1998年度日本オペレーションズ・リサーチ学会
秋季研究発表会

# 有向グラフにおける負サイクルの全列挙 *

02103040　防衛大学校　木下治信 †　KINOSHITA Harunobu
01700900　防衛大学校　山田武夫　YAMADA Takeo

## 1　はじめに

有向グラフ中のすべての単純サイクルを列挙する問題はTarjan等により研究されている[1],[2]が，枝に負数を含めた荷重が与えられた場合に，負サイクルのみを全列挙する問題は従来研究されていないと思われるので，本稿ではそのためのアルゴリズムを提示する．

## 2　アルゴリズム

$G=(V,E)$を節点集合$V$，枝集合$E\subseteq V\times V$からなる有向グラフとし，$G$の各枝には整数コスト$d: E\to Z$が与えられているものとする．グラフ$G$の単純サイクル$C$のコストを$d(C):=\sum_{e\in C}d(e)$とし，$d(C)<0$なる単純サイクルをここでは**負サイクル**と呼ぶ．$G$の負サイクルを全列挙することが本稿の問題である．

### 2.1　1つの負サイクルの検出

$G$中に負サイクルが存在する場合に，その1つを求め，存在しない場合にはそのことを確認する問題を**負サイクル検出問題**という．これについては，最短経路問題を解く**ラベル修正法**を利用した効率的なアルゴリズムが知られている[3]ので，以下ではこれを関数Find-an-NC()と記す．この関数は，負サイクルを見つけた場合にはそのサイクルを返し，負サイクルが存在しない場合は空集合$\emptyset$を返す．

### 2.2　アルゴリズムの概要

前項のアルゴリズムで，1つの負サイクル$C=\{e^1, e^2,\ldots,e^q\}$が求まったとすると，以下では問題を次のような条件を付加した部分問題の組$P^0, P^1,\ldots,P^{q-1}$に分割する．

**部分問題**$P^i$：　枝$e^1,e^2,\ldots,e^i$を含み，枝$e^{i+1}$を含まない負サイクルを全列挙せよ．

このように，いくつかの枝の集合$F$と$R$に対して「$F$のすべての枝を含み，$R$の枝を含まない」負サイクルを全列挙する問題を，以下では部分問題$P(F,R)$と記し，$F$，$R$の枝をそれぞれ**固定枝**，**禁止枝**と呼ぶ．また，このようなサイクルを$(F,R)$-**許容**なサイクルという．$P(F,R)$を解くアルゴリズムをFind-all-NCs$(F,R)$とする．

ここで，$F=\emptyset$の場合は，$P(\emptyset,R)$中に負サイクルが存在するか否かをFind-an-NC$(R)$により検出し，存在する場合には上述のように子問題を生成してそれぞれの子問題に再帰的にFind-all-NCs()を適用すれば良い．

これに対して，$F\neq\emptyset$の場合は，この部分問題中に負サイクルが存在するか否かを正確に判定することは難しい．ここで，$(F,R)$-許容なサイクルの**上下界値**　$\bar{z}(F,R)$，$\underline{z}(F,R)$を評価することが出来れば，$\bar{z}(F,R)<0$の場合は負サイクルが得られているので上述のように子問題を生成してFind-all-NCs()を再帰呼び出しし，$\underline{z}(F,R)\geq 0$の場合は負サイクルが存在しないので，このまま**return**すればよい．$\underline{z}<0<\bar{z}$の場合は，負サイクルの存否が不明であるので，$F$の先端から出る枝のそれぞれを固定した子問題を作り，それぞれへ分枝する．

### 2.3　全列挙アルゴリズム

以上をまとめると，Find-all-NCs()は次のように記述される．

**アルゴリズムFind-all-NCs()**
入力：　固定枝集合$F$，禁止枝集合$R$.
出力：　(F,R)-許容なすべての負サイクル.
**Step1**：　F=$\emptyset$のときは**Step2**へ，そうでないときは**Step3**へ進む.
**Step2**:　$C$=Find-an-NCs$(R)$とし，$C=\emptyset$ならば**return**;そうでなければ，

*日本大学会館(1998.10.15-16)
†E-mail:kinosita@cs.nda.ac.jp

・$C = \{e^1, e^2, \ldots, e^q\}$を出力し，

・$i = 1, 2, \ldots, q-1$について，

Find-all-NCs($F \cup \{e^1, e^2, \ldots, e^i\}, R \cup \{e^{i+1}\}$)を再帰呼び出しし，

**return**;

**Step3:** $\underline{z} = \underline{z}(F, R)$を計算し，$\underline{z} \geq 0$なら**return**;

**Step4:** $\bar{z} = \bar{z}(F, R)$を計算し，$\bar{z} < 0$ならば負サイクルが得られているので，これについて**Step2**と同様の処理を行い**return**;

**Step5:** $F$の終点から出る枝の集合を$\{e^1, e^2, \ldots, e^q\}$としたとき，$i = 1, 2, \ldots, q$について，再帰呼び出しFind-all-NCs($F \cup \{e^i\}, R$)を行い**return**;

### 2.4 上下界値の算出

問題$P(F, R)$において，$(F, R)$-許容な負サイクルを求めるには，$F$の終点$s$から始点$t$へ至る最短経路を求めれば良いが，グラフ中に負の荷重を持つ枝が存在するので，ダイクストラ法[3]等を直接適用することは出来ない．

しかし，この場合にもダイクストラ法は1つの経路を与えるので，これにより上界値$\bar{z}(F, R)$が得られる．一方，下界値は$s$から$t$への単純パスが必ず$|V|-1$ステップ以内であることから，再帰式

$$d_j(k) = max_{(i,j) \in E}\{d_i(k-1) + d(i,j)\}$$

により，$\underline{z}(F, R) = d_t(|V|-1)$として求められる．

## 3 計算例

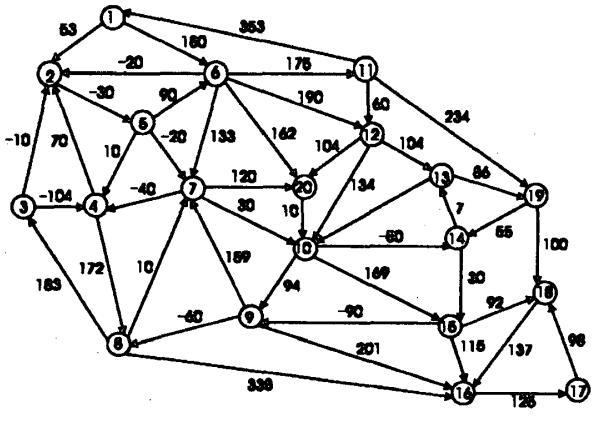


図 1: 例題NUSA1

図1のグラフについて，Find-all-NCs()を適用したところ，図2のように子問題が生成され，$2 \to 5 \to 7 \to 4 \to 2$，$7 \to 10 \to 14 \to 15 \to 9 \to 8 \to 7$，$2 \to 5 \to 7 \to 10 \to 14 \to 15 \to 9 \to 8 \to 3 \to 4 \to 2$，$2 \to 5 \to 7 \to 10 \to 14 \to 15 \to 9 \to 8 \to 3 \to 2$の4個の負サイクルが見つかった．図中，⊙はサイクルが見つかった部分問題を示し，サイクルを含まない子問題は×で表示した．この結果は，Tarjanのアルゴリズムにより全サイクルを列挙し，そのうち負サイクルのみを出力した結果と一致している．

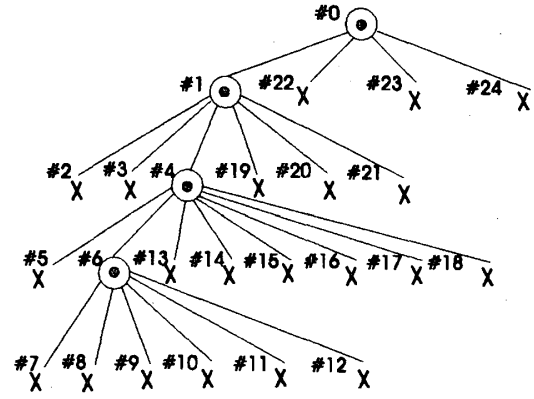


図 2: 計算の過程

## 4 むすび

Find-all-NCs()を適用した更なる実験例等については，研究発表会のときに示す．

## 参考文献


[1] R.C. Read and R.E. Tarjan: Bounds and backtrack algorithms for listing cycles, paths, and spanning trees, Networks 5: 237-252 (1975).

[2] R.E. Tarjan: Enumeration of the elementary circuit of a directed graph, SIAM J. Comput 2: 211-216 (1974).

[3] R.K. Ahuja, et al.: NETWORK FLOWS: Theory, algorithms, and applications, Prentice Hall (1993).