

goto 文最小のプログラムについて

早大 理工 岩田茂樹

1. はじめに

本研究は、制御の流れが goto 文だけによって変えられる Fortran 型言語において、与えられたフローチャートに対し、goto 文の数が最小となるようなプログラムを与えるアルゴリズムを考える。

2. 準備

有向グラフ $G = (N, E)$ は node の集合 N と edge の集合 E とからなる。有向グラフは loop や multiple edge を含まない。edge (u, v) は u を 去り, v に 入る。

本稿においては、すべての node について、その node を去る edge の数やその node に入る edge の数を 2 以下と仮定する。

edge が去らない node は 停止 node である。 u_1 から

u_n までの path は edge の列 $(u_1, u_2)(u_2, u_3) \dots (u_{n-1}, u_n)$ で、 $u_1, u_2, \dots, u_n$ がすべて異なる。 $u_1, u_2, \dots, u_n$ や $(u_1, u_2), (u_2, u_3), \dots, (u_{n-1}, u_n)$ は path の上にある。null path はただ1つの node からなる。edge の列 $e_1, e_2, \dots, e_n$ で、 $e_1, e_2, \dots, e_n$ と $e_2, e_3, \dots, e_n, e_1$ の両方が path のとき $e_1, e_2, \dots, e_n$ は cycle である。相異なる edge e_1, e_2 は node を共有するとき、隣接 する。edge の列 $e_1, e_2, \dots, e_n$ は (i) e_i と e_{i+1} ($1 \leq i < n$) が隣接する、かつ (ii) $e_1, e_2, \dots, e_n$ にあらわれる node で共有 node でないものはすべて異なる、のとき semipath である。edge の列 $e_1, e_2, \dots, e_n$ は、 $e_1, e_2, \dots, e_n$ と $e_2, e_3, \dots, e_n, e_1$ の両方が semipath のとき、semicycle である。edge e を含む cycle がないとき、 e は acyclic であり、有向グラフ G に cycle がないとき、 G は acyclic である。semipath や semicycle は、それらを構成している edge がすべて acyclic のとき acyclic である。有向グラフ (N, E) は、(i) $n_0 \in E$, n_0 から N のすべての node への path がある、かつ (ii) N 中に停止 node が1つ以上ある、のとき フローグラフ であり、 (N, E, n_0) で表わす。

定義1. G を有向グラフとする。 G の 包括集合 P とは、node の異なる path の集合であり、 G のすべての node が

P の path 上にあるものという。

goto 文以外の文を node におきかえることにより、フローチャートはフローグラフになる。我々は、Fortran 型プログラムが、代入文、if-goto 文、goto 文の列であり、そのフローグラフの包括集合がそのプログラムを表わしていると考え、フローグラフ $F = (N, E, n_0)$ の包括集合 P によって表わされた Fortran 型プログラムでは、goto 文は P の path 上にない edge に対応している。従って goto 文の数は、 P の path 上にない edge の数である。ここで P のある path が n_0 から始まっていることを仮定している。我々は、 P の path 上にない edge の数は、 $\#(P)$ と増減を共にすることに注目する。よって、最小の goto 文をもつ Fortran 型プログラムを見つける問題は、そのフローグラフの最小の要素からなる包括集合を見つける問題に帰着される。

定義 2. [2] $G = (N, E)$ を有向グラフとする。相異なる edge $(u, v), (w, x)$ は $u = w$ または $v = x$ のとき、交互隣接するという。semipath $e_1 e_2 \dots e_n$ は e_i と e_{i+1} ($1 \leq i < n$) が交互隣接のとき、交互 semipathといい、semicycle $e_1 e_2 \dots e_n$ は、 $e_1 e_2 \dots e_n, e_2 e_3 \dots e_n e_1$ の両方

が交互 semipath のとき, 交互 semicycle という.

補助定理 1. $G=(N, E)$ を有向グラフとし, E 中の acyclic edge e に対して, たかだか 1 つの edge f が交互隣接していない, とする. このとき, 次の性質を満足するような G の包括集合 P が存在する. (i) e は P の path 上にある, かつ (ii) G の任意の包括集合 P' に対して, $\#(P) \leq \#(P')$ である.

補助定理 2. G を有向グラフとし, $c = e_1 e_2 \dots e_{2n}$ を G の acyclic な交互 semicycle とし, e_1, e_2 の開始 node は同一であるとする. $i=1, 2$ に対して, 次の性質を満足するような G の包括集合 P が存在する. (i) e_i は P の path 上にある, かつ (ii) G の任意の包括集合 P' に対して $\#(P) \leq \#(P')$ である.

補助定理 3. G を有向グラフとし, e_1, e_2 を開始 node が同一であるような G の edge とする. もし, acyclic な交互 semipath $e_1 e_2 \dots e_k$ が存在し, $e_k = (w, x)$ のみが w を去るとき, 次の性質を満足するような G の包括集合 P が存在する. (i) e_1 は P の path 上にある, かつ (ii) G の

任意の包括集合 P' に対して $\#(P) \leq \#(P')$ である. もし, acyclic な交互 semipath $e_1, e_2, \dots, e_k$ が存在し, $e_k = (w, x)$ のみが x に入るとき, 次の性質を満足するような G の包括集合 P が存在する. (i) e_2 は P の path 上にある, かつ (ii) G の任意の包括集合 P' に対して $\#(P) \leq \#(P')$ である.

3. 1-ゲート フロー グラフ

アルゴリズム A. path の集合.

入力: フローグラフ $F = (N, E, n_0)$.

出力: F の path の集合 P .

方法: E を 3 つの edge の集合: $S =$ 選択される edge
 — P の path 上になる edge; $D =$ 除去される edge —
 P の path 上にならない edge; $U =$ 未決定の edge —
 S か D か分類できない edge ではじめは $U = E$; に分類
 することによって P をうる. S, D, U は互に排他的で,
 $S \cup D \cup U = E$ である.

A1. n_0 に入る edge を除去する.

A2. もし, 有向グラフ (N, U) において, 未決定の acyclic な edge e が存在し, かつ (i) たかだか 1 つの edge が e に交互隣接している, または (ii) acyclic な交互 semicycle で e を含むものがある, ならば e を選択する.

未決定な edge で e に交互隣接している edge は除く。
 もし、上の (i) または (ii) を満足する acyclic edge が存在しないときは、 P は有向グラフ (N, S) の path の集合であり、アルゴリズムは停止する。このステップを繰り返す。

定理 1. F をフローグラフとする。もし、アルゴリズム A を F に適用した結果えられる path の集合 P が F の包括集合ならば、 F の任意の包括集合 P' に対して $\#(P) \leq \#(P')$ である。

補助定理 4. $G = (N, E)$ を acyclic な有向グラフとする。もし E が空でないならば、 E 中の edge e が存在して (i) たかだか 1 つの edge が e に交互隣接している、かまたは (ii) e を含む交互 semicycle が存在する。

定理 1 と補助定理 4 より定理 2 をうる。

定理 2. F を acyclic なフローグラフとする。アルゴリズム A を F に適用した結果えられる path の集合 P は、 F の包括集合であり、 F の任意の包括集合 P' に対して、 $\#(P) \leq \#(P')$ である。

$F = (N, E, n_0)$ をフローグラフとし, $c \in F$ の cycle, $C \subseteq c$ にあらわれる node の集合とする. E 中の edge (u, v) はもし (i) $u \notin C, v \in C$ かつ (ii) C 中の node を通らない n_0 から u への path がある, ならば c の 入口 と呼ばれる. 同様に, E 中の edge (u, v) はもし (i) $u \in C, v \notin C$ かつ (ii) C 中の node を通らない v から F の停止 node への path がある, ならば c の 出口 と呼ばれる.

定義 3. $F = (N, E, n_0)$ をフローグラフとし, c を F の cycle とする. もし edge e が c の入口または出口のとき, e は c の ゲート であるという. もし, F のすべての cycle について, cycle 中の 1 つの edge のみが交互隣接しているようなゲートが存在するならば, F は 1-ゲート フローグラフ であるという.

1-ゲート フローグラフの定義と定理 1 より定理 3 をうる.

定理 3. F を 1-ゲート フローグラフとする. アルゴリズム A を F に適用した結果えられる path の集合 P は F の包括集合であり, F の任意の包括集合 P' に対して,

$\#(P) \leq \#(P')$ である.

4. ゲート フロ - グラフ

アルゴリズム B. path の集合 — 速いアルゴリズム 4.

入力: フロ - グラフ $F = (N, E, n_0)$.

出力: F の path の集合 $P = \{p^1, p^2, \dots, p^n\}$.

方法: はじめは edge の集合 E' は E である. 以下のステップで決定される edge を E' から除去する.

B1. $E' = E$, $u_i = n_0$.

B2. n_0 に入る edge を E' から除去する. $j = 0$.

B3. j を 1 増加する. $i = 0$.

B4. i を 1 増加する. u_i を去る E' 中の edge の数が, 0 なら B5 $\wedge$, 1 なら B6 $\wedge$, 2 なら B7 $\wedge$.

B5. $i \geq 2$ ならば, path p^j は選択された edge の列 $(u_1^j, u_2^j)(u_2^j, u_3^j) \dots (u_{i-1}^j, u_i^j)$ で, $i=1$ ならば path p^j は null path u_i である. もし, $p^1, p^2, \dots, p^j$ 上にはない node で, E' 中の edge が入っていない node があれば, その node を u_{i+1}^j とする. B3 $\wedge$ 戻る. そのような node がないときは, $n=j$ としてアルゴリズムは停止する.

B6. もし $(u_i^j, v) \in E'$ ならば, (u_i^j, v) を選択し, $u_{i+1}^j = v$ とする. (u_i^j, v) とそれに交互隣接する edge を

E' から除去する.

B7. $e_1 = (u_i^j, v_1) \in E'$, $e_2 = (u_i^j, v_2) \in E'$ とする.

もし交互 semicycle $e_1 e_2 \dots e_k$ がある場合や, $e_k = (w, x)$ のみが w を去るような交互 semipath $e_1 e_2 \dots e_k$ がある場合は, e_1 を選択し, $u_{i+1}^j = v_1$ とする. e_1 と e_1 に交互隣接する edge を E' から除去する. B4 へ戻る. もし, $e_k = (w, x)$ のみが x に入るような交互 semipath $e_1 e_2 \dots e_k$ がある場合は, e_2 を選択し, $u_{i+1}^j = v_2$ とする. e_2 と e_2 に交互隣接する edge を E' から除去する. B4 へ戻る.

定理4. F をフローグラフとする. もしアルゴリズム B を F に適用した結果えられる path の集合 P が F の包括集合ならば, F の任意の包括集合 P' に対して $\#(P) \leq \#(P')$ である.

定理5. F を acyclic なフローグラフとする. アルゴリズム B を F に適用した結果えられる path の集合 P は F の包括集合であり, F の任意の包括集合 P' に対して, $\#(P) \leq \#(P')$ である.

定義4. フローグラフ F のすべての cycle について,

cycle の各ゲートが cycle 中の 1 つの edge のみに交互隣接しているならば、 F は ゲート フローグラフ であるという。

注意. ゲート フローグラフのクラスは、1-ゲート フローグラフのクラスの部分集合である。

ゲート フローグラフの定義と定理 4 より定理 6 とうる。

定理 6. F をゲート フローグラフとする。アルゴリズム B を F に適用した結果えられる path の集合 P は F の包絡集合であり、 F の任意の包絡集合 P' に対して、
 $\#(P) \leq \#(P')$ である。

定理 7. すべてのフローグラフは、ゲートにダミー node を挿入することによってゲート フローグラフに帰着させることができる。

謝辞. 日頃より御指導いただく門倉敏夫教授、有益な討論をして下さった夜久竹夫氏、並井琢美氏に深謝する。

参考文献

- [1] Iwata, "Programs with minimal goto statements",
投稿中
- [2] Yaku, "Flowgraphs and covering subgraphs of
minimal branches", 投稿中
- [3] Harary, "Graph Theory" Addison-Wesley (1969)