Hindawi Publishing Corporation
Modelling and Simulation in Engineering
Volume 2014, Article ID 658408, 8 pages
http://dx.doi.org/10.1155/2014/658408

Research Article

Performance Analysis of Dijkstra-Based Weighted Sum Minimization Routing Algorithm for Wireless Mesh Networks

Nuha A. S. Alwan

Department of Computer Engineering, College of Engineering, Baghdad University, Baghdad, Iraq

Correspondence should be addressed to Nuha A. S. Alwan; n.alwan@ieee.org

Received 5 February 2014; Revised 19 April 2014; Accepted 22 April 2014; Published 6 May 2014

Academic Editor: Ricardo Perera

Multiobjective optimization methods for routing in static wireless mesh networks (WMNs), with more than one QoS measure to be optimized, are highly challenging. To optimize the performance for a given end-to-end route in a static network, the most common metrics that need to be optimized or bounded are the path capacity and the end-to-end delay. In this work, we focus on combining desirable properties of these two metrics by minimizing a weighted metrics sum via a Dijkstra-based algorithm. The approach is directed towards fast convergence rather than optimality. It is shown that the resulting algorithm provides more satisfactory results than simple Dijkstra-based pruning algorithms in terms of simultaneously achieving high capacity and small delay. The effect of changing the weighting factor on the proposed algorithm performance is investigated.

1. Introduction

Wireless mesh networks (WMNs) consist of mesh routers that are mainly stationary and provide wireless access to clients in a multihop environment. They find wide applications as community networks, enterprise networks, and last-mile access networks to the Internet. Being static, they have neither mobility nor power consumption issues [1]. Quality-of-service (QoS) routing in WMNs has been attracting considerable research for several years. A variety of routing metrics have been proposed for WMNs providing routing algorithms with high flexibility in best path selection. In general, the routing problem can be viewed as a multiobjective optimization problem with the most common metrics that need to be optimized or bounded being the path capacity or rate and the end-to-end delay. The path rate is the number of bits per second (bps) that can be sent along the source-destination path. The QoS metrics are generally optimized by making meaningful trade-offs due to their interconflicting nature.

It is worthwhile to note that minimizing delay and maximizing path capacity or throughput are concerned with individual application performance, that is, to optimize the performance for a given end-to-end path in a static network. In contrast, there are other metrics to be optimized for nonstatic WMNs such as mobile ad hoc networks, leading to network throughput maximization, energy consumption minimization, and equal distribution of traffic loads. These metrics are system-oriented objectives that focus on the performance of the network as a whole [2]. In this work, we focus on static WMNs.

We deal with the routing problem using the dynamic programming (DP) approach [3–5]. DP is a very powerful algorithmic paradigm, in which a problem is solved by identifying a collection of subproblems and tackling them one by one, smallest first, using the answer to small problems to solve larger ones [5]. In this context, the routing concept is closely related to finding the shortest path in a directed graph (DG) via DP techniques. Let $G(V, E)$ be the DG under consideration, where V is the set of vertices (network nodes) and E is the set of edges (network links) between the vertices. In WMNs, the edges are bidirectional and have costs assigned to them. The shortest path between any two vertices is that consisting of consecutive edges for which the overall cost is minimized. There are various applicable DP algorithms for routing when a single metric is to be minimized or optimized, such as the Dijkstra, Bellman-Ford, and Floyd-Warshall techniques [6, 7]. These differ in the processing time

and the amount of information that needs to be collected from other nodes, making each of them convenient for a specific routing scenario.

Multiconstrained optimization methods for path selection, with more than one metric to be optimized, are highly challenging and have been proved to be NP-complete [5, 8]. A problem is NP-complete if all decisions can be verified in polynomial time [5].

To achieve single-objective optimization for routing, Dijkstra's algorithm can be applied to a network. This algorithm finds the shortest path between two nodes by developing the path in order of increasing path length. It has the advantage of fast shortest path determination and a computational complexity of $O(N^2)$, where N is the number of nodes, rendering it efficient for use with relatively large networks.

As a first attempt at multiobjective optimization, one may consider a simple pruning procedure. This consists of optimizing one metric (path capacity for instance) using Dijksta's algorithm and then pruning the graph tree by discarding the branches (paths) that violate a bound set to another metric (delay for instance). There are many other instances of multiobjective optimization for routing in the literature. For example, in [9], a QoS multiconstrained routing algorithm based on simulated annealing (SA-RA) is proposed. This algorithm first uses an energy function to translate multiple QoS metrics into a single metric and then seeks to find a feasible path by simulated annealing. The latter is a general stochastic approximation method capable of handling multiple and conflicting requirements (multiobjective optimization) [10]. A genetic-algorithm-based routing method called GAMAN was proposed in [11] to cope with the multiconstrained QoS routing problem. The algorithm, however, suffers from untimely convergence as with most computationally complex GA algorithms, and the SA-RA method of [9] is shown to outperform it. However, even simulated annealing (SA) has the weakness of slow convergence time; typical values are 1 ms for 100 nodes and 20 seconds for 1000-node networks [10]. For this reason, we focus in this work on other facets of multiobjective optimization for QoS routing, namely, combining desirable properties of two metrics by minimizing a weighted metrics sum via a Dijkstra-based algorithm. It must be noted, however, that this approach is directed towards fast convergence rather than optimality; it is unable to sufficiently explore the solution space in order to obtain an optimal solution the way powerful GAs can.

The weighted sum method is the most common approach to multiobjective optimization. We will show that Dijkstra's algorithm with a weighted sum metric provides more satisfactory results than the simple pruning algorithm in terms of simultaneously achieving high path capacity and small end-to-end delay. We will also analyze the performance of the algorithm under consideration and study the effect of changing the weighting factor and source-destination pairs on routing optimality.

The rest of the paper is organized as follows: Section 2 discusses Dijkstra's algorithm with one metric (delay or capacity) and explains the concept of pruning as a step towards multiobjective optimization for QoS routing. Section 3 describes the weighted sum method and how to combine it with Dijkstra's algorithm and gives an insightful justification to its feasibility and viability. Various results demonstrating the algorithm performance are presented and discussed in Section 4 and conclusions are drawn in Section 5.

2. Dijkstra's Algorithm

Although Dijkstra's algorithm for the determination of shortest paths is well established in the literature [6], we find it convenient to give a brief explanation of it as follows.

(1) The set of nodes so far incorporated (T) consists of only the source node (s), and the initial path costs (w) to neighboring nodes are simply the link costs (L):

$$T = \{s\}$$
$$L(\text{node}) = w(s, \text{node}) \quad \text{for node} \neq s. \tag{1}$$

(2) The neighboring node not in T that has the least-cost path from node s is found and then that node is incorporated into T. Also incorporated is the edge that is incident on that node and a node in T that contributes to the path

$$\text{Find } x \notin T \quad \text{such that } L(x) = \min_{j \notin T} L(j). \tag{2}$$

Add x to T along with the edge that is incident on x and that contributes the least-cost component to $L(x)$.

(3) A comparison is made between the path cost from node s to any node in the network and the summation of path costs from node s to node x (x is another node in the network) and the link cost between node x and the node considered, and the minimum is chosen:

$$L(\text{node}) = \min\left[L(\text{node}), L(x) + w(x, \text{node})\right] \quad \forall\, \text{node} \notin T. \tag{3}$$

(4) Steps (2) and (3) are repeated until final paths have been assigned to all nodes in the network. The algorithm ends when all nodes have been taken. The running time of this algorithm is $O(N^2)$ where N is the number of nodes in the network.

The individual path costs in a path may be additive (as in end-to-end delay) to compute the overall path cost, or they can be compared to find their maximum or minimum value (as in capacity) and assign this value to the overall path. In some cases, where the link costs represent probability of error, for instance, the link costs are multiplied [12] to find the probability of error of the corresponding path.

Algorithm 1 is a pseudocode illustrating how Dijkstra's algorithm finds the optimal route capacity as a QoS measure. $P(vs, u)$ is the path from node vs to u, and $R(P(vs, u))$ is its rate. Likewise, luv is the link between node u and v and $r(luv)$

```
(1)  INPUT: no. of nodes n, source node vs, destination node vd, r(l) for all l.
(2)  OUTPUT: P*(vs, vd), R(P*(vs, vd))
(3)  /* Initialization */
(4)  FOR ALL nodes
(5)        Incorporated nodes = NIL
(6)        R(P(vs, node)) = 0
(7)        Parent(node) = NIL
(8)  END
(9)  R(P(vs, vs)) = ∞
(10) FOR i = 1 : (n − 1)
(11)       FOR ALL incorporated nodes
(12)             Rate(node) = R(P(vs, node))
(13)       END
(14)       r_max = max[rate(node)]
(15)       u = node corresponding to r_max.
(16)       /* u is the incorporated node */
(17)       FOR ALL nodes v
(18)             IF r(l_uv) < R(P(vs, u)) THEN
(19)                cap = r(l_uv)
(20)             ELSE cap = R(P(vs, u))
(21)             END IF
(22)             IF cap > R(P(vs, v)) THEN
(23)                R(P(vs, v)) = cap
(24)                parent(v) = u
(25)                /* P(vs, v) = P(vs, u) ⊕ l_uv */
(26)             END IF
(27)          END FOR
(28)          IF parent(vd) ≠ NIL THEN
(29)             P(vs, vd) = [vd]
(30)             t = vd
(31)             WHILE t ≠ vs
(32)                      p = parent(t)
(33)                      P(vs, vd) = [p P(vs, vd)]
(34)                      t = p
(35)             END
(36)          END IF
(37) END FOR
(38) P*(vs, vd) = P(vs, vd)
(39) RETURN P*(vs, vd), R(P*(vs, vd))
```

Algorithm 1: The Dijkstra-based routing algorithm (optimizing path capacity).

is its rate. The asterisk denotes optimality, and $\oplus$ denotes concatenation.

Delay optimization by Dijkstra's algorithm would be almost identical to the one that optimizes capacity except that the link costs are added rather than compared to find the overall path cost. For delay optimization, the total cost is actually "minimized" conforming with the general shortest path terminology.

As a step towards multiobjective optimization in routing, we may employ Dijkstra's technique to compute high-capacity paths while simultaneously bounding the end-to-end delay to an upper limit. This is achieved as follows. Before the decision to include a node in the set of incorporated nodes T, the delay of the path up to that node is considered, and if it is found to exceed the bound, this node is disregarded and the other node, that the first was favored to, is incorporated instead. This is called pruning. The paths that violate the delay bound are pruned off by comparing rates to decide between paths as in Algorithm 1 and then comparing delays. The latter comparison may change the decisions resulting from the former. Dijkstra with pruning retains all the corresponding advantages such as fast shortest path computation and relatively small computational complexity. However, its effectiveness in optimizing two metrics is limited as is shown in the section on results.

3. Dijkstra-Based Weighted Sum Minimization (DWSM) Algorithm

Before presenting the Dijkstra-based weighted sum minimization (DWSM) algorithm, which is the focus of this paper, we first discuss general concepts related to multiobjective optimization.

3.1. The Multiobjective Optimization Problem. This problem is stated as follows:

$$\begin{aligned} &\text{Minimize} \quad \mathbf{F}(\mathbf{x}) = \left[F_1(\mathbf{x}), F_2(\mathbf{x}), \ldots, F_k(\mathbf{x})\right]^T \\ &\text{Subject to} \quad g_j(\mathbf{x}) \le 0, \quad j = 1, 2, \ldots, m, \\ &\qquad\qquad h_l(\mathbf{x}) = 0, \quad l = 1, 2, \ldots, e, \end{aligned} \tag{4}$$

where k is the number of objective functions, m is the number of inequality constraints, and e is the number of equality constraints. $\mathbf{x} \in E^n$ is a vector of design variables (also called decision variables), where n is the number of independent variables [13]. In a network, the $\mathbf{x}$ represent the paths each with n links or n nodes depending on the context. $\mathbf{F}(\mathbf{x}) \in E^k$ is a vector of objective functions $F_i(\mathbf{x}) : E^n \rightarrow E^1$. $F_i(\mathbf{x})$ are also called objectives or cost functions. In a network, they represent path capacity, end-to-end delay, and so forth. These metrics differ with the paths so they are actually functions of the network paths denoted by the $\mathbf{x}$. The constraints can be thought of as bounds imposed on other metrics such as the number of hops for instance.

In contrast to single-objective optimization, there is no single global solution to multiobjective optimization problems. It is often necessary to determine a set of points that all fit a predetermined definition of an optimum. This is called Pareto optimality [13]. A point $\mathbf{x}$ (or path) is Pareto optimal if there is no other point that improves at least one objective function without detriment to another function. A point is weakly Pareto optimal if there is no anther point that improves all of the objective functions simultaneously. A Pareto optimal point is also weakly Pareto optimal, but the converse is not true.

3.2. The Weighted Sum Method. This is the most common approach to multiobjective optimization:

$$U = \sum_{i=1}^{k} w_i F_i(\mathbf{x}). \tag{5}$$

If all the weights are positive, the minimum of the above equation is Pareto optimal [14]. The weights are set such that

$$\sum_{i=1}^{k} w_i = 1. \tag{6}$$

The relative values of the weights reflect the relative importance or prioritization of the objectives. An a priori selection of the weights does not necessarily ensure that the final solution will be acceptable. The problem may have to be resolved with new weights.

3.3. The Proposed DWSM Algorithm. A routing metric is a function that assigns a cost to any given path and naturally to its individual links. Various combined routing metrics have been introduced in the literature but are tailored to specific application areas such as routing in a multiradio multihop WMNs [15] and routing for cognitive radio ad hoc networks [16]. These minimize delay and maximize probability of data delivery in nonstatic WMNs. They are not concerned with maximizing capacity or throughput for a given end-to-end path in a static WMN. Our proposed routing metric that characterizes a link in the network is simply of the form of a weighted sum of the link delay and the inverse of the link capacity. The performance of a metric designed such that the minimization of which minimizes delay and maximizes path throughput at the same time is not often evaluated or studied in the literature owing to the fact that throughput inverse and delay usually have different orders of magnitude [17]. In this work, however, we show that this optimization is possible in many cases and at least prove that it is superior to the simple pruning method of multiobjective optimization. The proposed metric is given by

$$\begin{aligned} U = {} & \beta\,(\text{link delay}) \\ & + (1-\beta)\ (\text{inverse of the link capacity}). \end{aligned} \tag{7}$$

The weight factor β is a tunable parameter subject to $0 \le \beta \le 1$ in accordance with (6).

The above weighted average can be viewed as an attempt to balance delay and capacity providing a tradeoff between the two. When combined with Dijkstra's algorithm, a shortest path is found that minimizes the metric along that path. The metric is additive along a path. If $\beta = 0.5$, equal weight is given to capacity and delay. Setting $\beta = 0$ selects links based solely upon capacity, whereas setting $\beta = 1$ selects links based on delay only; that is, only delay is optimized. Thus, the metric selects paths with less optimized delay (greater delay) and higher capacity (smaller second term) when β is low and vice versa. The total network capacity is maximized by using lower values of β at the expense of increased end-to-end delay. However, delay is still optimized though in a less pronounced manner than capacity and the selection is better than the single-metric Dijkstra's routing algorithm with regard to delay when the single-metric algorithm is optimized for capacity.

In the section on results, our simple routing metric of (7) will be experimented with illustrating the impact of varying β on the routing algorithm performance. It is the simplest improvement over the pruning method of multiobjective optimization and provides large diversity in yielding better throughputs and less end-to-end delays in contrast to the restricted nature of the pruning method of optimization. The resulting algorithm also retains the advantages of Dijkstra's technique of fast convergence and relatively small computational complexity, though it still does not provide the conceptual global optimum and only achieves Pareto optimality.

If we assume $\beta = 0$, the path metric is the addition of the capacity inverses of the path links. These are each designated by $1/Ai$ below. When the Dijkstra shortest path is found, this sum will have been minimized, and its inverse (designated by A below) is maximized:

$$A = \frac{1}{(1/A_1) + (1/A_2) + \cdots + (1/A_n)}. \tag{8}$$

Since $A <$ any Ai, $i = 1, 2, \ldots, n$, the maximum A will be less than the minimum link capacity. So the maximum

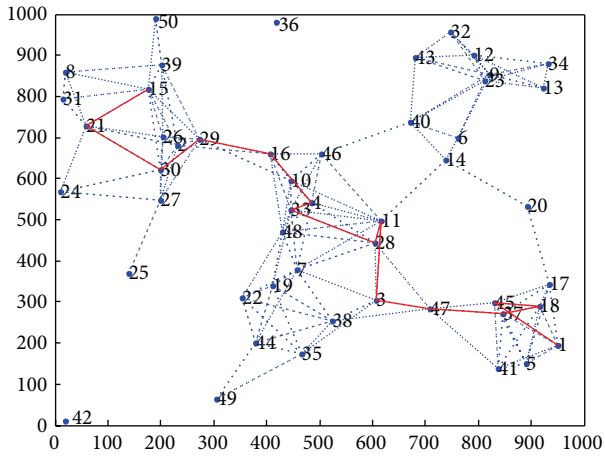

Figure 1: Network topology showing shortest route under single-objective Dijkstra's algorithm (capacity optimization). Path capacity = 5.7 Mbps, delay = 86.8771 ms.

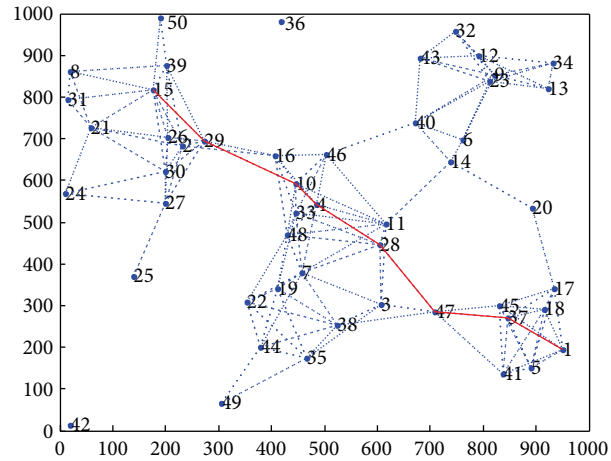

Figure 2: Network topology showing shortest route under single-objective Dijkstra's algorithm (delay optimization). Path capacity = 1.4 Mbps, delay = 51.7216 ms.

A is indicative of the optimum path capacity. The latter is maximized in Algorithm 1 by maximizing the minimum link capacity in the path. Rather, in our proposed algorithm, a quantity (A) is maximized that is less than the minimum link capacity. So it can readily be inferred that our chosen metric will almost also lead to maximization of the minimum link capacity which is the permissible path capacity. Thus, our metric is practically and conceptually feasible regarding path capacity as well as path delay. Minimization of path delay naturally occurs in a straightforward manner when using (7) as a routing metric.

4. Results

We model our WMN characterizing each link by link rate or capacity and link delay. In all experiments, the simulation program is coded in MATLAB 8.1. The experimental setup of this simulation analysis is based on the generation of 50 nodes randomly distributed on an area of $1000*1000\,m^2$. The random distribution is fixed, first, for the purpose of comparison among the algorithms and simulation scenarios. The network topology ensures that the node coverage area is 200 m. Thus some nodes are in the coverage area of others. The assigned link delay (in ms) and link capacity (in Mbps) follow uniform distributions. This characterization ensures link diversity in the network and does not affect the generality of the conclusions drawn.

For the chosen topology, the result of single-objective Dijkstra optimization described by Algorithm 1 of path capacity is shown in Figure 1, which illustrates the resulting shortest path between source node 1 and destination node 15. The optimized path capacity is 5.7 Mbps and the corresponding end-to-end delay is 86.8771 ms. Figure 2 shows the result of single-objective end-to-end delay optimization using the same algorithm but taking into account the difference in determining the overall metric along a path. The shortest path between nodes 1 and 15 is shown for which the optimized

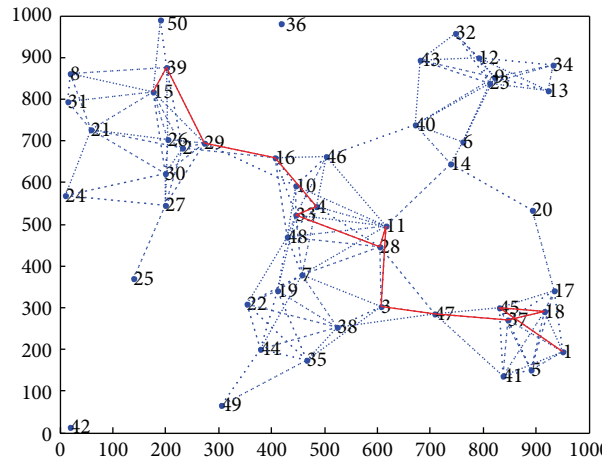

Figure 3: Network topology showing shortest route under Dijkstra's algorithm with pruning subject to delay bound = 80 ms. Optimized capacity = 5.8 Mbps, path delay = 78.2991 ms.

delay is 51.7216 ms and the corresponding path capacity is 1.4 Mbps.

The results for the pruning algorithm which involves optimization of capacity and bounding delay have been obtained for different values of delay bounds and for the same source and destination nodes used above. These are shown in Figures 3 and 4 with the values of path delay and optimized capacity and the delay bound indicated in the figures. As the delay bound decreases, the optimized capacity decreases.

The DWSM algorithm implemented with $\beta = 0$ and $\beta = 1$ obviously yields the same results as single-objective Dijkstra optimization for capacity and delay, respectively. Intermediate values of β are tried using the same source and destination nodes and the results are tabulated in Table 1 together with all the previous results for the purpose of comparison. We notice that the results obtained with DWSM are far better than those of pruning in terms of both delay

Table 1: End-to-end QoS measures for different optimization algorithms (topology as in Figures 1–5).

Optimization method	End-to-end delay (ms)	Path capacity (Mbps)	Number of hops
Single-objective optimization (capacity)	86.8771	5.8	14
Single-objective optimization (delay)	51.7216	1.4	7
Multiobjective optimization/pruning Delay bound = 80 ms	78.2991	5.8	13
Multiobjective optimization/pruning Delay bound = 75 ms	73.2357	2.6	12
DWSM $\beta = 0$	86.8771	5.8	14
DWSM $\beta = 1$	51.7216	1.4	7
DWSM $\beta = 0.1, 0.2$ path ⟨1 to 15⟩ (as in all previous entries)	53.1664	2.6	7
DWSM $\beta = 0.3$ and above path ⟨1 to 15⟩	51.7216	1.4	7
DWSM $\beta = 0$ path ⟨32 to 49⟩	75.6293	4.1	11
DWSM $\beta = 0.1$ path ⟨32 to 49⟩	53.8242	2.5	8
DWSM $\beta = 0.2$ and above path ⟨32 to 49⟩	53.3745	2.5	7

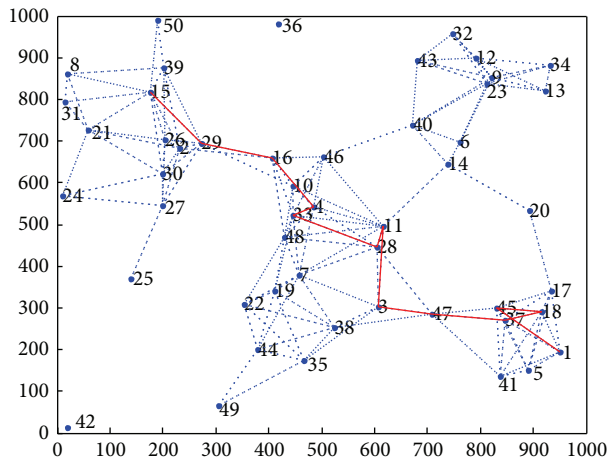

Figure 4: Network topology showing shortest route under Dijkstra's algorithm with pruning subject to delay bound = 75 ms. Optimized capacity = 2.6 Mbps, path delay = 73.2357 ms, and number of hops = 12.

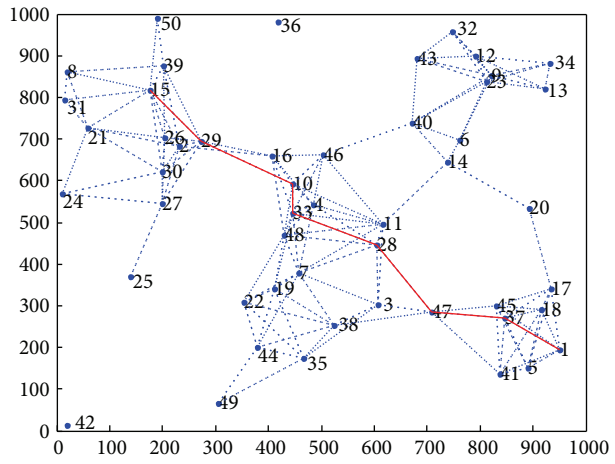

Figure 5: Network topology showing shortest route under DWSM algorithm with $\beta = 0.2$. Optimized capacity = 2.6 Mbps, path delay = 53.1664 ms, and number of hops = 7 (to be compared with Figure 4).

and capacity. For example, we obtain the same optimized path capacity as that with pruning under a 75 ms delay bound but with much shorter end-to-end delay. This case is portrayed in Figure 5 in comparison with Figure 4. In addition, we can even obtain better optimized shortest paths with fewer links (number of hops) with DWSM. This is also clear from Figure 5, as well as from Table 1. The minimization of the number of hops ensures reduced waste of network resources such as computational power. The last two rows in Table 1 show results for DWSM with the same previous values of β but with a different source-destination pair.

The values of delay and capacity indicated in the figures are typical of WMNs as may be found in [1, 18] to name but a few instances. Delay and capacity inverse values are not of the same order of magnitude, which is why we cannot vary β over

Table 2: End-to-end QoS measures for different optimization algorithms (same topology but delay modified).

Optimization method	End-to-end delay (ms)	Path capacity (Mbps)	Number of hops
Single-objective optimization (capacity)	17.3754	5.8	14
Single-objective optimization (delay)	10.3443	1.4	7
Multiobjective optimization/pruning Delay bound = 16 ms	15.6598	5.8	13
Multiobjective optimization/pruning Delay bound = 15 ms	14.6471	2.6	12
DWSM path ⟨1 to 15⟩ (as in all previous entries)			
$\beta = 0$	17.3754	5.8	14
$\beta = 0.2$	10.7347	2.6	7
$\beta = 0.5$	10.6333	2.6	7
$\beta = 0.6$	10.6333	2.6	7
$\beta = 0.7$	10.3443	1.4	7
$\beta = 1$	10.3443	1.4	7
DWSM path ⟨32 to 49⟩			
$\beta = 0$	12.3983	4.1	8
$\beta = 0.2$	11.0082	2.9	7
$\beta = 0.4$	10.7648	2.5	8
$\beta = 0.5$	10.6749	2.5	7
$\beta = 1$	10.6749	2.5	7

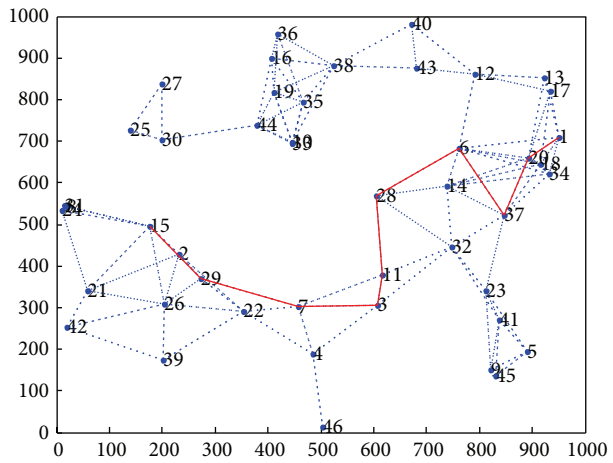

Figure 6: New network topology showing shortest route under Dijkstra's algorithm with pruning subject to delay bound = 70 ms. Optimized capacity = 2.3 Mbps, path delay = 68.2796 ms, and number of hops = 10.

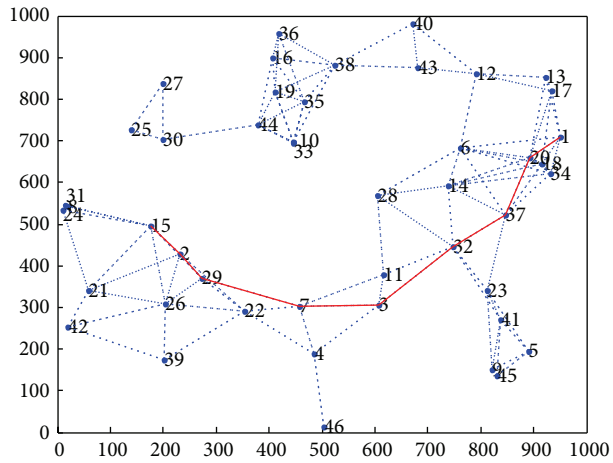

Figure 7: New network topology showing shortest route under DWSM algorithm with $\beta = 0.2$. Optimized capacity = 3.7 Mbps, path delay = 52.5050 ms, and number of hops = 8 (to be compared with Figure 6).

a wide range and yet get different results, even for different paths. The values of β that make a difference are between 0 and 0.3 only. Dividing all link delays of our topology by 5 still provides typical delay values but allows better DWSM performance. It allows us to vary β more freely to investigate more fully its impact on multiobjective optimization. All results are repeated for the delay-modified topology and are shown in Table 2. We find that the values of β that make a difference are between 0 and 0.7 for some paths; that is, the range of possible β variation for better multiobjective optimization has increased.

In analogy to the comparison between Figures 4 and 5, the multiobjective pruning and the DWSM algorithms may be compared by inspecting Figures 6 and 7 in which the two algorithms have been applied to a different topology for verification. This new topology has 46 nodes randomly distributed with a different sample realization and different uniformly distributed link capacities and delays. The nodes are also distributed on an area of $1000 * 1000\ m^2$. The merits of the DWSM algorithm are even more accentuated in this comparison. In Figure 7, with DWSM and $\beta = 0.2$, we obtain larger optimized path capacity, shorter delay, and fewer

links or number of hops than with the pruning algorithm manifested in Figure 6 under a delay bound of 70 ms.

Optimal weights are found experimentally with the aid of the tables. It is clear, by appropriately weighting the routing objectives in accordance with the network environment and to their importance with regard to performance, that the WMN routing performance improves significantly compared to single-objective and multiobjective pruning algorithms. The reason is that the mechanisms of the latter algorithms are almost unaware of the multiple performance objectives of interest.

5. Conclusions

This work dealt with combining desirable properties of path delay and capacity QoS measures by minimizing a weighted metrics sum via a Dijkstra-based algorithm (DWSM) to achieve multiobjective routing in static WMNs. It has been shown that the performance of the DWSM algorithm is far better than that of simple Dijkstra-based pruning in terms of both delay and capacity, in addition to minimizing the number of hops.

For the most typically common values of delay and capacity, the variation of the weighting factor β and its impact on DWSM performance are investigated, and it is found that varying β over a considerably wide range can provide diverse optimization results of both delay and capacity simultaneously. These results counterbalance remarks in the literature on the limited applicability of weighted sum minimization for multiobjective optimization routing due to capacity inverse and delay having different orders of magnitude.

Finally, it is noteworthy that although the DWSM algorithm is readily realizable in WMN environments and has the advantage of computational simplicity and consequently fast convergence, it cannot provide the flexibility and quality of solutions obtained with more powerful but computationally expensive techniques such as GAs.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

References

[1] J. W. Tsai and T. Moors, "Interference-aware multipath selection for reliable routing in wireless mesh networks," in *Proceedings of the IEEE international Conference on Mobile Adhoc and Sensor Systems (MASS '07)*, pp. 1–6, Pisa, Italy, October 2007.

[2] G. Parissidis, M. Karaliopoulos, R. Baumann, T. Spyropoulos, and B. Plattner, "Routing metrics for wireless mesh networks," in *Guide to Wireless Mesh Networks*, pp. 199–230, Springer, London, UK, 2009.

[3] A. Lew and H. Mauch, *Dynamic Programming: A Computational Tool*, Springer, Berlin, Germany, 2007.

[4] S. Sitarz, "Dynamic programming with ordered structures: theory, examples and applications," *Fuzzy Sets and Systems*, vol. 161, no. 20, pp. 2623–2641, 2010.

[5] S. Dasgupta, C. Papadimitriou, and U. Vazirani, *Algorithms*, McGraw-Hill, 2008.

[6] W. Stallings, *Data and Computer Communications*, Prentice-Hall, 9th edition, 2011.

[7] S. Glisic and B. Lorenzo, *Advanced Wireless Networks*, John Wiley & Sons, 2nd edition, 2009.

[8] G. Allard and P. Jacquent, "Heuristic for bandwidth reservation in multihop wireless networks," INRIA Rocquencourt 5075, 2004.

[9] L. Liu and G. Feng, "Simulated annealing based multi-constrained QoS routing in mobile ad hoc networks," *Wireless Personal Communications*, vol. 41, no. 3, pp. 393–405, 2007.

[10] K. Manousakis and A. J. McAuley, "Using stochastic approximation to design OSPF routing areas that satisfy multiple and diverse end-to-end performance requirements," in *Proceedings of the 6th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (Wiopt '08)*, pp. 394–402, Berlin, Germany, April 2008.

[11] L. Barolli, A. Koyana, and N. Shiratori, "A QoS routing method for ad-hoc networks based on genetic algorithm," in *Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA '03)*, pp. 175–179, Prague, Czech Republic, September 2003.

[12] T. K. Moon and W. C. Sterling, *Mathematical Methods and Algorithms for Signal Processing*, Prentice-Hall, Upper Saddle River, NJ, USA, 2000.

[13] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[14] L. A. Zadeh, "Optimality and non-scalar-valued performance criteria," *IEEE Transactions on Automatic Control*, vol. 8, no. 1, pp. 59–60, 1963.

[15] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (MobiCom '04)*, pp. 114–128, October 2004.

[16] M. Caleffi, I. F. Akyildiz, and L. Paura, "OPERA: optimized routing metric for cognitive radio ad hoc networks," *IEEE Transactions on Wireless Communications*, vol. 11, no. 8, 2012.

[17] J. Crichigno, J. Khoury, M. Y. Wu, and W. Shu, "A dynamic programming approach for routing in wireless mesh networks," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '08)*, pp. 1–5, December 2008.

[18] T.-S. Kim, G. Jakllari, S. V. Krishnamurthy, and M. Faloutsos, "A unified metric for routing and rate adaptation in multi-rate wireless mesh networks," in *Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS '11)*, pp. 242–251, October 2011.