权值为非负整数的稀疏图的高效 APSP 算法

黄跃峰, 钟耳顺

摘要

稀疏图是网络研究中常用的数学模型，我们提出了计算非负整数权稀疏图的所有结点之间最短路径(All pairs shortest paths，简称 APSP)的高效算法，在与网络相关的研究中，该算法对于揭示网络的拓扑结构具有较重要的意义。对于一个包含 n 个结点和 m 个弧段的非负整数权图，该算法计算所有结点之间最短路径的时间复杂度是 $O(mn+nC)$，其中 C 是弧段的最大权值。现有算法计算该问题的最小时间复杂度是 $O(mn+n^2\log\log(\min\{n,C\}))$，在实际应用中 C 常常与 n 无关，即对时间复杂度的影响小于 $n\log\log(\min\{n,C\})$，所以我们的算法时间复杂度更小。

我们还根据随机图理论进一步分析了算法的复杂度，根据证明随着 n 的增大该算法时间复杂度为 $O(mn+C\log n/(\log m-\log n))$ 的概率趋近于 100%。在一些实际问题中，网络普遍为 $m=O(n)$ 且 C 与 n 无关的稀疏图（例如交通网络），这时该算法时间复杂度为 $O(mn)$，即为 $O(n^2)$。APSP 算法的理论时间复杂度下限是 $O(mn)$，与其他算法相比，在解决实际问题时我们的算法更容易达到这一理论下限。

关键字

APSP; 最短路径; 稀疏图

A New APSP Algorithm for Integer-weighted Sparse Graphs

Yuefeng Huang[*1], Ershun Zhong[*2], Yunjin Li[*3]

Abstract

We present a new all pairs shortest paths algorithm for directed sparse graphs with non-negative integer weights. For a graph with n vertices and m edges, the algorithm runs in $O(mn+nC)$ time, where C is the maximal edge weight. In comparison with the best previous time bound of $O(mn+n^2\log\log(\min\{n,C\}))$, our algorithm runs faster for the case that C is lower than $n\log\log(\min\{n,C\})$.

We also prove that with high probability the time bound of this algorithm is

[*] The State Key Lab of Resources and Environmental Information System, Institute of Geographic Sciences and Natural Resources Research, CAS, China.
[1] E-mail: yuefenghuang.study@gmail.com
[2] E-mail: zhongershun@supermap.com
[3] E-mail: liyunjin@supermap.com

$O(mn + C\log n/(\log m - \log n)), (m/n = C_0 > 1)$ .Our algorithm is more probable to get to the theoretical lower time bound $O(mn)$ of any path-comparison based algorithm for APSP.

Keyword

APSP; shortest path; sparse graphs

1 Introduction

The all pairs shortest paths (APSP) problem is one of the best known algorithms in algorithm design and textbook algorithm. Nearly all APSP algorithms can be divided into two groups: those which assume dense graphs by matrix multiplication, and those which assume sparse graphs by repeating single source shortest path algorithm based on a series of comparison and addition operations. We consider here the case that the input is a directed sparse graph with non-negative integer weights.

1.1 Related work

For dense graphs with *n* vertices, matrix multiplication is generally applied in APSP. The classical Floyd-Warshall algorithm [1] runs in $O(n^3)$ time. Fredman [2] was the first to realize the possibility of a sub-cubic algorithm, and then a series of improvements appeared in a number of papers. Recently Chan [3] [4] proposed two algorithms with the time bound $O(n^3/\log n)$ and $O(n^3\log^3\log n/\log^2 n)$, and Han [5] announced an $O(n^3\log^{5/4}\log n/\log^{5/4} n)$ time algorithm. Finally, Yuster [6] gave the $O(n^{2.842})$ time algorithm for real vertex-weighted graphs.

For sparse graphs, a more efficient solution to APSP is to apply single source shortest path algorithm *n* times. For the case of non-negatively weighted graphs, Dijkstra's algorithm [7] with Fibonacci-heap [8] solves single source shortest path in $O(m + n\log n)$ time, where *m* is the count of edges. By repeated applications of that algorithm, APSP is solved in $O(mn + n^2\log n)$. Johnson's algorithm [9] preserved reduction from arbitrary weighted to non-negatively weighted graphs, assuming non-negative weight cycles. Combination with Dijkstra's algorithm implies an $O(mn + n^2\log n)$ time cost for general APSP algorithm. The component hierarchy model is an important approach for shortest path problems. Thorup [10], the inventor of that approach, showed that single source shortest path for undirected non-negative integer weighted graphs was solved in $O(m)$ time. Hagerup [11] generalized Thorup's approach to directed integer weighted graphs and gave an $O(mn + n^2\log\log n)$ APSP algorithm. Pettie and Ramachandran [12] adapted Thorup's approach to real weighted graphs and comparison-addition model, considering an $O(mn\log\alpha)$ APSP algorithm for undirected graphs, where $\alpha = \alpha(m,n)$ is the inverse-Ackermann function. An experimental study by Pettie [13] of a simplified version of this algorithm showed that it was decisively faster than Dijkstra's algorithm. Pettie [14] also showed a directed APSP algorithm with the comparison-addition complexity of $O(mn\log\alpha)$. After that, Pettie [15] gave the final version of the algorithm for real weighted directed graphs with the time bound of $O(mn + n^2\log\log n)$ in the comparison-addition model.

There are also time bounds in terms of *C*, where *C* is the maximal edge weight. Ahuja et al [16] provided a single source shortest path algorithm with the running time $O(m + n\sqrt{\log C})$, which means time bound of $O(mn + n^2\sqrt{\log C})$ for APSP. Cherkassky, Goldberg, and Silverstein [17] provided a lower time bound of $O(m + n\sqrt[3]{C}^{1+\varepsilon})$ for any fixed $\varepsilon > 0$, with the time bound of

$O(mn + n^2 \sqrt[3]{C}^{1+\varepsilon})$ for APSP. Raman [18] gave a new $O(m + n\sqrt[4]{C}^{1+\varepsilon})$ single source shortest path algorithm, with the time bound of $O(mn + n^2 \sqrt[4]{C}^{1+\varepsilon})$ for APSP. Thorup [19] gave the current best $O(m + n^2 \log\log\min\{n, C\})$ time bound. Our algorithm also gives the time bound in this term.

1.2 Main results

Traditional methods repeat single source shortest path *n* times, while we propose an algorithm which calculates all pairs shortest paths at the same time. The algorithm works in the manner of Dijkstra's algorithm. Every edge is relaxed for *n* times, and all the n^2 shortest paths are sorted by counting sort. The algorithm has the complexity as to Theorem 1.

Theorem 1. *Let $G=(V,E)$ be a non-negative integer weighted directed graph with $|V|=n$ and $|E|=m$. There is an algorithm that solves APSP problem in $O(mn+nC)$ time, where C is the maximal edge weight.*

The edges of any shortest path is not more than the edges of the diameter of the graph. We prove that the counting sort algorithm has a far less complexity with high probability by random graph theory. Our algorithm has a complexity with high probability as to Theorem 2.

Theorem 2. *Let $m/n = C_0 > 1$. With high probability, this algorithm solves APSP problem in $O(mn + C\log n/(\log m - \log n))$.*

As usual, we say that an event holds high probability or whp if its probability tends to 1 as $n \to \infty$.

In Section 2 we give a general description of the integer weight model, on which we perform our algorithm. In Section 3 we give the proof of Theorem 1. In Section 4 we give the proof of Theorem 2 by random graph theory. In Section 5, we give the lower time bound of path-comparison APSP algorithm in theory. Our algorithm is more probable to get close to that time bound. In Section 6 we discuss some open problems.

2 Preliminaries

Our computational model is word RAM. The word-size of the processor is determined, limiting how big integers can be operated on in constant time. We assume that the word-size *W* is bigger than the nC, so that we can address the different shortest path costs with single words. Besides direct and indirect addressing and conditional jumps, our algorithm only uses addiction and operation on a constant number of words in constant time, composing the minimal instructions set. The memory is divided into words, addressed 0, 1, 2... The time is the number of instructions performed and the space is the maximal address used.

The essential advantage of dealing with integer weights is that we can use them to compute addresses in bucket based algorithms directly, which is the key point of our algorithm. We note that our algorithm also works for real weighted graphs. The IEEE 754 floating point standard [20] is designed so that the ordering of float point numbers can be deduced by perceiving their representations as integers. Thus the time bound of our algorithm holds equally well for real weighted graphs.

3 The APSP algorithm

In this section, we prove Theorem 1. First, we give the exact description of our APSP algorithm. Second, we present the counting sort is powerful for this algorithm. In the end, we give the complexity.

A graph is a pair $G=(V,E)$ of sets such that $E \subseteq [V]^2$. The elements of E are 2-element subsets of V such that $E = \{(i,j) | i \in V, j \in V\}$. V is the vertices set and E is the edges set. The weight of a edge $e(i,j)$ is $w(e(i,j))$ or $w(i,j)$ for short. We propose an APSP algorithm for a directed graph with non-negative integer weight, then $e(i,j) \neq e(j,i)$ and $w(i,j) \geq 0$. The shortest path from the source vertex *i* to the destination vertex *j* is denoted by $\mathrm{sp}(i,j)$.

The algorithm is summarized as follows:

Algorithm.
S1: *initialize by putting all the edges into the candidate set* S_{cand} *of the shortest paths;*
S2: *select the minimum element from* S_{cand}, *output it as one of the shortest paths* $\mathrm{sp}(i,j)$*;*
S3: *if* $e(j,k) \in E$ *and* $\mathrm{sp}(i,k)$ *is not outputted, set* $w(i,k) = \min\{w(i,j) + w(j,k), w(i,k)\}$ *and put it into* S_{cand}, *that is a relaxing operation;*
S4: *if all the shortest paths are not got, then go to S2, else end.*

Lemma 1. *We get all the shortest paths from one vertex to all the other vertices in an order by the algorithm, and the order is the same as the one by Dijkstra's algorithm.*

Proof. For each vertex v_i, let $\mathrm{sp}(i,j) \leq \mathrm{sp}(i,k)$. $\mathrm{sp}(i,j)$ must be outputted before $\mathrm{sp}(i,k)$ in our algorithm by S2. Calculate the shortest paths from v_i by Dijkstra's algorithm. The vertex v_j must be selected into the shortest-paths tree before v_k if $\mathrm{sp}(i,j) \leq \mathrm{sp}(i,k)$. Lemma 1 is proved. □

Lemma 1 implies that our algorithm gets all pairs shortest paths of the graph the same as repetitions of single source shortest path algorithm *n* times.

Lemma 2. *Every edge is relaxed n times at most in our algorithm and all relaxations add* $O(mn)$ *complexity.*

Proof. Each edge $e(i,j)$ is relaxed iff the shortest path to v_j is outputted. There are n shortest paths to v_j, so it is relaxed exactly *n* times for *m* edges adding $O(mn)$ complexity. □

Theorem 1. *Our algorithm solves APSP problem in* $O(mn + nC)$ *time, where C is the maximal edge weight.*

Proof. There are n^2 elements in S_{cand} at most. Comparison sorts limit the complexity by the lower bound of $O(n^2 \log n)$. The maximum element is the maximum-length shortest path in the graph denoted by max-SP. It is less than nC, because any shortest path contains no more than n vertices. Counting sort works out the same order in $O(n^2 + nC)$. By Lemma 2, the total time is $O(mn + n^2 + nC)$. m is surely not less than n for a connected graph, so the time bound is $O(mn + nC)$. By Lemma 1, our algorithm solves APSP problem in $O(mn + nC)$. □

4 Probabilistic analysis

In this section, we prove that the complexity of our algorithm is smaller than we have proved in Section 3 by random graph theory. The maximum shortest path (max-SP) contains no more edges than the diameter of the graph. The second part of the complexity in Theorem 1 is decreased from nC to $diam(G)C$. Further more, the diameter of a graph with n vertices is proved to be $O(\log n / (\log m - \log n))$ with high probability.

Lemma 3. *Max-SP is no more than* $diam(G)C$*, where* $diam(G)$ *is the diameter of graph.*

Proof. Let *W* be the weight of the shortest path from source vertex *i* to destination vertex *j*. If the shortest path is the max-SP and $W > diam(G)C$, then $W / C > diam(G)$. That means there is a path from *i* to *j*, which has a longer distance than $diam(G)$. So there is not a shortest path more than $diam(G)C$. □

The two frequently occurring models for random graphs are $G(n,p)$ and $G(n,m)$. $G(n,p)$ consists of all graphs with *n* vertices in which the edges are chosen independently with the probability *p*. *p* is between 0 and 1. $G(n,m)$ consists of all graphs with *n* vertices and *m* edges, in which the graphs have the same probability. *m* is close to $pn(n-1)/2$. Bollobás [22] found that the two models are interchangeable. Lemma 4 gives this property.

Lemma 4 *(Bollobás [22]). If a property is convex and* $p(1-p)n(n-1)/2 \to \infty$*, then* $G(n,p)$ *has the property with high probability iff for every fixed x* $G(n,m)$ *has the same property with high probability, where* $m = \lfloor pn(n-1)/2 + x(p(1-p)n(n-1)/2)^{1/2} \rfloor$*. If x=0,* $m = \lfloor pn(n-1)/2 \rfloor$.

The diameter is a convex property. For a directed graph, *m* is the lower bound of $pn(n-1)$ and $p = m / n(n-1)$. If $n \to \infty$, $p(1-p)n(n-1)/2 = m(n^2 - n - m)/(n^2 - n) \to \infty$. So Lemma 4 works well for directed sparse graphs. In the left of this section, we use the model of $G(n,p)$ for convenience.

Lemma 5. *If* $\log n > np \ge c > 1$*,* $diam(G(n,p)) = O(\log n / \log(np))$ *with high probability.*

Proof. Chung and Lu [21] proved that if $\log n > np \ge c > 1$, $diam(G(n,p)) = \Theta(\log n / \log(np))$ with high probability, so Lemma 5 is proved. □

Definition 1 *(Bollobás [22]). A property will be said to be monotone increasing or simply monotone if whenever* G_1 *has the property and* $G_1 \subset G_2$ *then* G_2 *also has the property.*

Diameter is a monotone increasing property. If G_1 has a diameter with the distance of *D* and G_2 has the same vertex set as G_1 but more edges, G_2 is created by adding edges one by one from G_1. Certainly, the diameter of G_2 is not longer than *D*.

Lemma 6. *If* $np \ge c > 1$*,* $diam(G(n,p)) = O(\log n / \log(np))$ *with high probability.*

Proof. Bollobás [22] proved that if a property is monotone increasing and $p_1 \le p_2$, then $G(n,p_2)$ has the property with higher probability than $G(n,p_1)$. *If* $np \ge \log n$, Lemma 5 is also true. Lemma 6 is proved. □

Lemma 7. *If* $np \ge c > 1$*,* $diam(G(n,m)) = O(\log n / (\log m - \log n))$ *with high probability.*

Proof. By Lemma 4, Lemma 6, and $p = m / n(n-1)$, $diam(G(n,m)) = O(\log n / \log(m/(n-1)))$. Lemma 7 is proved. □

Theorem 2. *Let* $m / n = c_0 > 1$. *Our algorithm solves APSP problem in* $O(mn + C \log n / (\log m - \log n))$ *with high probability.*

Proof. By Theorem 1 and Lemma 7, this theorem is proved. □

5 Complexity

We mainly focus on APSP algorithm for sparse graphs, so single source shortest path algorithm makes a lot of sense. Most of the single source shortest path algorithms are adaptations to the original Dijkstra's algorithm. There are two actions in Dijkstra's algorithm that determine the complexity: looking up the smallest weight path, and relaxing its outgoing edges. All edges are relaxed once, adding $O(m)$ to the complexity of single source shortest path algorithm and $O(mn)$ to the complexity of APSP algorithm. Karger, Koller, and Phillips gave the lower time bound of APSP algorithm in Definition 2 and Theorem 3, which is corresponding to that.

Definition 2. *(Karger, Koller, and Phillips [24]) Any algorithm for all pairs shortest paths problem, which uses the edge weight function only in comparing the weights of paths in the graph, is path-comparison based APSP algorithm.*

Theorem 3. *(Karger, Koller, and Phillips [24]) Any path-comparison based APSP algorithm needs a lower bound of* $\Omega(mn)$ *in running time.*

Looking up the smallest weight path adds to the complexity. So different sorting algorithms have brought out a lot of developments in APSP algorithm. Our algorithm has a time bound $O(mn+nC)$. If $C \le m$, we get the lower time bound $O(mn)$, which is easily satisfied in practice in comparison to the other algorithms with the condition of $\log\log(\min\{n,C\}) \le m/n$. With high probability, our time bound decreases to $O(mn + C\log n/(\log m - \log n))$. Furthermore, $C \le mn(\log m - \log n)/\log n$ may be satisfied more easily. In conclusion, there is a higher probability for our algorithm to get to the lower time bound of $O(mn)$ in practice.

6 Discussion

The complexity of our algorithm lies on the maximum shortest path. Our algorithm works faster for denser graphs with bigger m/n. However, $O(mn)$ also increases with m/n. If $m \ge c_0 n$, the second component is less than $O(C\log n)$. If *m* gets nearer to *n*, the second component is more complex. Riordan and Wormald [24] dealt with the diameter of random graphs when *m* is very near to *n*, implying expected degree tending to 1. They proved that:

Theorem 4. *(Riordan and Wormald [24]) Let* $\varepsilon = \varepsilon(n)$ *satisfy* $0 < \varepsilon < 1/10$ *and* $\varepsilon^3 n \ge e^{(\log^* n)^4}$ *for all large enough n. Set* $\lambda = \lambda(n) = 1+\varepsilon$ *, and let* $\lambda_* < 1$ *satisfy* $\lambda_* e^{-\lambda_*} = \lambda e^{-\lambda}$*.Then*

$$diam(G(n,\lambda/n)) = \log(\varepsilon^3 n)/\log\lambda + 2\log(\varepsilon^3 n)/\log(1/\lambda_*) + O_p(1/\varepsilon).$$

$\log^* n$ *denotes the least k such that* $\log_k n < 1$*, where* $\log_k n = \log\log\ldots\log n$ *is the k-fold iterated logarithm of n.*

Theorem 4 is not considered in our algorithm for the moment. There may be a different method for the case. If the degree of a vertex is 1, it and its edge should be in some shortest path or not with no doubt. The vertex with the degree of 1 can be combined with its conjunction vertex and estimated in the algorithm as one vertex, because there are not any different edges for a shortest path to choose to get to its conjunction vertex. If *m* is very near to *n*, the graph is transformed into a really smaller graph in this way. The complexity of this case will be considered in the future.

7 References

[1] T. H. Corman, C. E. Leiserson, R. L. Rivest, and C. Staein. Introduction to algorithms. McGraw-Hill, 2nd ed., 2001.

[2] M. L. Fredman. New bounds on the complexity of the shortest path problem. SIAM Journal on Computing, 5:49-60, 1976.

[3] T. M. Chan. All pairs shortest path with real weights in $O(n^3/\log n)$ time. Algorithmica, 50:236-243, 2008.

[4] T. M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. Proceedings of the 39th ACM Symposium on Theory of Computing (STOC), 590-598, 2007.

[5] Y. Han. An $O(n^3(\log\log n/\log n)^{5/4})$ time algorithm for all pairs shortest paths. Algorithmica, 51:428-434, 2008.

[6] R. Yuster. Efficient algorithms on sets of permutations, dominance, and real-weighted APSP. Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 950-957, 2009.

[7] E. W. Dijkstra. A note on two problems in connexion with graphs. Numerische Mathematik, 269-271, 1959.

[8] M. L. Fredman, R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. Proceedings of the 25th Annual Symposium on Foundations of Computer Science, 338-346, 1984.

[9] D. B. Johnson. Efficient algorithms for shortest paths in sparse networks. Journal of the ACM (JACM), 24:1-13, 1977.

[10] M. Thorup. Undirected single-source shortest paths with positive integer weights in linear time. Journal of the ACM (JACM), 46:362-394, 1999.

[11] T. Hagerup. Improved shortest paths on the word RAM. Proceedings of the 27th International Colloquium on Automata, Languages and Programming, 61-72, 2000.

[12] S. Pettie, V. Ramachandran. A shortest path algorithm for real-weighted undirected graphs. SIAM Journal on Computing, 34:1398-1431, 2005.

[13] S. Pettie, V. Ramachandran, S. Sridhar. Experimental evaluation of a new shortest path algorithm. Revised Papers from the 4th International Workshop on Algorithm Engineering and Experiments, 26-142, 2002.

[14] S. Pettie. On the comparison-addition complexity of all-pairs shortest paths. Proceedings of the 13th International Symposium on Algorithms and Computation, 32-43, 2002.

[15] S. Pettie. A new approach to all-pairs shortest paths on real-weighted graphs. Theoretical computer science, 47-74, 2004.

[16] R. K. Ahuja, K. Mehlhorn, J. Orlin, R. E. Tarjan. Faster algorithms for the shortest path problem. Journal of the ACM (JACM), 37:213-223, 1990.

[17] B. V. Cherkassky, A. V. Goldberg, C. Silverstein. Buckets, heaps, lists, and monotone priority queues. Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms, 83-92, 1997.

[18] R. Raman. Recent results on the single-source shortest paths problem. ACM SIGACT News, 28:81-87, 1997.

[19] M. Thorup. Integer priority queues with decrease key in constant time and the single source shortest paths problem. Proceedings of the thirty-fifth annual ACM symposium on Theory of

computing, 149-158, 2003.

[20] IEEE. Standard for binary floating-point arithmetic. ACM Sigplan Notices, 22:9-25, 1985.

[21] F. Chung, L. Lu. The diameter of sparse random graphs. Advances in Applied Mathematics, 26:257-279, 2001.

[22] B. Bollobás. Random graphs. Cambridge University Press, 36-39, 2001.

[23] D. R. Karger, D. Koller, S. J. Phillips. Finding the hidden-path: time bounds for all-pairs shortest paths. SIAM Journal on Computing, 22:1190-1217, 1993.

[24]O. Riordan, N. Wormald. The diameter of sparse random graphs. Arxiv preprint arXiv:0808.4067, 2008.

作者简介 黄跃峰，1981 年 6 月 15 日出生，中科院地理所博士研究生，主要研究领域：地理信息软件技术及海量空间信息计算方法．通信地址：北京市海淀区西三旗建材城西路 31 号，太伟科研楼 B 座 3 层，100096．电子邮件：huangyuefeng@supermap.com．