3B-1

有向非循環グラフにおける最短N経路探索アルゴリズム*

蓮井洋志†

室蘭工業大学情報工学科‡

1 はじめに

グラフの最短路探索のアルゴリズムとしては最良優先探索、Dijkstra 法、ヒューリスティクス探索、ビームサーチなどがある。最良探索は正確な解析であるが、グラフが大きくなると時間がかり過ぎる。計算量が多項式時間でなく、指数時間である。Dijkstra 法は正確な解析が可能であるが、最短路しか求まらない。ヒューリスティクス探索は、最短路しか求まらないだけでなく、高速だが、不正確な情報をもとに枝刈りをするために正確な解析となりえない場合がある。ビームサーチは N 経路を探索できる。しかし、ビーム幅が大きいと計算量が大きくなり過ぎるし、ビーム幅が小さいと正確な解が得られない。

本研究では、Dijkstra 法を活用した Dijkstra-Hasui 法を提案する。この方法は短い方から N 個の最短路を高速に探索できる。この探索のことをグラフの最短 N 経路探索と呼ぶ。最短 N 経路を厳密に求めるには全解探索をするアルゴリズムである深さ優先探索、幅優先探索、最良優先探索などしかなかった。これらの方法はグラフが大きくなると探索の速度が極度に悪くなる。それは計算量が指数時間であるためである。それに対して、本研究の方法は多項式時間で解析ができる。一番探索するのに不利と考えられる、完全有向非循環グラフの場合、グラフの頂点の数を $MaxVertexNumber$ とすると、$\sum_{i=1}^{N}(MaxVertexNumber - i) \times (MaxVertexNumber - i - 1)$ の計算量で探索できる計算となる。

多くの有向非循環グラフに対して探索を行ったところ、Dijkstra-Hasui 法はすべての場合において、厳密解を求めていることがわかった。

本稿では、まず 2 章で Dijkstra-Hasui 法について、3 章で最短 N 経路が正しく探索できているかの統計的評価について、4 章ではその評価結果を考察する。

*N-Best Shortest Path Search Algorithm for Directed Acyclic Graph

†Hiroshi Hasui

‡Department of Computer Science and Systems Engineering in Muroran Institute of Technology

2 Dijkstra-Hasui 法

Dijkstra 法はグラフの最短路を探索するアルゴリズムである。開始頂点からあらゆる他の頂点までの最短路を探索する。動的計画法のひとつで計算量は $O(MaxVertexNumber^2)$ である。

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
// 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) を実行する
(4) routes.route[0]->dp に-1 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) に戻る
   (5-1-2-1) r1 に route の dp までの頂点番号をいれる
   (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
```

(1) の処理は Dijkstra 法で求まる。グラフの目標接点から逆にたどって他の頂点までの最短路を全て

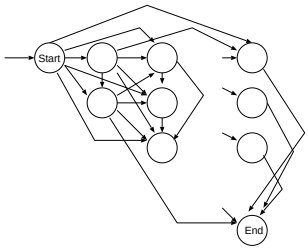

図 1: 有向非循環完全結合グラフ

求める。1 度の探索ですべての最短路が求まる。その経路を全て routetable に登録する。(2)〜(5) の処理は開始頂点から途中の頂点までの短い方から数えた経路をたどる方法である。途中の頂点からは (1) で求めた経路テーブル routetable から検索する。その結果を加えた経路を経路テーブル routes に登録する。routes の短い方から N 個の経路を最短 N 経路とする。

最短路の途中までの経路に nextpoint を加えた部分経路に、nextpoint から目的頂点までの最短路を加えた経路を解候補とする。その解候補の短い方から N 個を結果とする。そのためにこのアルゴリズムで解析できた解は正しいと推測できる。

3 最短 N 経路探索実験

3.1 実験方法

実験 1 ：Dijkstra-Hasui 法でグラフの最短 N 経路を探索する。$N = 16$ の場合のみ行った。その後、最良優先探索で探索した結果と比較する。グラフは日本語形態素解析システム WordClass の解析途中のものを活用する。グラフはすべて有向非循環グラフである。143 個のグラフに対して行った。

実験 2 ：有向非循環完全結合グラフを Dijkstra-Hasui 法で解析する。もっとも手間を多くかかるグラフであることが推測できる。この有向非循環完全結合グラフとは図 1 のように一番結合する辺が多い有向非循環グラフである。グラフの大きさが 40, 80, 160, 320 の有向非循環完全結合グラフに対して Dijkstra-Hasui 法を適用した。

3.2 実験結果

実験 1 ：表 1 にグラフの大きさ、枝の数、および解析速度を表す。 最良優先探索はグラフの大きさによっては 10 分以内で解析できなかったものがある。そういったグラフは対象外とした。

大きさ	枝の数	最良優先探索	Dijksrtra-Hasui 法
45	103	0.00	0.01
62	122	0.03	0.00
81	241	12.17	0.00
155	554	12.16	0.02
142	305	173.56	0.01

表 1: 最短 N 経路探索の速度の比較

大きさ	解析速度
40	0.11
80	1.080
160	8.890
320	50.43

表 2: 完全結合グラフの解析速度

すべてのグラフで最良優先探索と Dijkstra-Hasui 法は同じ解析結果となった。最良優先探索は厳密解を求めるものであるため、Dijkstra-Hasui 法の解は全て正しいといえる。大規模なグラフでは最良優先探索が有限時間内にできないために正しい解がでているかどうかは分らないが、この結果からみて正しいとみなしてかまわないのではないかと考える。

実験 2 ：表 2 に完全結合グラフの大きさとその解析速度を示す。Dijkstra-Hasui 法はこういった極限状態のグラフに対しても有限時間内に解析できることが分った。

4 考察

実用的な速度で厳密解を探索することのできる方法のひとつである。Nagato[1] は、形態素解析用の N Best Search Algorithm を発表しているが、この方法では N=2〜3 の場合しか正しい解が見当たらない。しかし、本法は時間が多少余計にかかるが正しい解を探索できる。

実験 2 をみると解析速度はグラフの大きさの 3 乗に比例している。計算量は指数時間ではなく、多項式時間であることが推測できる。

参考文献

[1] 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).