A note on the practical performance of the auction algorithm for shortest paths

Jesper Larsen
DIKU Department of Computer Science
University of Copenhagen
Universitetsparken 1
DK 2100 Copenhagen Ø
e-mail: friberg@diku.dk

February 3, 1997

Abstract

The performance of the auction algorithms for the shortest paths has been investigated in four papers with differing conclusions. In the following I report a series of experiments with the code from the two most recent papers. The experiments clearly show that the auction algorithm is inferior to state-of-the-art shortest paths algorithms

Keywords: shortest path problem, auction algorithm, performance results.

1 Introduction

The (sequential) auction algorithms for the shortest path problem have been the subject of experiments which have been reported in at least five papers.

The first paper [Ber91] introduced the auction algorithm for the shortest paths. This algorithm had pseudo-polynomial time-complexity. The paper also introduced the concepts of best and second-best neighbour; improvements that enhanced the practical performance.

In [PS91] graph reduction was introduced for the first time in the auction algorithm (here we call it *simple* reduction). This resulted in a polynomial auction algorithm. No experimental testing was reported in this paper.

In [BPS92] and [BPS95] Bertsekas et al. improved the polynomial time-complexity even more with their strong graph reduction (hereafter referred to as *extended* reduction).

The present author worked with the auction algorithm in a joint MSc. thesis [LP95]. We introduced an even stronger graph reduction scheme (called the *improved* reduction). Although this theoretically has a time-complexity worse than

the extended graph reduction, experimentally it proved to be as good as the extended graph reduction and sometimes even better.

In [Ber91] no computational results of the one-to-all auction algorithm was provided, only a statement that it was slower than the S-HEAP code of [GP88]. The main conclusion for the best one-to-all auction algorithm of [BPS92] and [BPS95] is that is marginally outperforms S-HEAP for dense graphs, while the oppostite is the case for more sparse graphs although this is not documented in detail.

In this note I report on comparison of performance tests between the codes developed in [BPS92], [BPS95] and [LP95].

I also adress the equally important question originating from the different conclusions reached in the mentioned papers: are the auction algorithms of [BPS95] better than a state-of-the-art shortest path algorithm?

The algorithms tested are:

(LP95) The auction algorithm with best neighbour improvement (as introduced in [Ber91]) and extended reduction (as suggested in [BPS92]).

(DiH) A heap implementation of Dijkstras algorithm (see e.g. [GP88]). The Dijkstra algorithm using binary heaps is used as reference algorithm in [BPS92].

(DiD) Dijkstras algorithm with double buckets as described in [CGR93]. This algorithm is used as the reference algorithm in [LP95] as it performs substantially better than (DiH).

(BPS) According to the test results reported in [BPS95], this is the best of the proposed algorithms in [BPS95]. It is an auction algorithm with extended reduction and best neighbour improvement.

The remaining three algorithms of [BPS92] were not tested here as they performed worse than (BPS).

2 Experimental setup and results

Throughout this section n denotes the number of nodes and m the number of arcs in a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

I tested the algorithms with graphs generated by the programs `sprand` and `spgrid` provided by Cherkassky et al. in connection with [CGR93]. These programs are available via ftp at Stanford University.

I used the following test cases:

RAND4 This is a class of random graphs with n nodes and $4n$ arcs. A node has an average of four outgoing arcs. The RAND4 class represents sparse graphs. The arc lengths are chosen at random from the interval $[0; 10000]$. The generator program guarantees that all arcs can be reached from the source. Graphs have been generated for values of n equal to 8192, 32768 and 65536.

RAND14 Here, the number of arcs is $\frac{n^2}{4}$ (where n is the number of nodes), and hence the graphs are dense. Again it is guaranteed that the graph is connected. As in the RAND4 class, the arc lengths randomly chosen from the interval $[0; 10000]$. Graphs were generated for n equal to 512, 1024 and 1536.

GRIDW A graph of the GRIDW class is a rectangular grid, 16 "node-layers" wide and with x nodes in each layer, in all, a total of 16384 nodes plus a source node connected to all nodes in the first layer. The arcs form a two-way cycle in each layer. Between two adjacent nodes (or the first and the last node) there are two arcs, one in each direction. In addition, each node in a layer has an outgoing arc to the adjacent node in the next layer, except for nodes in the last layer. Finally, the source node is connected to the nodes of the first layer. The graph has a total of 49152 arcs. The lengths of the arcs are selected at random from the interval $[0; 10000]$. Graphs were generated for x equal to 256, 512 and 1024.

GRIDH The basis of a GRIDH-class graph is a GRIDW- graph with only a single cycle in each layer. Here graphs consists of a rectangular grid, 16 "node-layers" wide with x nodes in each layer. In addition, there is a collection of arcs connecting randomly selected pairs of nodes on the cycle. The lengths of the arcs inside a layer are small and non-negative (here selected at random from the interval $[0; 100]$). Additionally, arcs from lower to higher numbered layers are added. If an arc from layer τ_1 to layer τ_2 is included the randomly selected length (chosen in the interval $[1000; 10000]$) is multiplied by $(\tau_2 - \tau_1)^2$. These graphs were generated for x equal to 128, 256 and 512.

COMP This is the class of complete graphs. The length are randomly choosen from the interval $[1; 10000]$. Here we have generated graphs for $n = 512$, $n = 768$ and $n = 1024$ nodes.

The experiments were performed on a Hewlett-Packard Apollo 9000 series 700 Model 730 computer at the Department of Computer Science at the University of Copenhagen. It has a 99 MHz PA-RISC 7100 processor and 80 MB of main memory. The running times reported in Table 1 are averages taken over 5 runs.

It should be noted that while (LP95), (DiH) and (DiD) are written in C and compiled with `gcc`, (BPS) is written in Fortran and compiled with `f77`.

The running times clearly indicate that the Dijkstra codes are substantially better than both of the auction codes. The Dijkstra codes are always at least an order of magnitude better than the best of the two tested auction codes. The (DiD) code is always the fastest one-to-all shortest path algorithm.

The variation in the running times of the auction codes indicate no clear winner. While (BPS) is best on the random graphs, (LP95) is best on the grid graphs. Here further research is necessary to determine if there is a clear winner.

Test cases	n or x	(LP95)	(BPS)	(DiH)	(DiD)
RAND14	512	0.700	0.187	0.063	0.017
	1024	2.067	0.477	0.234	0.074
	1536	6.087	0.910	0.510	0.170
RAND4	8192	5.041	7.307	0.143	0.037
	32768	23.617	29.743	0.840	0.290
	65536	46.817	59.670	2.337	0.680
GRIDW	256	1.067	2.783	0.050	0.014
	512	3.533	9.633	0.117	0.040
	1024	11.983	36.383	0.257	0.083
GRIDH	128	5.750	10.112	0.057	0.027
	256	24.687	44.283	0.146	0.043
	512	106.980	188.233	0.303	0.113
COMP	512	3.201	0.457	0.227	0.077
	768	7.766	0.823	0.497	0.190
	1024	14.917	1.337	0.920	0.320

Table 1: The running times in seconds of the algorithms.

The algorithms are programmed in two different languages and differences in performance may be due to this rather than to the different algorithms. The differences observed are, however, too large to originate in the difference in programming language.

3 Conclusion

Based on my experiments it is difficult to envisage that an auction code can be made faster than a code based on a Dijkstra-like algorithm. The results clearly indicates a huge advantage for the latter codes. The most valuable refinements of the original auction algorithm has definitely been the reductions and further research in more effective reductions may lead to more efficient algorithms.

Acknowledgements

I would like to thank Stefano Pallottino from the University of Pisa for providing me with the code from [BPS92] and for his comments, and I would also like to thank Jens Clausen from the Department of Computer Science at the University of Copenhagen for comments upon the experiments and the draft of this note.

References

[Ber91] Dimitri P. Bertsekas. An auction algorithm for shortest paths. *SIAM Journal on Optimization*, 1:425 – 447, 1991.

[BPS92] Dimitri P. Bertsekas, Steffano Pallottino, and Maria Grazia Scutellá. Polynomial auction algorithms for shortest paths. Technical Report TR-16/92, Dipartimento di Informatica, University of Pisa, 1992.

[BPS95] Dimitri P. Bertsekas, Steffano Pallottino, and Maria Grazia Scutellá. Polynomial auction algorithms for shortest paths. *Computational Optimization and Applications*, 4(2):99 – 125, 1995.

[CGR93] Boris V. Cherkassky, Andrew V. Goldberg, and Tomasz Radzik. Shortest path algorithms: Theory and experimental evaluation. Draft, July 1993.

[GP88] Giorgio Gallo and Stefano Pallottino. Shortest path algorithms. *Annals of Operations Resaerch*, 13:3 – 79, 1988.

[LP95] Jesper Larsen and Ib Pedersen. The auction approach for the shortest path problem: Theory and experiments. Technical report, Department of Computer Science, University of Copenhagen (DIKU), February 1995. Master thesis.

[PS91] Stefano Pallottino and Maria Grazia Scutellá. Strongly polynomial auction algorithms for shortest path. Technical Report TR-19/91, Dipartimento di Informatica, University of Pisa, 1991.