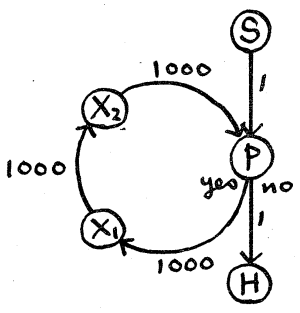
数理解析研究所講究録
第296巻 1977年 82-88

流れ図の最大道グラフ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 工学部　本多波雄

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、最大道グラフをみつけ



(a)

```
      S
L1:   P
      GO TO L2 ( 1 )
      X1
      X2
      GO TO L1 (1000)
L2:   H
```

(b)

```
      S
L1:   P
      GO TO L2 (1000)
      H
L2:   X1
      X2
      GO TO L1 (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。

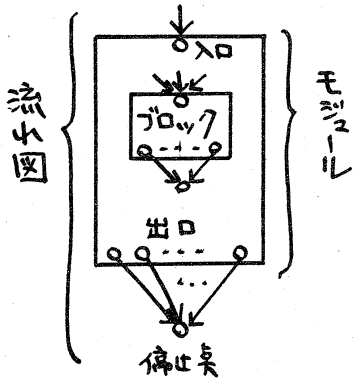


図2

$\mathcal{G}(\mathcal{A})$の各元に停止点をつけてできる流れ図の集合を$\underline{\text{flow}(\mathcal{A})}$で示す。$v$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

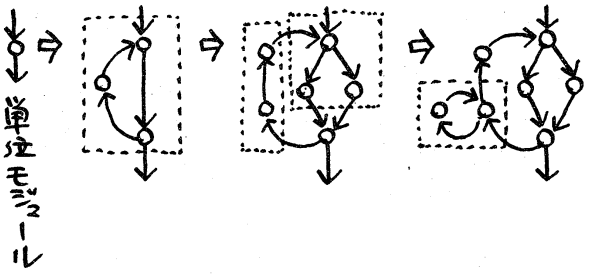
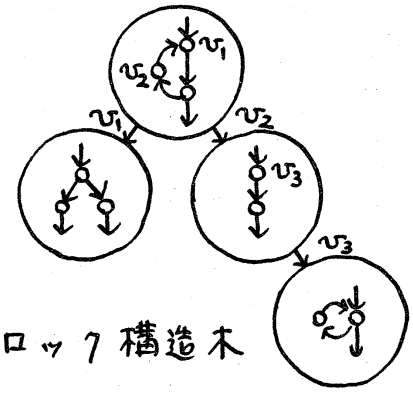


図 3



ブロック構造木

(U)の元である。flow(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 Π=(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。$cost(\Pi,\delta) \triangleq \sum_{e\in E''} \delta(e)$ が最大の道グラフを(G,δ)の最大道グラフという。

Fが流れ図，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、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] A⊂U，$B_0 \in \mathcal{G}(A)$ とする。B_0 が $A_0 \in U$ をブロックとして含むとき、$A_0 \in A$，B_1=(B_0のA_0を一点に縮めたもの)$\in \mathcal{G}(A)$である。さらに B_1 が $A_1 \in U$ をブロックとして含むとき、$A_1 \in A$，B_2=(B_1のA_1を一点に縮めたもの)$\in \mathcal{G}(A)$である。以下同様に $B_3, B_4, \cdots$ とつくってゆくと、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れ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つくりかえる。条件：（異びのサクセッサリストの先頭の点を v_{top} とする。$\{(v, v_{top}) \mid v \neq$ 停止点$\}$ からなるパーシャルグラフは、停止点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）をおこない、サーチの終了した順に点を並べると、任意のブロックを構成している点は入口を最後にして一列にならぶ。

［アルゴリズム］ 新しくつくりかえられたサクセッサリスト表現に対してDFSをおこない、ある点のサーチが終了するたびに、その点を入口とするブロックで v の元であるものが存在するか否かを調べて、もし存在すれば、それを一点に縮めていく。 ■

このアルゴリズムは、任意の流れ図 $F=(V, E)$ に対して、F を flow(v) 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），flow（丹下）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。丹下は、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，$u_1,\cdots,u_k$，出口を$v_1,\cdots,v_m$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っていればよいかを考える。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。$g\in$ testshell(G_1)とし，G_1^gを図4のように，G_1とgを組み合わせたグラフとする。(G_1^g,δ_1^g)の最大道グラフのG_1の内部を testpg(G_1,g)とする。testpg(G_1,g)の中で，$u_1,\cdots,u_k,v_1,\cdots,v_m$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,g)とする。

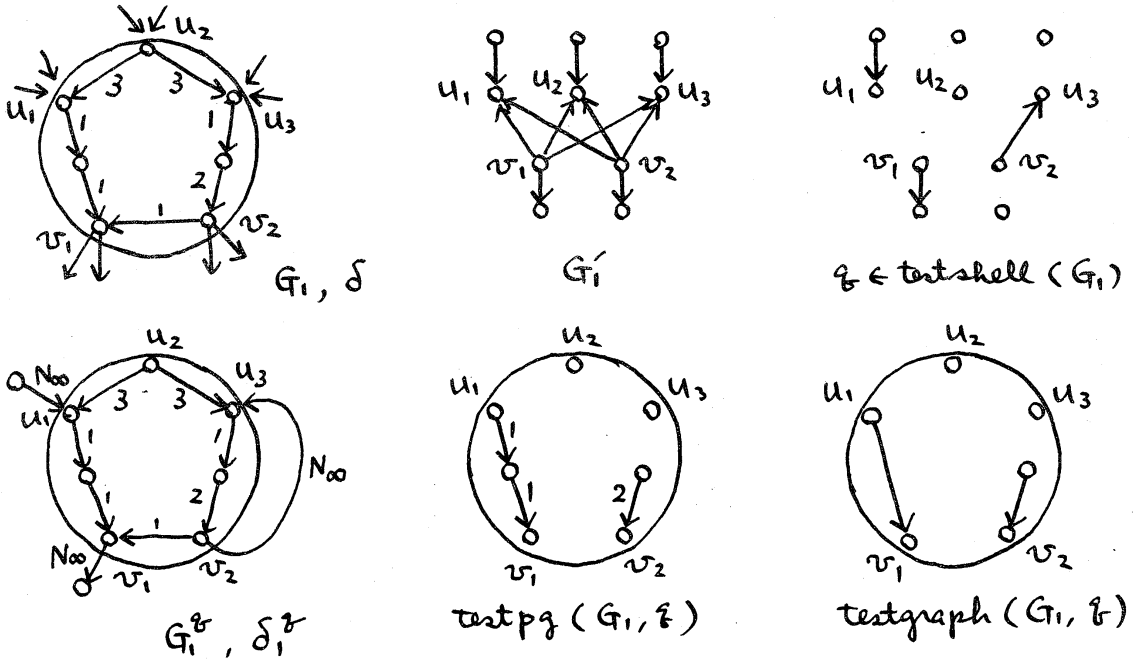


図 4

$G=(V,E)$を有向グラフ，δをGの枝コスト関数．G_1，G_2，…，G_ℓを互いに共通枝をもたないGのサブグラフとする。$g_i \in testshell(G_i)$とし、各G_iをtestgraph(G_i, g_i)でおきかえたグラフを$G[g_1, g_2, \cdots, g_\ell]$とする。testgraph$(G_i, g_i)$の内部の枝のコストを$N_\infty$，その他の枝のコストは$\delta$と同じにしたものを$\delta[g_1, g_2, \cdots, g_\ell]$とし，$(G[g_1, g_2, \cdots, g_\ell], \delta[g_1, g_2, \cdots, g_\ell])$のMPGを$\Pi G[g_1, g_2, \cdots, g_\ell]$とする。$value(\Pi G[g_1, g_2, \cdots, g_\ell]) \triangleq cost(\Pi G[g_1, g_2, \cdots, g_\ell], \delta[g_1, g_2, \cdots, g_\ell]) - N_\infty \sum_{i=1}^{\ell}$(testgraph$(G_i, g_i)$の枝の数$) + \sum_{i=1}^{\ell} cost(testpg(G_i, g_i), \delta)$とし，$\{value(\Pi G[g_1, g_2, \cdots, g_\ell]) \mid g_i \in testshell(G_i)\}$の中で最大のものを$value(\Pi G[\gamma_1, \gamma_2, \cdots, \gamma_\ell])$とする。$E'' = \left((\Pi G[\gamma_1, \gamma_2, \cdots, \gamma_\ell]の枝) - \bigcup_{i=1}^{\ell} (testgraph(G_i, \gamma_i)の枝)\right) \cup \bigcup_{i=1}^{\ell} (testpg(G_i, \gamma_i)の枝)$とすると、Gのパーシャルグラフ$(V, E'')$は$(G, \delta)$の最大道グラフである。

以上述べた方法で、入力流れ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をつくっていくと、最後には、与えられた流れ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は入力流れ図を生成するのに使用したモジュールの枝の数の最大個である。また、flow（モジュールの有限集合），flow（分岐）に対しては O(|V|) の時間で最大道がうつ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|·|E| + |V|·M·g(kM)) の時間で最大道がうつをつくることができる。ここで g(m) は枝の個数 m 個の有向グラフの最大道がうつをつくるのに要する時間である。

6 NP完全性

すべての点の入次数，出次数が2以下であり、大きさ3以上の有向閉路をもたない流れ図のクラスを $\mathcal{F}_2$ とする。一般的な流れ図を対象にした場合、k本の道からなる道がうつ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、k個の被覆点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年度電気関係学会関西支部連合大会 G8-12

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