情報学基礎11－4
（1988. 11. 21）

パスジェネレータの自動生成によるプログラムのテスト法に関して

柳沢隆夫
芝浦工業大学

本論文は、プログラムの自動的なテストパス発生において生じる、種々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、有向グラフの最も多くの指定された辺を含むパス集合を決定することである。

On Program Testing and Generation of Program Path

Takao Yanagisawa
Shibaura Institute of Technology
Oomiya-shi, Saitama-ken, Japan

In this paper we consider algorithm for problems that arise in automatic test path generation for program : the problem of determining a path which contain the most specified edges of a directed graph.

1. はじめに.

プログラムの全ての経路を導出して行う完全なテストは、普通のプログラムでも、その経路の数が膨大となるので、全ての経路の部分集合を選んでテストする方法が考えられている。[1][2][3] (図.1)

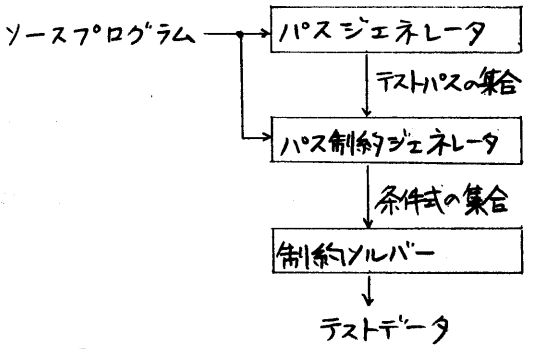


図. 1

ところで、この部分集合の選び方として、全体的に網羅するものが行われているが、部分的なテストパスを導出する方法に注目したものが見当らない。そこで、本研究は、テスト上、1番重要と思われる命令を最多に含む、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[4][5]、実用上、有用と思われるので、さらに、テストが必らずしも必要とされない命令は、最少になるテストパスを求める問題の解法について述べる。

この問題は、プログラムをグラフ表現し(命令を節で、命令間の直接の制御を有向辺で表わす)、重要と思われる有向辺(以後、e_iで表わす)に正の大きな重み、必らずしも必要でない有向辺に負の重みを付けたときの、辺の重みの合計が最大となる、S-Tパスを求める問題と等価となる。

このため、本研究は、グラフ理論を応用して、このS-Tパスの導出を行う。

2. e_iを最多に含み、かつ最短なパスの導出法

先ず最初に、プログラムグラフに有向サイクルが含まれていない場合の導出法を述べる。この問題は、前記のように最長パスを求める問題(辺の重みの符号を付けかえることにより、最短パスを求める問題となる)となるため、既存の最長(最短)パスを求める解法を、プログラムグラフに応用することにより解けばよい。

手順.1
※LABL-1は始節からの最長距離
※LABL-2は始節からの最長パスにおける1つ前の節番号

step.0 各節のラベルを0クリアする

step.1 入次数0の節を選び、この節と、その節より出ている有向辺(e_k)を、グラフより除去する。

step.2 除去された有向辺の頭の節のラベルに次のことを行う。

1). LABL-1 ← max{(e_kの重み + e_kの尻のLABL-1), e_kの頭のLABL-1}

2). 上記で、LABL-1が改訂されたときに限り
LABL-2 ← e_kの尻の節番号

step.3 終節にラベルが付いたら、最長パスをLABL-2を用いて構成し、終了する。そうでないならばGO TO step.1

この方法で求められる理由は、SからTへのパスの中で、最長のものの中にテスト上1番重要と思われる命令が多く含まれていることと、テスト上1番重要と思われるものを多く含んでいるものの中で、テストが必らずしも必要とされない命令を、1番少く含むものが最長のパスとなるからである。

2.1 プログラムに有向サイクルが含まれるのを許した場合の最適解法

次に、より一般的な、プログラムに有向サイクルが含まれる場合の、S－Tパスの導出法について述べる。

方針として、このS－Tパスは、プログラムグラフの強連結成分を、その成分内のテスト上重要とされる有向辺（以後e_iと呼ぶ）を全て含み、かつ最短なパス（最短なハミルトンパスを求める問題に帰せしめて解く（手順2））に置き換えたグラフにおいて、前記と同様に最長パスを求めることにより求める方法が考えられる。

手順．2

step．1 強連結成分を検出する。

step．2 e_iを含む各々の強連結成分について、次のことを行う

step(1)、強連結成分の入口と出口並にその強連結成分に含まれる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

step(2)、入口節から、出口節への最短なハミルトンパスを求める。

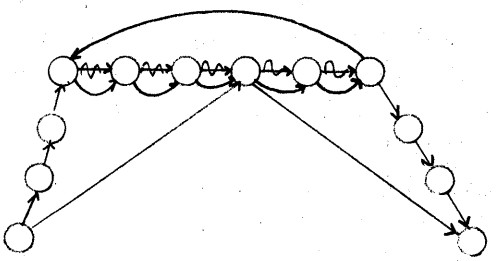
step(3)、ハミルトンパスの節間のもとのグラフ上での最短経路を求める。

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路を求める。

又、プログラムが構造化されていないときは、強連結成分の入口と出口の全ての組合せについてハミルトンパスを求めて、これを置きかえる必要がある。

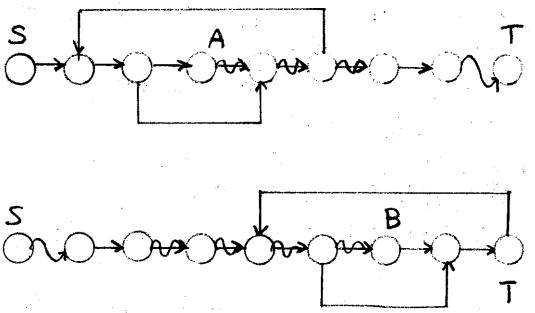
ハミルトンパスを求める問題の解法は、いくつか提案されているが、グラフの節の数が多い場合、計算量が多大に増加する。強連結成分内のe_iが連続して存在している場合（プログラムが構造化されている場合）のみ、強連結成分の入口からe_iの連続の尾の辺への最短路と、e_iの連続の頭から強連結成分の出口への最短路を求める簡単な方法で、e_iを含む最短なパスが求められる。

図．2は、構造化されていないと上記の方法が適用出来ないことを示している。



図．2

図．3は構造化されているプログラムにおいては、図．2のような経路が存在しないことを示している。（e_iの連続の途中へ入るSからの経路は、e_iの連続の尾を通るためには、e_iに入り込んだときより、さらに前に戻る必要があるため、結局、SからAへのパスを含むことになる。又同様に、e_iの連続の途中から出た経路は、e_iの連続の頭を通るためには、e_iから出たときより、さらに前に戻る必要があり、結局、BからTへのパスを含むことになる。



図．3

第2章として、強連結成分内の e_i を含み、かつ最短なパスが、最少コストフローを導出して、次にこのフローを単位増加フローに分解したものに対応していることを利用して、これを導出する方法が考えられる。

手順. 3

step. 0 e_i を含むレベルパスの入口と出口にフロー下限1を設ける。
e_i を除いた形の1つ上のレベルパスの入口と出口にもフローの下限1を設ける。
強連結成分の入口と出口に辺を設けフローの上、下限1を設ける。
e_i にフローの下限1を与え、成分内の全ての有向辺にコスト0を与える。

Step. 1 最少コストフローを導出する。

Step. 2 ラベリング法を用いて、フロー分解、結合して、e_i を含む最短なパスを導出する。

2.2 プログラムに有向サイクルが含まれるのを許した場合の近似解法。

最後に、H.F.S法を応用して、より効率的なアルゴリズムで、S-Tパスを求める方法について述べる。

手順. 4

step. 1. 未テスト辺を節とする到達可能グラフの導出

step. 2 Sからの最短距離を求め、上記の辺へ重み付ける

step. 3 到達可能グラフのSより、優先順位を、H, F, S法に与えて、辺の操作を行う。リターン辺を除去する。

step4. SよりTへの最長パスの導出.

Step5 欠落辺の最短パスを導出する

ここで、Warshallのグラフの全ての2節間の最短距離を求めるアルゴリズムを応用すると、step. 1は置数が0でないとき、1とすることにより、到達可能行列は導出され、step. 2はそのまま応用され、step 5は、上記のアルゴリズムを導出すると、付随して導出される。

このアルゴリズムは、e_i 間の到達関係を保存しながら、有向サイクルをグラフより除去してしまう方法である。この方法は、グラフを到達可能グラフに変換した後、H. F. S法を適用してそのとき生じた、リターン辺を除去して、有向サイクルを除去することにより行われる。その結果、有向サイクルの無くなったグラフに最長(最短)アルゴリズムを応用して解が求めることが出来る。しかし、この方法は、最適近似解法で、必ずしも最短なパスは導出しない。この方法は、前記の第1章と比較すると、最短なハミルトンパスを求める替りに、H, F, S法を適用したことになり、計算量は多大に減少する。第2章と比較すると、最少コストフローとそのフロー分解を求めるかわりに、H, F, S法を適用していることになり、計算量は減少する。

3. 参考文献

[1]. R. K. Deb, On Generation of Test Data and Minimal Cover of Directed Graph, IFIP Congress Proceding, pp 13-16, 1977.

[2]. 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 No 5, pp 520-529, 1979.

[3]. R. E. Prather, Theory of Program Testing - An Overview, THE BELL SYSTEM TECHNICAL JOURNAL, Vol. 62, No. 10, part 2, pp 3073-3105, 1983.

[4] 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, ソフトウエア工学 54-2, 1987.

[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の自動生成について, 情報処理学会研究報告, アルゴリズム 3-6, 1988.