1B-2

最短N経路探索アルゴリズムDijkstra-Hasui法の検証*

蓮井洋志†
室蘭工業大学情報工学科‡

1 はじめに

交通ネットワークの最短路探索は、カーナビゲーションシステムにおいて、欠かせない技術である。最短N経路を求められれば、より面白い経路をドライバーに提示できる。また、形態素解析の分野では、N-Best探索アルゴリズムがの大切である。N個の解析結果を保持したまま、次の処理にそのデータを送れれば、正しい解析ができる。音声認識の分野でもグラフの探索は大切な問題である。

しかし、N-Best探索はNの値を大きくすると、メモリの占有量が大きくなり探索速度も落ちる。ラティス構造のN-Best探索アルゴリズムの一つに動的計画法[1]が提案されている。この方法は、有向非循環グラフ一般に応用できる。しかし、Nが大きくなるとメモリ消費量が大きくなり、探索速度も遅くなる。高速でメモリの占有量の小さいアルゴリズムを開発することは、大規模なグラフをN-Best探索するために必要なことである。

我々は、The Forward DP Backward A* N-Best Search Algorithm[2]を一般のグラフに応用した、高速でしかもメモリ消費量の小さいアルゴリズムDijkstra-Hasui法[3]を提案した。しかし、それが必ず最短N経路を探索できていることを証明できなかった。本稿では、Dijkstra-Hasui法がグラフのN-Best解を探索できることを証明し、その解析速度とプロセスのメモリ占有量を動的計画法と比較実験する。

2 Dijkstra-Hasui法

Dijkstra法を使った最短N経路アルゴリズムを開発した。Dijkstra-Hasui法と呼ぶ。このアルゴリズムを実行するために必要な変数を以下に示す。

```
// 目的解：
RouteTable  routes;        // 最短 N 経路候補
// ワーキング変数：
RouteTable  routetable;
          // 他頂点から目標頂点までの最短路
Vertexs     nextpoint, dp;// 頂点番号変数
int         k;            // 整数変数
Route       *r1, *r2, *r; // 経路変数
Route       *route;       // 最短経路変数
```

Dijkstra-Hasuiアルゴリズムを以下に示す。

```
BestNSearch(Vertexs start, Vertexs end)
(1) end から他の全ての頂点までの最短路を routetable に保存する
(2) routetable の中の end から start までの最短路を検索し、それを routes に登録する
(3) routes が空ならば return グラフはつながらない;
    それ以外ならば (4) にすすむ
(4) routes.route[0]->dp に start をいれる
(5) k = 0, ..., N まで以下をループ
 (5-1) routes の中にある k 番目に短い経路 route の最初の頂点から route の最後の頂点まで以下をループ
  (5-1-1) dp に route->dp を代入し、dp まで頂点を進める
  (5-1-2) nextpoint に dp を中心として展開する頂点番号をいれる。nextpoint は route の経路と違う頂点だとする。dp が end ならば探索を (5) に戻す
   (5-1-2-1) r1 に route の dp までの頂点番号列をいれる
   (5-1-2-2) r1 に nextpoint をいれる
   (5-1-2-3) routetable から nextpoint から end までの最短路を検索し、それを r2 に入れる。
   (5-1-2-4) r に r1 と r2 をつなげていれる
   (5-1-2-5) r の経路の長さを求める
   (5-1-2-6) routes に r を登録する
  (5-1-3) route->dp に同経路の dp の次の頂点番号をいれる
(6) routes の中の短い方から N 個を解として返す
```

このアルゴリズムはまず、最短路をDijkstra法で求める。その最短路の中の分岐点で1辺外れた頂点からの最短路とそれまでの経路を加えたものを最短N経路候補とする。i番目に短い経路が見付かったら、その経路から1辺それた頂点からの最短路とそれまでの経路を加えたものが次の最短N経路候補である。それらの候補の中でi+1番目に短い候補をi+1番目に短い経路とする。これをN回繰り返す。

3 Dijkstra-Hasui法の証明

グラフを $g(v_i, a_j, S, E)$ と定義する。v_i とは頂点を表し、a_j とはその頂点間の長さの属性である。S は開始頂点、E は終了頂点を表す。ここで、S から E の最短路を $v_{11}, v_{21}, ..., v_{n1}$ とする。このグラフの最短路は r_1、k番目に短い経路は r_k とする。

S から展開された頂点 $v_{11}, v_{12}, ..., v_{1k}$ を通る経路の集合が S から E までの経路の集合である。

これを v_{11} に応用すると、v_{11} を通る経路の集合は S から v_{11} を通り、$v_{21}, v_{22}, ..., v_{2l}$ を通る経路の集合

*Verification of N-Best Shortest Path Search Algorithm, Dijkstra-Hasui Method
†Hiroshi Hasui
‡Department of Computer Science and Systems Engineering in Muroran Institute of Technology

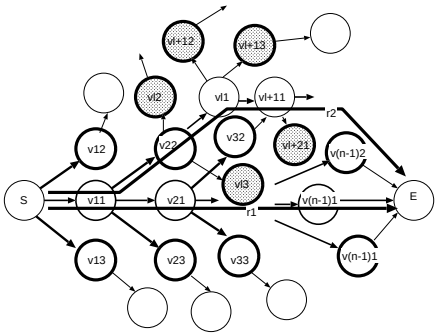

図 1: 最短 2 経路とそれ以外の経路に分割したグラフ

である。つまり S から E までの全経路の集合は、S から $v_{12}, ..., v_{1k}$ を通る経路と S, v_{11} を通ってしかも $v_{21}, ..., v_{2l}$ を通る経路の集合と一致する。

これを再帰的に繰り返すと最短路 r_1 とそれから 1 辺それた頂点 V_1 を通る経路の集合 R_1 の和集合が S から E までの全経路の集合であることがわかると思う。

図 1 に最短路とそれ以外の経路への分割したグラフを示す。真ん中の経路が最短路である。太い丸で囲まれた白抜きの頂点を通る経路の集合が最短路より長い経路の集合である。

R_1 の中の一番短い経路が 2 番目に短い経路である。つまり、V_1 の各頂点から E までの最短路にその頂点までの最短路の経路を加えあわせた経路集合の中に必ず 2 番目に短い経路が含まれていると言える。

次に、R_1 に含まれない経路の中で、2 番目に短い経路から、1 辺それた頂点、図 1 では太丸で囲まれた灰色の頂点であるが、それ を通る経路の集合 R_2 を考えると、$\{r_1 + r_2 + R_1 - r_2 + R_2\}$ が S から E までの全経路である。図 1 でいうと、$S, v_{11}, v_{22}, v_{l2}, ..., E$ などが R_2 に含まれる。

そうすると、そこから最短路と 2 番目に短い経路を取り除いた $\{R_1 - r_2 + R_2\}$ が、2 番目に短い経路より長い経路の集合である。

つまり、最短路かあるいは 2 番目に短い経路から 1 辺それた経路の中で 2 番目に短い経路が 3 番目に短い経路である。

これを繰り返し行うことで N-Best 解を探索することができる。$\{R_1 + R_2 + ... + R_{N-1} - r_2 - ... - r_{N-1}\}$ が N 番目に短い経路より長い経路の集合である。つまり、その中で一番短い経路が N 番目の最短路であることがわかると思う。$N-1$ 個の最短路から 1 辺それた頂点から、E までの最短路に、それまでの経路を加えた経路の中で $N-1$ 番短い経路が N 番目に短い経路である。これは、アルゴリズムどおりの探索である。

方法	Dijkstra-Hasui 法		久光・新田法	
大きさ	速度	メモリ	速度	メモリ
40	0.19	1681	0.68	2083
80	1.21	1683	3.96	3339
120	3.52	1716	12.71	5383
160	7.39	1746	29.93	8222
200	12.80	1749	57.80	11884
240	21.61	1780	98.28	16306
280	33.50	1814	152.11	21553
320	46.30	1846	207.47	27693
640	291.78	2242	2023.73	104749

表 1: 有向非循環完全結合グラフの解析速度

4 他手法との比較評価実験

$N = 256$ のときの非循環有向完全結合グラフの解析速度と解析終了時のメモリ占有量を Dijkstra-Hasui 法と動的計画法で比較した。このグラフの辺の距離は一番多くの頂点を通らないと最短路にならないようになっている。最大計算量がこのグラフから推測できる。

表 1 にその結果を示す。CPU が Pentium 4 3.4GHz でメモリが 512MB のパソコンで行った。単位は速度は秒、メモリは KB で表している。これを見ると、Dijkstra-Hasui 法の方が動的計画法より 3-6 倍位高速で、しかもメモリの占有量が少ないことがわかる。動的計画法は 640 頂点のグラフの探索は 100MB 以上のメモリがないと探索できないが、Dijkstra-Hasui 法では 3MB くらいでできる。

しかし、$N = 27$ 以下の場合、Dijkstra-Hasui 法より動的計画法のほうが高速だった。メモリの消費量もほぼ変わらなかった。N が小さいとき動的計画法を使い、大きいと Dijkstra-Hasui 法を使うのが良いと言える。

参考文献

[1] 久光徹, 新田義彦. ゆう度付き形態素解析用の汎用アルゴリズムとそれを利用したゆう度基準の比較, 電子情報通信学会 D-II, Vol. J77-D-II, No. 5, pp. 959-969, (1994).

[2] M. NAGATA. A Stochastic Japanese Morphological Analyzer Using a Forward-DP Backward-A* N-Best Search Algorithm, *Proceedings of the 15th International Conference on Computational Linguistic* , pp. 201-207, (1994).

[3] 蓮井洋志. 有向非循環グラフの最短 N 経路探索アルゴリズム, 情報処理学会全国大会講演論文集, Vol. 68, No. 1, pp. 187-188, (2006)