Turkish Journal of Electrical Engineering and Computer Sciences

Volume 25 | Number 2 Article 44

1-1-2017

Maximum size of the pareto cost sets for multi-constrained optimal routing

DERYA YILTAŞ KAPLAN

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 1211 – 1222

doi:10.3906/elk-1508-263

Maximum size of the pareto cost sets for multi-constrained optimal routing

Derya YILTAŞ KAPLAN*
Department of Computer Engineering, Faculty of Engineering, İstanbul University, İstanbul, Turkey

Received: 31.08.2015 • **Accepted/Published Online:** 13.04.2016 • **Final Version:** 10.04.2017

Abstract: Routing under multiple independent constrains in point-to-point networks has been studied for over 10 years. Its NP-hardness keeps pushing researchers to study approximate algorithms and heuristics, and many results have been published in these years. To the best of our knowledge, the nature of its average case has been explored only for the self-adaptive multiple constraints routing algorithm (SAMCRA), which is an algorithm about multiple constraints routing. In this paper, we simplify SAMCRA into a format that is convenient for our average case analysis. This variant algorithm gives optimal solutions also for very large dimensional networks such as with more than 1000 nodes. Although it runs in exponential time in the worst case, we prove that its average case time complexity is bounded by a polynomial function of the number of nodes in the network. Lastly, we give empirical results that align with our theoretical work.

Key words: Algorithm, computer network, expected polynomial time, multi-constrained routing, pareto frontier, pareto optimisation, quality of service, routing algorithm

1. Introduction

Quality of service (QoS) is a generic term for describing the goodness of network performance in the literature. It includes simultaneous optimisation of some dependent or independent metrics. In the context of network routing, the metrics can be end-to-end delay, jitter, packet delivery ratio, packet loss rate, throughput, bandwidth, number of hops, link stability, etc. [1–4]. The details of metrics diversification based on the network communication layers can be observed clearly in [3]. QoS metrics are inevitably affected by the network traffic, and different route selections in the network layer improve or worsen the performance dramatically.

1.1. Definitions and problem formulation

The network in question is a connected, and undirected, static point-to-point arbitrary graph $G = (V,E)$, where $V = \{1,2,...,n\}$ is the set of nodes and E the edge set. Each edge has a k-dimensional cost $(c_1, c_2, \ldots, c_k)$, where $c_q \in R^+$, q = 1,...,k, can be jitter, packet loss rate, bit-error rate, link stability, etc., and they are independent. The cost of an edge can be considered as a vector, and for any path the total cost is the vector sum of all vectors along the path.

Recall that the edge cost is k-dimensional, and hence the total cost of a path is also k-dimensional. Given an arbitrary set of k-dimensional points, linear order is not guaranteed. For any two points $(c_1, c_2, \ldots, c_k)$ and $(d_1, d_2, \ldots, d_k)$, if (1) $[\forall q \in [1,k]\ c_q \leq d_q]$ and (2) $[\exists q \in [1,k]\ c_q < d_q]$, then $(c_1, c_2, \ldots, c_k)$ outperforms $(d_1, d_2, \ldots, d_k)$. With only condition (1), we can say that $(c_1, c_2, \ldots, c_k)$ is at least as good as, or not worse

*Correspondence: deryayiltas@gmail.com

than, $(d_1, d_2, \ldots, d_k)$. With only condition (2), we do not know their order. In short, if [$\exists$q,r$\in$[1,k] •q$\neq$r, $c_q < d_q$ and $d_r < c_r$], then their order is undetermined, else one must outperform the other. If the order of any two elements in a set of k-dimensional points is undetermined, then the set forms a pareto frontier. A pareto frontier is sorted in two-dimensional, but not higher dimensional space.

We now define the problem. The input is the graph G(V,E), and a source node $s \in V$, k real numbers $C_1^{\max}$, $C_2^{\max}$, ..., $C_k^{\max}$, and the output is a path from s to each node for which total cost $\left(C_1^T, C_2^T, \cdots, C_k^T\right)$ is not worse than $(C_1^{\max}, C_2^{\max}, \cdots, C_k^{\max})$, and $F\left(C_1^T, C_2^T, \cdots, C_k^T\right)$ is optimised, where F is a user predefined function that maps the total cost to a value. Intuitively, this is a one-to-all shortest paths problem with k-dimensional edge cost and k-dimensional constraints. For completeness, if there is no path satisfying both constraints, then the output is false.

1.2. Background

Finding the multiple metric-constrained paths is known as the multi-constrained path problem (MCPP) in the literature. It can also be turned into an optimality problem [5,6]. Several methods have been proposed to solve the MCPP. Djarallah et al. [5] represented the metrics as a vector structure of the weight parameters. Using this vector, they treated the metrics as noncomposite numbers, which require discrete computation for each metric. Another multiple metric method is applied in [7]. In this method, after a user demands the QoS values specifically, each service provider tries to find an appropriate domain link with respect to these demanding values. Bertrand et al. [8] calculated the sum of the metric values of the links in a relaxation step by taking into account the additive metrics.

Shin et al. [9] found a multi-constrained routing solution by starting from some subpaths and then extending them into the whole paths. Xue and Ganz [10] made the routing process based on only two metrics, namely delay and bandwidth. In the same way, Yuan and Liu [11] assumed a limited number of independent metrics. They required all optimal paths to be stored and supplied the QoS requirements by using each metric's constraint.

There are also some studies that analyse the time complexity for the MCPP methods. Xue et al. summarised a few works before 2007, including the NP-hardness nature of the problem when the number of constraints is no less than two [12]. They also gave three polynomial time approximate solutions to the problem. For optimal algorithms, exponential time complexity is expected. The well-known optimal algorithm self-adaptive multiple constraints routing algorithm (SAMCRA), which is given by Van Mieghem et al. [13], has no exception. However, Van Mieghem and Kuipers [14] claimed that polynomial time can be achieved in most cases. They proved the polynomial complexity in the worst case by the help of simulation results in [15].

1.3. Our contribution

SAMCRA is a routing algorithm that is able to obtain all possible paths between a source and destination pair in an initial search-space. It also gives the shortest path even after the search-space reduction. All paths in the result set obey to the bounds of the constraints [13]. We simplify SAMCRA for the ease of average case analysis. This simplicity helps us to use the same algorithm for two or more constraints. We prove that for optimal multi-constrained routing expected time is a polynomial function to the number of nodes in the network, as long as the number of constraints is a constant. We give experimental work for various networks with node numbers from 100 up to 6000 and 2 to 128 constraints. We point out that there are not many papers using such large number of nodes and network dimensions in the literature. We choose the constraint

or metric values from various ranges and number sets such as integers or floating points, not only from the same ranges. We represent the interaction between the processing time and the number of nodes, and compute the effect of the edge probability on the processing time. The empirical results are consistent with our average case analysis. Our analysis also covers a wide range of computations by considering the whole operations along all nodes comparatively due to their final cost sets. We also assign the edge values based on two different classifications that handle the same and different metric values in opposite directions of the full-duplex edges respectively. For each computation the values cover various edge probabilities such as 1 that defines a fully connected or complete graph. We also extend the network management to cope with the relationship between the edge and node numbers in our biased and unbiased graphs. These graphs represent whether the workload of the network is distributed on any specific area of the network or among the whole network well-balanced respectively. We note that these complete experiments have not been done in the literature. Van Mieghem and Kuipers [15] presented their practical solution for networks with nodes up to 800 with constraint numbers 2, 4, and 8, and also they used node numbers around 1000 only for hop count statistics. They computed their function's statistics for the constraint numbers 16 and 32 only for a network with 25 nodes. They also chose the link weights from the range [0, 1] as uniformly distributed random variables. Their theoretical analysis and practical solution cover different parameters than ours.

1.4. Organisation of the paper

The rest of this paper is designed as follows. In Section 2, we give a variant of SAMCRA as an expected polynomial time algorithm for optimal multi-constrained routing. We give the probabilistic analysis of the problem in Section 3. In Section 4, we present the experimental results that we obtained from several conditions such as along various network topologies with different number of nodes and for different constraint numbers. Finally, we give the conclusions of this study in Section 5 and then the acknowledgements part.

2. A variant algorithm

Associate each node with a set of pareto costs, which are the costs from the useful paths originated from s. The value of a cost is represented by the k-tuple, (c_1, c_2, ..., c_k) as mentioned in Subsection 1.1. By "useful path", we refer to a path whose cost is not outperformed by others. We need only pareto sets because the costs of underperforming paths are not useful in any case. The set for s is initially $\{(0, 0, ..., 0)\}$, and the sets for the other nodes are initially empty. Let the set in node i be S_i, for all i in V.

We now give a simple, but optimal, algorithm called the exponential polynomial time algorithm (ExPoTA), which is given in Figure 1.

The algorithm in Figure 1 runs in expected polynomial time and contains $n - 1$ rounds. Each round can be considered as one step going out from each node to all its neighbours and the cost sets S_* are updated on the way. The updating of the cost sets follows its requirement: no element can outperform the other, and each one is unique. After $n - 1$ rounds, s can reach all nodes, leaving all cost sets optimal.

For k = 1 and k = 2, ExPoTA is not efficient, because the linear cost structure inside each S_i, $i = 1$ to$n$,is not used. However, the aim of this paper is to study S_i for general value of k and the algorithm is good enough for our analysis. The algorithm runs in polynomial time of the size of S_i, $i = 1$ to n (steps 3.1.1.1 and 3.1.1.1.1.3). In the worst case, the sizes can grow exponentially; however, in the next section, we argue that they are bounded by linear function of n in the average case.

```
1           For each node i in V, i ≠ s.  // s is the source.
1.1             Create empty set S_i of paths with pareto costs from s to i.
2           Initialize S_s = {(0,0, ... , 0)}.
3           For round = 1 to n-1
3.1             For each node i in V
3.1.1               if S_i is not empty
3.1.1.1                 For each element (c1, c2, ..., ck) in S_i,
3.1.1.1.1                   For each neighbour j of i
3.1.1.1.1.1                     Let the cost of edge (i,j) be (d1, d2, ..., dk).
3.1.1.1.1.2                     kill = false;  // kill is a boolean variable
3.1.1.1.1.3                     For each element (b1, b2, ..., bk) in S_j,        (*)
                                    If cq+dq ≥ bq , for all q∈[1,k]
                                    kill = true;
                                    Quit the For-loop labeled with (*) //Break
                                    If cq+dq ≤ bq , for all q∈[1,k]
                                    remove (b1, b2, ..., bk) from S_j;
3.1.1.1.1.4                     If kill = false
                                    Store (c1+d1, c2+d2, ..., ck+dk) in S_j;
```

Figure 1. Pseudocode of ExPoTA.

2.1. Probabilistic analysis

In this section, we will argue that the size of S_i, $i = 1$ to$n$,will be bounded by a linear function of n, for small value of k, assuming that the two costs $(c_1 + d_1, c_2 + d_2, ..., c_k + d_k)$ and $(b_1, b_2, ..., b_k)$ in step 3.1.1.1.1.3 are randomly distributed in the k-dimensional space.

The probability that $(b_1, b_2, ..., b_k)$ is removed from S_j is $\frac{1}{2^k}$, because $c_q + d_q \leq b_q$ is true for q = 1, 2, ..., k. Similarly, the probability that $(c_1 + d_1, c_2 + d_2, ..., c_k + d_k)$ is not stored in S_j is also $\frac{1}{2^k}$. If there are n elements in S_j before comparison, the probability that $(c_1 + d_1, c_2 + d_2, ..., c_k + d_k)$ is stored in S_j is $\left(1 - \frac{1}{2^k}\right)^n$ and the probability that no element in S_j is removed is also $\left(1 - \frac{1}{2^k}\right)^n$. Hence, for one iteration of the for-loop in step 3.1.1.1.1, the probability that the size of S_j increases by one is $\left(1 - \frac{1}{2^k}\right)^{2n}$, which is very close to zero for small k. On the other hand, the probability for a decrease in the size of S_j is much higher, and it equals the probability that $(c_1 + d_1, c_2 + d_2, ..., c_k + d_k)$ outperforms any two or more elements in S_j. It is $\binom{n}{2}\left(\frac{1}{2^k}\right)^2\left(1 - \frac{1}{2^k}\right)^{n-2} + \binom{n}{3}\left(\frac{1}{2^k}\right)^3\left(1 - \frac{1}{2^k}\right)^{n-3} + \cdots + \binom{n}{n}\left(\frac{1}{2^k}\right)^n\left(1 - \frac{1}{2^k}\right)^0 = 1 - \binom{n}{0}\left(\frac{1}{2^k}\right)^0\left(1 - \frac{1}{2^k}\right)^n - \binom{n}{1}\left(\frac{1}{2^k}\right)^1\left(1 - \frac{1}{2^k}\right)^{n-1}$, which is very close to one for large n and small k. It

is obvious that when k is small, if the size of S_j reaches n, then it is likely to drop, rather than to rise. This explains why Max($|S_j|$), j=1,... n, is bounded by n in our experimental result shown below.

For theoretical interest, we briefly look into the case for larger k. When k tends to $\log_2 n$, the probability for increasing the size of S_j tends to $\left(\frac{1}{e}\right)^2$, and that for decreasing is $1-\left(\frac{2}{e}\right)$; and this implies a significant chance of keeping unchanged. When k tends to $2\log_2 n$, the two probabilities become $\left(\frac{1}{\sqrt[n]{e^2}}\right) \approx 1$ and $1-\frac{1}{\sqrt[n]{e}}\left(1+\frac{n}{n^2-1}\right) \approx 0$, respectively. However, they will become $\left(\frac{1}{e}\right)^{2f(n)} \approx 0$ and $1-\left(\frac{1}{e}\right)^{f(n)}\left(1+\frac{n}{n^2-1}\right) \approx 1$ again, respectively, when the size of S_j reaches $n^2 \times f(n)$, where $f(n)$ is any monotonic increasing function of n (like $\log_2 n$, $\log_2 \log_2 n$, $\cdots$) and will tend to infinity with n. Extending this argument, we conclude that the size of S_j will become steady at $O(2^k)$.

3. Empirical results

We built our C programs on MS Visual Studio 2012 environment. We performed our tests on a laptop with Intel Core i5 CPU 2.67 GHz, 4.00 GB RAM and a PC with Intel Core i7-2600 CPU 3.40 GHz, 8.00 GB RAM. We computed all experimental results over a sample of mesh network with full-duplex edges. We constructed all edges randomly between any two nodes. We executed various computations depending on the properties explained in the following subsections.

3.1. Processing time

We provided each edge in the network with numerical values of delay and packet loss rate. Delay values are randomly distributed between [1, 250] and the packet loss rates should be randomly chosen from one of the values 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, and 10^{-8}. The value bounds are widely used similar to these selections in network applications. Packet loss rates are also very small in general. For this reason we chose discrete values for packet loss rates to diversify successive numbers for concrete comparisons.

We computed the *log* function of the average processing time of ExPoTA for each node number varying between 100 and 1000, and used 10 different instances for each one. We illustrated the graphical representation as in Figure 2. The average processing time for 100 nodes converges to 0; for this reason the *log* result of that point is negative. The results become exponential with the increase in the number of nodes in the network. This implies that the time is bounded by a polynomial function.

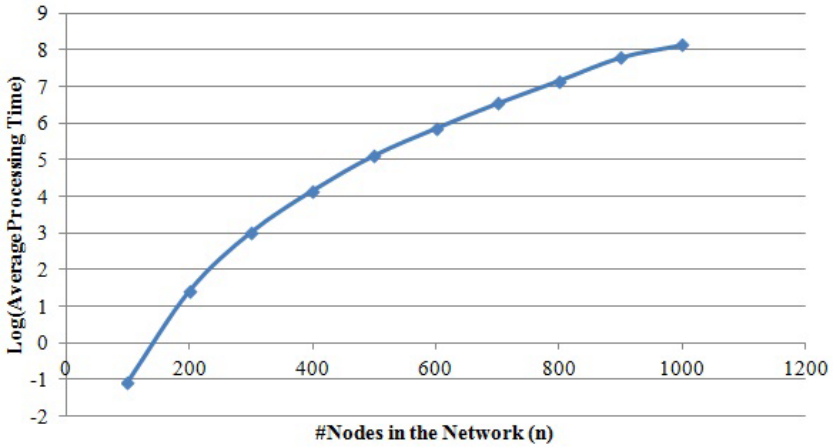

Figure 2. Log(average processing time) vs #nodes for 2-metrics (edge probability = 0.5).

We also analysed the average processing time according to the decrease in the 'edge probability' between two nodes. Edge probability represents the probability that there is a connection between the node pairs. For Figure 2, we built all edges with a probability of 0.5. After the implementation of Figure 2, we tried the edge probabilities of 0.5, 0.2, 0.05, 0.02, 0.005, and 0.002 between the node pairs. We planned the network scenarios with 500 and 1000 nodes. We computed the average processing time of 10 different network instances for each edge probability in a node number in Figure 3. As seen in Figure 3, in both of the networks, the average processing time is proportional to the edge probability. This is because the edge number in the network generally increases with the edge probability. Therefore, the process numbers related to the edges and total durations also increase.

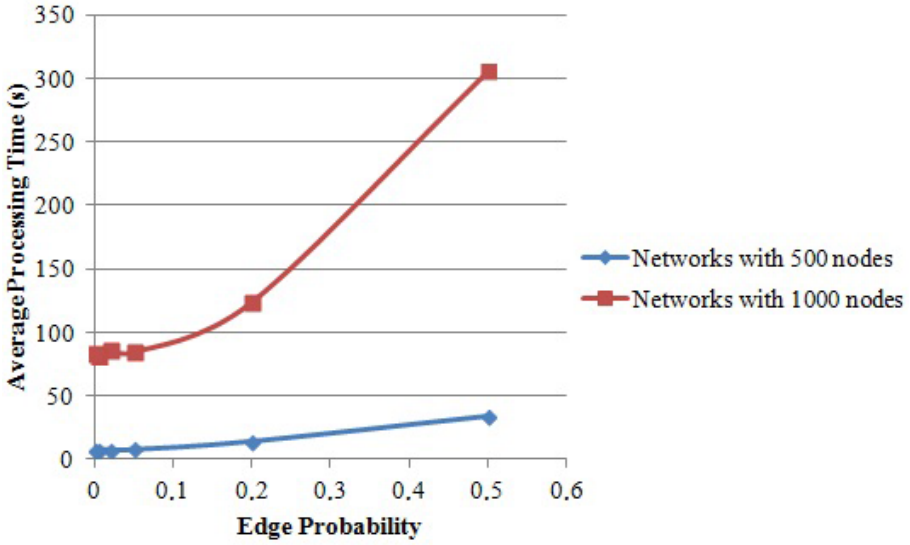

Figure 3. Average processing times for different edge probabilities.

3.2. Different number of metrics

We extended our experimental analysis in Subsection 4.1 considering additional metrics such as jitter, bandwidth requirement, and throughput. The values of all metrics can be chosen from discrete or continuous numbers inside the programs. For example, jitter values are randomly distributed in [1, 50], bandwidth requirements are within [1, 100] randomly, and throughput values are chosen from several different discrete numbers. As a note, we experienced that the nature of the value bounds would not have any impact on the results. We classified our tests executing with 2, 3, and 4 metrics, consecutively. 2-Metrics cover delay and packet loss rate as mentioned above. Additional to these metrics, 3- and 4-metrics involve the new ones with respect to attaining the relevant metric number. We constructed several networks with various numbers of nodes n from 100 up to 6000. We built all network edges with a probability of 0.5. This means that there are lots of edges in each network type. Table 1 represents the maximum number of pareto costs staying all along S_i sets, denoted as Max($|S_i|$) in the designed networks. Here we note that S_i stores the pareto points at the ith node as explained in Section 2. All points must be processed for each round for any algorithm; therefore, the average case and worst case time complexities depend very much on the Max($|S_i|$) rather than other algorithmic issues. In fact, their sizes depend on the input (graph, source node), but not the algorithm used. The importance of such sets is also demonstrated in [15]. The authors proved that the upper bound on the expected size is $<(\mathrm{N} \ln \mathrm{N})^{k-1}$, where k is the dimension. This result affects their time complexity analysis very much.

Table 1. Max($|S_i|$) values for different number of metrics (edge probability = 0.5).

	With 2-metrics	With 3-metrics	With 4-metrics
n	Max(|Si|)		
100	10	25	31
200	10	29	51
300	11	31	63
400	12	36	63
500	14	47	73
600	11	46	78
700	11	55	79
800	12	44	69
900	10	40	75
1000	11	42	79
2000	11	50	87
3000	10	52	94
4000	10	54	112
5000	11	54	113
6000	10	63	123

As seen in Table 1, the metric number raises Max($|S_i|$) for each n. In other words, when the metric number is getting larger, Max($|S_i|$) will increase with greater probability. This is caused by the fact that the outperformance of any cost with larger number of metrics occurs rarely with respect to the comparisons among all metric values. Furthermore, in Table 1, the increase in Max($|S_i|$) related to the number of nodes can be especially seen along about 3- or 4-metrics. There are some fluctuations in 2-metrics, because in 2-metric classification the comparison between the costs does not have much more difference even in different node numbers in the networks. Thus, sometimes there can be an increase and sometimes a decrease with small units for 2-metrics.

There is also an observable difference for 2-metrics and the others in Table 1 according to both the Max($|S_i|$) results for any specific node number and the changing ratio between the results of any two consecutive node numbers. The reason is that there is a linear order for 2-metrics, but not for 3- and 4-metrics through the cost comparisons. In 2-metrics, the increasing order in one metric is exactly the decreasing order of the other, because the elements in S_i are pareto as mentioned above. This means indirectly that the outperformance of any cost can be seen very easily and frequently during the cost comparisons in 2-metrics. Adversely, in 3- and 4-metrics, the outperformance of any cost occurs very rarely in terms of almost all metrics linearly. For example, in 3-metrics, sometimes the delay and packet loss rate of a cost outperform all other costs while jitter does not bring any support. On the other hand, for some other costs in S_i, the outperforming metrics may be the packet loss rate and jitter excluding the delay. Therefore, it is not frequent to find a cost value that outperforms the others and removes them from the result set S_i of any node i. For this reason Max($|S_i|$) values of 3- and 4-metrics are larger than that of 2-metrics.

We implemented the triangular network structure in [15] to compare with our random networks that give the results in Table 1. Because the triangular type is referred to as 'chain topology' in [15] we use this name hereafter. We performed 2, 3, and 4 metrics on the chain topologies. We chose the metric values randomly as the same bounds as in our random networks. Figure 4 represents the comparison between our random networks and the chain topologies. In Figure 4, we refer to our network topologies working with our algorithm ExPoTA as Random and the ones in [15] as Chain. Figure 4 shows that we have different findings of the same problem

in the area of multi-constrained routing. We note that our network topologies give smaller Max($|S_i|$) values and also these topologies are more suitable for real network applications covering randomised (dis)connections.

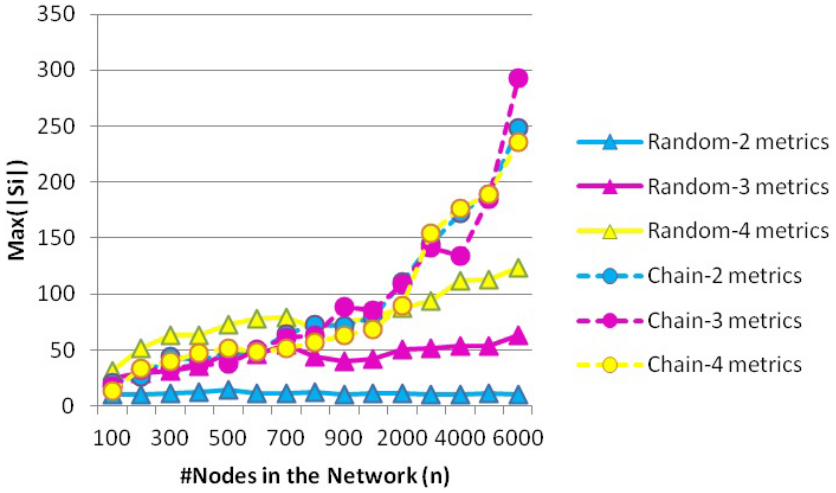

Figure 4. The comparison of random and chain topologies.

Moreover, in this subsection, we completed new trials for ExPoTA with very large number of metrics. As we mentioned in Subsection 1.3, when we compare with any other multi-constrained study in the literature, it is a favoured ability of ExPoTA to settle a lot of metrics simultaneously. We set a network with 500 nodes. Table 2 gives the results for the experiments covering 4, 8, 16, 32, 64, and 128 metrics. As seen in Table 2, the results converge to some similar values after a number of metrics such as 16-metrics as explained in Section 3.

Table 2. Max($|S_i|$) values for large numbers of metrics.

	Max(|Si|)					
Edge probability	With 4-metrics	With 8-metrics	With 16-metrics	With 32-metrics	With 64-metrics	With 128-metrics
1	83	181	499	499	499	499
1 / 2	76	193	276	289	287	285
1 / 4	61	142	163	164	158	160
1 / 8	44	82	84	84	85	86
1 / 16	32	46	48	53	53	57

3.3. Network classification depending on edge properties

We constructed two different network classes, namely Type-I and Type-II, for the next practices. Type-I covers the edges having the same metric values through both transmission directions in between any node pairs. We also used this class in Table 1 samples. On the other hand, Type-II has the edges with different metric values in both directions. We used 3-metrics on these network structures and found Max($|S_i|$) under the edge probabilities of 1, 1 / 2, 1 / 4, 1 / 8, and 1 / 16, respectively. The edge probability 1 provides a fully connected (complete) graph. Tables 3 and 4 give the results for Type-I and Type-II network, respectively. There is not much difference between the results of these two network classes. The main difference appears during the change of the edge probability. Tables 3 and 4 prove that Max($|S_i|$) is directly proportional to the edge probability. In other words, Max($|S_i|$) decreases with the reduction in the edge probability in any network. Additionally as

seen in Tables 3 and 4, when the edge probability is double, Max($|S_i|$) is less than double for any n between 100 and 6000.

Table 3. Max($|S_i|$) values depending on the edge probabilities (type-I networks).

	Edge probability = 1	Edge probability = 1 / 2	Edge probability = 1 / 4	Edge probability = 1 / 8	Edge probability = 1 / 16
n	Max($\|S_i\|$)				
100	46	27	14	9	6
200	92	51	30	18	11
300	123	72	40	25	15
400	168	91	54	32	19
500	208	111	67	40	23
600	250	134	77	44	26
700	282	150	86	50	28
800	322	176	99	52	32
900	370	193	110	62	35
1000	400	217	118	65	38
2000	783	406	222	120	69
3000	1160	610	331	175	99
4000	1526	793	420	225	128
5000	1887	980	512	273	144
6000	2219	1173	608	328	175

Table 4. Max($|S_i|$) values depending on the edge probabilities (type-II networks).

	Edge probability = 1	Edge probability = 1 / 2	Edge probability = 1 / 4	Edge probability = 1 / 8	Edge probability = 1 / 16
n	Max($\|S_i\|$)				
100	46	31	18	10	6
200	89	52	34	18	12
300	127	75	40	26	15
400	169	97	52	30	19
500	211	111	63	37	22
600	246	136	73	42	25
700	291	154	85	49	30
800	324	176	94	53	32
900	361	190	111	62	39
1000	399	222	116	67	37
2000	776	413	217	116	68
3000	1171	601	324	174	97
4000	1529	797	418	232	122
5000	1904	991	507	271	148
6000	2236	1161	614	324	184

3.4. Different edge numbers and edge distributions

We carried out two different experiment sets considering the 3-metrics, namely delay, packet loss rate, and jitter. The edge probability is 0.5. We controlled the edge numbers along the network to become multiples of the node numbers. In the first experiment set, we chose the edges starting from the smaller node indices that are small

values of i in V as aforementioned. We say that this graph is biased because the nodes with smaller indices have more connections on average. In the second experiment set, we chose the edges randomly from the set E. This new graph is unbiased in which there is equity between the edge selections along the whole network. Table 5 represents the average results per edge number $|E|$ giving biased results before the comma and unbiased results after the comma. We note that we obtained the average results based on 5 different executions of the program each time. We also note as a detail that the processing time of this experiment for each n becomes reasonably smaller than the previous executions because of the edge number bounds.

As can be seen in Table 5, for each n, the average Max($|S_i|$) values increase with the rise in the $|E|$ values. Sometimes there may also be exceptions as for biased results in $|\mathrm{E}| = 4 \times \mathrm{n}$ with n = 200, 800, and 900, which represents the values not larger than that of $|\mathrm{E}| = 3 \times \mathrm{n}$. The unbiased results are dramatically smaller than those of biased form because of the fair distribution in the edge selections. In this unbiased form, the edges in between various nodes are selected; thus every time the same nodes do not take place in the algorithm steps. Hence the size of any set S_i does not become too large.

Table 5. Average results of Max($|S_i|$) for biased , unbiased graphs with respect to various edge numbers.

	$\|\mathrm{E}\| = \mathrm{n}$	$\|\mathrm{E}\| = 2 \times \mathrm{n}$	$\|\mathrm{E}\| = 3 \times \mathrm{n}$	$\|\mathrm{E}\| = 4 \times \mathrm{n}$
n	Average of Max($\|S_i\|$)			
100	16 , 3	21 , 5	22 , 7	22 , 8
200	26 , 3	43 , 6	43 , 8	40 , 8
300	47 , 3	59 , 7	64 , 8	65 , 10
400	52 , 3	77 , 7	79 , 8	84 , 9
500	66 , 4	91 , 7	99 , 9	106 , 9
600	97 , 4	114 , 7	118 , 9	123 , 10
700	121 , 4	135 , 7	140 , 9	141 , 11
800	137 , 5	157 , 8	165 , 9	161 , 10
900	141 , 4	166 , 7	171 , 9	171 , 11
1000	156 , 5	168 , 7	199 , 9	201 , 11
2000	332 , 5	377 , 8	394 , 9	396 , 11
3000	409 , 5	562 , 8	584 , 9	577 , 12
4000	550 , 5	763 , 8	781 , 10	784 , 11
5000	740 , 5	867 , 8	943 , 10	957 , 12
6000	921 , 6	1146 , 8	1154 , 10	1159 , 12

4. Discussion

In this paper, we present an expected polynomial time algorithm ExPoTA as a variant of SAMCRA for k-constraint optimal routing in networks. The novelty of this paper is not in the ExPoTA, but in the analysis of the origin of the time complexities for all algorithms of multi-constrained routing. ExPoTA is an optimal algorithm. We skipped the proofs (of optimality and exponential time complexity in the worst case), as it is already guaranteed by Van Mieghem and Kuipers [15]. For the analysis, we noticed that the average case and worst case time complexities depend very much on the Max($|\mathrm{Si}|$) rather than other algorithmic issues. As mentioned above, Van Mieghem and Kuipers [15] proved that the upper bound on the expected size is $<(\mathrm{N} \ln \mathrm{N})^{k-1}$, where k is the dimension. It is the best existing result. In our paper, we investigate the expected size by experiment, and the empirical result suggests a much lower value. We think that the empirical result is a motivation for finding a better upper bound.

Our computer program results demonstrate that the average case time complexity of our algorithm is bounded by a polynomial function of the number of nodes in the network and the processing time is exponential in the worst case.

We point out the conclusions of this paper as a summary as follows:

(1) The processing time is exponential in the worst case; it is indeed polynomial on average. The time complexity depends on Max(|Si)|). Therefore, we shift our focus to Max(|Si|).

(2) Max($|S_i|$) results for any specific node number and the changing ratio between the results of any two consecutive node numbers are different in between various metric numbers such as 2, 3, and 4 metrics. After a saturation point as a large number of metrics, the results converge to each other such as in 16, 32, 64, and 128 metrics.

(3) When the same metric bounds are used for both the chain topology mentioned by Van Mieghem and Kuipers [15] and this paper's random topology, the former results in larger Max(|Si|) values entirely for different metric numbers.

(4) To change the parameter values to similar or different values in both directions for an edge does not influence the results much.

(5) When the edge probability is double, Max(|Si|) is less than double for any number n between 100 and 6000 in both Type-I and Type-II networks. It is reasonable to consider the performance for high edge-probability only. Additionally, when the edge numbers are bounded based on the node numbers of the networks, the processing time of the experiments become reasonably smaller.

(6) The average Max(|Si|) values increase with the rise in the $|E|$values in both biased and unbiased graphs. However, the values in unbiased graphs are much smaller than those in biased graphs. One reason may be that various nodes take place in the algorithm steps of unbiased graphs; thus the costs do not cumulate in the same nodes especially the ones with smaller indices every time.

Acknowledgements

The author would like to sincerely thank Savio SH Tse for sharing his comments during the preparation of this study. This work was supported by the Scientific Research Projects Coordination Unit of İstanbul University. Project Number: 10590.

References

[1] Wang CF, Ding JW, Chen TC. A routing protocol for mobile ad-hoc networks using the profit optimization model. Int J Commun Syst 2014; 27: 2851-2869.

[2] Santhi G, Nachiappan A. A survey of QoS routing protocols for mobile ad hoc networks. International Journal of Computer Science & Information Technology (IJCSIT) 2014; 2: 125-136.

[3] Kandari S, Pandey MK. Constraint based QoS routing in MANET. In: International Conference on Advances in Engineering and Technology; 29–30 March 2014; Singapore. New York, NY, USA: IRED. pp. 63-68.

[4] Tanwar S, Kumar N, Rodrigues JJPC. A systematic review on heterogeneous routing protocols for wireless sensor network. J Netw Comput Appl 2015; 53: 39-56.

[5] Djarallah NB, Sauze NL, Pouyllau H, Lahoud S, Cousin B. Distributed E2E QoS-based path computation algorithm over multiple inter-domain route. In: 3PGCIC 2011 Sixth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing; 26–28 October 2011; Barcelona, Catalonia, Spain. New York, NY, USA: IEEE. pp. 169-176.

[6] Ali NB, Molnár M, Belghith A. Multi-constrained QoS Multicast Routing Optimization. Research Report, Institut De Recherche En Informatique Et Systèmes Aléatoires, Rennes Cedex, France, 2008.

[7] Yampolskiy M, Hommel W, Lichtinger B, Fritz W, Hamm MK. Multi-domain end-to-end (E2E) routing with multiple QoS parameters-considering real world user requirements and service provider constraints. In: Second International Conference on Evolving Internet (INTERNET); 20–25 September 2010; Valencia, Spain. Red Hook, NY, USA: IARIA. pp. 9-18.

[8] Bertrand G, Lahoud S, Texier G, Molnár M. Computation of multi-constrained paths in multi-domain MPLS-TE networks. In: Next Generation Internet Networks; 1–3 July 2009; Aveiro, Portugal. New York, NY, USA: IEEE. pp. 1-8.

[9] Shin DW, Chong EKP, Siegel HJ. Multi-postpath-based lookahead multi-constraint QoS routing. J Franklin I 2012; 349: 1106-1124.

[10] Xue Q, Ganz A. Ad hoc QoS on-demand routing (AQOR) in mobile ad hoc networks. J Parallel Distr Com 2003; 63: 154-165.

[11] Yuan X, Liu X. Heuristic algorithms for multi-constrained quality of service routing. In: IEEE INFOCOM 2001 Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies; 22–26 April 2001, Anchorage, AK. New York, NY, USA: IEEE. pp. 844-853.

[12] Xue G, Sen A, Zhang W, Tang J, Thulasiraman K. Finding a path subject to many additive QoS constraints. IEEE ACM T Network 2007; 15: 201-211.

[13] Van Mieghem P, De Neve H, Kuipers F. Hop-by-hop quality of service routing. Comput Netw 2001; 37: 407-423.

[14] Van Mieghem P, Kuipers FA. Concepts of exact QoS routing algorithms. IEEE ACM T Network 2004; 12: 851-864.

[15] Van Mieghem P, Kuipers FA. On the complexity of QoS routing. Comput Commun 2003; 26: 376-387.