社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－AL－109 (4)
2006／11／21

平面グラフの分枝幅決定アルゴリズムの効率的実装

玉木久夫　　吉武由実

明治大学理工学部情報科学科
〒214-8571　神奈川県川崎市多摩区東三田 1-1-1
tamaki@cs.meiji.ac.jp, yyoshi@cs.meiji.ac.jp

あらまし　平面グラフの分枝幅を決定する Seymour と Thomas のアルゴリズムを実装した。その際、このアルゴリズムの背後にあるネズミ捕りゲームの性質を用いた 3 種類の改良を試みた。改良を実装に組み込んだ結果、メモリ節約が可能となり、素朴な方法で扱うことができなかった辺数の多い例題を扱うことができるようになった。多くの例題で、これまでの研究結果と比べて、この実装が高速であることを確認した。

Efficient Implementation of the Algorithm for Computing the Branchwidth of Planar Graphs

Hisao Tamaki , Yumi Yoshitake

Department of Computer Science,Meiji University
1-1-1 Higashi-mita, Tama-ku, Kawasaki-shi, Kanagawa 214-8571

Abstract In this paper,we implement an algorithm of Seymour and Thomas for computing the branchwidth of planar graphs. We propose three improvements based on some observations on the ratcatcher game underlying their algorithm.The improved implementation can save memory usage and deal with the graphs that have too many edges to be dealt with by naive implementations.Experiments show that our implementation is faster on large instances than previously published ones.

1　はじめに

グラフの分枝分割 (branch decomposition) と分枝幅 (branch width) の概念は Robertson と Seymour[1] により、有名な木分割と木幅の概念と関連して導入された。木幅と分枝幅は線形の関係にあることが知られている。グラフ G の最適な分枝分割が与えられているとする。すると、グラフ G の木分割の幅について上界を知ることができる。そして、多くの組み合わせ問題に対して、グラフのサイズに関して線形時間で、分枝分割幅に関して指数時間で動作するアルゴリズムが存在する。したがって、与えられたグラフの分枝幅を決定する問題や最適な分枝分割を構成する問題は重要である。

与えられたグラフ G の分枝幅を決定する問題は NP 完全問題であるのだが、G が平面グラフであるとするならば、Seymor と Thomas[2] が考案した ST アルゴリズムによって $O(n^2)$ 時間で解くことができる。ST アルゴリズムでは、平面グラフ最

適分枝分割構成には$O(n^4)$の計算時間が必要である。Gu-Tamaki[3]のアルゴリズムによって、計算時間は$O(n^3)$となった。どちらのアルゴリズムにおいても、STアルゴリズムによる分枝幅決定が核となる。STアルゴリズムを実装するためには、使用メモリ量が$O(n^2)$必要である。そのため、グラフGの辺数が10000を超えるとメモリ量が実行する上で制約となってしまう。Hicksは[4]で、メモリを節約するために、基礎情報を格納せずに再計算する手法を提案している。

この論文でも、STアルゴリズムの効率的な実装と実験結果について述べる。我々は、メモリ節約を再計算によらず、出来るだけ局所的に計算を完結させる方針により達成した。そのためにST法の背後にあるネズミ捕りゲームの性質を用いている。これにより、素朴な方法ではメモリ不足のために解く事ができない例を扱うことができるようになった。そして、Hicks[4]の実験データのうち、入手可能であった6.5割（頂点数51個から14051個までの23例）について実験を行い、頂点数の大きい例題についてHicks[4]の実装よりも高速であることを確認した。

2 諸定義

グラフGの分枝分割Bは、Bの葉頂点の集合が$E(G)$であり、Bの任意の内部頂点の次数が3であるような木である。Bの各辺eは、$E(G)$の2分割(E_1, E_2)と自然に対応する。ここで、E_1とE_2はBをeによって切断して得られる2つの部分木における葉の集合を表す。E_1とE_2の共有する頂点の個数をBにおける辺eの幅と呼ぶ。Bの幅とは、Bの辺の幅の最大値のことであり、Gの分枝幅とは、Gの分枝分割を全て考えたときのそれらの幅の最小値のことである。

STアルゴリズムでは、平面グラフの分枝分割の問題を次に述べる刻み分割の問題に帰着する事によって解く[2]。分枝分割がグラフの辺の集合の階層的分割を表すのに対し、刻み分割は同様な表現でグラフの頂点の集合の階層的分割を表わす。

グラフGの刻み分割Cは、Cの葉頂点の集合が$V(G)$であり、Cの任意の内部頂点の次数が3であるような木である。Gの刻み分割Cの各辺eは、$V(G)$の2分割$(V1, V2)$と自然に対応する。ここで、$V1$と$V2$はCをeによって切断して得られる2つの部分木における葉の集合を表す。$V1$と$V2$を結ぶ辺の本数をCにおける辺eの幅と呼ぶ。刻み分割Cの幅とは、Cの辺の中で一番大きい辺の幅のことであり、グラフGの刻み幅とは、Gの刻み分割を全て考えたときのそれらの幅の最小値のことである。

3 平面グラフの刻み幅決定アルゴリズム

平面グラフGの分枝分割問題は実際はGを変換して得られる多重平面グラフの刻み分割に帰着される。以下では,記述の簡明さのために単純平面グラフについての刻み幅決定について述べるが、多重平面グラフへの一般化は容易である。刻み幅決定をするために、ネズミ捕りゲーム$RC(G,k)$を考える。

3.1 ネズミ捕りゲーム$RC(G,k)$

プレイヤーはネズミとネズミ捕り機の二人である。ゲームをするボードは家の間取り図で、平面グラフGと相当している。部屋は、Gの面に相当し、壁は辺に相当している。

1. ネズミ捕り機が好きな部屋を選ぶ
2. それに応じてネズミは好きな部屋の隅(頂点)を選ぶ。（ネズミとネズミ捕り機は常に、お互いのいる場所をわかっている）
3. ネズミ捕り機は自分がいる部屋の周りの部屋の中から、次へ行く部屋を選び、そこへ向かうためにドア（今いる面と次に行く面の間の辺）へ向かう。ここで、ネズミ捕り機はノイズを発生させる。ノイズは壁をある程度突き抜ける。突き抜ける条件は後述する。

4. ネズミが移動する。ネズミはノイズが突き抜けている壁を通ることは出来ない。しかしノイズが突き抜けていない壁ならば、いくつでも伝って、今とは異なる隅に行くことが出来る。そのまま、今と同じ隅にいてよい。

5. ネズミ捕り機は予定していた次の部屋に行く。この時点で気が変わって、今までいた部屋に戻ることは出来ない。ネズミ捕り機はノイズを常に出し続けている。

6. もしネズミが, 次数が k 以下の隅にいて、ネズミ捕り機が、ネズミがいる隅と接している部屋にたどり着いたならネズミ捕り機の勝ちである。たどり着いていないなら、3に戻る。

ネズミ捕り機の出すノイズが壁 e を貫く条件を示す。G の双対グラフを G* とする。G の面 r に対応した G^* の頂点を r^*,G の辺 e に対応した G^* の辺を e^* とする。ネズミ捕り機が扉 e にいるとき、壁 f をノイズが突き抜けるということは、G*において e^* と f^* を通る長さ k 以下の閉じた歩道があるということである。同様に, ネズミ捕り機が部屋 r にいるとき、壁 f をノイズが突き抜けるということは、G^* において r^* と f^* を通る長さ k 以下の閉じた歩道があるということである。ネズミが勝つということは、ネズミがネズミ捕り機に捕まらずに逃げ続けることが出来るということである。

定理 3.1 (Seymour and Thomas[2]). 平面グラフ G が幅 k 以下で刻み分割可能であることの必要十分条件は $RC(G,k)$ においてネズミ捕り機が必勝であることである。

3.2 ゲームのグラフ化

$RC(G,k)$ のゲームの状態を頂点、状態遷移を辺とする二部グラフ $H(G,k)$ を次のように定義する。ネズミ捕り機が面 r にいるときにノイズが通過する辺を G から取り除くことによってできる G の部分グラフを G_r とする。同様に、ネズミ捕り機が辺 e にいるときにノイズが通過する辺を取り除くことによってできる G の部分グラフを G_e とする。G_r、G_e の連結成分の集合をそれぞれ、C_e、C_r と表す。G の辺 e と面 r が接しているとき、$E(G_r) \subseteq E(G_e)$ であり、従って、$V(G)$ の分割 C_r は C_e の細分である。よってネズミ捕り機が面 r から辺 e に移動したいときに、ネズミが移動できる頂点集合 $c \in C_e$ は、ネズミがどの $c' \in C_r$ の中にいたかだけに依存し、c' のどの頂点にいたかには依存しない。つまり、ネズミ捕りが面 r にいるとき、ゲームの状態は (r,c') によって表すことができる。

$H(G,k)$ の頂点集合は S,T であり、$R(G)$ を G の面の集合とすると、$S = \{(r,c)|r \in R(G), c \in C_r\}$ はネズミ捕り機が面にいるときの状態の集合である。$T = \{(e,c)|e \in R(G), c \in C_e\}$ はネズミ捕り機が辺にいるときの状態の集合である。そして、$(r,c) \in S$ と $(e,c') \in T$ の間に $H(G,k)$ の辺があるのは、これらの状態間で遷移が可能であるときである。すなわち、r と e が接していて、かつ、$c \subseteq c'$ であるときである。S の頂点 (r,c) からは、ネズミ捕り機が次に向かう面 r' と r の間の辺 e を選択することによって、T の頂点 (e,c') へ遷移する。一方、T の頂点 (e,c) からの遷移はネズミの選択した頂点 v とネズミ捕り機が移動する先である面 r' によって、$v \in c''$ である S の頂点 (r',c'') と決まる。

3.2.1 ネズミ捕りゲームの勝敗判定

もし、グラフ G の次数が k より大きい場合、ネズミは k よりも大きい次数を持つ頂点にいれば捕まることはないので、ネズミが勝つことが判る。よってこれ以後はグラフ G の次数が k 以下である場合の判定法を述べる。$RC(G,k)$ における、ネズミ捕り機必勝の状態を表す集合を以下のように帰納的に定義する。

1. $\hat{T}_0 = \phi$

2. $\hat{S}_0 = \{(r,\{v\})|G$ の面 r と頂点 v は接している $\}$

3. $\hat{T}_{i+1} = \{(e,c)|e$ と接しているある面 r と $c' \subseteq c$ なるすべての $c' \in C_r$に対して $(r,c') \in \hat{S}_i\} \cup \hat{T}_i$

4. $\hat{S}_{i+1} = \{(r,c)|r$ と接しているある辺 e と $c' \supseteq c$ なるある $c' \in C_e$ に対して $(e,c') \in \hat{T}_{i+1}\} \cup \hat{S}_i$

$\hat{S} = \bigcup_i S_i, \hat{T} = \bigcup_i T_i$ とおく。各 $\hat{S}_i$ に属する状態がネズミ捕り機必勝状態であることを数学的帰納法により証明する。$\hat{S}_0$ はネズミ捕り機がネズミを捕まえゲームが終了する状態の集合であるので、$i = 0$ のときは主張はなりたつ。次に、$\hat{S}_i$ がネズミ捕り機必勝状態の集合であると仮定する。状態 (r,c) が $\hat{S}_{i+1} - \hat{S}_i$ に属すならば、規則4より r と接するある辺 e と $c' \supseteq c$ なる $c' \in C_e$ があって、$(e,c') \in \hat{T}_{i+1}$ である。もし $(e,c') \in \hat{T}_i$ であるならば、$(r,c) \in \hat{S}_i$ であるので、$(e,c') \in \hat{T}_{i+1} - \hat{T}_i$ である。規則3より、e と接するある面 r' があって $c'' \subseteq c'$ なる すべての $c'' \in C_{r'}$ に対して $(r',c'') \in \hat{S}_i$ である。もし $r = r'$ ならば $(r,c) \in \hat{S}_i$ となって仮定に反するので、r' は e に関して r の反対側の面である。状態 (r,c) において、ネズミ捕り機が r' への移動を選択するとき、ネズミの選択後の状態はいずれも $\hat{S}_i$ に属し、帰納法の仮定によりその状態はネズミ捕り機必勝である。これにより帰納法の証明は完了する。

次に、$S - \hat{S}$ に属す状態はネズミ必勝であることを示す。実際、もし (r,c) が $\hat{S}$ にも属さないとすると、規則4が適用できないことにより、r と接するどの辺 e に対しても状態 (e,c_e), $c_e \supseteq c$, $c_e \in C_e$ は $\hat{T}$ に属さない。規則3が適用できないことより、r と辺 e をはさんで隣接する面 r_e に対してある $c'_e \in C_{r'}$、$c'_e \subseteq c_e$ が存在して状態 (r_e,c_e) は $\hat{S}$ に属さない。したがって、ネズミ捕り機がどの辺 e を選択してもネズミは c'_e を選択することにより、ふたたび安全な状態に落ち着くことができる。

勝敗判定アルゴリズムの実装においては、$H(G,k)$ から頂点を削除することにより $\hat{S}$、$\hat{T}$ を表現する。実装を工夫することにより、この計算は $O(n^2)$ で実行することができる [2]。また、次の観察により、$\hat{S}$ を全て計算しなくても判定できる場合がある。

観察 3.1. ある面 r に対して、すべての $(r,c), c \in C_r$ が $\hat{S}$ に属するならば、$S = \hat{S}$ である。つまり、ゲームはネズミ捕り必勝である。

4 試みた改良

今回、3つの改良を試みた。改良を述べるために以下の定義を行う。R を G^* において連結な面の集合とし、E_R を R の面と接する辺の集合とする。S,T の部分集合 S_R, T_R を $S_R = \{(r,c)|r \in R, c \in C_r\}, T_R = \{(e,c)|e \in E_R, c \in C_e\}$ と定義し、H_R を $S_R \cup T_R$ によって誘導される $H(G,k)$ の部分グラフとする。$\hat{S},\hat{T}$ の構成規則を H_R の中だけで、適用して得られる $\hat{S},\hat{T}$ の部分集合を $\hat{S}_R, \hat{T}_R$ で表す。$\hat{S}_R$ はネズミ捕りが R, E_R を動いて勝つことのできる状態の集合を表している。

面 $r \in R$ に対して $(r,c) \in S_R - \hat{S}_R$ なる $c \in C_r$ が唯一であるとき、c を R における面 r のサバイバル成分と呼ぶ。サバイバル成分が $\hat{S}$ に属さないことは、ネズミが勝つために必要である。$\hat{S}$ の帰納的定義において、$\hat{S}_0$ を $\hat{S}_0 \cup X \subseteq S$ で、$\hat{T}_0$ を $Y \subseteq T$ で置き換えて得られる集合を、それぞれ $\hat{S}(X,Y), \hat{T}(X,Y)$ で表す。$\hat{S}_R(X,Y), \hat{T}_R(X,Y)$ も同様に定義する。

4.1 改良1. 帰納的に計算することなく、簡単な計算により、ネズミ捕り機必勝であることが判る状態を見つける

ネズミ捕り機が面 r にいて、ネズミが頂点 v から一歩も動けずにネズミ捕り機に近づかれて、捕まってしまうための十分条件の真偽を調べる。

定理 4.1. 次数が k 以下である平面グラフ G において、r を G の面、v を G の頂点とする。以下の条件を満たす、ある二つの面 s,t があるならば、$(r,v) \in \hat{S}$ である。

1. G^* において、r^* から最短距離で s^* まで行き、v^* の周りを時計回りして t^* まで行き、r^* へ最短距離で戻ってくる閉じた歩道 C_1 の長さが k 以下である
2. r^* から最短距離で s^* まで行き、v^* の周りを反時計回りして t^* まで行き、r^* へ最短距離で戻ってくる閉じた歩道 C_2 の長さが k 以下である

証明. G の面 r と頂点 v に対して、定理の条件を満たす s,t,C_1,C_2 が存在すると仮定する。v に接続するどの辺も、その双対が C_1 または C_2 に属するので、C_1 と C_2 の共有する任意の辺 e^* に対して、v は G_e の孤立頂点である。つまり、ネズミ捕り機は、r から C_1 と C_2 の共有する辺のみを通って s に達することができるので、$(r,v) \in \hat{S}$ である。 □

実装の容易さのため、v と接する面の中で、2つの面 r,s を決めておく。r,s は G^* において、v^* の周りを時計回りに r^* から G 実行する際は、v と接する面の中で二つの面 s,t を決めておく。s,t は G^* において、v^* の周りを、時計回りに s^* から t^* まで通ったときの距離と、反時計回りに s^* から t^* まで通ったときとの最大値が最小になるように選ぶ。定理の条件を満たした状態の集合を X とし、$\hat{S}(X,\phi)$ を帰納的に計算することにより $\hat{S}$ を求める。

4.2 改良2. $H(G,R)$, および、$\hat{S},\hat{T}$ の漸増的な構成

G のある面 r_0 から、幅優先探索で面の集合 R を次第に増大させながら、$S_R,T_R,\hat{S}_R,\hat{T}_R$ を計算していく。

前に述べたように, $\hat{S},\hat{T}$ は S,T から要素を削除することによって表現するので、使用メモリ量の節約が期待できる。実験により、改良1と組み合わせたときに効果が著しいことが判った。観察3.1から、$H(G,k)$ 全体を構成せずともアルゴリズムを停止することができる場合があるので、この方法は計算時間の改良にもなる可能性がある。

4.3 改良3. サバイバル成分の特別扱い

以下の定理を用いる。

定理 4.2. R を G^* において連結な面の集合とし、各 $r \in R$ がサバイバル成分 C_r を持つと仮定する。$\tilde{E}_R$ によって R に属する面と R に属さない面を隔てる辺の集合を表す。R^* を G^* から除いて作られる各連結成分 Q に対して $Y_Q = \{(e,c)|e \in \tilde{E}_R \cap \tilde{E}_Q, (e,c) \in \tilde{T}_R\}$ と定義する。$\hat{S} = S$ と次の条件は同値である。

ある辺 $e \in \tilde{E}_R$ と e に接する面 $r \in R$ があって、C_r を包含する C_e の要素を c_e とするとき、$(e,c_e) \in \hat{T}_Q(\phi,Y_Q)$ である。但し Q は G^* から R^* を取り除いたときにできる連結成分のうち、e を境界に持つような集合である。

証明. 定理の条件を満たす e,r が存在するとき、明らかに $(r,c_r) \in \hat{S}$ であり、観察3.1より、$\hat{S} = S$ である。$\hat{S} = S$ とする。$\hat{S}$ を構成する帰納法の結果は規則3,4の実行順に依存しないので、次のような実行順で $\hat{S} = S$ が示せるはずである。

1. $\hat{S}_R$ を計算する
2. R^* を G^* から除いて得られる連結成分 Q^* の全てに対して $\hat{S}(\phi,Y_Q)$ を計算する
3. 更に規則3,4を適応できる限り適用する

3のステップが進行するためには定理の条件を満たす、r,e がなくてはならない。 □

上の定理より、サバイバル成分を持つ連結な面の集合 R と E_R については $\tilde{E}_R$ に属す辺 e についてのみの状態集合だけを保持すればよく、メモリ節約効果が期待できる。

5 実験

今回、素朴な方法と、改良1、改良2、改良1,2の組み合わせ、改良2,3の組み合わせ、改良1,2,3の組み合わせをそれぞれ組み込んだ方法の計6種類の実装を行った。改良3のみの改良では、あまり効果がないと判断し実装を行わなかった。

実行環境はPentium4 3.2GHzのCPUを使用した。使用言語はjavaである。今回、Hicks[4]の結果と比較するため、Hicks[4]が実験で使用したメモリ量600MByteをVMの最大使用メモリ量とした。実験に使用したグラフはTSPLIBにある点集合をdelaunay三角形分割をしたものであり、Hicks[4]が使用したグラフと同じものである。

表 4 は、例題ごとに、与えられたグラフの分枝幅を決定するための計算時間を表している。このとき、Tamaki[5] の計算時間 $O(n)$ の刻み幅発見的アルゴリズムによって刻み幅の上界 k を得てから、$RC(G,(k-1))$ でネズミが勝つまで k を減らしていく。k を減らした回数を反復回数とした。表にある × はメモリ不足により計算できなかったことを示す。

結果を考察すると、改良 1 はメモリ使用量、計算時間に関して効果が大きい。改良 2 のみでは、あまりメモリ使用量に関しても、計算時間に関しても効果が小さい。しかし、改良 1,2 を組み合わせると、改良 1 のみよりも効果が大きくなる。表 1,2 は k が刻み幅近辺であるときの k の値別の計算時間である。改良 2,3 を組み合わせることによって、改良 2 のみの場合と比較して、ネズミ捕り機が勝つ場合の計算時間を小さくすることができたことが判る。これは、本来ならば、$H(G,k)$ を全て計算しなくては $\hat{S},\hat{T}$ に属することが計算できないが、改良 1 により、部分的に $H(G,k)$ を見るだけで $\hat{S}$ に属するかどうかの伝播が伝わるからである。改良 2,3 の組み合わせの効果は少ない。改良 1,2,3 の組み合わせと改良 1,2 の組み合わせはあまり効果が変わらないように思われるが、実験した中でグラフのサイズが最も大きい例題 brd14051 などでは改良 1,2,3 を組み合わせの計算時間が改良 1,2 の組み合わせよりも短い。これは、メモリ不足によるガーベジコレクションの回数が改良 1,2,3 組み合わせの方が少なかったからであると考える。

改良 1,2,3 の組み合わせと Hicks[4] の comrat アルゴリズムの計算時間とを比較する。表 3 は Hicks[4] のアルゴリズムごとの実行結果を表す。このとき CPU は SGI Power Challenge 194MHz を 6 台使用している。単純に比較すると、辺数 393 から辺数 42128 のテストパタンに関して 20 倍から 89 倍高速であった。CPU の周波数の比 16.5 を考慮しても高速効果があったと考える。(Hicks[4] は CPU を 6 台使用していることも考慮したい)

表 1: 改良 2,3 による、幅 k 以下分枝分割可能性判定の計算時間

例題名	k の値ごとの計算時間 (sec)		
	分枝幅 + 1	分枝幅	分枝幅 − 1
pr2392	22.1	24.6	111
fl3795	65.5	78.9	323
rl5934	303	322	882

表 2: 改良 2 による、幅 k 以下分枝分割可能性判定の計算時間

例題名	k の値ごとの計算時間 (sec,k は ksec)		
	分枝幅 + 1	分枝幅	分枝幅 − 1
pr2392	104	106	111
fl3795	352	367	392
rl5934	939	1.00k	1.05k

6 結論と今後

結果から、改良 1,2 の組み合わせはとても有用であることがわかる。改良 3 は単独では効果が少ないが、メモリ不足の場合には使ってもよいと思われる。平面グラフの分枝分割を求めるためには、更にメモリが必要であり、14000 頂点のグラフの分枝分割を構成するためにはもっとメモリの節約が必要であると思われる。今後は、実際に、分枝分割を構成するアルゴリズムを実装し、実装上の改良を試みたい。

謝辞 御討論頂いた Qian-Ping Gu 教授に感謝いたします。

表 3: Hicks[4] による分枝幅決定計算時間 (使用言語:C++ 使用 CPU:SGI Power Challenge 194MHz 6 台 使用メモリ:600MByte)

例題名	計算時間 (sec)	
	comrat	memrat
eil51	1	2
lin105	7	9
ch130	13	18
pr144	12	19
kroB150	18	24
tsp225	29	46
pr226	22	31
rd400	112	136
u574	336	298
p654	198	276
d657	396	545
pr1002	448	562
rl1323	1519	1590
d1655	1608	2206
rl1889	3207	4704
u2152	3207	4704
pr2392	3813	5167
fl3795	18469	17142
fnl4461	35933	51035
rl5934	73468	66461
pla7397	65197	53564
usa13509	×	413861
brd14051	×	594408

表 4: 例題ごとの分枝幅決定計算時間 (使用言語:java 使用 CPU:pentium4 3.2GHz メモリ:600Mbyte)

例題名	辺数	分枝幅	反復回	計算時間 (sec,m は msec,k は ksec)					
				素朴な方法	改良 1	改良 2	改良 1,2	改良 2,3	改良 1,2,3
eil51	140	8	1	937m	156m	875m	281m	922m	265m
lin105	292	8	2	4.44	438m	4.61	641m	4.77	625m
ch130	377	10	4	23.1	1.17	24.0	781m	25.4	735m
pr144	393	9	1	4.50	406m	5.73	531m	5.99	500m
kroB150	436	10	2	11.7	859m	12.5	860m	13.0	828m
tsp225	622	12	2	37.1	1.47	35.2	1.28	37.8	1.16
pr226	586	7	1	7.64	906m	9.00	1.17	9.38	1.13
rd400	1183	17	2	×	5.06	161	3.78	166	3.58
u574	1708	17	2	×	12.4	260	11.6	276	11.2
p654	1806	10	3	×	25.5	×	11.2	×	11.3
d657	1958	22	1	×	5.94	×	7.25	×	7.28
pr1002	2972	21	5	×	82.9	×	28.7	×	27.9
rl1323	3950	22	3	×	115	×	65.8	×	75.9
d1655	4890	29	3	×	125	×	58.7	×	57.8
rl1889	5631	22	3	×	296	×	184	×	188
u2152	6312	31	2	×	96.5	×	99.1	×	76.2
pr2392	7125	29	3	×	400	×	156	×	158
fl3795	11326	25	6	×	1.74k	×	552	×	502
fnl4461	13359	48	2	×	937	×	540	×	622
rl5934	17770	41	4	×	3.31k	×	1.78k	×	1.69k
pla7397	21865	33	3	×	×	×	1.90k	×	2.37k
usa13509	40503	63	6	×	×	×	9.27k	×	9.23k
brd14051	42128	68	2	×	×	×	7.41k	×	6.68k

参考文献

[1] N. Robertson and P.D. Seymour, Graph minors X. Obstructions to tree-decomposition, Journal of Combonatorial Theory, Series B, 153-190, 1991.

[2] P.D.Seymour and R.Thomas, Call Routing and the Ratcatcher. Combinatorica, 14(3), 217-241,1994.

[3] Qian-Ping Gu and Hisao Tamaki, Optimal Branch-decomposition of planar graphs in $O(n^3)$time. ACALP 2005, 373-384.

[4] Illya V.Hicks,Planar Branch Decomposition I:The Ratcatcher,INFORMS Journal on Computing. Vol.17, No.4, Fall2005, pp.402-412.

[5] Hisao Tamaki, A liner time heuristic for the branch-decomposition of planar graphs. ESA2003, 765-775.