

木パッキング問題について (On the Tree Packing Problems)

京都大学工学部 増山 繁

Shigeru MASUYAMA

Faculty of Engineering, Kyoto University

あらまし 本稿では、木を木に最大個数詰める最大木パッキング問題[10]について調べる。まず、根付き有向木に根付き有向木を詰める場合に、最大節点パッキング(節点の重なりも辺の重なりも許さない)を求める $O(n^2)$ (n は T の節点数)のアルゴリズム、及び、最大辺パッキング(節点の重なりは許すが辺の重なりは許さない)を求める多項式時間のアルゴリズムを与えた。それに基づき、無向木への無向木の最大節点パッキングを求める $O(n^2)$ (以下では、 n は節点数)のアルゴリズム、及び、最大辺パッキングを求める多項式時間のアルゴリズムを与えた。また、根付き楕形木を根付き有向木に詰める場合、 k 星を無向木に詰める場合など、いくつかの特別の場合に対しては、 $O(n)$ のアルゴリズムを得た。

1. はじめに

木パッキング問題は、与えられた木 T に木 T_0 を互いに節点も辺も重ならないようにして最大個数詰める最大節点パッキング問題と、節点は互いに重なってもよいが辺は重ならないようにして最大個数詰める最大辺パッキング問題からなる。節点パッキング問題と、それに関連したグラフの因子分解問題は、グラフ理論における古典的な問題のひとつであって(たとえば[5]参照)、従来よく研究されており、特に、 k 鎖節点パッキング問題[9]では、 $k=1$ の場合はグラフ上のマッチング問題[8]となることから、グラフのマッチング問題の一般化とみなせる。グラフのマッチング問題に対しては効率のよい解法が知られている[8]が、 k 鎖節点パッキング問題は、 $k \geq 2$ のとき、NP完全である[4]。なお、NP完全な問題に対しては多項式時間のアルゴリズムが存在しないと一般に信じられている。更にKirkpatrickとHell[7]により、グラフを節点パッキングする問題においては、節点と辺のみからなる部分グラフを詰める場合、つまり、実質的にマッチングの場合を除いてはNP完全であることも示されている。一方、辺パッキング問題に関しては、文献[12]でグラフの構造を入れて一般化したビンパッキング問題の性質を調べ、また、Jüngerら[6]により、グラフが連結ならば、辺の集合の、長さ2の鎖への分解が可能であることが示されているくらいで、それ以外には k 鎖を詰める場合も含め、ほとんど論じられていない。

本稿では、木の最大節点パッキング問題と、最大辺パッキング問題、について新たに得た結果[10]について紹介する。詳しくは[10]を参照されたい。

2. 諸定義

本節では、以下の議論に必要な定義を与える。なお、よく知られたグラフ理論の用語については、定義せずに用いた場合もある。詳しくは、文献 [1, 2, 5] 等を参照されたい。

(無向)木とは、閉路を持たない(無向)グラフである。次数(それに接する辺数)が1の節点を葉という。楕形(k_1, \dots, k_h)木とは長さ $h-1$ の道と葉のみとからなっており、道上の i 番目の節点(軸)が丁度 k_i 個の葉を持っている木である。 k 星とは楕形(k)木のことである。

根付き有向木 $T=(V, E, r)$ 、但し、 V は節点の集合、 E は辺の集合、 r は根、は、次の条件を満たす有向木である。

1. 根 r は親をもたない。
2. 根以外のすべての節点は唯一の親をもつ。
3. 根以外のすべての節点は根から道により到達できる。

但し、 $(i, j) \in E$ のとき、 i は j の親、 j は i の子という。子を持たない節点を葉という。また、 i から j へ道があるとき、 i は j の先祖、 j は i の子孫という。なお、そのような道は唯一であることを注意せよ。根付き有向木の高さとは、根から最も遠い葉までの距離をいう。

根付き有向木 $T=(V, E, r)$ の節点 v に対し、 v を根とし、 v の子孫すべてと、 T でそれらを結ぶ辺とからなる部分木を、 $T(v)$ と記す。

根付き楕形(k_1, \dots, k_h)木とは、楕形(k_1, \dots, k_h)木の最初の軸(k_1 個の葉をもつもの)を根として根付き有向木にしたものである。

$$S = \{T_1, \dots, T_t\},$$

但し、 $T_i=(V_i, E_i)$ 、 $i=1, \dots, t$ 、 T_i は(無向または有向木) T の部分木、 T の節点パッキングとは、任意の $i \neq j$ 、 $i=1, \dots, t$ 、に対し

$$V_i \cap V_j = \emptyset$$

が成り立つことであり、 S が T の辺パッキングとは、任意の $i \neq j$ 、 $i=1, \dots, t$ 、に対し

$$E_i \cap E_j = \emptyset$$

が成り立つことである。以下、本稿では、 S の要素がすべてある T_0 に同型な場合のみを論ずる。最大辺(節点)パッキング問題とは、与えられた T に対して、 $|S|$ を最大にする辺(節点)パッキングを求める問題である。但し、 $|S|$ は集合 S の要素数を表わす。

3. 根付き有向木上の木パッキング

まず、根付き有向木 $T=(V, E, r)$ 、 r は根、への、根付き有向木 T_0 の最大節点パッキング問題を解くアルゴリズム TREE-N を導く。 r から出発して幅優先探索 [1] を行ない、訪れた順に節点に番号を付す。以下では、節点と、それに付された番号とを同一視する。次に、番号最大の節点から順に訪れ(つまり、根から遠い方の節点から順に訪れて)、以下の操作を根に到るまで行う。

節点 v を訪れたとき、 T_0 を、 v を根とする T の部分木 $T(v)$ に詰めることができるかどうか調べるため、Procedure CHECK($T_0, T(v)$) を呼び、YES ならば T_0 を $T(v)$ に詰め、 T から T_0 を詰めた部分の節点と辺、及び、その部分に接する辺をすべて取り除き、NO ならば、そのままにして、次の節点を訪れる。

Procedure CHECK($T_0, T(v)$)は以下のように実行される。 T_0 を $T(v)$ に詰めることができるかどうか調べるため、 T_0 の根 r_0 の子 $w_i, i=1, \dots, d_0$, 及び v の子 $v_j, j=1, \dots, d$ に対し、CHECK($T_0(w_i), T(v_j)$)を再帰的に呼び、2部グラフ $G=(N_1, N_2, E)$, ここで、 $N_1=\{T_0(w_i) \mid i=1, \dots, d_0\}$, $N_2=\{T(v_j) \mid j=1, \dots, d\}$, $E=\{(T_0(w_i), T(v_j)) \mid T_0(w_i) \text{は} T(v_j) \text{に詰めることができる}\}$, を作る。Gに N_1 の節点をすべて覆うマッチングが存在するときYESを、存在しないときNOを出力する。以上のアイデアを形式的に書くと次のアルゴリズムRTREE-NとProcedure CHECK(T, T')を得る。

アルゴリズムRTREE-N

入力: 根付き有向木 $T=(V, E, r)$, 及び、 T に詰められる根付き有向木 T_0 .

出力: T に節点パッキングできる根付き有向木 T_0 の最大個数 N .

ステップ1. T を根 r から順に、訪問順に番号を振りながら幅優先探索を行う。 $N \leftarrow 0, v \leftarrow n(, n=|V|)$.

ステップ2. T の節点を番号の大きい方から順に訪れ、以下の操作を行う。

節点 v を訪れたとき、Procedure CHECK($T_0, T(v)$)を呼ぶ。もし、答がYESなら、 $N \leftarrow N+1$ とし、 T_0 が詰められた部分のすべての節点と辺、及びその部分に接するすべての辺を T から取り除き、 $v \leftarrow v-1$ とする。さもなければ、 $v \leftarrow v-1$ とする。

ステップ3. もし、 v が0ならステップ4へ、さもなければ、ステップ2へ。

ステップ4. 停止。□

Procedure CHECK(T, T')

入力: 根付き有向木 $T=(V, E, r)$ と根付き有向木 $T'=(V', E', r')$.

出力: もし、 T を T' に詰めることができるならYES, さもなければ、NO.

T の根 r の子 w_1, \dots, w_d , 及び、 T' の根 r' の子 v_1, \dots, v_d に対し、CHECK($T(w_i), T'(v_j)$)を再帰的に呼び、2部グラフ $G=(N_1, N_2, E)$, ここで、 $N_1=\{T(w_i) \mid i=1, \dots, d\}$, $N_2=\{T'(v_j) \mid j=1, \dots, d\}$, $E=\{(T(w_i), T'(v_j)) \mid T(w_i) \text{は} T'(v_j) \text{に詰めることができる}\}$, を作る。そして、 N_1 の節点をすべて覆うマッチングが存在するときYESを、存在しないときNOを出力して、RETURN. □

定理1. アルゴリズムRTREE-Nにより、 $O(n^2)$ で、 T に節点パッキングできる根付き有向木 T_0 の最大個数が求まる。但し、 n は T の節点数。

(略証)アルゴリズムの正当性は、アルゴリズムのステップ2の実行に関する帰納法で示せる。また、ステップ2で T の各節点 v を1回ずつ訪れ、節点 v を訪れた際のProcedure CHECKの再帰的呼出しは、高々 v から h 以内(h は T_0 の高さ)の子孫に及ぶだけであることに注意すると、 T の各節点 v でProcedure CHECK($T_0, T(v)$)の実行に伴い、解かなければならない最大マッチング問題の回数は高々 h 回であり、最大マッチング問題を解くのに要する時間が $O(|N_1|^2|N_2|)$ である[8]ことと併せて、全体の計算時間が $O(n^2)$ であることが分る。但し、 T_0 の節点数は定数とみなすので、 $|N_1|$ 及び h は定数であることに注意せよ。□

アルゴリズムRTREE-Nでは、ステップ2で節点 v を訪れたときには、 T_0 が $T(v)$ に詰められるかどうか調べるだけでよかった。それに対し、 T_0 の T への最大辺パッキングを求める場合には、次の2点で難しくなる。

(1) $T(v)$ に T_0 をなるべく多く詰めなければならない。

(2) そのような最大個数の辺パッキングのうち、後で v の先祖を訪れたときに"有利"となる可能性のあるものはすべて残す必要がある。

まず、(1)を解決するため、次の問題Aを考える。

問題A. アルゴリズムAの直前で定義された2部グラフ $G=(N_1, N_2, E)$ が与えられたとき、次の2つの条件を満たす最大個数の辺の集合 M_1, \dots, M_k を求めよ。

条件1. $M_i, i=1, \dots, k$, はそれぞれ、 N_1 の節点をすべて被覆するマッチングである。

条件2. N_2 の各節点は、 $M_1 \cup \dots \cup M_k$ の辺により、高々一回しか被覆されない。□

問題Aは次の問題A'が解ければ k に関して2分探索($0 \leq k \leq D, D(\leq |N_2|)$ は、 N_1 に属する節点の G での最大次数)することで解くことができる。

問題A'. 2部グラフ $G=(N_1, N_2, E)$ が与えられたとき、問題Aのふたつの条件を満たす辺の集合 M_1, \dots, M_k が存在するかどうか決定せよ。□

問題A'は容易に分るように、次の問題Bと等価である。

問題B. 2部グラフ $G=(N_1, N_2, E)$ に対して、 N_1 の節点を全て含み、それらの次数がすべて k であり、 N_2 に属する節点の次数がすべて1である G の部分グラフ G_0 (次数制限マッチング)が存在するか? なお、 G_0 は N_2 の節点を必ずしもすべて含む必要はない。□

よく知られているように[2], 問題Bは容易に、 G に対し、節点 s, t , 及び、各 $x \in N_1$ に対し、辺 (s, x) を、また、各 $y \in N_2$ に対し、辺 (y, t) を付加し、更に各 $x \in N_1$ に対し、 $c(s, x)=k$, 各 $x \in N_1, y \in N_2$ に対し、 $c(x, y)=\infty$, また、各 $y \in N_2$ に対し、 $c(y, t)=1$, 但し、 $c(i, j)$ は辺 (i, j) の容量、として得られるネットワーク S 上の最大フロー問題に帰着することができ、たとえば、SleatorとTarjanの方法[11]($O(mn \log n)$), 但し、 n, m は、それぞれネットワークの節点数と辺数)を用いると、 $O(|N_1| |N_2| (|N_1| + |N_2|) \log(|N_1| + |N_2|))$ 時間で解くことができる。

次に、上記の(2)を解決するため、動的計画法を用いる。すなわち、ステップ2で節点 v を訪れたとき、 v の子 $v_i, i=1, \dots, d$, に対して残されている v_i を根とする各部分木 $T(v_i, j)$ から $v_i, i=1, \dots, d$ に対してひとつずつ選んで得られる各組合せそれぞれに対して最大個数の辺パッキングを求め、全体での最大 k_M を求める。ところが、節点 v の子の数を d とすると、 k_M を求めるためには、 c^d 個の v の子を根とする部分木の組合せを考えなければならない。そのままでは、計算時間を多項式時間に押えるためには、 T の最大次数を定数としなければならない。そこで、問題Bの代わりに、次の問題B'を考える。

問題B'. $N'_1 = \{T_0(w_i) | w_i \text{は } T_0 \text{の根 } r_0 \text{の子}\}, N'_2 = \{T(v_i, j) | v_i \text{は } v \text{の子}\}, E' = \{(T_0(w_i), T(v_p, j)) | T_0(w_i) \text{は } T(v_p, j) \text{に詰めることができる}\}$ として、2部グラフ $G''=(N'_1, N'_2, E')$ を作り、更に、節点 $s, t, 1, 2, \dots, d$ を追加し、また、各 $x \in N'_1$ に対し、辺 $(s, x), j=1, \dots, d$ に対し、辺 $(j, t), v$ の子 v_j を根とする部分木に対応する各 $y \in N'_2$ に対し、辺 (y, j) , また、各辺の容量を、各 $x \in N'_1$ に

し、 $c(s, x) = k$, 各 $x \in N'_1$, $y \in N'_2$ に対し、 $c(x, y) = \infty$, 各 $y \in N'_2$ に対し、 $c(y, j) = \infty$, $j = 1, \dots, d$ に対し、 $c(j, t) = 1$, として得られる、ネットワーク S' が実行可能流れを持つか。□

問題 B' は、最大フロー問題を解くことによって、解くことができる。その解を k_M とする。次に、 T_0 の各節点 u に対して、次の操作を行う。 u の子 $u_1, \dots, u_{t(u)}$, $t(u)$ は u の子の数、に対して v の各子 v_i , $i = 1, \dots, d$, に対応する部分木を丁度一個ずつ、計 $t(u)$ 個だけ選び、それらからなる部分木に $T_0(u)$ が詰められるかどうか調べる。詰められるとき、それら計 $t(u)$ 個を除いた残りの v の子を根とする部分木をひとつずつ選んだすべての組合せに対して k_M 個のパッキングが可能か問題 B' と同様にして調べ、詰められる組合せが存在する場合はひと組だけ残して T_0 を k_M 個詰め、その部分の辺を取り除く。そのようにして得られた部分木のうちで重複を省き、他に含まれない $T(v, j)$ のみ残せばよい。

以上のアイデアを形式的に書くと、次のアルゴリズム $RTREE-E$ を得る。

アルゴリズム $RTREE-E$

入力: 根付き有向木 $T = (V, E, r)$, 及び、 T に詰められる根付き有向木 T_0 .

出力: T に辺パッキングできる根付き有向木 T_0 の最大個数 N .

ステップ 1. T を根 r から順に、訪問順に番号を振りながら幅優先探索を行う。 $N \leftarrow 0$, $v \leftarrow n$ ($n = |V|$).

ステップ 2. T の節点を番号の大きい方から順に訪れ、以下の操作を行う。

節点 v を訪れたとき、 v の子 v_i , $i = 1, \dots, d$ に対して残されている v_i を根とする部分木 $T(v_i, j)$ に対して、問題 B' を k に関する 2 分探索を行ないながら解くことにより、 k の最大値 k_M を求める。次に問題 B' の直後に述べたようにして部分木 $T(v, j)$ を生成し、 $N \leftarrow N + k_M$, $v \leftarrow v - 1$ とする。

ステップ 3. もし、 v が 0 ならステップ 4 へ、さもなければ、ステップ 2 へ。

ステップ 4. 停止。□

定理 2. アルゴリズム $RTREE-E$ により、根付き有向木 T_0 の最大辺パッキングが多項式時間で求まる。□

もし、 T_0 が根付き楕形 (k_1, \dots, k_h) 木 (h は定数) なら、アルゴリズム $RTREE-N$ のステップ 2 において次の Procedure $COMBCHECK(k_1, \dots, k_h, T(v))$ を用いることによって、計算の複雑度を $O(n)$ に下げることができる。このようにして得られるアルゴリズムを $RCOMB-N$ と呼ぶ。

Procedure $COMBCHECK(k_1, \dots, k_h, T(v))$

v の子を v_1, \dots, v_d とする。 $i = 1, \dots, d$ に対して Procedure $COMBCHECK(k_2, \dots, k_h, T(v_i))$ を再帰的に呼ぶ。少なくともひとつ YES, かつ、 $k_1 \leq d - 1$ なら YES, さもなければ、NO。□

定理3. アルゴリズムRCOMB-Nにより, $O(n)$ で根付き櫛形 (k_1, \dots, k_h) 木の最大節点パッキングが求まる. \square

最大辺パッキング問題の場合も, 根付き櫛形 (k_1, \dots, k_h) 木を詰める場合は, 以下のよう
に計算複雑度を下げることが可能である.

アルゴリズムRCOMB-E

入力: 根付き有向木 $T=(V, E, r)$, 及び, T に詰められる根付き櫛形 (k_1, \dots, k_h)
木 T_0 .

出力: T に辺パッキングできる T_0 の最大個数 N .

ステップ1. T を根 r から順に, 訪問順に番号を振りながら幅優先探索を行う. $N \leftarrow 0, v \leftarrow r (n = |V|)$.

ステップ2. T の節点を番号の大きい方から順に訪れ, 以下の操作を行う.

節点 v を訪れたとき, n_1 を, v の子 v_i のうち, それを根とする部分木 $T(v_i, j)$ で, 根
付き櫛形 (k_2, \dots, k_h) 木を詰めることのできるものがあるものの個数, n_2 を, そうでない
 v の子の個数とする. すると,

$n_1 \leq \lfloor n_2 / k_1 \rfloor$ ならば n_1 個の根付き櫛形 (k_1, \dots, k_h) 木を詰める. さもなければ,
 $\lfloor n_2 / k_1 \rfloor + ((n_2 - k_1 \lfloor n_2 / k_1 \rfloor) + \lfloor (n_1 - \lfloor n_2 / k_1 \rfloor) / (k_1 + 1) \rfloor)$ 個の根付き櫛形 $(k_1, \dots,$
 $k_h)$ 木を詰める. そのような v における最大個数の詰め方から, 根付き櫛形 $(k_{h-i+1}, \dots,$
 $k_h)$ 木が詰められる部分木 $T(v_i, i)$ があればそれぞれひとつずつ残す.

ステップ3. もし, v が 0 ならステップ4へ, さもなければ, ステップ2へ.

ステップ4. 停止. \square

定理4. アルゴリズムRCOMB-Eにより, $O(n)$ で根付き櫛形 (k_1, \dots, k_h) 木の最大
辺パッキングが求まる. \square

4. 無向木への無向木のパッキング

本節では, まず, 無向木 T への無向木 T_0 の最大節点パッキングを求めるアルゴリズムTRE
EE-Nを導入する. TREE-Nでは, まず, T の任意の節点 r を根として選び, r から訪
問順に番号を振りながら幅優先探索を行う. 以下では得られた根付き有向木も T で表わす. 次
に, T を, 番号の大きな節点から順に訪れ, 以下の操作を行う. 節点 v を訪れたときを考え
る. 根付き有向木上のパッキングの場合には, T_0 の根が指定されていたが, 無向木の場合に
は, T_0 のどの節点を根として T に詰めてもよい. そこで, T_0 の各節点 u に対し, u を根とす
る根付き有向木 T_0^u を作り, CHECK($T_0^u, T(v)$)をそれぞれ実行する. ひとつでもYE
Sなら, 詰められる. 詰められる場合には, T_0 を詰めた部分の節点, 辺, 及び, それに隣接
する全ての辺を T から除去して, すべてNOならそのまま, いずれも, 次の節点を訪れる.

以上のアイデアを形式的に書くと, 次のアルゴリズムTREE-Nを得る.

アルゴリズムTREE-N

入力: 無向木 $T=(V, E)$, 及び, T に詰められる無向木 T_0 .

出力: T に節点パッキングできる無向木 T_0 の最大個数 N .

ステップ1. T の節点 r を任意に選び, r から順に, 訪問順に番号を振りながら幅優先探索を行い, 得られた根付き有向木も T と呼ぶ. r を根とする根付き有向木をつくる. $N \leftarrow 0$, $v \leftarrow n$ ($n = |V|$).

ステップ2. T_0 の各節点 u に対してProcedure CHECK(T_0^u , $T(v)$)を呼ぶ. もし, 少なくともひとつ答がYESなら, $N \leftarrow N + 1$ とし, T_0 を詰めた部分のすべての節点と辺, 及びその部分に接するすべての辺を T から取り除き, $v \leftarrow v - 1$ とする. さもなければ, $v \leftarrow v - 1$ とする.

ステップ3. もし, v が0ならステップ4へ, さもなければ, ステップ2へ.

ステップ4. 停止. \square

定理1と同様にして, 次の定理を得る.

定理5. アルゴリズムTREE-Nにより, $O(n^2)$ で T に詰められる無向木 T_0 の最大個数が求まる. \square

次に, 無向木 T への無向木 T_0 の最大辺パッキングを求める多項式時間のアルゴリズムTREE-Eを導く.

ステップ2で節点 v を訪れたとき, 根付き有向木の場合と異なり, T_0 のどの節点を根として詰めるかという自由度がある. そこで, RTREE-Eで述べた問題Aの代わりに, 次の問題 A_0 を考える.

問題 A_0 . 問題Aの2部グラフ $G = (N_1, N_2, E)$ に対して, $N_{1u} = \{T_0^u(w_i) \mid w_i \text{は } u \text{の子}\}$, $u = 1, \dots, n_0$. 但し, n_0 は T_0 の節点数, を作り, その和集合を N'_1 とする.

$T_0^u(w_i) \in N'_1$ が $x \in N_2$ に詰められるときかつそのときに限り, x と辺で結ぶ, として得られるグラフを $G' = (N'_1, N_2, E')$ とする. そのとき, 次の条件を満たす辺の集合 M_{ui} , $i = 1, \dots, k_u$, を最大個数とる.

条件1. M_{ui} は N_{1u} , かつ, それのみに属する節点全てを被覆するマッチングである.

条件2. N_2 の各節点は, $\cup M_{ui}$ の辺により, 高々一回しか被覆できない. \square

問題 A_0 は次の問題 A_0' が解ければ k に関して2分探索($0 \leq k \leq D$, D は, N_1 に属する節点の最大次数)することで解くことができる. ただし, $k = k_1 + \dots + k_{n_0}$ を満たす組 k_1, \dots, k_{n_0} すべてについて解かなければならない.

問題 A_0' . 2部グラフ $G' = (N'_1, N_2, E)$ が与えられたとき, $k = k_1 + \dots + k_{n_0}$ を満たす組 k_1, \dots, k_{n_0} について問題 A_0 の2つの条件を満たす辺の集合 M_{u1}, \dots, M_{uku} が存在するかどうか決定せよ. \square

問題 A_0' は容易に分るように, 次の問題 B_0 と等価であり, また, 問題 B_0 は, 問題Bと同様にして最大フロー問題に帰着して解ける.

問題 B_0 . 2部グラフ $G' = (N'_1, N_2, E)$ に対して, N_{1u} , $u = 1, \dots, n_0$ の節点を全て含み, それらの次数がすべて k_u であり, N_2 に属する節点の次数がすべて1である次数制限マッチングを持つか? \square

更に, 問題 B' と同様にして, 計算時間を多項式に保つため, 次の問題 B'_0 を考える.

問題B'₀. $N'_2 = \{T(v_p, j) \mid v_p \text{ は } v \text{ の子}\}$, $E' = \{(T_0(w_i), T(v_p, j)) \mid T_0(w_i) \text{ は } T(v_p, j) \text{ に詰めることができる}\}$ として, 2部グラフ $G' = (N'_1, N'_2, E')$ を作り, 更に, 節点 $s, t, 1, 2, \dots, d$ を追加し, また, 各 $x \in N'_1$ に対し, 辺 (s, x) , $j = 1, \dots, d$ に対し, (j, t) , v の子 v_j を根とする部分木に対応する各 $y \in N'_2$ に対し, (y, j) , また, 各辺の容量を, 各 $x \in N'_1$ に対し, $c(s, x) = k$, 各 $x \in N'_1, y \in N'_2$ に対し, $c(v, y) = \infty$, 各 $y \in N'_2$ に対し, $c(y, j) = \infty$, $j = 1, \dots, d$ に対し, $c(j, t) = 1$, として, ネットワーク S' が実行可能流れを持つか. □

問題B'₀は, 最大フロー問題を解くことによって, 解ける. その解を k_M とする. あとは, 問題B'の直後に述べたようにして部分木 $T(v, j)$ を生成する.

アルゴリズム TREE-E

入力: 無向木 $T = (V, E)$, 及び, T に詰められる無向木 T_0 .

出力: T に辺パッキングできる無向木 T_0 の最大個数.

ステップ1. T の節点 r を任意に選び, r から順に, 訪問順に番号を振りながら幅優先探索を行う. $N \leftarrow 0, v \leftarrow n$ ($n = |V|$). 得られた根付き有向木も T と呼ぶ.

ステップ2. T の節点を番号の大きい方から順に訪れ, 以下の操作を行う.

節点 v を訪れたとき, v の子 $w_i, i = 1, \dots, d$ に対して残されている w_i を根とする部分木 $T(w_i, j)$ に対して, 問題B'₀を k に関する2分探索を行ないながら解くことにより, k の最大値 k_M を求める. あとは, 問題B'の直後に述べたようにして部分木 $T(v, j)$ を生成する. $v \leftarrow v - 1$ とする.

ステップ3. もし, v が0ならステップ4へ, さもなければ, ステップ2へ.

ステップ4. 停止. □

定理6. アルゴリズム TREE-Eにより, 無向木 T に対する無向木 T' の最大辺パッキングが多項式時間で求まる. □

次に, 無向木に, 特に k 星を詰める場合, 節点パッキング, 辺パッキングのいずれも $O(n)$ で解けることを示す.

まず, k 星の最大節点パッキングを $O(n)$ で求めるアルゴリズム k STAR-Nから述べる. やはり, まず, T の任意の節点 r を根として選び, r から順に番号を振りながら幅優先探索を行い, 根付き有向木とする(それも T と呼ぶ). 次に, T の節点を番号の大きな方から順に訪れ, 以下の操作を行う.

節点 v を訪れたとき, k 星を詰めることができるかどうか調べる. そのとき, $T(v)$ への k 星の詰め方には, 以下の2通りがあることに注意せよ.

型1. 葉が v に重なる詰め方.

型2. 軸が v に重なる詰め方.

k 星の型1かまたは型2のいずれかの詰め方が可能かどうか調べる. 可能なら詰め, 詰めた部分の節点及び辺, 及び, その部分に隣接する辺をすべて T から除去し, 次の節点を訪れる. 紙数の都合で詳細は略す.

次に、 k 星の最大辺パッキングを $O(n)$ で求めるアルゴリズム kSTAR-E も以下のようにして得るが、紙数の都合で詳細は略す。ステップ2で節点 v を訪れたとき、まず、型1の詰め方をできるだけ多く行う。次に、残りの部分木に、型2の詰め方をなるべく多く行う。

本節を閉じるにあたり、楕形 (k_1, k_2) 木の無向木への節点及び辺パッキングが、いずれも $O(n)$ で解けることを示そう。

まず、節点パッキングについては、ステップ2で、節点 v を訪れたとき v の子、 w_1, \dots, w_d のうち、 k_1 個以上の子を持つものがありかつ $d-1 \geq k_2$ 、または、 k_2 個以上の子を持つものがありかつ $d-1 \geq k_1$ 、なら詰める。詳細は略す。

次に、楕形 (k_1, k_2) 木の無向木 T への最大辺パッキングを $O(n)$ で求めるアルゴリズム 2COMB-E では、節点 v を訪れた際、なるべく多く、根付き楕形 (k_1, k_2) 木、又は、根付き楕形 (k_2, k_1) 木を詰めるが、そのとき、 $k_1 \geq k_2$ とすると、まず、根付き楕形 (k_2, k_1) 木を優先し、次に根付き楕形 (k_1, k_2) 木を詰める。詳細は略す。

5. むすび

今後の課題として、グラフパッキング問題で、今までに得た以外の効率良く解ける部分クラスを求め、多項式時間で解ける限界を明らかにすること、良い近似解法を開発し、その性能評価を行うこと、などがある。

謝辞 末筆ではあるが、日頃ご指導賜わる京都大学工学部の茨木俊秀教授と長谷川利治教授、及び、関西大学工学部の三根久教授(京都大学名誉教授)に深謝する。また、第3節の問題Aが問題Bに帰着できることを示唆していただいた、京都大学工学部の永持仁氏に謝意を表す。なお、本研究は一部文部省科学研究費によるものである。

文 献

- [1] Aho, A. V., Hopcroft, J. E. and Ullman, J. D., The Design and Analysis of Computer Algorithms, Addison-Wesley (1974) (邦訳, 野崎, 野下, 共訳, アルゴリズムの設計と解析 I, II, サイエンス社(昭和52)).
- [2] Berge, C., Graphes et Hypergraphes, Dunod (1970); C. ベルジュ著, 伊理他訳, グラフの理論 I, サイエンス社(1976).
- [3] Boesch, F. T. and Gimpel, J. F., "Covering the points of a digraph with point-disjoint paths and its application to code optimization", J. Assoc. Comput. Mach., 24, pp.192-198 (1977).
- [4] Garey, M. R. and Johnson, D. S., Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H., Freeman and Company, San Fransisco, (1979).
- [5] Harary, F., Graph Theory, Addison-Wesley, Reading Mass.; 池田貞雄(訳), グラフ理論, 共立出版(昭和46).
- [6] Jünger, M., Reinelt, G. and Pulleyblank, W. R., "On partitioning the edges of graphs into connected subgraphs", J. Graph Theory, Vol.9, pp.539-549 (1985).

- [7] Kirkpatrick, D. G. and Hell, P., "On the complexity of general graph factor problems", SIAM J. Comput. Vol.12, No.3, pp.601-609(1983).
- [8] Lawler, E. L., Combinatorial Optimization: Networks and Matroids, Holt Rinehalt and Winston(1976).
- [9] Masuyama, S. and Ibaraki, T., "Computational complexity of chain packing problems", Technical Report #87001, Department of Applied Mathematics & Physics, Faculty of Engineering, Kyoto University, Kyoto 606, Japan, (1987).
- [10] Masuyama, S., "On the tree packing problems", manuscript under preparation for the Technical Report of Department of Applied Mathematics & Physics, Faculty of Engineering, Kyoto University, Kyoto 606, Japan, (1987).
- [11] Sleater, D. D. and Tarjan, R. E., "A data structures for dynamic trees", Proc. of the 13th Annual ACM Symposium on Theory of Computing, pp.114-122 (1981).
- [12] Zhang, Ze-Zeng, Masuyama, S., Ibaraki, T. and Mine, H., "Graph packing over a rooted tree", to appear in Memoirs of the Faculty of Engineering, Kyoto University, Vol.49, No.2, Kyoto, Japan.