情報処理学会論文誌　Vol. 50　No. 10　2462–2470 (Oct. 2009)

シミュレーテッドアニーリングプログラミングにおける探索に有効な部分木とその活用方法 ——餌集め問題における検討

三木光範[†1]　上田祐一郎[†2]
廣安知之[†3]　松井勇樹[†2]

シミュレーテッドアニーリングを木構造が扱えるように拡張したシミュレーテッドアニーリングプログラミング（SAP）という自動プログラミング手法の改良を行う．従来の SAP における次状態生成では，ランダムに生成した部分木をランダムに選択した交換点に挿入する．そこで，探索に有効に働く部分木（有効部分木）を発見できれば，それを活用することで探索性能の向上が期待できる．本研究では，代表的な餌集め問題である Santa Fe trail 問題において，SAP の有効部分木を確認した．また，餌集め問題における SAP の有効部分木には問題に依存するものと依存しないものが存在することを確認した．そして，問題に依存しない有効部分木を終端記号として用いることで，有効部分木を活用することの有効性を示した．

Effective Subtrees in Simulated Annealing Programming for Artificial Ant Problems

Mitsunori Miki,[†1] Yuichiro Ueda,[†2]
Tomoyuki Hiroyasu[†3] and Yuki Matsui[†2]

Simulated Annealing Programming (SAP), an automatic programming method, is an extension method of Simulated Annealing (SA) that allows SA to handle tree structures. In this method, the point to exchange subtrees is chosen randomly, and a subtree to insert is also generated randomly. If we can discover some effective subtrees, the performance of SAP will be improved using those subtrees. In this research, we discovered some effective subtrees in Santa Fe trail problem. These subtrees can be classified into two kinds. One group is independent of problems, another is depending on specific problems. The effective subtrees which are independent of problems can be used as additional terminal nodes, and the performance of the proposed method is found to be very effective.

1. はじめに

ロボットの行動を制御するプログラムなどを，計算機を用いて自動設計する自動プログラミングの研究が注目されている．これは，あらかじめ人が想定できないような状況にも対応できるプログラムや，複数のロボットが協調動作するような複雑なプログラムを，容易に設計することができるためである．このような自動プログラミングの主な手法として，遺伝的プログラミング（Genetic Programming: GP）[1] が提案されている．

GP は代表的な自動プログラミング手法である．GP の研究は数多くされており，ADF[2] や頻出部分木発見手法[3]，ノードの使用頻度に基づく交叉[4] などによって，探索の効率化を実現している．しかし GP では，探索が進むにつれてプログラムのサイズが劇的に増大する「ブロート」が生じる．ブロートは探索の停滞や実行時間の増大につながり，GP の最大の問題点とされている[5]．

一方，ブロートを防いだ自動プログラミング手法の 1 つとして，著者らはシミュレーテッドアニーリングプログラミング（Simulated Annealing Programming: SAP）[6] を提案した．SAP は，ブロートの発生原因である交叉を用いないことで，ブロートが生じることなく GP と同等の性能が得られる[6]．しかし，従来の SAP における次状態の生成方法に関しては検討がされておらず，ランダムに生成した部分木をランダムに選択した交換点に挿入する方法がとられている．このため，探索が効率的に進まない可能性がある．

そこで著者らは，探索に有効に働く部分木（以降，有効部分木と呼ぶ）に注目した．SAP の探索において，もし有効部分木が存在すれば，この部分木を探索に活用することで，SAP の性能向上が期待できる．本研究では，SAP の探索における有効部分木がどのようなものかを確認し，その問題例依存性について検討を行った．そして，餌集め問題において，問題例に依存しない有効部分木の活用方法とその有効性を示した．なお，ここでの問題例とは，餌集め問題において，用いるフィールドが異なる問題を示すものとする．

†1 同志社大学理工学部
Faculty of Science and Engineering, Doshisha University
†2 同志社大学大学院工学研究科
Graduate School of Engineering, Doshisha University
†3 同志社大学生命医科学部
Faculty of Life and Medical Sciences, Doshisha University

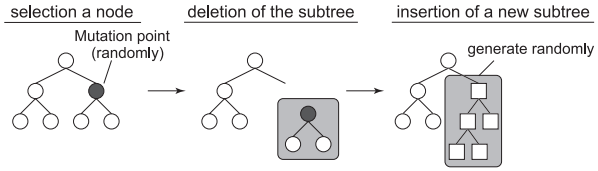


図 1　SAP の生成処理
Fig. 1　Generation method in SAP.

2. シミュレーテッドアニーリングプログラミング（SAP）

SAP は，SA を木構造が扱えるように拡張した手法である．以下にアルゴリズムを示す．

STEP 1　初期解の生成

初期解をランダムに生成し，その評価を行う．

STEP 2　生成処理

現在の解に対して GP の突然変異と同様の操作を行うことで新しい解候補を生成し，それを評価する．具体的には，**図 1** に示したように，現在の解に対してランダムに突然変異点（交換点）を選択し，その点を根とする部分木を削除する．その後，ランダムに部分木を生成し，削除した部分木に挿入する．

STEP 3　受理判定，状態遷移

現在の解の評価関数値 E と新しい解候補の評価関数値 E' との差分 ΔE（$= E' - E$），および温度 T を基に，新しい解候補を受理するかの判定（受理判定）を行う．受理判定には式 (1) に示す Metropolis 基準[7] を用いる．

$$P_{AC} = \begin{cases} 1 & \text{if } \Delta E \leq 0 \\ \exp\left(-\frac{\Delta E}{T}\right) & \text{otherwise} \end{cases} \tag{1}$$

STEP 4　クーリング

STEP 2 および 3 を一定期間繰り返した後，温度 T を小さくする．クーリング後の温度 T_{k+1} は，式 (2) によって決定する．

$$T_{k+1} = \gamma T_k \ (0.8 \leq \gamma < 1.0) \tag{2}$$

ここで，γ は冷却率であり，T_k は現在の温度である．

STEP 5　終了判定

STEP 2〜4 を定めた回数行えば，探索を終了する．

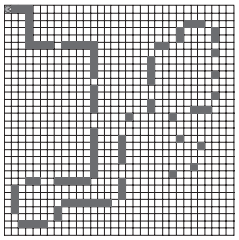


図 2　Santa Fe trail 問題
Fig. 2　Santa Fe trail problem.

3. Santa Fe trail 問題

本研究では，GP の代表的なベンチマーク問題である Santa Fe trail 問題[1] を対象とした．Santa Fe trail 問題とは，1 匹の人工蟻が**図 2** に示す 32×32 のマス目上に配置された餌を，限られたエネルギー内にできるだけ多く獲得するプログラムを生成する問題である[1]．なお，すべての餌を獲得すれば探索が成功したとする．非終端記号は {IfFoodAhead，Progn2，Progn3}，終端記号は {move，left，right} である．IfFoodAhead は引数を 2 つ持ち，人工蟻の 1 マス前方に餌があれば第 1 引数を，なければ第 2 引数を実行する．Progn$N$ は引数を N 個持ち，第 1 引数，第 2 引数，$\cdots$，第 N 引数の順に実行する．本問題は，IfFoodAhead の連鎖により実行されないノードが含まれる可能性がある問題，すなわち構文的イントロンが発生する問題である．また，評価関数 E_{val} は，餌の総数から獲得した餌の数 F を引いたものであり，0 を最適解とする最小化問題である．

4. 探索に有効に働く部分木（有効部分木）

4.1 概　　要

GP の探索では，解集団内に有効な積木構造（Building Blocks）が生成され，この積木構造が交叉によって組み合わされていくことで最適解に到達すると考えられている[1]．すなわち，GP の探索において有効に働く部分木構造が存在する．一方 SAP においても解を木構造として表現するため，GP と同様に探索に有効に働く部分木構造が存在すると考えられる．これを有効部分木と呼ぶ．特に SAP は交叉を用いず突然変異を基に探索を進めるため，有効部分木の存在や特徴を明らかにし，探索過程においてそれらに注目したメカニズムを取り入れることが，探索性能を向上させるうえで非常に重要となる．

(a) completed subtrees　　(b) uncompleted subtrees

図 3　対象とする部分木構造
Fig. 3　Target structure of subtrees.

本研究では，有効部分木の存在を確認するために，最適解に共通して出現する部分木（以降，最適解頻出部分木と呼ぶ）に注目する．これは，もし最適解頻出部分木が存在すれば，その部分木は最適解を得るために必要な部分木である可能性が高いと考えられ，最適解を探索するうえで有効部分木となることが期待できるためである．

4.2　最適解頻出部分木の確認

4.2.1　概　　要

前節で述べたように，本研究では有効部分木の有無を検討するために，最適解に共通して出現する最適解頻出部分木に注目する．なお，本研究において対象とした部分木は，**図 3** (a) に示すような子ノードがすべて付いた「完全な部分木（木構造の末端部に出現）」であり，図 3 (b) に示すような子ノードがすべて付いていない「不完全な部分木（木構造の中間部に出現）」は対象としていない．これは，不完全な部分木ではその動作が子ノードに依存するため，子ノードが 1 種類変化すると部分木の意味は大きく変化する．このことから，部分木に付く子ノードも含めた完全な部分木として検討する必要があるためである．

4.2.2　解 析 結 果

Santa Fe trail 問題に SAP を適用して得た最適解 100 個に対して，部分木の出現頻度解析を行った．ここで，**表 1** に本解析の結果として出現頻度が高かった部分木のうち，上位 10 種類を示す．表 1 より，高い頻度で最適解に出現している部分木が複数確認できることから，最適解頻出部分木が存在するといえる．すなわち，これらの最適解頻出部分木は最適解を得るためには必要であると考えられる．したがって，最適解頻出部分木が SAP の探索における有効部分木となることが期待できる．

4.3　最適解頻出部分木の活用と有効部分木

前節で述べたように，Santa Fe trail 問題における最適解頻出部分木は存在する．そして，この最適解頻出部分木が有効部分木となる可能性がある．すなわち，この最適解頻出部分木を探索中で活用することで，探索性能の向上が期待できる．そこで本研究では，この最適解頻出部分木を活用することで探索性能が向上するか検討を行った．

表 1　最適解頻出部分木（Santa Fe trail 問題）
Table 1　Ranking of subtrees (Santa Fe trail problem).

Appearance ratio	Subtrees
69/100	Progn2 - move - move
52/100	IfFoodAhead - move - left
35/100	IfFoodAhead - Progn2 - move - move - right
25/100	IfFoodAhead - Progn2 - move - move - left
22/100	Progn3 - move - move - right
20/100	Progn3 - right - left - left
17/100	Progn2 - move - right
17/100	IfFoodAhead - move - right
16/100	Progn3 - move - move - left
15/100	Progn2 - right - left

4.3.1　概　　要

本研究では最適解頻出部分木の活用方法として，最適解頻出部分木を終端記号に含めるという方法をとる．部分木を終端記号に含めるという活用方法は，ADF をはじめとする GP の多くの研究で用いられており[2),8),9)]，本研究でもそれらを参考にしている．ただし，具体的な含め方として，次の 2 種類を検討した．

- **カプセル化**
  部分木を 1 つのノードとして扱う．すなわち交換点を選択する際，この部分木に属するノードの選択を許さない．これにより，部分木は保護される．
- **非カプセル化**
  部分木として扱う．すなわち交換点を選択する際，この部分木に属するノードの選択を許す．これにより，部分木は保護されない．

4.3.2　比 較 結 果

最適解頻出部分木を活用することによって探索性能が向上するか，一般的な SAP と比較を行うことで検討した．なお，対象問題は Santa Fe trail 問題である．また，本検討において終端記号に含めた最適解頻出部分木は，表 1 に示す部分木の中から出現頻度が 30%を超す部分木 3 種類に，頻度順位 3 位の部分木とほぼ同等の意味を表す 4 番目の部分木を含む 4 種類とした．

この比較結果として，各手法を 100 試行行った際の成功率の履歴を**図 4** に示す．なお，

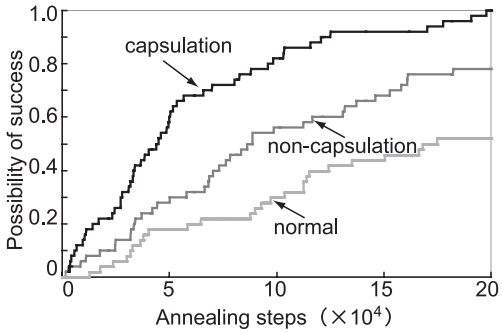


図 4　最適解頻出部分木の有効性
Fig. 4　Effectiveness of frequently-appearing subtrees.

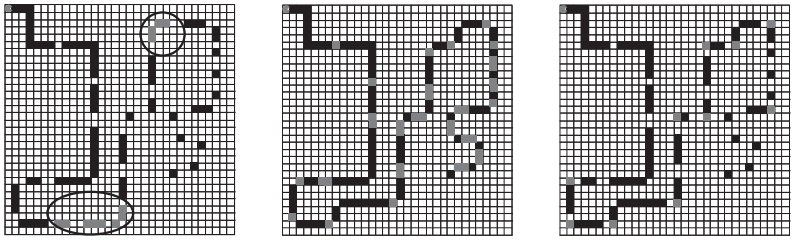
図 4 の横軸は探索回数，縦軸は成功率を示す．図 4 より，終端記号に最適解頻出部分木を含めると探索性能は向上することが分かる．さらに，この最適解頻出部分木の扱い方に関しては，保護しない方法より保護した方法の方が，より大きく性能向上することが分かる．すなわち，探索に有効に働く可能性が高い部分木は，カプセル化という方法で保護することが重要といえる．

以上より，最適解頻出部分木は SAP の探索における有効部分木といえる．そして，この有効部分木を保護（カプセル化）し，終端記号に加えることで，探索性能の向上を実現できるといえる．

5. 有効部分木の問題例依存性

前章で述べたように，SAP の探索における有効部分木が存在し，この有効部分木の活用方法としては，カプセル化して終端記号に含める方法が良いことを示した．しかし，前章では Santa Fe trail 問題における検討しかしていない．そのため，前章で用いた有効部分木が同種の問題に有効に働く部分木なのか，それとも Santa Fe trail 問題にのみ有効に働く部分木なのかは明確ではない．

もし，有効部分木が問題例に依存しないのであれば，あらかじめ発見した有効部分木をライブラリとして用意することで，他の同種の問題に活用することが可能となる．一方，有効部分木が問題例に依存するのであれば，各問題例で有効部分木を発見する方法を検討する必要がある．すなわち，有効部分木の問題例依存性を検討することは重要といえる．本章では，有効部分木の問題例依存性に注目し，その特徴を明らかにする．



(a) Field A　(b) Field B　(c) Field C

図 5　各フィールド
Fig. 5　Test fields.

表 2　フィールド変更による評価関数値の変化
Table 2　Change in the evaluation function value for change in the field.

Santa Fe trail problem	Field A	Field B	Field C
0	32	42	60
41	0	69	53
78	77	0	93
4	18	71	0

5.1 対象問題

本章では，有効部分木の問題例依存性について検討するため，**図 5** に示すような 3 種類のフィールドを対象とする．これらのフィールドは，Santa Fe trail 問題の餌の配置を少し変更したものであり，Santa Fe trail 問題との違いは以下のとおりである．

- **Field A**：ルートを少し変更したフィールド（円領域の部分）
- **Field B**：餌の間の空白を埋めたフィールド
- **Field C**：曲り角の法則を変更，つまり曲り角とその隣のマスで餌を交換したフィールド

なお，Santa Fe trail 問題のような餌集め問題では，餌のある配置における最適解が餌の配置を少し変更した場合に最適解になるとは限らず，評価関数値が著しく悪化する場合が多い．すなわち，餌の配置を少し変更するだけで，問題例の特徴は大きく異なるといえる．**表 2** に，Santa Fe trail 問題および図 5 の各問題例における最適解の 1 つを，他の各問題例に適用した場合の評価関数値を示す．これより，Santa Fe trail 問題の最適解は図 5 の各問

題例を解くことができない．一方，図 5 の各問題例の最適解は他の問題例を解くことができず，このことから，図 5 に示す各問題例の特徴は異なるといえる．このため，図 5 に示すような問題例を対象とすることで，有効部分木の問題例依存性を検討できると考えられる．

5.2 Santa Fe trail 問題における有効部分木の他の問題例への活用

有効部分木の問題例依存性を検討するために，4.3 節で述べた Santa Fe trail 問題に対する有効部分木（以降，S 型有効部分木と呼ぶ）4 種類を活用した SAP と一般的な SAP との比較を行う．なお，有効部分木の活用方法は，カプセル化して終端記号に含める方法とし，対象問題は図 5 に示す各フィールドとする．

図 6 に比較結果として，100 試行の成功率の履歴を示す．なお，図 6 は横軸に探索回数，縦軸に成功率を示す．

図 6 より，Field A のように一般的な SAP ではあまり性能が良くない問題例においても，Field B のように一般的な SAP でも性能が良い問題例においても，ともに S 型有効部分木を活用した SAP は一般的な SAP より高性能であった．しかし，Field C のように，S 型有効部分木を活用しても性能が向上しない問題例もあった．しかしどの問題例においても，S 型有効部分木を活用することで性能が悪化することはなかった．

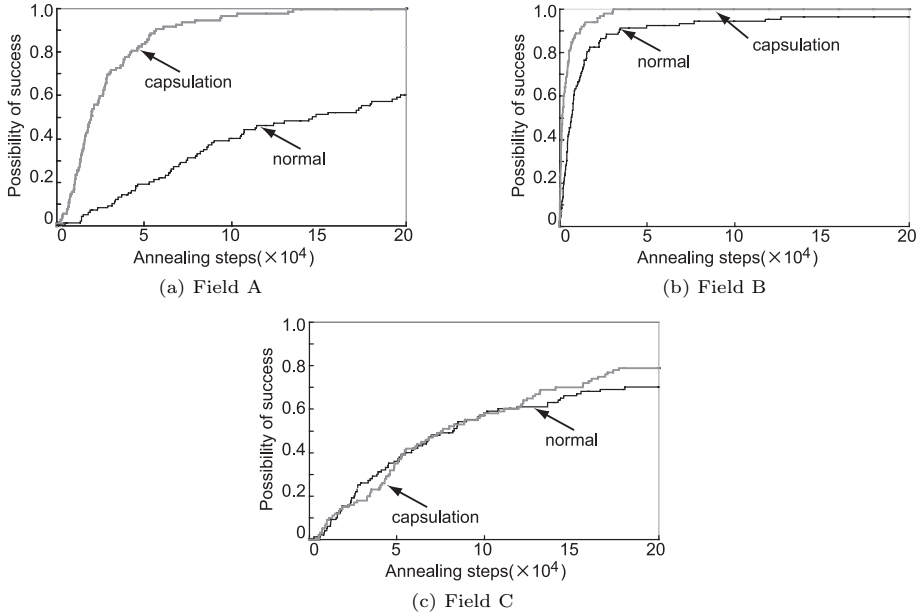


図 6　Santa Fe trail 問題における有効部分木の有効性
Fig. 6　Effectiveness of effective subtrees in Santa Fe trail problem.

5.3 問題例依存性

前節で述べたように，S 型有効部分木を活用した SAP は一般的な SAP と比較すると，どの問題例においても性能悪化はしなかった．しかし，S 型有効部分木が有効に働く問題例と有効に働かない問題例が存在した．そこで，S 型有効部分木が有効に働かなかった Field C に一般的な SAP を適用し，得た最適解 100 個に対して再度，部分木の出現頻度解析を行った．

ここで**表 3** に本解析の結果として，出現頻度が高かった部分木のうち上位 10 種類を示す．表 3 を表 1 と比較すると，{Progn2 - move - move} のように S 型有効部分木と共通したものも存在するが，{Progn2 - move - left} や {IfFoodAhead - Progn2 - move - left - left} のように異なるものも多く存在することが確認できる．すなわち，同じ餌集め問題においても，有効部分木は問題例に依存するものと依存しないものが存在すると考えられる．

そこで，本解析で対象とした問題例 4 種類の最適解 100 個に対して部分木の出現頻度解析を行い，4 種類の問題例すべてで共通して出現した部分木の一部，および 1 種類の問題例でのみ出現した部分木の一部を**表 4** に示す．

表 4 に示すように，上で述べた {Progn2 - move - move} をはじめ，多くの部分木が 4 種類すべての問題例で共通して出現した．なお，4 種類すべての問題例で共通して出現した部分木は 17 個あった．また，1 種類の問題例にしか出現しなかった部分木も多くあった．な

表 3　最適解頻出部分木（Field C）
Table 3　Ranking of subtrees (Field C).

Appearance ratio	Subtrees
82/100	Progn2 - move - move
52/100	Progn2 - move - left
49/100	IfFoodAhead - Progn2 - move - left - left
44/100	Progn2 - move - right
43/100	IfFoodAhead - Progn2 - move - right - right
41/100	IfFoodAhead - left - right
38/100	IfFoodAhead - Progn2 - move - move - right
37/100	IfFoodAhead - right - left
35/100	IfFoodAhead - Progn2 - move - move - left
12/100	IfFoodAhead - move - left

お，1 種類の問題例にしか出現しなかった部分木は 635 個あった．このことから有効部分木には，問題例に依存しない部分木と依存する部分木が存在すると考えられる．

また，表 3 に示す部分木の中から 4.3.2 項と同様に出現頻度が上位 4 種類の部分木（以降，C 型有効部分木と呼ぶ）を活用し，再度一般的な SAP と比較を行った．なお，有効部分木の活用方法は，カプセル化して終端記号に含める方法とし，対象問題は Field C とする．この結果として 100 試行の成功率の履歴を**図 7** に示す．なお，図 7 は横軸に探索回数，縦軸に成功率を示す．図 7 より，S 型有効部分木を活用しても性能が向上しなかった Field C においても，その問題例における有効部分木（C 型有効部分木）を活用することで探索性能が向上することを確認できた．

表 4　各フィールドにおける最適解頻出部分木
Table 4　Appearance of subtrees in each field.

Subtrees (in all fields)	Subtrees (in one fields)
Progn2 - move - move	IfFoodAhead - move - Progn3 - right - move - right
IfFoodAhead - move - left	Progn3 - right - move - right
IfFoodAhead - move - right	IfFoodAhead - move - Progn2 - left - move
IfFoodAhead - Progn2 - move - move - left	IfFoodAhead - Progn2 - move - right - move
IfFoodAhead - Progn2 - move - move - right	IfFoodAhead - right - IfFoodAhead - left - left
Progn2 - IfFoodAhead - move - right - move	Progn2 - IfFoodAhead - move - left - right
Progn3 - move - move - move	Progn3 - move - left - right
Progn2 - right - right	Progn3 - left - right - right
Progn2 - left - left	IfFoodAhead - move - Progn3 - left - left - move
IfFoodAhead - right - Progn2 - right - right	Progn3 - left - left - move

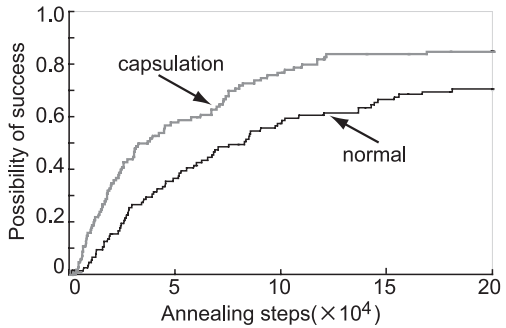


図 7　有効部分木の有効性
Fig. 7　Effectiveness of effective subtrees.

以上より，同じ餌集め問題を対象とした場合でも，SAP の探索における有効部分木は問題例に依存しないものと依存するものがともに存在するといえる．そして，各問題例に対して適切な有効部分木を活用することで，探索性能は向上するといえる．

問題例に依存しない有効部分木と問題例に依存する有効部分木の発見方法と活用方法については，以下の方法が良いと考えられる．なお本研究では，このうち，問題例に依存しない有効部分木について，次章以降で述べる．

- **問題例に依存しない有効部分木**
  これは同種の問題に有効に働くと考えられるため，各問題例で適切な有効部分木と見なせる．したがって，ライブラリとしてあらかじめ用意して活用すればよい．なおこの部分木は，あらかじめ簡単な問題例を複数解くことで発見できると考えられるが，この詳細については次章で述べる．
- **問題例に依存する有効部分木**
  これは各問題例で適切なものが異なるため，探索過程で発見しながら活用すればよい．

6. 問題例に依存しない有効部分木

6.1 問題例に依存しない有効部分木の抽出方法

前章で述べたとおり，問題例に依存しない有効部分木に関しては，あらかじめライブラリとして用意することが可能であると考えられる．そして，このライブラリを作成するために本研究では，簡単な問題例を複数解き，各問題例における最適解で共通して出現する部分木に注目する．ここでいう簡単な問題例とは，Santa Fe trail 問題を細かく分けたような問題例を意味する．すなわち，**図 8** に示すような，直線や曲り角，餌の間に空白が存在するもの，などである．こういった簡単な問題例における最適解に注目することで，問題例に依存しない有効部分木を発見できると考える理由は，以下の 3 点によるものである．

- Santa Fe trail 問題のような餌集め問題では，餌の配置が少し変わるだけで，問題例の特徴は大きく異なる．このため，並びの違う簡単な問題例を複数解くことは，様々な特

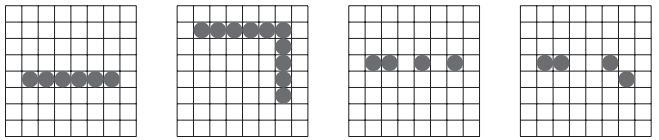


図 8　簡単なフィールド例
Fig. 8　Simple fields.

徴の問題例を解くことを意味する．

- 同種の複雑な問題例は，直線や曲り角などの簡単な問題例の組合せと考えることができる．すなわち，これら簡単な問題例における最適解に共通して出現する部分木は，同種の問題で有効に働くと考えることができる．
- 問題例が簡単なため，非常に少ない探索回数で各問題例における最適解を得られる．

6.2 問題例に依存しない有効部分木の抽出

ここで，簡単な問題例 15 種類を一般的な SAP で解き，それぞれで得た 100 個の最適解に対して部分木の出現頻度解析を行った．そのうえで，各問題例で共通して出現した部分木のうち，上位 10 種類を**表 5** に示す．

表 5 に示すように，有効部分木の多くが「前方に餌がある場合，前進する」という直感的に有効と考えられる部分木であるのに対し，「前方に餌がある場合，前進しない」，「右（左）を向いた後，左（右）を向く」などの直感的に有効でないと判断できる部分木は含まれていないことが分かる．このことから，簡単な問題例を複数解くことにより，問題構造から有効と考えられる部分木を抽出することができるといえる．一方，表 5 において，「前進する，前進する」や「前進する，前進する，前進する」のように直感では有効か判断できない部分木が有効部分木として抽出されていることから，有効部分木の抽出に直感だけを用いることはできないといえる．

また表 5 より，プログラムサイズが小さい単純な部分木が，多くの簡単な問題例における最適解で共通して出現していることが確認できる．特に，上位 3 つの部分木に関しては，半数を超える簡単な問題例における最適解で共通して出現している．したがって，このようなプログラムサイズの小さい単純な部分木が，同種の問題で有効に働く，問題例に依存しない有効部分木と考えられる．

表 5　問題例に依存しない有効部分木
Table 5　Effective subtrees independent of problems.

Appearance ratio (in 15 problems)	Subtrees
12/15	IfFoodAhead - move - right
11/15	Progn2 - move - move
9/15	IfFoodAhead - move - left
6/15	IfFoodAhead - move - move
5/15	IfFoodAhead - Progn2 - move - move - right
4/15	Progn2 - move - IfFoodAhead - move - right
4/15	Progn3 - move - move - move
3/15	Progn2 - move - IfFoodAhead - move - left
3/15	Progn2 - move - right
3/15	IfFoodAhead - move - Progn2 - move - right

7. 数 値 実 験

7.1 概　　要

前章で述べたように，表 5 に示した部分木のうち特に上位 3 種類の部分木が問題例に依存しない有効部分木であると考えられる．この真偽を検討するために，これら 3 種類の部分木を活用した SAP と通常の SAP との性能比較を行う．なお，活用方法はカプセル化して終端記号に含める方法とし，対象問題は Santa Fe trail 問題および，図 5 に示す問題の計 4 種類の問題例とする．またパラメータは，探索回数 20 万回，温度は 4 で固定する[6]．

7.2 結　　果

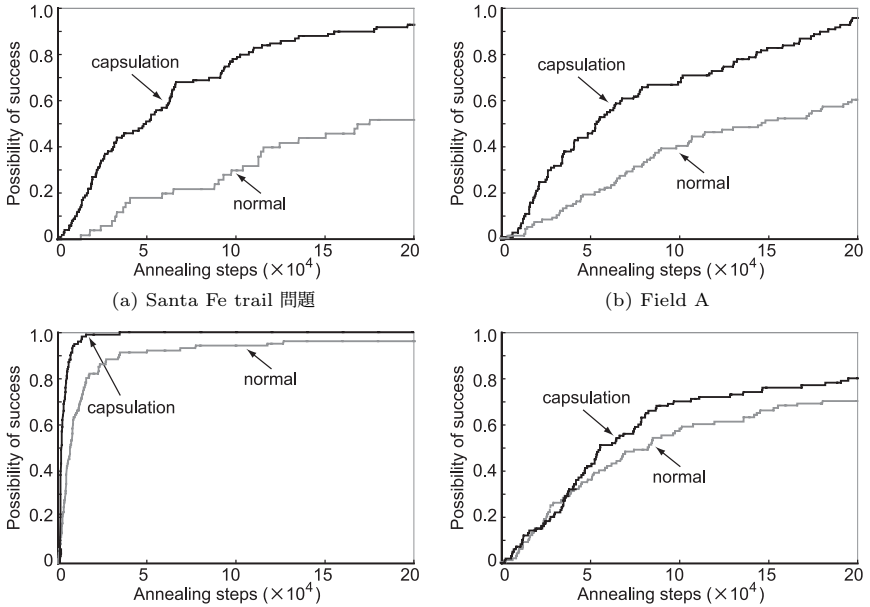
比較結果として，**図 9** に各手法の成功率の履歴を示す．なお，図 9 は横軸に探索回数，縦軸に成功率を示す．

図 9 より，すべての問題例において，問題例に依存しない有効部分木と考えられる部分木 3 種類を活用した SAP の方が通常の SAP よりも高い探索性能を得られることが確認できる．このことから，問題例に依存しない有効部分木が存在するということ，さらには，簡単な問題例における最適解に共通して出現する部分木が，問題例に依存しない有効部分木となることがいえる．そして，問題例に依存しない有効部分木は，本実験で活用したようなプログラムサイズの小さい単純な部分木であるといえる．

8. 考　　察

前章より，簡単な問題例における最適解に共通して出現する部分木を数個活用することで，探索性能を向上させることが可能なことを示した．また，これらの部分木はどれもプログラムサイズが 3 程度の非常に小さな部分木であった．しかし，プログラムサイズが 3 程度の小さな部分木であれば，どのような部分木を活用しても有効であるのか，それとも前章で用いたような，厳選した数種類の部分木を活用することが有効であるのか，といった疑問が残る．

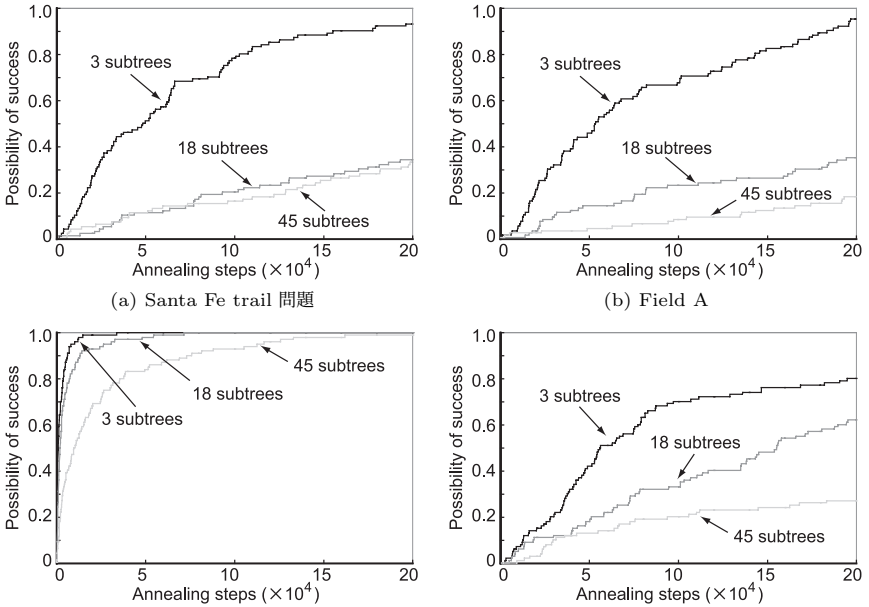
ここでは，プログラムサイズが 3 の部分木すべて（18 種類），および深さが 2 の部分木すべて（45 種類）を活用した SAP との比較を行う．なお，活用方法はカプセル化して終端記号に含める方法とし，対象問題は Santa Fe trail 問題および，図 5 に示す問題の計 4 種類の問題例とする．



(a) Santa Fe trail 問題　(b) Field A

(c) Field B　(d) Field C

図 9　問題例に依存しない有効部分木の活用結果
Fig. 9　Effectiveness of effective subtrees independent of problems.



(a) Santa Fe trail 問題　(b) Field A

(c) Field B　(d) Field C

図 10　問題例に依存しない有効部分木の活用結果
Fig. 10　Effectiveness of effective subtrees independent of problems.

比較結果として，**図 10** に各手法の成功率の履歴を示す．なお，図 10 は横軸に探索回数，縦軸に成功率を示す．

図 10 より，すべての問題例において，厳選した 3 種類の部分木を活用した SAP の方が高い探索性能であることが確認できる．すなわち，プログラムサイズが 3 程度の小さい部分木ならどのような部分木でも有効に働くというわけではなく，活用する部分木は厳選する必要があるといえる．この厳選の方法としては，簡単な問題例における最適解に共通して出現する部分木に注目し，その中から数種類を選択すればよい．これはいいかえると，問題例に依存しない有効部分木は，簡単な問題例における最適解に共通して出現する簡単な部分木であり，この簡単な部分木を数種類活用するだけで，探索性能を向上できるといえる．以上より，餌集め問題における問題例に依存しない有効部分木とその活用による有効性を示すことができた．

9. ま と め

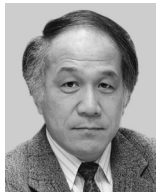
本研究では，SAP の探索における有効部分木を確認し，問題例に依存しない有効部分木と問題例に依存する有効部分木が存在することを確認した．そして，餌集め問題において，問題例に依存しない有効部分木を，簡単な問題例を複数解くことで発見し，これを活用することの有効性を示した．問題例に依存しない有効部分木は，プログラムサイズの小さい単純なものであるが，これを数個活用することで，SAP の探索性能は大きく向上する．これらの部分木をあらかじめライブラリとして用意することで，用いる終端，および非終端記号が同じ問題に対しては，探索性能を向上させることが可能であるといえる．また，異なる行動規則を持つ問題に対しても，ここで述べたように「発見した有効部分木を，あらかじめライブラリとして用意し，探索に活用する」という方法が探索性能を向上させることがで

きるといえるだろう．なお，この活用方法はSAP独自のアルゴリズムに依存するものではない．このため，SAPのみでなくGPや他の自動プログラミング手法にも応用可能であり，その有効性が期待できるだろう．

参 考 文 献

1) Koza, J.R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press (1992).
2) Koza, J.R.: *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press (1994).
3) Asai, T., Abe, K., Kawasoe, S., Arimura, H. and Arikawa, S.: Efficient Substructure Discovery from Large Semistructured Data, *Proc. 2nd SIAM Intl. Conf. on Data Mining*, pp.158–174 (2002).
4) Katagami, D. and Yamada, S.: Speedup of Evolutionary Behavior Learning with Crossover Depending on the Usage Frequency of a Node, *IEEE International Conference on Systems, Man, and Cybernetics*, Vol.5, pp.601–606 (1999).
5) 伊庭斉志：遺伝的プログラミング入門，東京大学出版会 (2001).
6) 藤田佳久，三木光範，橋本雅文，廣安知之：シミュレーテッドアニーリングを用いた自動プログラミング，情報処理学会論文誌，Vol.48, pp.88–102 (2007).
7) Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E.: Equation of State Calculation by Fast Computing Machines, *Journal of Chemical Physics*, Vol.21, pp.1087–1092 (1953).
8) Roberts, S.C., Howard, D. and Koza, J.R.: Evolving Modules in Genetic Programming by Subtree Encapsulation, *Lecture Notes in Computer Science*, Vol.2038, pp.160–175 (2001).
9) Rosca, J.P. and Ballard, D.H.: Hierarchical Self-Organization in Genetic Programming, *Proc. 11th International Conference on Machine Learning*, pp.251–258 (1994).

(平成21年1月6日受付)
(平成21年7月2日採録)



三木　光範（正会員）

1950年生．1978年大阪市立大学大学院工学研究科博士課程修了，工学博士．大阪府立大学工学部航空宇宙工学科助教授等を経て，1994年同志社大学工学部教授．進化的計算手法とその並列化，および知的なシステムの設計に関する研究に従事．著書は『工学問題を解決する適応化・知能化・最適化法』（技法堂出版）等多数．IEEE，人工知能学会等各会員．知的オフィス環境コンソーシアム会長．

上田祐一郎（学生会員）

1984年生．2007年同志社大学工学部知識工学科卒業．同年同志社大学大学院工学研究科修士課程入学．シミュレーテッドアニーリング等の研究に従事．

廣安　知之（正会員）

1997年早稲田大学大学院理工学研究科後期博士課程修了．同志社大学工学部准教授を経て2008年同志社大学生命医科学部教授．創発的計算，最適設計，並列処理等の研究に従事．IEEE，電子情報通信学会，計測自動制御学会，日本機械学会，超並列計算研究会，日本計算工学会各会員．

松井　勇樹

1985年生．2008年同志社大学工学部知識工学科卒業．同年同志社大学大学院工学研究科修士課程入学．シミュレーテッドアニーリングプログラミングの研究に従事．