プログラムテストに用いるパスジェネレータの自動生成について

柳沢隆夫
芝浦工業大学　工業経営学科

本論文は、プログラムの自動的なテストパス発生に於いて生じる　2つの問題のためのアルゴリズムを考慮している。
これらの問題は、有向グラフの最も必要とされる辺を含むパスを決定することと、有向グラフの指定された辺を通る最少のパス集合を決定することである。

On automatic generation of the path generater for program testing

Takao Yanagisawa
Depertment of Industrial managiment, Shibaura Institute of Technology.
Oomiya-shi, Saitama-ken, Japan

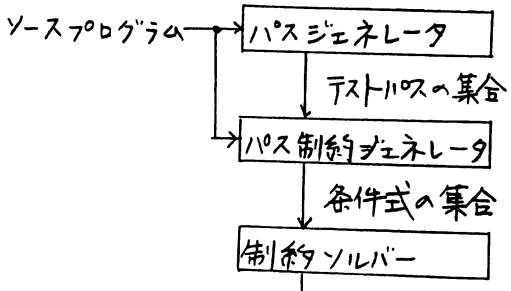
In this paper we consider algorithm for tow problems that arise in dutomatic test path generation for program: the problem of determining a path which contain the most necessitated edges of a directed graph and the problem of determining a minimal set of path which is through d specified edges of a directed graph.

1. はじめに

プログラムテストは、プログラムの信頼性を高めるために行われる。本研究は、プログラムテストに用いるテストパス導出に関する次の2つの問題（問題A，問題B）の解法を行っている。

1.1 問題Aについて

プログラムの全ての経路集合を求めて、そして、この経路を通るテストデータを算出して行うテスト法（図．1）



図．1

は、普通のプログラムでも、そのような経路の数は膨大となり、完全なテストは実行不可能となることがあるため全ての経路集合の部分集合を求めて、テストを行う方法が考えられている。この部分集合の選び方にある基準に合致するものを選ぶものがあり、基準にも色々なものがある[1]。

ところで、上記の部分集合の選び方として、プログラムの重要性（使用頻度の大，小など）などにより、テストの回数を決めて行うものや、あるいは数回のテストの後、もう1度だけテストをしたいようなことがある。しかしこの問題に注目した研究は余り見当らない。ここでのテストは重要と思われる部分を1番多く含むテスト経路を導出して、効果的にテストすることが必要と思われる。

そこで本研究は、あと1度だけテストを行うことを対象に、テスト上重要と思われる部分（ブランチ，命令）を最大に含む、S－Tパス（スタートからエンドまでの経路）を求める問題を扱い、又実用上、有用と思われるので、上記の内で、テストが必要とされない部分（ブランチ，命令）は最少に含むテストパスを求める解法について述べる。

1.2 問題Bについて

全ての経路集合の部分集合を選ぶ基準として、プログラムの各々のブランチを少くとも1度は実行するテスト経路集合を選ぶものがある。

このテスト経路の導出法については[3][4]いくつかの研究が報告されている。しかし、この研究はプログラムに有向サイクルが含まれない場合に限定して行っている。

一方、有向サイクルが含まれることを許したプログラムグラフに対する最少パス集合の導出についても、これまで若干行われている[5]（Prather は、正則式を用いて導出した。この方法は簡潔で、計算時間が少くて解けるが、プログラムが構造化されていない場合適用出来ない）

そこで、本研究は 有向サイクルが含まれることを許したプログラムグラフ（構造化されていないものも含む）の各有向辺を含む最少のパス集合を、グラフ理論を応用して求める問題を扱い、又有用性を増すために、上記のパス集合の重複部分を最少にしたパス集合（最短のパス集合）を求める解法について述べる。

ところで、プログラムテスト法として、先ず プログラムグラフのテストパスを決定し、次に、これを通る入力データを導出して行うものは、テストパスが不実行パスであったときは破局となる。

この危険性を柔らげるものとして、先ず、プログラムのスタートと各ブラ

ンチの直ぐ後に、カウンターを挿入し、いくつかの入力データをランダムに入力してテストを行い、次に、残りのカウンター値が0を示している未だ未テストの部分にテスターを振り向けるテスト法が考えられている。

しかし、このテスト法を具体的に定めた研究は余り見受けられない。ここでのテストは、最少回のテストで全ての未テスト辺を実行するテスト経路を導出して、効果的にテストすることが考えられる。

そこで、本研究は、未テストブランチを全て含み、かつ最少のS－T経路集合を求める問題を扱い、又実用上、有用と思われるので、上記の内で、即にテストされているブランチあるいは命令を最少に含むテストパスの導出解法について述べる。

2. 問題Aの解法

2.1 問題の定式化

テスト上重要と思われる部分（ブランチ，命令）を最大に含むS－T経路であり、かつ、上記の内で、テストが必要とされない部分を最少に含むS－T経路を求める問題は、プログラムを有向グラフ表現（スタート，エンド，各命令、各ブランチを節とし、その間に直接の制御があるときは有向辺を設けることによりグラフを構成する。）し、テスト上重要と思われる部分に対応したこのグラフの部分を最大に含みかつ、必ずしも必要とされない部分を最少に含むS－Tパス（同一有向サイクルを2度通らないものとする）を求める問題となる。

2.2 有向サイクルを含まないプログラムグラフの場合の解法

重要と思われるブランチ，あるいは命令に対応した節より出る有向辺(e_i)に正の重み付けをし、テストの必要とされないブランチあるいは命令より出る有向辺に負の重み付けをしたグラフに於いて、Sより下への最長パスを求めることにより、問題Aの解は導出される。　これにより求められる理由は、Sから下へのパスの中で、最長のものの中に　テスト上一番重要と思われる部分が含まれていることと、テスト上1番重要とされる部分を含んでいるもののの中で、テストの必要とされない部分を1番少く含むものが最長のパスとなるからである。

2.3 有向サイクルを含むのを許したプログラムグラフの場合の解法

2－2で述べた最長パスを求めることにより求める方法は、有向サイクルを廻る度に長さが増大する場合があり、ここでは適用出来ない。次に、いくつかの導出法を述べるが、それぞれの方法の効率性と適用条件についても述べて行く。

先ず、方1案として、既存のアルゴリズムを直接的に適用して導出するもの（図．2）が考えられる。

しかし、この方法は、ハミルトンパスの最短なものを求めるアルゴリズムを含んでおり、強連結成分内のe_iの数が多いとき（10以上のとき）、余り効率のよい方法は見い出されていない又、プログラムが構造化されていないときは、入口と出口の組合せの対によりe_0を含む最短なパスの長さが異るため、入口と出口の組合せによりその手続きを行って、その結果導出されたパスを埋め込んだグラフ（図．3）に対して、最長パスを求める解法を用いなければならない。このため、この方法は強連結成分内のe_0の数が少ないときに実用上は有用となる。

ところで、もし強連結成分内にe_iが直列に連続して生じている場合は、プログラムが構造化されているときに限り、図．3の方法のその部分を、図4にかえることにより、ハミルトンパ

強連結成分を検出する

↓

e_i を含む各々の強連結成分について、次のことを行う。 ＊

1). 強連結成分の入口と出口並にその成分に含まれる e_i を節とし、その到達関係を辺とするグラフを構成し、辺に最短距離を重み付ける

2). 入口節から出口節へのハミルトンパスの最少のものを求める。

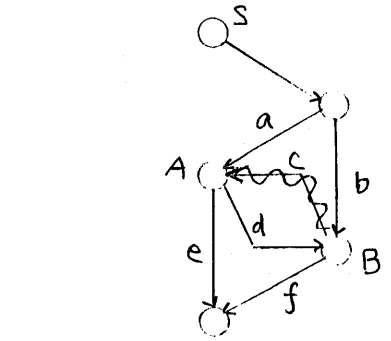
3). ハミルトンパスの節間のもとのグラフ上での最短経路を求める。

e_i を含まない各々の強連結成分については、入口と出口間の最短経路を求める。

↓

強連結成分を前のステップで導出したパスに置き換えたプログラムグラフに対して、e_i に正の重み、他に負の重みを付けて最長パスを導出する ＊＊

図. 2



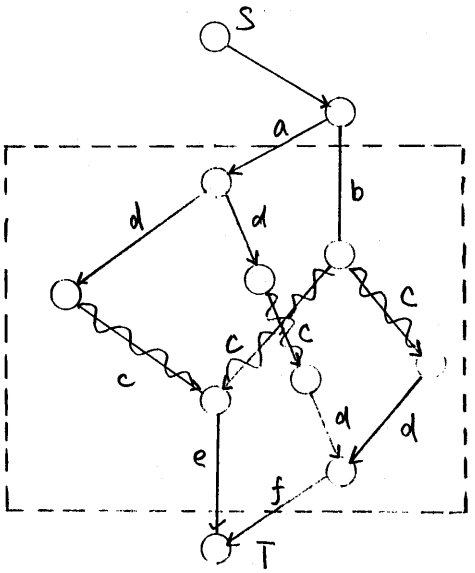


図. 3

スを求めなくてすむ効率的な解法が得られる。（構造化されていないときは導出が出来ない、図. 5）

強連結成分内の e_i の連続性のチェックは、強連結成分の検出の際にその辺を記憶し、グラフの隣接行列を用いて調べることにより チェックが可能となる。

次に、第2案は、Pamiseのプログラムグラフのレベルパス分解[10]を応用して、1段づつ求めて行く方法が考えられる。（図. 6）（構造化されていないときは、例えば、図. 7のようなグラフが生起する場合が生じ適用が不能となる）

e_i を含む各々の強連結成分について入口から e_i の連続の尾の辺へ，e_i の連続の頭の辺から出口への最短路を求める。

図. 4

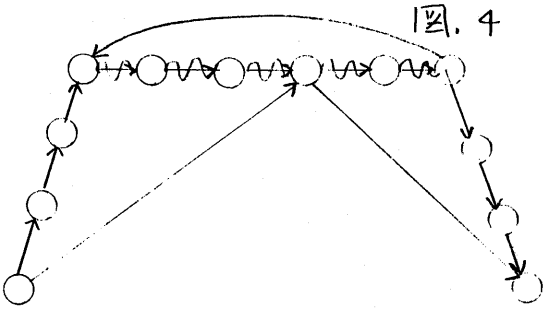


図. 5

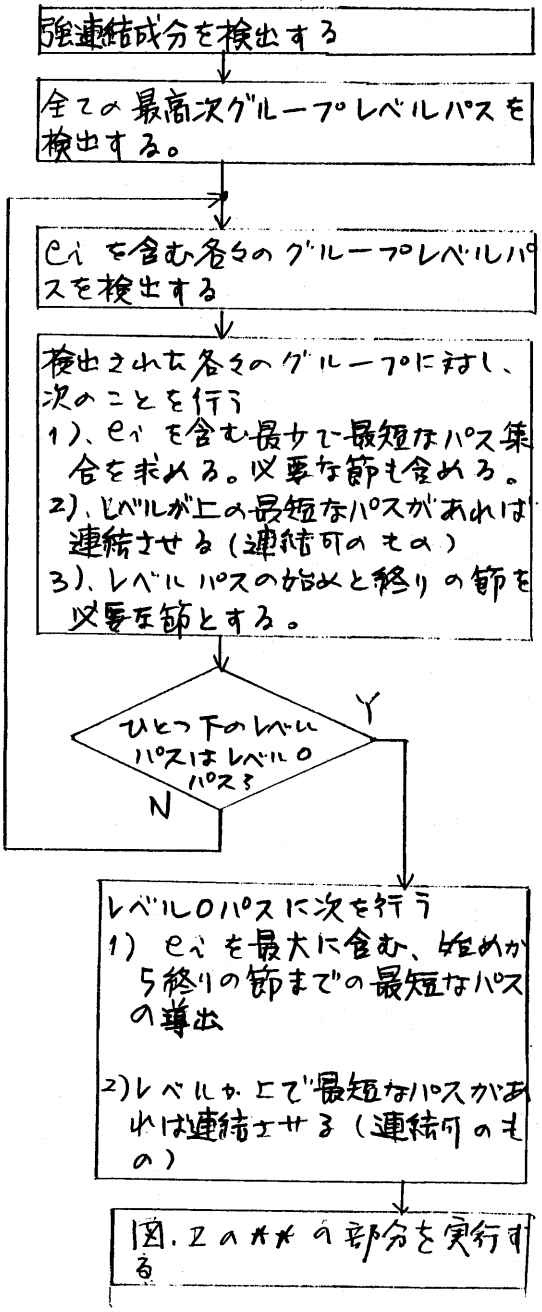


図. 6

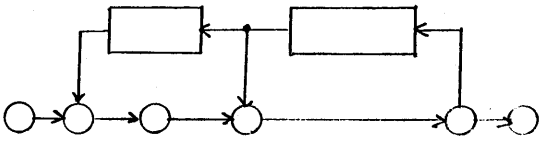


図. 7

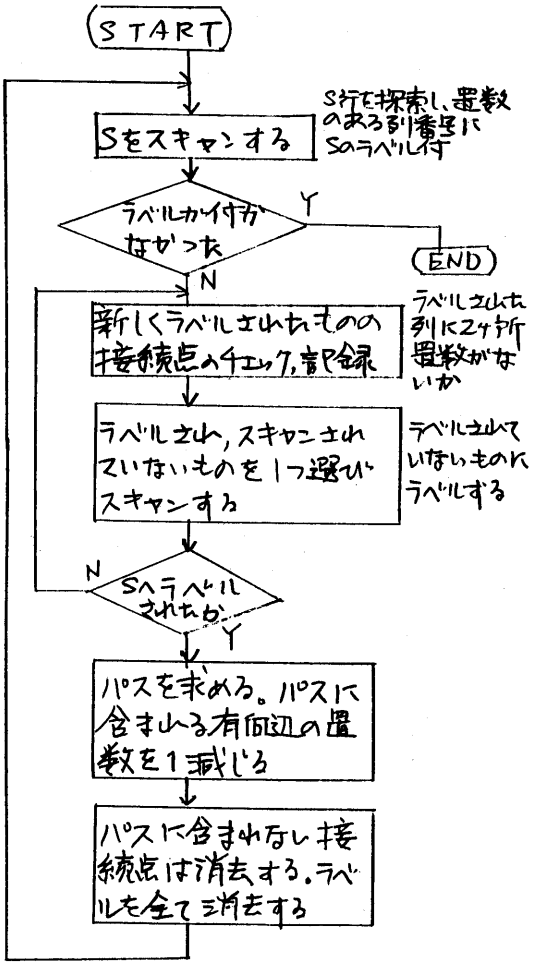


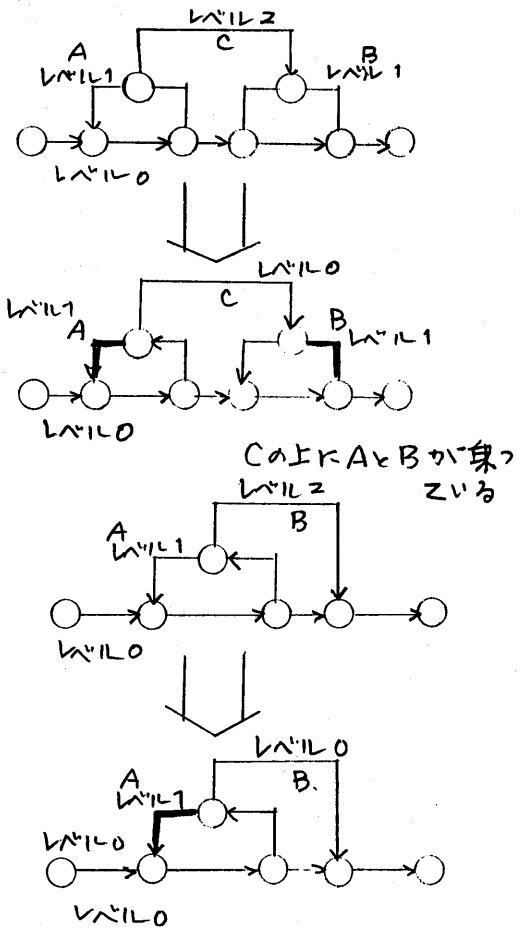
図. 8

3. 問題Bの解法

3.1 最少で最短なパス集合の導出

各辺にフローの下限：1，コスト率：1を与えたプログラムグラフに、S－T最少フロー時の最少コストフローを構築し、次に、このフローを単位順道フローと単位順閉路フローに分解し、そして結合することにより最少で最短なS－Tパス集合を求めることが出来る。分解，結合の手続は、図. 8で示され

るが、この方法で求められる理由は、プログラムグラフをレベルパスに分解すると、各レベルパスは、その1つ下のレベルの、同一のレベルパスの上に両端点を持っているためである。異るレベルパスの上に、あるいは、レベルの異るパスの上に端点を、おれかもっているように見えても、図、9のような構造となっている。

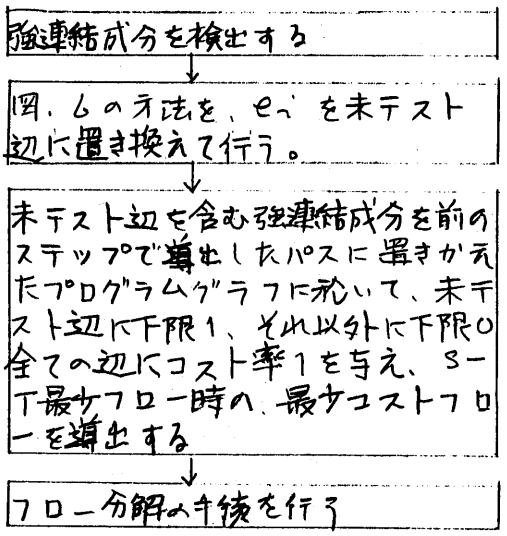


図、9

3.2 未テスト辺を含む最少で最短なS－Tパス集合の導出

有向サイクルを含まないプログラムグラフの場合は、未テスト辺にフローの下限1、各辺にコスト率1を与えたプログラムグラフに、S－T最少フロー時の最少コストフローを構築し、次に、フロー分解の手続きを適用することにより、未テスト辺を含む、最少で、最短なパス集合を求めることが出来る。

有向サイクルを含むのを許したプログラムグラフの場合は、図、2を応用して、強連結成分内の未テスト辺を導出してもよいが、未テスト辺の数が多い場合は、図、6の方法を応用して、図、10のようにして求めると効率が改善される。



図、10

4. 参考文献

[1]. R. E. Prather, Theory of Program Testing-An Overview, THE BELL SYSTEM TECHNICAL JOURNAL, Vol. 62, No. 10, part2, pp3073-3105, 1983

[2]. 柳沢隆夫, プログラムテストに用いるパスジェネレータへの一考察, 情報処理学会研究報告, ソフトウェア工学 54-2, 1987.

[3] R. K. Deb, On Generation of Test Data and Minimal Cover of Directed Graph, IFIP Congress Proceding, pp13-16, 1977.

[4] S. C. Ntafos and S. Louis Hakimi, On Path Cover Problems in Digraphs and Applications to Program Testing, IEEE Transaction On Software Engineering, Vol. SE-5 No5, pp520-529, 1979.

[5] 柳沢隆夫, プログラムテストに用いるパスジェネレータへの一考察, 情報処理学会研究報告, ソフトウェア工学 55-1, 1987

[6]. M. R. Paige, Program Graphs, an Algebra, and Their Implication for Programming, IEEE Transaction On Software Engineering, Vol. SE-1 No.3, 1975.