計算過程のグラフコーディングにおける実行可能性判定について

5K－10

内藤 昭三, 山本 公洋, 伊藤 正樹
NTT ソフトウェア研究所

1 はじめに

逆導出原理と遺伝的アルゴリズムを用いた規則集合獲得手法 GA-CIGOL を提案した [1]. そこでは, 逆導出の計算過程をグラフコーディングし, 遺伝的アルゴリズムの枠組に基づき, 交差・突然変異の操作を行なうことにより新たな逆導出計算過程に対応するグラフを生成した. この場合, 新たに生成されるグラフコーディングされた逆導出計算過程が, 全域的に実行可能である保証はない. 実行可能でない逆導出演算に対応するグラフノードは致死であるとして, そのようなグラフは棄却するなり, 実行可能になるように修正するなりの対応が必要になる. 本報告では, グラフコーディングの実行可能性の判定アルゴリズムを提案し, その複雑性を考察する. アルゴリズムの並列化についても考察する.

2 GA-CIGOL におけるコーディング

逆導出の適用過程を有向グラフ (染色体に相当) にコーディングする. ノード (遺伝子に相当) が, 一回の逆導出演算 (およびその演算結果) に対応し, アークが逆導出演算への入出力関係に対応する. 入力アークを持たないノードは, 初期データとして与えられる事例集合中の一事例に対応する.

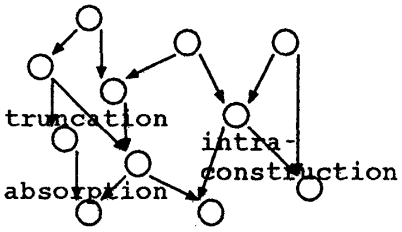

図 1: コーディング

遺伝操作には, 交叉と突然変異の 2 種類を使用する.

交叉では, 2 つのグラフをそれぞれ 2 分割し, 分割グラフの片側を交換して, 新しいグラフを構成する (図2参照). また, 突然変異では, 任意にノードおよびアークの生成・削除を行なう (図 3参照). 交叉の遺伝操作だけからは, 交叉される前の 2 つの部分グラフ間にアークが生じないので, 非連結なグラフを生じる. 突然変異を組み合わせることにより, 部分グラフ間のアークを生成してやることが有効であると考えられる.

[1]Complexity on Deciding whether Graph Coding of Computation is Executable
Shozo NAITO, Kimihiro YAMAMOTO and Masaki ITOH
NTT Software Laboratories

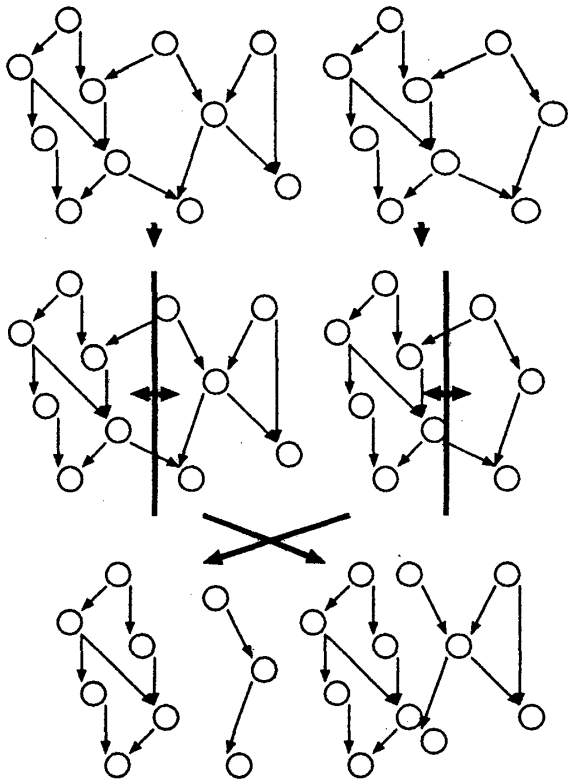

図 2: 交叉

交叉と突然変異の遺伝操作を行なうと, 実行可能でないノード (致死遺伝子に相当) を含むグラフが発生する可能性がある. このような実行可能でないノードを含むグラフに対しては, 実行可能な部分だけの結果を用いる, 棄却し次世代に残さない, 実行不可能な部分を削除したり修正することにより全域実行可能なグラフに変更する, などの対応策が必要である. GA-CIGOL (version2) では, 最後の方法を用いている [1].

3 実行可能性の判定

グラフが全域実行可能であるためには以下の 2 条件が必要十分である.

1. in-degree を持たないノード (入力ノード) が全て初期データとして与えられる事例集合の要素であること
2. グラフが Directed Acyclic Graph (DAG) であること

この条件に基づく判定アルゴリズムを図 4に示す. このアルゴリズムは, DAG のノードに対するトポロジ

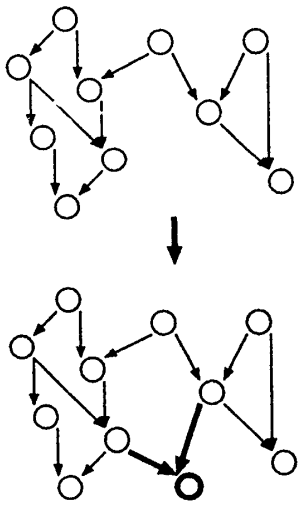

図 3: 突然変異

カルソートを与えるアルゴリズム [2] を使用している [2]. ただし, 全域実行可能でない場合への対応策を考慮して, いくつかのパラメータを導入した. NI は, 初期データとして与えられる事例に対応していないインプットノードの集合である. また, R はインプットノードが事例に対応していると仮定した場合に, 実行可能となるノードからなる部分グラフを構成するノードである. NI を実例に変更し, R からなる部分グラフを構成することにより, 全域実行可能なグラフに修正することができる.

アークを探索することにより, 全てのノードの in-degree c(i) を線形時間で計算できる. また, for loop では, ノード i ∈ L に対して, アーク (i,j) を一回たどるだけであり, トータルの計算量は, たかだか $O(n+e)$ である. ここで, n および e は, それぞれ ノード数, および アーク数である.

次に, 並列アルゴリズムについて考察する [3]. 有効グラフがループを持つか否かは, 推移閉包を求めることにより判定できるので, CRCW PRAM 上で, $O(logn)$ 時間, $O(n^3logn)$ 回の演算により判定可能である (Corollary 5.2 in [4]). ただし, これは図 4 のアルゴリズムの直接の並列化ではない.

4 むすび

グラフコーディングした計算過程の実行可能性を判定するアルゴリズムを与えた. このコーディングは, 逆導出原理と遺伝的アルゴリズムを用いた規則集合獲得手法 GA-CIGOL で使われている [1]. そこでは, 致死が生じたときには, 致死部分を削除し, 全域実行可能な部分グラフを構成する修正を行なった.

しかしながら, 致死遺伝子は無意味 (no operation) なだけで, コーディング長を冗長にする程度のマイナス要因でしかない (ヒトゲノムでも, なんの意味も持たないような塩基配列部イントロンがある). このような部分も, 遺伝操作により意味を持つようになる場合もある. このような可能性を残しておくために, 致死遺伝子を含む一見冗長なコーディングが有効かもしれない.

また, ここでは実行可能性に DAG という強い条件を含めたが, ループを含むような一般のプログラムでもループ部分に初期データが与えられれば, 実行可能であるとみなすこともできる. ループを含むデータフロープログラムの実行可能性検証に対する拡張なども興味深い.

```
W:=all nodes; グラフの全ノード
R:=empty ; 実行可能ノードの集合
L:=L'=empty ; 新たに実行可能となったノード
I:=empty ; in-degree が 0 のノード
NI:=empty ; 実例でないインプットノード
b:='undecided'
for each node i ∈ W do
    c(i):= in-degree of node i
L:=I:={i ∈ W|c(i) = 0}
for each i ∈ W do
    if i is not a given example node
      then append i to NI
if NI is not empty then
    b:='some inputs not example'
while L is not empty do
begin
    R:=R ∪ L;
    for each node i ∈ L do
      for each arc (i,j) do
          begin
              c(j):=c(j)-1;
              if c(j)=0 then append j to L'
          end;
    L:=L';
    L':=empty
end;
if  b:='undecided' then
    begin
    if R=W then b:='globally executable'
      else b:='partially executable'
    end;
return b.
```

図 4: アルゴリズム

参考文献

[1] 山本公洋, 鈴木英明, 内藤昭三, 伊藤正樹: 逆導出原理と遺伝的アルゴリズムを用いた規則集合獲得手法 GA-CIGOL, 情報処理学会論文誌, vol.36, no.9, 1995 (掲載予定).

[2] J. van Leeuwen: Graph Algorithms, in J. van Leeuwen (ed) Handbook of Thoretical Computer Science, Chapter10, MIT Press, 1990.

[3] Alan Gibbons and Wojciech Rytter: Efficient Parallel Algorithms, Cambridge University Press, 1988.

[4] Joseph JaJa: An Introduction to Parallel Algorithms, Addison-Wesley, 1992.