langle a, b, c, d, e\rangle$ が $\sigma_{\min}$ 回以上集団 Δ に出現した場合を考える．このとき，$\langle a, b, c, d, e\rangle$ の部分列全て ($\langle a\rangle$, $\langle a, b\rangle$, $\langle a, c\rangle$, ...) が頻出パターンである．従ってこのような場合は染色体長に対し指数倍個のパターンが存在することになり，一般にそれらを列挙するのは実時間では困難である．一方，$\sigma(s) > \sigma(\langle a, b, c, d, e\rangle)$ でない限り，s ($\subset \langle a, b, c, d, e\rangle$) は飽和パターンとはならないため，頻出する飽和パターンの数は抑えられる．更に，BIDE ではパターン候補が飽和しているかどうかを動的に検査することで，探索空間の枝刈りを積極的に行う．進化過程の後半では染色体が互いに似てくるため，現在のところ GAP においては飽和パターンの利用が欠かせないと思われる．

2.1.3 制約を利用した効率化

パターンの探索を更に効率的に行うため，BIDE は 2 種類の制約を利用する．一つは染色体表現における制約である．例えば，ビット表現，順列表現の双方において，各パターンは一つの染色体に一回しか出現しない．更にビット表現においては，$l < l'$ のときに座位-遺伝子のペア (l, a) は他のペア (l', a') の後に出現することはなく，また (l, a) と $(l + 1, a')$ の間に他の座位-遺伝子のペアが出現することもない．この性質を利用して，BIDE は飽和パターンの検査を一部省略している．もう一つの制約はパターンの形に関するものである．この制約は最適化問題の性質に応じてユーザが指定する．現在の実装では元のアルゴリズム [15] に修正を施し，パターンの最小長 $L_{\min}$，最大長 $L_{\max}$，パターンに含まれるギャップの最小幅 $G_{\min}$，最大幅 $G_{\max}$ という 4 種類の制約を指定可能にしている．このうち $L_{\max}$, $G_{\min}$, $G_{\max}$ は BIDE の高速化に貢献する．例えば $L_{\min} = 2$ を指定するのはどの最適化問題でも有効であると考えられ，また順列表現において $G_{\max} = 1$ あるいは 2 を指定すれば局所的なパターンを抽出できる．

2.1.4 top-K パターン発見手法の導入

抽出されるパターンの数は最小サポート $\sigma_{\min}$ に敏感であることがよく知られている．$\sigma_{\min}$ が小さすぎると抽出されるパターンが極端に多くなり，また $\sigma_{\min}$ が大きすぎるとパターンが何も抽出されなくなってしまう．GA では進化過程の途中で適切な $\sigma_{\min}$ の値が変わると思われるため，$\sigma_{\min}$ を自動的に調整できることが望ましい．このため我々は minimum support raising [17] と呼ばれる，実装容易な top-K パターン発見手法を導入する．この top-K パターン発見手法では，$\sigma_{\min}$ を自動的に調整して，最も頻出な K 個のパターンのみが返ってくる．具体的に言うと，我々は非常に小さな最小サポート $\sigma_{\min}^{(0)}$（すなわち $\sigma_{\min} := \sigma_{\min}^{(0)}$）から出発し，深さ優先探索の途中で K 個パターンを見つけたら，最小サポートを $\sigma_{\min} := \sigma(s)$ と更新する．ここで s は上位 K 個のパターン候補の中で最も出現回数の少ないパターンである．そして，新しいパターン候補が発見される度にこの更新が繰り返され，最小サポートが増加する．以降では $\sigma_{\min}^{(0)}$ を初期最小サポートと呼ぶ．

多くの場合，最小サポートは素早く引き上げられ，探索空間を枝刈りする機会が増すため，top-K パターン発見におけるオーバヘッドはそれ程大きくないと考えられる．それよりも $\sigma_{\min}$ の代わりに K と $\sigma_{\min}^{(0)}$ を指定する簡便さのメリットの方が大きい．また，BIDE はパターンが飽和であるかどうかの検査を動的に行い，上位 K 個の候補には飽和パターンしか含まれないため，BIDE と組み合わせた場合に minimum support raising を行うことは安全である．

2.2 交叉点分布の修正に基づく部分解保護

推定部分解を得た後に問題になるのは，その推定部分解を次世代でどう継承し，組み合わせるかである．そのために GAP の保護ステップでは，親染色体における推定部分解の出現位置を元に，可能な交叉点上の確率分布（以下，交叉点分布）を適宜修正することを考える．このようなソフトなアプローチをとるのは，(特に進化の初期段階で）推定された部分解が誤っている可能性があるためである．GAP の交叉操作は，交叉点分布が交叉の度に親染色体に依存して決定し直される点で CDC (context-dependent crossover) [14] に類似しており，CDC と同様に座位が重複する部分解を自然に扱うことができる．

この 2.2 節の以降では，一点交叉，二点交叉，オリジナルのエッジ交叉 (edge recombination)，位置に基づく交叉 (position-based crossover) の各々についてどのように交叉点分布を修正するかを記述する．ここで話を簡単にするために，我々は各交叉オペレータの交叉点分布に対して共通の修正手続きを行う．すなわち，望ま

しくない交叉点から確率 mass を割り引き (discount), 割り引いた確率 mass を他の（望ましい）交叉点に再配分する．この修正手続きは Sebag と Schoenauer の交叉制御手法 [18] に類似している．

各交叉オペレータに対する修正手続きを具体的に説明する前に，記法を追加する．まず，抽出ステップでパターン（推定部分解）$\pi = \langle \beta_1, \beta_2, \ldots, \beta_m \rangle$ が得られたとする．π が染色体 c に出現するとき ($\pi \subseteq c$)，$\mathrm{Occ}(\pi, c)$ は π が占める c 中の座位の集合を表す．すなわち，$c = \langle \alpha_1, \alpha_2, \ldots, \alpha_n \rangle$ かつ $\beta_1 = \alpha_{i_1}$, $\beta_1 = \alpha_{i_1}$, $\ldots$, $\beta_m = \alpha_{i_m}$ が成り立つとき，$\mathrm{Occ}(\pi, c) = \{i_1, i_2, \ldots, i_m\}$ となる．例えば，$\pi = \langle b, d \rangle$ および $c = \langle a, b, c, d \rangle$ に対して，$\mathrm{Occ}(\pi, c) = \{2, 4\}$ が成り立つ．

2.2.1 一点交叉

以上の準備の下で，一点交叉の場合における交叉点分布の修正手続きを説明する．まず，長さ n である二つの親染色体 c_1, c_2 を考える．すると一点交叉では $N = (n-1)$ 箇所の交叉点 $\{\phi_1, \phi_2, \ldots, \phi_N\}$ が存在する．ここで ϕ_i は第 i 座位と第 $(i+1)$ 座位の間にある交叉点である．また，ϕ_i が交叉点として選ばれる確率を p_i と書く．このとき，$\{p_i \mid 1 \leq i \leq N\}$ は可能な交叉点の集合 $\{\phi_i \mid 1 \leq i \leq N\}$ 上の確率分布，すなわち交叉点分布である．

ここで，K 個のパターン $\Pi = \{\pi_1, \pi_2, \ldots, \pi_K\}$ が抽出ステップで得られているとする．もし，Π 中のどのパターンも親染色体に出現しなければ，通常どおり一点交叉が行われる．また，もし Π のパターンにより片方の親の座位が全て占められてしまった場合は交叉自体がスキップされる．そのいずれでもない場合，交叉点分布が次のように修正される．まず，各親染色体 c_u ($u = 1, 2$) に対し，Π 中のパターンが占める最左の座位 $i^{(u)}_{\mathrm{left}}$ と最右の座位 $i^{(u)}_{\mathrm{right}}$ が $i^{(u)}_{\mathrm{left}} = \mathrm{minOcc}(\Pi, c_u)$, $i^{(u)}_{\mathrm{right}} = \mathrm{maxOcc}(\Pi, c_u)$ と計算される．ここで $\mathrm{Occ}(\Pi, c)$ は

$$\mathrm{Occ}(\Pi, c) = \bigcup_{1 \leq k \leq K : \pi_k \subseteq c} \mathrm{Occ}(\pi_k, c). \qquad (1)$$

と定義される．そして，第 $i^{(u)}_{\mathrm{left}}$ 座位から第 $i^{(u)}_{\mathrm{right}}$ 座位までを保護しようと考える．そのために $\Phi = \{\phi_i \mid i^{(1)}_{\mathrm{left}} \leq i < i^{(1)}_{\mathrm{right}}\} \cup \{\phi_i \mid i^{(2)}_{\mathrm{left}} \leq i < i^{(2)}_{\mathrm{right}}\}$ を求め，Φ に含まれる交叉点の確率を以下のように割り引く：

$$p_i := \frac{\delta}{N} \quad (\phi_i \in \Phi). \qquad (2)$$

ここで δ ($0 < \delta < 1$) は，ユーザが指定する制御パラメータであり，割引率と呼ばれる．割り引かれた確率 mass $h(1-\delta)/N$ は以下のように再配分される：

$$p_i := \frac{1}{N} + \frac{1}{N-h} \cdot \frac{h(1-\delta)}{N} = \frac{N - h\delta}{N(N-h)} \quad (\phi_i \notin \Phi). \qquad (3)$$

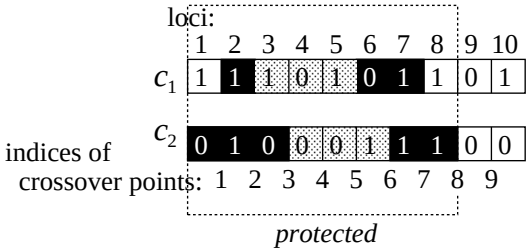

図 2: 一点交叉に対する部分解保護．

ここで $h = |\Phi|$ は保護される交叉点の数である．

以上の処理を図 2 を用いて説明する．まず長さ 10 の親染色体 c_1, c_2 を考える．このとき 9 箇所の交叉点が存在し ($N = 9$)，図 2 では 1 から 9 まで番号が振られている．c_1 と c_2 で黒く塗りつぶされた各ビットは Π 中のパターンのどれかに占められていることを表し，$\mathrm{Occ}(\Pi, c_1) = \{2, 6, 7\}$ および $\mathrm{Occ}(\Pi, c_2) = \{1, 2, 3, 7, 8\}$ に対応する．一方，網掛けのビットはパターン内のギャップに対応する．このとき，定義により $i^{(1)}_{\mathrm{left}} = \min\{2, 6, 7\} = 2$, $i^{(1)}_{\mathrm{right}} = 7$, $i^{(2)}_{\mathrm{left}} = 1$ および $i^{(2)}_{\mathrm{right}} = 8$ が成り立つ．従って座位集合 $\Phi = \{\phi_1, \phi_2, \ldots, \phi_7\}$ に対応する交叉点確率が割り引かれる ($h = |\Phi| = 7$)．そして δ の値（例えば $\delta = 0.5$）と式 2, 3 に基づき，交叉点分布が修正される．

上で述べた一点交叉における部分解保護には，図 2 における網掛けのビットは推定部分解に含まれていないにも関わらず黒塗りのビットとともに過剰に保護されてしまうという問題がある．この問題は次節で説明する二点交叉における交叉点分布の修正手続きで解消される．

2.2.2 二点交叉

二点交叉の場合，長さ n の親染色体 c_1, c_2 に対して，$N = n(n+1)/2$ 箇所の交叉点 $\{\phi_{ij} \mid 0 \leq i < j \leq n\}$ が存在する．ここで ϕ_{ij} は第 $(i+1)$ 座位から第 j 座位の部分を交換することに対応する．ここでは簡単のため，交叉点部分布の修正手続きを例で説明する．図 3 は，長さ $n = 10$ の親染色体に対して与える交叉点の番号と保護したい座位 $\mathrm{Occ}(\Pi, c_1) \cup \mathrm{Occ}(\Pi, c_2) = \{2, 6, 7\} \cup \{1, 2, 3, 7, 8\} = \{1, 2, 3, 6, 7, 8\}$. を示している．一点交叉の場合と異なり，我々は第 9, 10 座位のビットだけではなく第 4, 5 座位のビットも交換できることが分かる．更に，$\Phi_{\mathrm{full}} = \{\phi_{ij} \mid 0 \leq i < j \leq n\}$ を全ての交叉点から成る集合，$\overline{\Phi}_1 = \{\phi_{ij} \mid i, j \in \{3, 4, 5\}, i < j\}$ を保護対象の座位の間にある交叉点の集合，$\overline{\Phi}_2 = \{\phi_{ij} \mid i, j \in \{0, 8, 9, 10\}, i < j\}$ を保護対象の座位の外側にある交叉点の集合としたとき，保護対象の交叉点 Φ は $\Phi = \Phi_{\mathrm{full}} \setminus (\overline{\Phi}_1 \cup \overline{\Phi}_2)$ で与えられる．このとき一点交

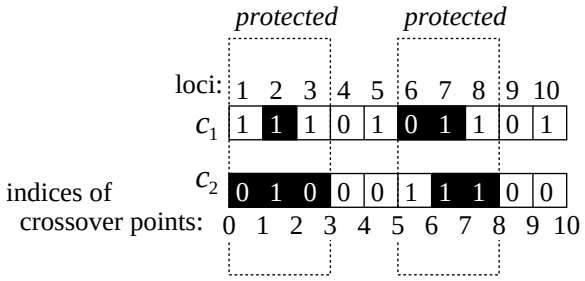

図 3: 二点交叉に対する部分解保護.

叉の場合と同様，交叉点分布 $\{p_{ij} \mid 0 \leq i < j \leq n\}$ が

$$p_{ij} := \frac{\delta}{N} \quad (\phi_{ij} \in \Phi) \tag{4}$$

$$p_{ij} := \frac{N - h\delta}{N(N-h)} \quad (\phi_{ij} \notin \Phi) \tag{5}$$

と修正される．ここで δ は割引率，$h = |\Phi|$ である．ここで述べた Φ を求めるため手続きを一般化するのは容易である．図 2 と図 3 を比較すると，二点交叉では保護対象が最小範囲で済んでいることが分かる．

2.2.3 エッジ交叉

エッジ交叉は順列表現用に提案された交叉オペレータである [19]．ここではまず長さ n の親染色体を考える．この親染色体には $\{1, 2, \ldots, n\}$ から選ばれた遺伝子がちょうど一回ずつ出現する．例えば長さ $n = 6$ の場合，$\langle 2, 4, 3, 1, 5, 6\rangle$ は順列表現の染色体である．以降では簡単のため，巡回セールスマン問題を考慮して，各遺伝子が都市の番号を表すとする．エッジ交叉では二つの親染色体に対して，まずエッジ写像 (edge map) と呼ばれる表 M を構築する．$M[\gamma]$ は都市 γ とエッジで繋がる都市の集合を指し，γ のエッジリスト (edge list) と呼ばれる．

例えば [19] に従い，二つの親染色体 $\langle 1, 2, 3, 4, 5, 6\rangle$, $\langle 2, 4, 3, 1, 5, 6\rangle$ を考える．このとき，エッジ写像 M は以下のように構築される：

都市 γ	エッジリスト $M[\gamma]$
1	{2, 3, 5, 6}
2	{1, 3, 4, 6}
3	{1, 2, 4}
4	{2, 3, 5}
5	{1, 4, 6}
6	{1, 2, 5}

このエッジ写像 M を用いて，エッジ交叉は図 4 のように進められる．図 4 における交叉の結果，$\langle \alpha_1, \alpha_2, \ldots, \alpha_n\rangle$ が子染色体として得られる．図 4 のステップ 4 と 5 に確率的な選択があるため，GAP ではパターン（推定部分解）Π から得られる情報を用いてこれらの確率的選択における確率分布の修正を図る．

1. 親染色体のうち，一方から最初に訪問する都市 γ を選ぶ．$t := 1$ とし，また $\alpha_1 := \gamma$ とおく．
2. すべての $1 \leq \gamma' \leq n$ に対し，（可能なら）$M[\gamma']$ から γ を削除する．
3. もし $M[\gamma] \neq \emptyset$（別の都市へのエッジがある）ならば，ステップ 4 に進む; そうでなければステップ 5 に進む.
4. $M[\gamma]$ に含まれる都市の中でエッジリストのサイズが最小なものを選ぶ．もし複数あればランダム一つ選ぶ．選ばれた都市を γ とおき，ステップ 6 に進む.
5. もし $t = n$（未訪問の都市がない）ならば，子染色体 $\langle \alpha_1, \alpha_2, \ldots, \alpha_n\rangle$ を返す; そうでなければ，$\{1, 2, \ldots, n\} \setminus \{\alpha_1, \ldots, \alpha_t\}$ から未訪問の都市をランダムに選ぶ．選ばれた都市を γ とおき，ステップ 6 に進む.
6. $t := t + 1$ とした後，$\alpha_t := \gamma$ とする．ステップ 2 に進む.

図 4: エッジ交叉の手続き.

具体的には，まず頻出エッジ写像 (frequent edge map) と呼ばれる別の表 M^* を構築する．集団 Δ に含まれる染色体 $c = \langle \alpha_1, \alpha_2, \ldots, \alpha_n\rangle$ を考えたとき，Π 中のパターンの一部となっているエッジ集合は $E_c^* = \{(\alpha_i, \alpha_{i+1}) \mid (i, i+1) \in \mathrm{Occ}(\Pi, c)\}$ として得られる（$\mathrm{Occ}(\Pi, c)$ は式 1 で定義したもの）．E_c^* 中のエッジは c の頻出エッジと呼ばれる．そして都市 γ に対する頻出エッジのリストは $M^*[\gamma] = \bigcup_{c \in \Delta} \{\gamma' \mid (\gamma, \gamma') \in E_c^* \text{ or } (\gamma', \gamma) \in E_c^*\}$ と求められる．

更に M^* に基づき，ステップ 4 および 5 において次に訪問する都市の選択に関する確率分布を修正する．現在訪問中の都市を γ とし，$\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_N\}$ を次に訪問する都市の候補集合とする．また，確率 p_i で次の都市 γ_i を選択することにすると，$\{p_1, p_2, \ldots, p_N\}$ は確率分布を為す．元々のエッジ交叉では $p_i = 1/N$ と固定していたことに注意する．一方 GAP では，頻出エッジのリストに含まれる都市 $\gamma_i \in M^*[\gamma]$ に対し，その γ_i が「選ばれない」確率の割り引きを行う．すなわち，

$$p_i := 1 - \left(1 - \frac{1}{N}\right)\delta \quad (\gamma_i \in M^*[\gamma]) \tag{6}$$

$$p_i := 1 - \left(1 - \frac{1}{N}\right)\frac{N - h\delta}{N - h} \quad (\gamma_i \notin M^*[\gamma]) \tag{7}$$

が実行される．δ は割引率，また $h = |M^*[\gamma]|$ である．

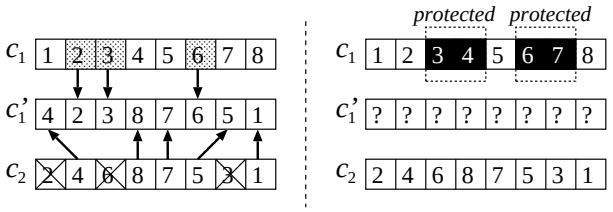

図 5: 位置に基づく交叉に対する部分解保護.

2.2.4 位置に基づく交叉

位置に基づく交叉は順列表現に対する別の交叉オペレータである [19]. 図 5（左）は親染色体 c_1, c_2 から位置に基づく交叉により子染色体 c_1' が生成される様子を示したものである（[19] の例を用いている）. 位置に基づく交叉では次のステップ (i)∼(iii) により子染色体を生成する. (i) まず片方の親 c_1 で幾つかの座位を選択する（図では網掛けになっている箇所）. (ii) 選択された都市は位置を変えずにそのまま子染色体にコピーされる. (iii) 残りの遺伝子はもう一方の親 c_2 から順序を変えずに子染色体にコピーされる. もう一つの子染色体は c_1 と c_2 の役割を変えてステップ (i)∼(iii) をもう一度繰り返すことにより生成される.

GAP では，パターン（推定部分解）Π からの情報に基づき，ステップ (i) における座位の選択確率にバイアスを与えることを考える. ここで，長さ n の親染色体 c_1 からおおよそ $(n \cdot r_{\mathrm{pos}})$ 箇所の座位を選択することを考える. r_{pos} は新たに導入される制御パラメータであり，本論文では位置保存率 (position-keeping rate) と呼ぶ. もし何もバイアスを与えないのであれば，座位 i を選択するかどうかを各々独立に確率 $p_i = r_{\mathrm{pos}}$ で決めればよい $(1 \leq i \leq n)$. それに対し，保護すべき座位の集合 $\mathrm{Occ}(\Pi, c_1)$ （式 1）を用いて，確率 p_i を

$$p_i := \kappa \cdot (1 - (1 - r_{\mathrm{pos}})\delta) \quad (i \in \mathrm{Occ}(\Pi, c_1)) \tag{8}$$

$$p_i := \kappa \cdot r_{\mathrm{pos}}\delta \qquad (i \notin \mathrm{Occ}(\Pi, c_1)) \tag{9}$$

と修正する. ここで κ は平均 $(n \cdot r_{\mathrm{pos}})$ 箇所の座位を選ぶように調整される定数である. 例えば，$\mathrm{Occ}(\Pi, c_1) = \{3, 4, 6, 7\}$ が得られたとき，図 5（右）で黒く塗られた座位の選択確率が大きくなる. 式 8 においては座位 i が選ばれない確率が割り引きされ，一方式 9 では座位 i が選ばれる確率が割り引きされる.

3 実験

本節で説明する比較実験では，Royal Road 問題と巡回セールスマン問題の 2 種類を取り上げる. Royal Road 問題ではビット表現の染色体を用い，巡回セールスマン問題では順列表現の染色体を用いる. Royal Road 問題は Mitchell らによって導入された [2]. GA の巡回セールスマン問題への適用に関しては，[19] で Larrañaga らが詳細な解説を行っている. また，[19] では標準的なベンチマークデータを用いて，様々な交叉・突然変異オペレータに対する比較実験の結果も詳細に報告されている. 両方の問題において，基本的に我々は標準的な GA（standard GA, 以下 SGA）と GAP を比較し，推定部分解の保護が探索性能を向上させるかどうかを観察する. 加えて Royal Road 問題に対しては，確率モデル構築に基づく GA として広く知られる BOA (Bayesian Optimization Algorithm) [10] とも比較する.

3.1 Royal Road 問題

Royal Road 問題は，染色体 c に対する適合度関数 $F(c) = \sum_{s \in S} w(s)\sigma(s, c)$ を最適化する問題である. ここで S はユーザ定義のスキーマの集合であり，$w(s)$ はスキーマ $s \in S$ に対する重みである. また $\sigma(s, c)$ はスキーマ s が染色体 c に出現すれば 1, そうでなければ 0 をとる.

Royal Road 問題を用いた比較実験では，“tight” な符号化に基づく適合度関数と “loose” な符号化に基づく適合度関数の 2 種類を用意する. これらの 2 種類の適合度関数は 15 種類のスキーマをもつ点が共通であり，スキーマ集合と重みは図 6 で与えられる. この図から，“tight” な符号化では隣り合うビットからスキーマが構成され，スキーマの破壊が起こりにくくなっている. それに対し，“loose” な符号化ではスキーマが染色体内に散在しており，交叉によるスキーマの破壊が起こりやすい.

比較実験においては，集団サイズ |Pop| を 64 から 512 まで変え，進化計算手法として SGA（一点交叉），SGA（二点交叉），GAP（一点交叉），GAP（二点交叉），BOA の 5 つを比較した. Mitchell らの実験では集団サイズは 128 に固定され，スケーリングを導入したルーレット選択が用いられている. それに対し，BOA では切捨て選択を用いるため，今回の比較実験では全ての手法で切捨て選択を用いている. 更に SGA と GAP では一様突然変異を用いる. 進化サイクルは最適値 256 を見つけるか，5,000 世代に達するまで繰り返される. また，全ての進化計算手法において，疑似乱数の種を 100 回変えて評価を行った.

ユーザ定義の制御パラメータの設定を表 1 に示す. 表 1 にあるように，幾つかの制御パラメータでは 2 つ以上の値を試しており，本論文では最も良い探索性能を示したパラメータを用いた結果を示す. GAP においては初期最小サポート $\max\{5, (0.2 \cdot r_{\mathrm{mine}}|\mathrm{Pop}|)\}$ を用いた top-K パターン発見手法を用いる. この初期最

$s_1 = 11111111**********; w(s_1) = 8$
$s_2 = ********11111111**********; w(s_2) = 8$
$s_3 = ****************11111111**********; w(s_3) = 8$
$s_4 = ************************11111111**********; w(s_4) = 8$
$s_5 = ********************************11111111**********; w(s_5) = 8$
$s_6 = **11111111**********; w(s_6) = 8$
$s_7 = **11111111********; w(s_7) = 8$
$s_8 = **11111111; w(s_8) = 8$
$s_9 = 1111111111111111**********; w(s_9) = 16$
$s_{10} = ****************1111111111111111**********; w(s_{10}) = 16$
$s_{11} = ********************************1111111111111111**********; w(s_{11}) = 16$
$s_{12} = **1111111111111111; w(s_{12}) = 16$
$s_{13} = 11111111111111111111111111111111**********; w(s_{13}) = 32$
$s_{14} = ********************************11111111111111111111111111111111; w(s_{14}) = 32$
$s_{15} = 1111111111111111111111111111111111111111111111111111111111111111; w(s_{15}) = 64$

$s_1 = 1******1******1******1******1******1******1******1******1******; w(s_1) = 8$
$s_2 = *1******1******1******1******1******1******1******1******1*****; w(s_2) = 8$
$s_2 = **1******1******1******1******1******1******1******1******1****; w(s_3) = 8$
$s_4 = ***1******1******1******1******1******1******1******1******1***; w(s_4) = 8$
$s_5 = ****1******1******1******1******1******1******1******1******1**; w(s_5) = 8$
$s_6 = *****1******1******1******1******1******1******1******1******1*; w(s_6) = 8$
$s_7 = ******1******1******1******1******1******1******1******1******1; w(s_7) = 8$
$s_8 = *******1******1******1******1******1******1******1******1*******; w(s_8) = 8$
$s_9 = 1***1***1***1***1***1***1***1***1***1***1***1***1***1***1***1***; w(s_9) = 16$
$s_{10} = *1***1***1***1***1***1***1***1***1***1***1***1***1***1***1***1**; w(s_{10}) = 16$
$s_{11} = **1***1***1***1***1***1***1***1***1***1***1***1***1***1***1***1*; w(s_{11}) = 16$
$s_{12} = ***1***1***1***1***1***1***1***1***1***1***1***1***1***1***1***1; w(s_{12}) = 16$
$s_{13} = 1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*; w(s_{13}) = 32$
$s_{14} = *1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1*1; w(s_{14}) = 32$
$s_{15} = 1111111111111111111111111111111111111111111111111111111111111111; w(s_{15}) = 64$

図 6: （上）“tight” な符号化を用いた Royal Road 問題,（下）“loose” な符号化を用いた Royal Road 問題.

表 1: Royal Road 問題に対する制御パラメータ.

手法	制御パラメータ	値
SGA	交叉確率	0.7
GAP	突然変異確率	0.01
	切捨て率 (r_{sel})	0.5
GAP	最大パターン数 (K)	30
	最小パターン長 ($L_{\min}$)	3
	抽出用切捨て率 (r_{mine})	0.05, 0.1, 0.2
	割引率 (δ)	0.1, 0.3, 0.5, 0.7, 0.9
	初期最小サポート ($\sigma^{(0)}_{\min}$)	$\max\{5, (0.2 \cdot r_{\mathrm{mine}}\|\mathrm{Pop}\|)\}$
BOA	切捨て率	0.01, 0.05, 0.1, 0.2
	親変数の最大個数	1, 2, 5
	疑似カウント (C)	0.01, 0.05, 0.1, 0.5, 1

小サポートの設定は，適合度が非常に高い集団 Δ_{mine}（図 1）のうち少なくとも 20%に出現したパターンのみを用いることを意味する．一方，集団サイズが小さい場合は，初期最小サポートは 5 に固定される．

BOA の制御パラメータも表 1 に示されている．この表の中で疑似カウント C は集団内の染色体数に関する事前知識を指しており，元論文 [10] で $m'(x_i, \pi_{X_i})$ として参照されていたものである．我々の実装では，疑似カウントを考慮して Bayesian ネットワークの条件付き確率を推定している[2]．更に BOA では，非循環性 (acyclicity) を満たすまでエッジの追加を最大 1,000 回試み，モデルスコア（BD metric）が向上するまでネットワーク構造の生成を最大 100 回行う．

表 2（上）は “tight” な符号化を用いた Royal Road 問題に対する実験結果である．ここで表中の各エントリは，最適値を見つけるまでの平均世代数およびその標準誤差であり，“1PTX” および “2PTX” はそれぞれ一点交叉，二点交叉を指す．この表の全体を通して二点交叉の方が一点交叉より優れていることが分かり，より大きい集団の方が探索性能が向上している．これは予想された結果であるが，その一方で，一点交叉を用いた場合に GAP の性能が SGA より劣化していることも分かる．先にも述べたように，一点交叉では推定部分解を過剰に保護する点に問題があると思われる．二点交叉を用いた場合は，全ての集団サイズで GAP の方が SGA より優れた性能を示している．BOA は着実な性能を示しているが，集団サイズの増加に対して性能の向上が鈍くなっている．

[2] より具体的には，我々は条件付き確率を

$$p(x_i \mid \pi_{X_i}) := \frac{m(x_i, \pi_{X_i}) + C}{m(\pi_{X_i}) + d_i C},$$

によって推定する．ここで x_i は Bayesian ネットワークにおける i 番目の変数 X_i がとる値であり，π_{X_i} は変数 X_i の親変数の値（ベクトル）である．また $m(x_i, \pi_{X_i})$ は，x_i および π_{X_i} を満たす染色体の数である．更に d_i は X_i が取り得る値の個数である．

表 2: Royal Road 問題における比較結果.

Tight	SGA		GAP		BOA
|Pop|	1PTX	2PTX	1PTX	2PTX	
64	684.6 ± 36.9	548.41 ± 35.6	624.5 ± 31.4	538.2 ± 27.5	384.9 ± 20.6
128	266.5 ± 16.0	207.5 ± 14.5	284.3 ± 18.8	186.0 ± 13.7	309.8 ± 14.3
256	89.5 ± 7.1	74.4 ± 6.7	132.3 ± 11.2	55.6 ± 4.7	155.8 ± 7.9
512	28.0 ± 3.5	28.9 ± 2.4	49.3 ± 4.9	23.7 ± 0.8	85.4 ± 3.2

Loose	SGA		GAP		BOA
|Pop|	1PTX	2PTX	1PTX	2PTX	
64	578.2 ± 26.3	537.3 ± 23.5	577.9 ± 27.5	541.7 ± 26.0	371.0 ± 21.6
128	294.4 ± 13.4	288.6 ± 11.7	325.4 ± 13.6	285.6 ± 12.0	312.2 ± 14.1
256	185.8 ± 7.1	164.4 ± 7.6	208.8 ± 6.5	151.8 ± 6.1	145.5 ± 5.6
512	110.4 ± 4.0	97.9 ± 3.1	135.6 ± 5.3	95.0 ± 3.7	78.8 ± 2.9

表 2（下）は “loose” な符号化を用いた Royal Road 問題に対する実験結果である．この表からは，まず “tight” な符号化を用いた場合と比べて SGA と GAP の性能が劣化していることが分かる．一点交叉や二点交叉ではその性能が符号化のスタイルに大きく依存するためであると考えられる．それに対し，BOA では元々全ての座位を区別なく扱うため，“loose” な符号化に対する性能の劣化が見られない．しかしそれでも，二点交叉の場合は，集団サイズが大きい場合において GAP は SGA より性能が高く，集団サイズ 128, 256 では BOA と同等の性能をもつ．集団サイズが小さい場合に GAP の性能が伸びないのは，部分解を正しく推定するために必要となる高適合な染色体の数が不足するためであると考えられる．

3.2 巡回セールスマン問題

巡回セールスマン問題を用いた比較実験では，エッジ交叉（2.2.3 節）と位置に基づく交叉（2.2.4 節）という 2 種類の交叉オペレータを用いる．そして進化計算手法を TSPLIB95[3] で提供される Grötschels48, Grötschels120 を使って比較する．Grötschels48 には 48 都市が含まれ，最適解は 5,046 である．一方 Grötschels120 には 120 都市が含まれ，最適解は 6,942 である．

この実験では，集団サイズを 200 から 2,000 まで変えて SGA（エッジ交叉），SGA（位置に基づく交叉），GAP（エッジ交叉），GAP（位置に基づく交叉）の 4 種類を比較する．SGA と GAP ともに切捨て選択を用い，突然変異手法として inverse mutation [19] を用いた．進化サイクルは，最適値が見つかるか，最後の 50 世代で最良個体の適合度に変化が見られなくなるまで繰り返す．実験に使用した制御パラメータを表 3 に示す．巡回セールスマン問題においては，局所的なパターンを部分解として考え，新たに最大ギャップ幅の制約を設けた．また，Royal Road 問題の場合と同様に，疑似乱数の種を 100 回変えて評価を行った．この実験では順列表現に対する確率モデル構築 GA 手法である EHBSA(Edge Histogram Based Sampling Algorithm) [11] との比較は行っていない．これは，現在の GAP が完全な世代交代を行っているのに対し，元論文の EHBSA では安定状態 (steady-state) 戦略をとっているためである．

表 3: 巡回セールスマン問題に対する制御パラメータ.

手法	制御パラメータ	値
SGA GAP	交叉確率	0.7
	突然変異確率	0.05
	切捨て率 (r_{sel})	0.5
	位置保存率 (r_{pos})	0.25 (for PX only)
GAP	最大パターン数 (K)	30
	最小パターン長 ($L_{\min}$)	2
	最大ギャップ幅 ($G_{\max}$)	0, 1, 2
	抽出用切捨て率 (r_{mine})	0.01, 0.05, 0.1
	割引率 (δ)	0.1, 0.3, 0.5, 0.7, 0.9
	初期最小サポート ($\sigma^{(0)}_{\min}$)	$\max\{5, (0.2 \cdot r_{\mathrm{mine}}\|\mathrm{Pop}\|)\}$

Grötschels48 に対する実験結果を表 4 に示す．表中の “ER” および “PX” はそれぞれエッジ交叉，位置に基づく交叉を指す．表 4（上）の各エントリは，100 回の試行における最良個体の適合度の平均である．一方，表 4（下）の各エントリは，進化サイクルが停止するまでに要した世代数の平均である（制御パラメータの設定は表 4（上）と同じ）．これらの表から，ほとんどの場合において，GAP が最良個体の適合度を向上させ，停止までに要する世代数を減らしていることが分かる．従ってこの実験結果より，GAP は進化を早めることに成功したと言える．停止までの世代数は適合度関数の評価回数に比例するため，GAP は適合度関数評価にかかるコストを軽減することが分かる．同様の傾向は表 5 に示す Grötschels120 の結果でも観察できる．

[3] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

表 4: Grötschels48 問題における比較結果.

Fitness	SGA		GAP	
|Pop|	ER	PX	ER	PX
200	5441.7 ± 14.3	5665.1 ± 31.6	5462.1 ± 16.0	5663.2 ± 32.2
500	5152.0 ± 5.5	5326.2 ± 21.8	5143.7 ± 5.0	5262.0 ± 14.4
1000	5111.2 ± 4.2	5224.7 ± 12.7	5103.9 ± 4.2	5223.8 ± 11.7
2000	5098.3 ± 2.9	5189.0 ± 9.4	5081.5 ± 2.4	5192.1 ± 8.9

Gens.	SGA		GAP	
|Pop|	ER	PX	ER	PX
200	157.8 ± 3.0	318.3 ± 25.7	154.1 ± 2.7	303.0 ± 21.1
500	165.1 ± 2.3	406.2 ± 45.9	163.8 ± 2.7	287.2 ± 12.6
1000	164.6 ± 1.5	358.4 ± 39.8	146.1 ± 2.2	422.5 ± 50.8
2000	166.1 ± 1.9	445.8 ± 55.0	135.8 ± 1.7	489.4 ± 60.9

表 5: Grötschels120 問題における比較結果.

Fitness	SGA		GAP	
|Pop|	ER	PX	ER	PX
200	9434.5 ± 34.8	9742.3 ± 69.1	9542.3 ± 38.5	9730.2 ± 85.4
500	8411.0 ± 29.1	8075.4 ± 34.2	8432.3 ± 23.1	8071.6 ± 33.6
1000	8258.1 ± 28.9	7733.1 ± 29.2	8254.1 ± 27.9	7694.2 ± 23.7
2000	8148.2 ± 34.8	7579.0 ± 21.1	8103.7 ± 21.6	7570.5 ± 21.9

Gens.	SGA		GAP	
|Pop|	ER	PX	ER	PX
200	557.0 ± 8.7	488.2 ± 10.9	544.3 ± 8.9	513.5 ± 21.2
500	874.7 ± 20.1	528.2 ± 17.8	856.9 ± 19.8	490.5 ± 9.2
1000	1039.9 ± 20.8	574.2 ± 31.1	928.9 ± 20.4	553.6 ± 22.9
2000	1275.0 ± 29.4	649.7 ± 40.0	1044.3 ± 18.8	580.5 ± 29.4

4 関連研究

遺伝的プログラミングでは部分解が座位に依存せず，染色体も不定サイズであることが一般的であるため，パターンに基づく部分解の抽出・保護を目的とした手法は遺伝的プログラミング分野で多くみられる（例えば [20, 21, 22, 23, 24]）．本論文はこれらの部分解抽出・保護手法が系列状の染色体にも適用可能であることを示している．GAP では可変長染色体を必要とする最適化問題も扱うことができる．また GA 研究においては，確率モデル構築 GA，摂動に基づく手法（例えば [8, 9]）で機械学習手法が導入されており，Sebag と Schoenauer は帰納学習に基づいて交叉を制御する手法を提案している [18]．それに対し，(我々の知る範囲において）GAP は頻出パターン発見手法を GA に適用した最初の試みとなっている．

5 おわりに

本論文では，GAP (GA with patterns) と呼ばれる，Gero と Kazakov の遺伝子工学アプローチを拡張した枠組みを提案した．GAP では系列パターン発見手法（BIDE および top-K パターン発見）を用い，高適合な染色体から部分解を推定し，交叉点分布を適宜修正することで，望ましくない交叉からの部分解の保護を細粒度（遺伝子単位）で行う．GAP では座位依存表現（ビット表現）と順列表現の両方を取り扱うことができ，実験結果から GAP によって進化を早めることで適合度関数の評価回数の軽減を図ることができる．

GAP には改良の余地がまだ多く残っている．頻出パターンを用いる代わりに顕在パターン (emerging pattern) [25] を導入することが考えられる．GA の文脈で考えたとき，高適合な染色体には多く現れるが，低適合な染色体にはそれほど多く現れないパターンを顕在パターンとして考えることができる．顕在パターンを利用することで，それ程重要ではないパターン（部分解）を除去することができ，より正確な部分解の推定が期待される．加えて，現在の GAP では各世代で部分解の抽出を行っているが，Sebag と Schoenauer が示唆するように [18]，部分解の抽出を数世代に一回行うようにして効率化を図るのが望ましいと考えられる．更に，本論文では実装を簡単にするため，GAP に系列パターン発見手法を用いてきた．しかしビット表現された染色体は座位-遺伝子のペアの集合と見なせるため，ビット表現をより効率的に扱うためには，高速化された飽和アイテム集合発見手法（例えば LCM [26]）を導入する必要がある．

参考文献

[1] D. E. Goldberg: “Genetic Algorithms: in Search, Optimization and Machine Learning”, Addison-Wesley (1989).

[2] M. Mitchell, S. Forrest and J. H. Holland: “The royal road for genetic algorithms: fitness landscapes and GA performance”, Proc. of the 1st European Conf. on Artificial Life (ECAL-92), pp. 245–254 (1992).

[3] M. Munetomo and D. E. Goldberg: “Linkage identification by non-monotonicity detection for overlapping functions”, Evolutionary Computation, **7**, 4, pp. 377–398 (1999).

[4] T.-L. Yu, D. E. Goldberg, K. Sastry, C. F. Lima and M. Pelikan: “Dependency structure matrix, genetic algorithms, and effective recombination”, Evolutionary Computation, **17**, 4, pp. 595–626 (2009).

[5] D. E. Goldberg, B. Korb and K. Deb: “Messy genetic algorithms: motivation, analysis, and first results”, Complex Systems, **3**, pp. 493–530 (1989).

[6] G. R. Harik and D. E. Goldberg: “Learning linkage”, IlliGAL Report No.96006, Dept. of General Engineering, University of Illinois at Urbana-Champaign (1996).

[7] M. Munetomo and D. E. Goldberg: "Identifying linkage by nonlinearity check", IlliGAL Report No.98012, Dept. of General Engineering, University of Illinois at Urbana-Champaign (1998).

[8] M. Tsuji, M. Munemoto and K. Akama: "Linkage identification by fitness difference clustering", Evolutionary Computation, **14**, 4, pp. 383–409 (2006).

[9] C.-Y. Chuang and Y.-p. Chen: "Linkage identification by perturbation and decision tree induction", Proc. of the 2007 IEEE Congress on Evolutionary Computation (CEC-07), pp. 357–363 (2007).

[10] M. Pelikan, D. E. Goldberg and E. Cantú-Paz: "Linkage problem, distribution estimation, and Bayesian networks", Evolutionary Computation, **8**, 3, pp. 311–340 (2000).

[11] S. Tsutsui: "Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram", Proc. of the 7th Int'l Conf. on Parallel Problem Solving from Nature (PPSN VII), pp. 224–233 (2002).

[12] J. S. Gero and V. A. Kazakov: "Evolving building blocks for genetic algorithms using genetic engineering", Proc. of the 1995 IEEE Int'l Conf. on Evolutionary Computation, pp. 340–345 (1995).

[13] J. Smith and T. C. Fogarty: "Recombination strategy adaptation via evolution of gene linkage", Proc. of the 1996 IEEE Int'l Conf. on Evolutionary Computation, pp. 826–831 (1996).

[14] M. Tsuji, M. Munemoto and K. Akama: "A crossover for complex building blocks overlapping", Proc. of the 2006 Genetic and Evolutioary Computation Conf. (GECCO-06), pp. 1337–1344 (2006).

[15] J. Wang and J. Han: "BIDE: Efficient mining of frequent closed sequences", Proc. of the 20th Int'l Conf. on Data Engineering (ICDE-04), pp. 79–90 (2004).

[16] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal and M.-C. Hsu: "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth", Proc. of the 17th Int'l Conf. on Data Engineering (ICDE-01), pp. 215–224 (2001).

[17] P. Tzvekov, X. Yan and J. Han: "TSP: Mining top-K closed sequential patterns", Proc. of the 3rd Int'l Conf. on Data Mining (ICDM-03), pp. 347–354 (2003).

[18] M. Sebag and M. Schoenauer: "Controlling crossover through inductive learning", Proc. of the 3rd Int'l Conf. on Parallel Problem Solving from Nature (PPSN III), pp. 209–218 (1994).

[19] P. Larrañaga, C. Kuijpers, R. Murga, I. Inza and S. Dizdarevich: "Genetic algorithms for the travelling salesman problem: A review of representations and operators", Artificial Intelligence Review, 13, pp. 129–170 (1999).

[20] P. J. Angeline and J. B. Pollack: "The evolutionary induction of subroutines", Proc. of the 14th Conf. of the Cognitive Science Society, pp. 236–241 (1992).

[21] W. A. Tackett: "Mining the genetic program", IEEE Expert, **10**, 3, pp. 28–38 (1995).

[22] J. P. Rosca and D. H. Ballard: "Discovery of subroutines in genetic programming", Advances in Genetic Programming II, MIT Press (1996).

[23] R. D. Howard and J. R. Koza: "Evolving modules in genetic programming by subtree encapsulation", Proc. of the 4th European Conf. on Genetic Programming (EuroGP 2001), pp. 160–175 (2001).

[24] Y. Kameya, J. Kumagai and Y. Kurata: "Accelerating genetic programming by frequent subtree mining", Proc. of the 2008 Genetic and Evolutioary Computation Conf. (GECCO-08), pp. 1203–1210 (2008).

[25] G. Dong and J. Li: "Efficient mining of emerging patterns: discovering trends and differences", Proc. of the 5th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, pp. 43–52 (1999).

[26] T. Uno, M. Kiyomi and H. Arimura: "LCM ver. 2: efficient mining algorithms for frequent/closed/maximal itemsets", Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementation (FIMI-04) (2004).