2001年度日本オペレーションズ・リサーチ学会
春季研究発表会

平面グラフの全ての面を認識するアルゴリズム*

01700900　防衛大学校情報工学科　山田 武夫† YAMADA Takeo

1　はじめに

図1のような, 平面グラフ $G=(V,E)$ の全ての面をコンピュータに認識させる問題を考える. これはグラフ理論 [1] や計算幾何学 [2] の分野で基本的な問題と思われるが, この方面の教科書にはあまり取り上げられていないようである. 本稿では, そのためのアルゴリズムとして双対グラフ法, 平面走査法の2つの方法を示す.

注意: 平面グラフとは「平面上に枝が交わらずに描画されたグラフ」を意味する. 平面的グラフという言葉もあるが, これは「平面グラフとして描画可能なグラフ」のことである. 同型なグラフであっても, 平面への描画の仕方によって面の構造が異なることがあるので, '平面的グラフ' では問題の設定が不正確である. ∎

以下では, 簡単のため次の仮定を置く.

A$_1$. 枝はすべて直線である.

A$_2$. 面はすべて凸で, 非退化.

A$_3$. 節点はX座標の昇順に整列済みで, X座標は全て異なる.

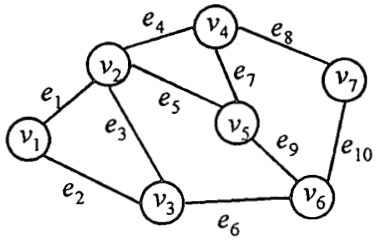

図 1: 平面グラフ

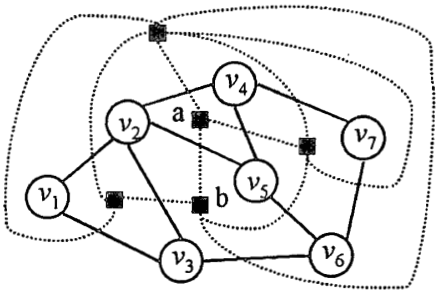

図 2: 双対グラフ

2　双対グラフ法

図1の双対グラフは図2の破線のようになる. 双対グラフの節点はもとのグラフの面と1対1に対応するので, 双対グラフの節点を（例えば深さ優先方式で）探索しようというのが本節の方法である.

まず, 枝 $(v,v')\in E$に対して, その右側の面を認識するアルゴリズムを考える. このためには (v,v') から出発し, 最先端節点で常に反時計方向に隣の枝を選んで前進するという操作を, v に戻るまで反復すればよい. これをアルゴリズム **Find_a_Face** (v,v') と呼ぶ. **Find_a_Face**(v,v') によって, 枝 (v,v') の右側の面を認識することを, 枝 (v,v') の精査という.

全ての面を認識するアルゴリズムを構成するために, 各枝 $e\in E$に $\{0,1,2\}$ の値を取るラベル $l(e)$ を付す. ここに, $l(e)=k$ は e の両側の面のうち k 個までが認識済みであることを示す.

また, スタック S を用意し, 今後精査すべき枝の集合を保持する.

以上の準備のもとで, アルゴリズムは次のように記述される.

Find_all_Faces:

1. $l(e):=0$; $\forall e\in E$, $S:=\emptyset$ とする.
2. 任意の枝 $(v,v')\in E$を Sにプッシュする.
3. $S\neq\emptyset$ である限り, 以下を実行する.
   3-1. Sの先頭の要素 $e=(v,v')$ をポップする.
   3-2. $l(e)<2$ であれば, 以下を実行する.
   (i) e を精査して新しい面を発見する.
   (ii) 上で前進中にたどった枝のラベルを1だけ増やす. さらに, これらの枝のラベルが1以下なら, 向きを逆転して Sにプッシュする.

*法政大学, 2001.5.1-2

†E-mail: yamada@nda.ac.jp

例えば, 図 1 で枝 (v_2, v_4) を精査すると, 図 3 の影を付した面が見つかり, スタックは図中の S のようになる. 次に, 枝 (v_2, v_5) をポップしてこれを精査するとこの枝の右側の面が見つかり, それとともにスタックの内容も図 4 のように変化する. これは, 図 2 で a 点からその隣接点を探索して b 点を見いだしたことに相当する. この操作を反復すると, すべての面が得られる.

上のアルゴリズムで, 精査中に遭遇した枝を向きを逆転してスタックに積み上げる理由は, その枝の左側の面を発見するには, 枝を逆に辿らなければならないことによる. また, 上記のアルゴリズムではスタックを用いたため, 双対グラフ上の動作としては節点を深さ優先方式で探索していることになる. もし, スタックの代わりにキューを用いれば幅優先探索となる.

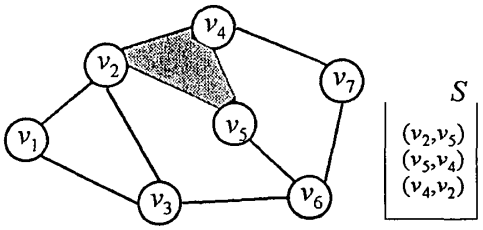

図 3: 双対グラフ法（ステップ 1）

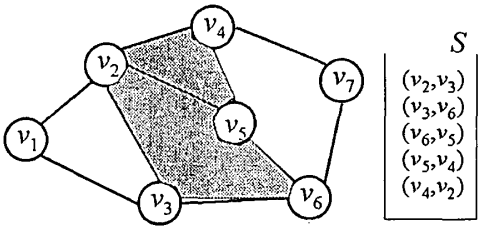

図 4: 双対グラフ法（ステップ 2）

3 平面走査法

これは, Y 軸に平行なバーを最左端から最右端まで動かし, グラフ G の節点, 枝との交差を連続的に追跡して全ての面を認識しようとする方法である. バーはそれが含まれる面と交差する枝を, 上から下に順に記憶するものとする. これをバーの状態と呼ぶ. 例えば, 図 5 の (a) ではバーの状態は外周面 f_0 のみである. バーが (b), (c) の位置に来た場合も含めて状態を示すと, 下のようになる.

(a) f_0

(b) $f_0 - e_1 - f_1 - e_2 - f_0$

(c) $f_0 - e_4 - f_2 - e_5 - f_3 - e_3 - f_1 - e_2 - f_0$

このように, バーの状態は両端が常に f_0 で, (バーが節点上を通過する場合を除いて) 面と枝が交互にあらわれる. 節点 i の左 (右) 隣接枝集合とは, i に左 (右) 方向から隣接する枝の集合をいう. すると, バーが節点 i を通過する前後で状態は次のように変化することが分かる.

- i の左隣接枝集合とそれらに挟まれた面が削除され,
- i の右隣接枝集合とそれらの間の面が新たに挿入される.

例えば, 図 5 で v_2 を通過する前後では (b) の e_1 が (c) では $e_4 - f_2 - e_5 - f_3 - e_3$ に置き換えられている. そこで, 図 5 の (a) から出発し, 節点 $1, 2, \ldots, n$ の順に, これらを通過するごとにバーの状態を更新していけば, 全ての面が認識される.

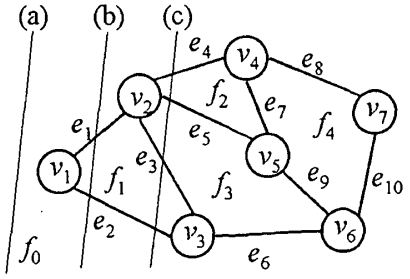

図 5: 平面走査法

4 まとめ

平面グラフのすべての面を認識する 2 つのアルゴリズムを提示した. これらの方法は, 若干の修正により平面グラフの双対グラフの導出にも利用することができる. 今後, これらの方法の計算量を検討する.

謝辞. アルゴリズムの画像表示部分については本校研究生善家大輔君 (現筑波大学大学院) と卒研生谷口史晃君の助力を得た.

参考文献

[1] F. ハラリー (池田訳), グラフ理論, 共立出版, 1971.

[2] 今井浩, 今井桂子, 計算幾何学, 共立出版, 1994.