

流れ図の最大道づうつをみつける

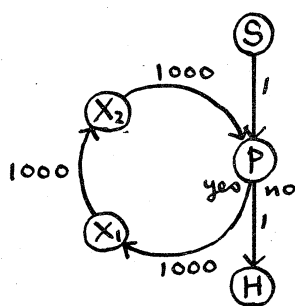
アルゴリズムについて

東北大 通研 永松 正博

名大 エ学部 本多 淑雄

1 まえがき

流れ図をシーケンシャルな形のプログラムに変換することを考える。図1に例を示す。流れ図の1回の実行において各枝を通過する回数が(a)の各枝に書いてある数値であるとする。(b)の書き方の場合、1001回、(c)の書き方の場合、2000回 GO TO 文が実行される。ここでは、GO TO 文の実行回数が最小のシーケンシャルなプログラム記述をみつける問題と同じ問題であるところの、最大道づうつをみつけ



(a)

L_1 : S
 P
 GO TO L_2 (1)
 X_1
 X_2
 GO TO L_1 (1000)
 L_2 : H

(b)

L_1 : S
 $\overline{P}$
 GO TO L_2 (1000)
 H
 L_2 : X_1
 X_2
 GO TO L_1 (1000)

(c)

図 1

る問題について述べる。

2 代入による流れ図の生成

モジュールとは1個の入口と1個以上の出口をもつ有向グラフで、モジュールに停止点をつけたものを流れ図とよぶ。モジュールのサブグラフで、入口が1個、出てゆく先が1個であるものをブロックという。 $\mathcal{A}$

がモジュールの集合のとき、 $\mathcal{G}(\mathcal{A})$ をつぎのように定義する。

(i) (単位モジュール) $\in \mathcal{G}(\mathcal{A})$

(ii) $B \in \mathcal{G}(\mathcal{A})$, v が B の単位ブロック, $A \in \mathcal{A}$ のとき、 B において v に A を代入してできるモジュールも $\mathcal{G}(\mathcal{A})$ の元である。

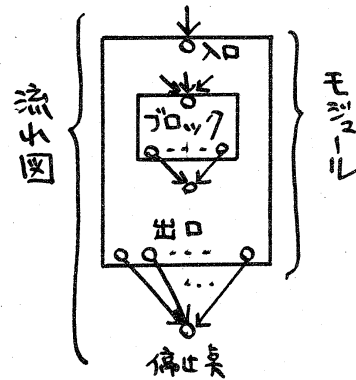


図 2

$\mathcal{G}(\mathcal{A})$ の各元に停止点をつけてできる流れ図の集合を flow $(\mathcal{A})$ と示す。uを自分自身と単位ブロック以外にはブロックをもたないモジュールの集合とする。任意の流れ図は flow

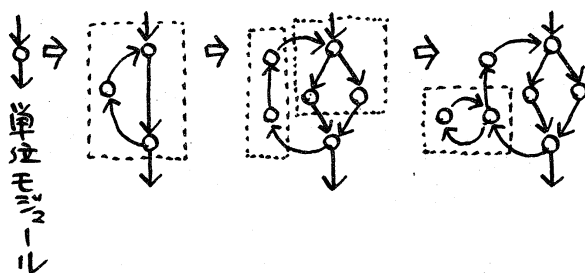
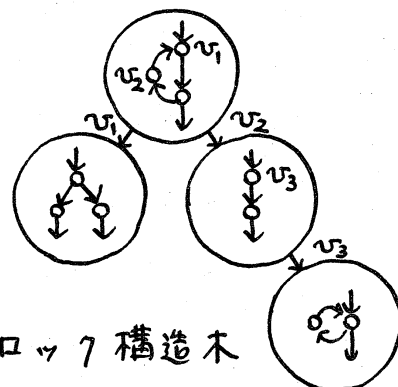


図 3



ブロック構造木

(U)の元である。 $\text{flow}(\mathcal{A})$ の元が与えられたとき、それが上記の定義で、どのようなモジュールの代入の過程を経て生成されたかを一種の導出木の形で表現できる。これをブロック構造木とよぶ。

3 最大道グラフ

$G = (V, E)$ を有向グラフ, δ を E から非負実数への関数(枝コスト関数)とする。 G のパーシャルグラフ $\Pi = (V, E')$ のすべてのコンポーネントが単純道であるとき、 Π を G の道グラフであるという。 $\text{cost}(\Pi, \delta) \triangleq \sum_{e \in E'} \delta(e)$ が最大の道グラフを (G, δ) の最大道グラフという。

F が流束図, δ が F の各枝の通過回数を表わすとき、 (F, δ) の最大道グラフは、GOTO文の実行回数が最小のシーテンシャルなプログラムに対応する。

4 ブロック構造木をつくるアルゴリズム

[補題] $\mathcal{A} \in \mathcal{U}$, $B_0 \in \mathcal{B}(\mathcal{A})$ とする。 B_0 が $A_0 \in \mathcal{U}$ をブロックとして含むとき、 $A_0 \in \mathcal{A}$, $B_1 = (B_0$ の A_0 を一点に縮めたもの) $\in \mathcal{B}(\mathcal{A})$ である。さらに B_1 が $A_1 \in \mathcal{U}$ をブロックとして含むとき、 $A_1 \in \mathcal{A}$, $B_2 = (B_1$ の A_1 を一点に縮めたもの) $\in \mathcal{B}(\mathcal{A})$ である。以下同様に $B_3, B_4, \dots$ とつづけてゆくと、ある m において、 B_m は単位モジュールになる。

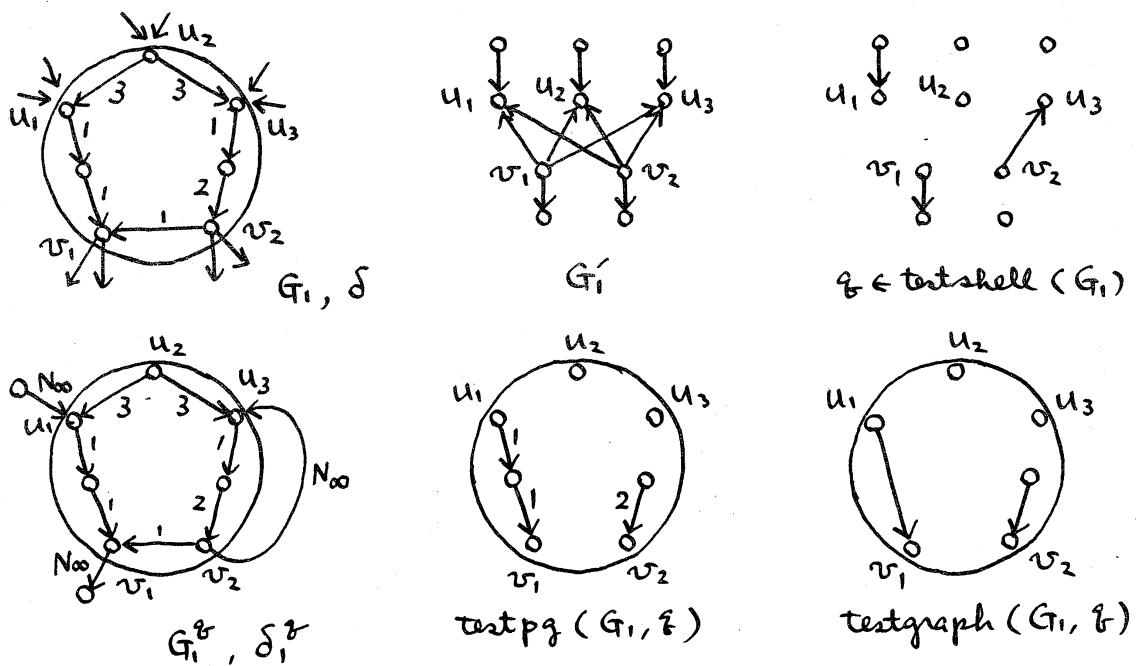
これが、ブロック構造木をつくるアルゴリズムの基本となるものである。このアルゴリズムの入力は流小図のサクセッサリスト表現であるが、まず、つぎの条件を満足するように作りかえる。条件：(あるサクセッサリストの先頭の素を v_{top} とする。 $\{(v, v_{top}) \mid v \text{ キ 終 止 素}\}$ からなるパシアルグラフは、停止素を根とする流入木である。) この条件を満足しているとき、デプスファーストサーチ (DFS) をおこなない、サーチの終了した順に素を並べると、任意のブロックを構成している素は入口を最後にして一列になる。

[アルゴリズム] 新しく作りかえられたサクセッサリスト表現に対して DFS をおこなない、ある素のサーチが終了するたびに、その素を入口とするブロックで π の π であるものが存在するか否かを調べて、もし存在すれば、それを一素に縮めていく。 ■

このアルゴリズムは、任意の流小図 $F = (V, E)$ に対して、 F を $\text{flow}(\mathcal{U})$ の元とみなして、そのブロック構造木を $O(|E| \cdot |V|)$ の時間でつくる。また、このアルゴリズムをすこし変えると、 flow (モジュールの有限集合), $\text{flow}(\mathcal{M})$ に対して、 $O(|V|)$ の時間でブロック構造木をつくるアルゴリズムが得られる。 $\mathcal{M}$ は、ある簡単な構造をしたモジュールの無限集合である。

5 最大道グラフをみつけるアルゴリズム

$G = (V, E)$ を有向グラフ, δ を枝コスト関数, N_∞ を適当な大きな数, $G_1 = (V_1, E_1)$ を G のサブグラフ, G_1 の入口を $u_1, \dots, u_k$, 出口を $v_1, \dots, v_m$ とする。 G の最大道グラフをつくるのに、 G_1 の内部でどのようなことがわか、それによいかを考える。図4のようなグラフ G_1 をつくる。 G_1 の道グラフの集合を testshell(G_1) と書く。 $\mathcal{G} \in \text{testshell}(G_1)$ とし、 $G_1^{\mathcal{G}}$ を図4のように、 G_1 と $\mathcal{G}$ を組み合わせたグラフとする。 $(G_1^{\mathcal{G}}, \delta_1^{\mathcal{G}})$ の最大道グラフの G_1 の内部を testpg($G_1, \mathcal{G}$) とする。 testpg($G_1, \mathcal{G}$) の中で、 $u_1, \dots, u_k, v_1, \dots, v_m$ 以外の点で、道の途中にある点を短絡してできるグラフを testgraph($G_1, \mathcal{G}$) とする。



$G=(V, E)$ を有向グラフ, δ を G の枝コスト関数. $G_1, G_2, \dots, G_\ell$ を互いに共通枝をもたない G のサブグラフとする. $g_i \in \text{testshell}(G_i)$ とし, 各 G_i を $\text{testgraph}(G_i, g_i)$ で置きかえたグラフを $G[g_1, g_2, \dots, g_\ell]$ とする. $\text{testgraph}(G_i, g_i)$ の内部の枝のコストを N_∞ , その他の枝のコストは δ と同じにしたものを $\delta[g_1, g_2, \dots, g_\ell]$ とし, $(G[g_1, g_2, \dots, g_\ell], \delta[g_1, g_2, \dots, g_\ell])$ の M.P.G. を $\Pi G[g_1, g_2, \dots, g_\ell]$ とする. $\text{value}(\Pi G[g_1, g_2, \dots, g_\ell]) \triangleq \text{cost}(\Pi G[g_1, g_2, \dots, g_\ell], \delta[g_1, g_2, \dots, g_\ell]) - N_\infty \sum_{i=1}^{\ell} (\text{testgraph}(G_i, g_i) \text{ の枝の数}) + \sum_{i=1}^{\ell} \text{cost}(\text{testpg}(G_i, g_i), \delta)$ とし, $\{\text{value}(\Pi G[g_1, g_2, \dots, g_\ell]) \mid g_i \in \text{testshell}(G_i)\}$ の中で最大のものを $\text{value}(\Pi G[y_1, y_2, \dots, y_\ell])$ とする. $E'' = ((\Pi G[y_1, y_2, \dots, y_\ell] \text{ の枝}) - \bigcup_{i=1}^{\ell} (\text{testgraph}(G_i, y_i) \text{ の枝})) \cup \bigcup_{i=1}^{\ell} (\text{testpg}(G_i, y_i) \text{ の枝})$ とすると, G のパーシャルグラフ (V, E'') は (G, δ) の最大道グラフである.

以上述べた方法で、入力流小図のブロック構造木の葉の方からポストオーダーで、 testpg , testgraph をつくってゆくと、最後には、与えられた流小図の最大道グラフをつくることができる。これがアルゴリズムの基本的な考え方である。このアルゴリズムをつかうと、 $O(|V| \cdot |E| + |V| k^M)$ の時間で最大道グラフをつくることができる。ここで、 k は

ある定数、 M は入力流小図を生成する α に使用したモジュールの枝の数の最大値である。また、 flow (モジュールの有限集合), $\text{flow}(\alpha, \gamma)$ に対しては $O(|V|)$ の時間で最大道グラフをつくることができる。また、枝コスト関数が1の場合は、 $O(|V| \cdot |E| + |V| \cdot M \cdot g(KM))$ の時間で最大道グラフをつくることができる。ここで $g(n)$ は枝の個数 n 個の有向グラフの最大道グラフをつくるのに要する時間である。

6 NP完全性

すべての葉の入次数が、出次数が2以下であり、大きき3以上の有向閉路をもたない流小図のクラスを $\mathcal{F}_2$ とする。一般的な流小図を対象にした場合、根本の道からなる道グラフをもつか否かを判定する問題はNP完全である。 $\mathcal{F}_2$ を対象にした場合も、この問題がNP完全であることが証明できる。証明の方法は、無向グラフに対する、長径の被覆集合をもつか否かを判定する問題が、この問題に多項式時間で帰着することをつかう。

文 献

岩田: "GOTO文最小プログラムについて", 京大数解研シンポ「組合せ構造とグラフ理論」予稿集 P79, 1975

奥井, 谷口, 都倉, 岩: "フローチャートからプログラムへの変換", 昭和46年度電気通信学会全国支部連合大会 G8-12

西岡, 高見沢, 斎藤: "GOTO文最小プログラム記述の計算手続について" 信学技報 AL75-82; 1976