Non-Dominance in QoS Routing: an Implementational Perspective

F.A. Kuipers and P. Van Mieghem

***Abstract*— In QoS routing, the problem of finding paths subject to multiple constraints, is NP-complete. Therefore, efficient search-space reducing techniques are needed. The concept of dominance is such a technique. Contrary to the popularity of using dominance verification, the dominance implementation issues are hardly studied. This letter provides such a study.**

I. Introduction

Finding paths that can guarantee multiple constraints (QoS routing) is NP-complete. This is reflected in the search-space, which may grow exponentially large. To reduce the size of the search-space, the technique of non-dominance could be used. In this letter we examine three possible ways to implement the non-dominance concept. First we explain the notation that is used.

A network is represented as a graph $G = (V, E)$ consisting of a set V of $N = |V|$ nodes and a set E of $M = |E|$ links. Each link $(u, v) \in E$ from node u to node v is characterized by an m-dimensional link weight vector $\vec{w}(u, v)$, where $w_i > 0$. The m components refer to QoS measures such as delay. $\vec{L}$ denotes the constraints vector. The possible QoS measures belong to two different classes: additive (e.g., delay) and min-max (e.g., bandwidth) QoS measures. Without loss of generality [10], all QoS measures are assumed to be additive.

The rest of this letter is organized as follows: Section II discusses non-dominance and related work. The complexity is investigated in Section III. Section IV discusses a possible improvement for verifying non-dominance, while Section V evaluates this gain via simulations. Section VI ends with the conclusions.

II. Definition of Non-Dominance

The concept of non-dominance is widely used in the field of multi-objective optimization, where it is often referred to as Pareto-optimality.

Definition 1: A path P is called *non-dominated* if there does not exist a path P' for which $w_i(P') \leq w_i(P)$ for all link weight components i except for at least one j for which $w_j(P') < w_j(P)$.

Theorem 1: If for all m QoS measures, there is no negative cycle in the graph G, then a walk containing a loop is always dominated by the same walk without the loop.

Proof: If no negative cycles appear in G, then traversing a cycle (loop) Q will never decrease any weights and therefore walk P will always dominate walk $P + Q$, since $w_i(P) \leq w_i(P + Q) = w_i(P) + w_i(Q)$, for all $i = 1, ..., m$. ■

The amount of papers that specifically target or use non-dominance for QoS routing is relatively small. Hansen [3] was among the first to study the bi-objective ($m = 2$) shortest path problem and demonstrated that the number of non-dominated paths can grow exponentially with the size of the network. Martins [6] generalized the algorithm of Hansen to handle $m \geq 2$ measures. Henig [4] described methods to obtain non-dominated and extreme non-dominated paths and compared the expected complexity of these methods. Among the large set of QoS routing algorithms, the SAMCRA [11] algorithm is one of the few algorithms that employs the concept of non-dominance.

A drawback of checking for dominance is the involved complexity, which could be quadratic in the number of non-dominated vectors. Little attention has been paid to improve this complexity. For instance, checking if a new vector is dominated could be done more efficiently with a proper data structure [8] or through clever sorting [1].

III. Complexity of dominance

The non-dominance property is very strong when the number of QoS measures m is small or when the QoS measures are positively correlated. There exists a straightforward way to check whether a new path is dominated by a set of other paths, which is to check whether the new path is dominated by the first path and continuing down the list of paths until the new path is either dominated or all paths are examined (and hence the new path is non-dominated). The worst-case complexity of checking for dominance in this straightforward way equals $O(k_{ND}^2 m)$, where k_{ND} refers to the number of non-dominated paths stored at a node.

If we would not check for dominance and only check for loops (Theorem 1), then for each path at most $N - 1$ hops should be traced. This approach has a complexity of $O(k_{NO} N)$, where k_{NO} refers to the number of loop-free paths stored at a node, because for each examined path at most $N-1$ hops need to be checked. In this case dominated paths (that are loop-free) may also be stored and hence $k_{NO} \geq k_{ND}$. To understand the complexity of checking for dominance, it is therefore essential to know k_{ND}. Simulations (not displayed), for $m = 2$ in the class of square lattices, indicate that the time to check for dominance is much smaller than only to check for loops. The maximum number of loop-free paths k_{NO} between two nodes in any graph is upper bounded by $\lfloor e(N-2)! \rfloor$. This upper bound is precisely attained in the complete graph [9].

The worst-case amount of non-dominated paths k_{ND} is determined by the granularity of the constraints. In that case the constraints L_i can be expressed as an integer number of a basic unit.

Theorem 2: If all weight components have a finite granularity, the number of non-dominated paths k_{ND} within the constraints cannot exceed $\frac{\prod_{i=1}^{m} L_i}{\max_{1\leq i\leq m} L_i}$.

Proof: See [10]. ■

The worst-case number of partial paths is

$$k_{ND} \leq \min\left[\frac{\prod_{i=1}^{m} L_i}{\max_{1\leq i\leq m} L_i}, \lfloor e(N-2)!\rfloor\right] \qquad (1)$$

The first argument of the min-operator applies only to k_{ND} in the case of a finite granularity. The second argument of the min-operator denotes the maximum number of loop-free paths $k_{NO} \geq k_{ND}$ that exists between two nodes in any graph (see [9]) and applies in case the granularity is infinitely small or, equivalently, for real values of w_i.

IV. Efficient dominance checking

In the previous section we have discussed the straightforward way of checking for non-dominance, which we further refer to as "normal dominance." We have also mentioned the use of only loop-detection, which we omit from further discussion since we did not observe an improvement in running time. In this section we will apply sorting to see if we can improve the complexity of checking for dominance. The idea for sorting was previously applied by Climaco and Martins [1], but the description of their algorithm is not precise in explaining how to sort. Our goal is to provide a more detailed analysis of the efficiency gain via sorted dominance verification. The concept of sorting has mainly potential for the case $m = 2$, to which we confine.

It is easier to check for dominance among a sorted set of paths. We will discuss two length functions: the "semi-linear" length

$$l(P) = \begin{cases} w_1(P), \text{ if } w_i(P) \leq L_i,\ i = 1, ..., m \\ \infty, \text{ else} \end{cases}$$

where $w_i(P) = \sum_{(u,v)\in P} w_i(u,v)$ and the non-linear length as used by SAMCRA [11]

$$l(P) = \max_{i=1,...,m}\left(\frac{w_i(P)}{L_i}\right)$$

In case of the "semi-linear" length, the k-th shortest path P_k from source node s to node u will have weight $w_1(P_k) \geq w_1(P_{k-1})$. P_k is non-dominated if $w_2(P_k) < w_2(P_{k-1})$. Given $k-1$ shortest non-dominated paths with $w_1(P_j) \geq w_1(P_{j-1})$, then $w_2(P_j) < w_2(P_{j-1})\ \forall j \leq k-1$. If $w_2(P_k) \geq w_2(P_{k-1})$ then path P_k is dominated by path P_{k-1}, else P_k is non-dominated by the $k-1$ paths. In conclusion, to check whether P_k is dominated, we only need to verify whether $w_2(P_k) < w_2(P_{k-1})$.

In case of the non-linear length, the paths P_j^* are ordered in non-decreasing length such that for all $j \leq k$

$$\max\left(\frac{w_1(P_j^*)}{L_1}, \frac{w_2(P_j^*)}{L_2}\right) \geq \max\left(\frac{w_1(P_{j-1}^*)}{L_1}, \frac{w_2(P_{j-1}^*)}{L_2}\right)$$

where P_j^* is the j-th shortest path according to the non-linear length, which is not necessarily equal to P_j given by the semi-linear length. If $\frac{w_1(P_k^*)}{L_1}$ $\left(\frac{w_2(P_k^*)}{L_2}\right)$ is the maximum, we only need to compare $w_2(P_k^*)$ with $w_2(P_{k-1}^*)$ $\big(w_1(P_k^*)$ with $w_1(P_{k-1}^*)\big)$ to check for dominance. This requires a complexity of $O(1)$.

The complexity of sorting depends on the way of implementation. Sorting an array of k components requires a complexity of $O(k \log_2 k)$. Hence sorting the paths each time a new non-dominated path has been added requires $O(k_{ND}^2 \log_2 k_{ND})$ in total, which is worse than the $O(k_{ND}^2)$ without sorting. Another way to obtain a sorted list is to directly insert new paths in the correct place. Finding (and inserting) the correct place within k components requires $O(\log k)$, when using sorted heaps. In that case, checking for dominance is possible in $O(k \log k)$. We use red-black trees [2] and sort them based on the "semi-linear" length function. We call this "red-black dominance." It may happen that a new path dominates one or more already stored paths. However, because the new path is only verified against one previously stored path, other possibly dominated paths are not recognized. "Normal dominance" checks all stored paths and hence $k_{RB} \geq k_{ND}$.

An alternative approach emerges when we are searching in a best-first manner (like Dijkstra or SAMCRA [11]); then the length of paths extracted from the queue are non-decreasing (or non-increasing if maximization is strived for). These extracted paths are already sorted by length. We call this "efficient dominance." By using the ordering acquired from best-first search, it is possible to verify for dominance in $O(1)$ time for each of the k_{ED} paths, leading to a total complexity of $O(k_{ED})$. We have that $k_{ED} \geq k_{RB} \geq k_{ND}$.

V. Evaluation of dominance verification

We have simulated with the three approaches red-black (RB), efficient (ED) and normal (ND) dominance. All simulations were performed in the class of two-dimensional lattices, with source and destination chosen in opposite corners. This class along with negatively correlated link weights provides a worst-case scenario [5]. We have simulated with a correlation $\rho = -1$ and $\rho = 0$, where the link weights were uniformly distributed in the range $(0, 1]$ and with two types of constraints "loose" and "strict." In the case of strict constraints only one feasible path is available. For loose constraints $L_i = N$, for $i = 1, .., m$. For each simulation run 100 graphs were generated, after which a multi-constrained path between the source and destination was computed by three SAMCRA-based algorithms. For "red-black dominance" SAMCRA was equipped with red-black trees at each node to check for dominance. In "efficient dominance" the ordering in (the non-linear) length of the extracted paths in SAMCRA was used to check for dominance. The algorithm for "normal dominance" is the SAMCRA algorithm as proposed in [11]. Note that all algorithms used the same (non-linear) length function, use Fibonacci heaps [2] to structure the queues and were equipped with a second search-space reducing technique called look-ahead [11].

During the simulations the maximum number of paths k stored at a node and the execution time were maintained. The results are illustrated in Figure 1, with ratios $r[k_j] = \frac{E[k_j]}{E[k_{ND}]}$ and $r[Time[j]] = \frac{Time[j]}{Time[ND]}$, for $j = RB, ED$.

For $\rho = -1$, the expected queue-size of "normal dominance" is smallest, but as N grows, "red-black dominance" and "efficient dominance" approximate this value and hence the ratios approach 1. The time-ratios for "red-black dominance" and "efficient dominance" on the other hand become smaller than 1, illustrating that sorting indeed results in a gain in time-complexity even though there is a slight loss in space-complexity. Unfortunately, the gain in complexity is relatively small. The increase in space-complexity produces a $O(kN\log(kN))$ increase in the Fibonacci heap, which counteracts the gain in dominance verification. Thus, it seems that there is a one-to-one trade-off between space- and time-complexity.

For $\rho = 0$, there are more dominated paths and therefore "normal dominance" does not need to check its entire list every time. In this case "normal dominance" performs best. The results for strict and loose constraints are similar in behavior.

VI. Conclusions

QoS routing is an NP-complete problem and therefore efficient search-space reducing techniques are needed. The non-dominance concept is such a technique that is much used in the field of multi-objective optimization, but its strength for QoS routing is still not fully studied. With this letter we have provided a coverage of the non-dominance concept for QoS routing and we have presented efficient ways of checking for dominance when there are only two ($m = 2$) QoS measures. The efficiency gain has been evaluated via simulations. There is a one-to-one trade-off between space- and time-complexity, making all implementations about equally powerful. However, only "normal dominance" is simple and valid for all m, which made it the preferred choice for SAMCRA.

References

[1] J. Climaco and E. Martins, "A bicriterion shortest path algorithm," European Journal of Operational Research, 11:399-404, 1982.

[2] T. H. Cormen, C. E. Leiserson and R. L. Rivest, *An Introduction to Algorithms*, MIT Press, Boston, 1991.

[3] P. Hansen, "Bicriterion path problems," in G. Fandel and T. Gal (eds.), *Multiple criteria decision making: theory and applications*, lecture notes in economics and mathematical systems 177, pp. 109-127, Springer, Heidelberg, 1980.

[4] M.I. Henig, "The shortest path problem with two objective functions," European J. of Operational Research, 1985, vol. 25, pp. 281-291.

[5] F. A. Kuipers and P. Van Mieghem, "The Impact of Correlated Link Weights on QoS Routing," Proc. of IEEE INFOCOM, 2003.

[6] E.Q.V. Martins, "On a multicriteria shortest path problem," European J. of Operational Research, 16, pp. 236-245, 1984.

[7] H. L. Royden, *Real Analysis*, Macmillan Publishing Company, New York, third edition, 1988.

[8] M. Sun and R. Steuer, "Quad trees and linear lists for identifying nondominated criterion vectors," INFORMS Journal on Computing, 8(4):367-375, 1996.

[9] P. Van Mieghem, "Paths in the simple Random Graph and the Waxman Graph," Probability in the Engineering and Informational Sciences (PEIS), vol. 15, pp. 535-555, 2001.

[10] P. Van Mieghem and F. A. Kuipers, "On the Complexity of QoS Routing," Computer Communications, vol. 26, No. 4, pp. 376-387, March 2003.

[11] P. Van Mieghem and F.A. Kuipers, "Concepts of Exact Quality of Service Algorithms," to appear in IEEE/ACM Transaction on Networking.

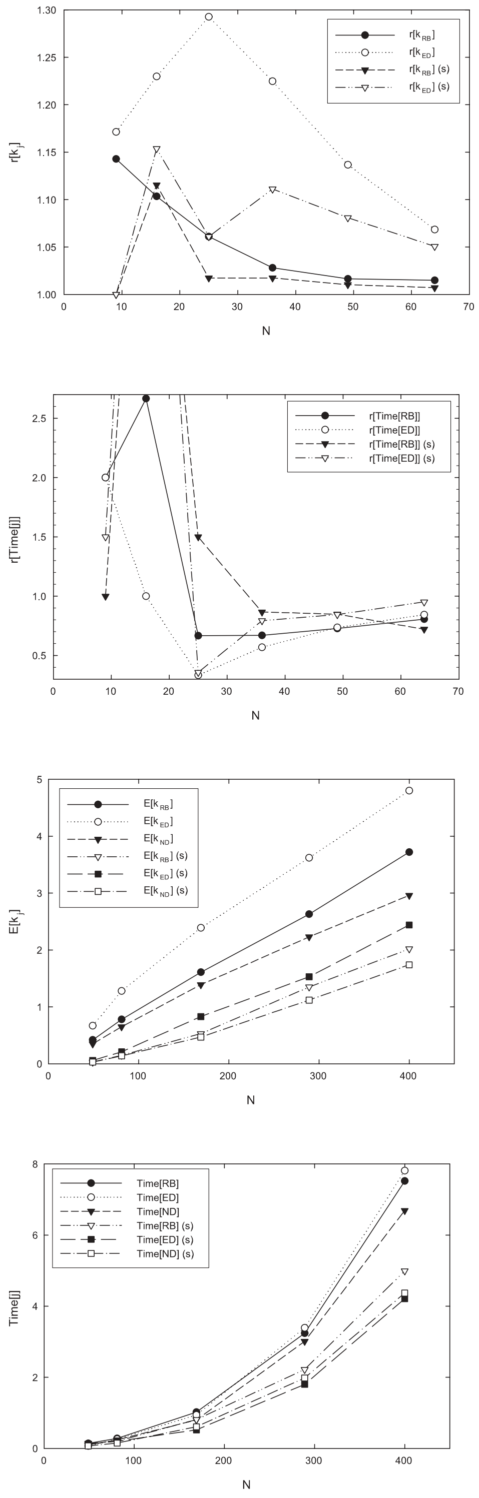

Fig. 1. The results (s denotes strict constraints) for the class of lattices as a function of the number of nodes N ($m = 2$). The ratio of k (1st) and the ratio of time (2nd) for $\rho = -1$. The expected queuesize $E[k]$ (3rd) and the expected time $Time[j]$ in seconds (4th) for $\rho = 0$.