# Performance Evaluation of Delay-Constrained Least-Cost QoS Routing Algorithms Based on Linear and Nonlinear Lagrange Relaxation

Gang Feng, Christos Douligeris†, Kia Makki, Niki Pissinou
Telecommunications and Information Technology Institute, Florida International University, Miami, FL 33174.
†Department of Informatics, University of Piraeus, Piraeus, 18534, Greece.

***Abstract*—The development of efficient quality of service (QoS) routing algorithms in a high-speed network environment is a very important and at the same time very difficult task due to the need to provide divergent services with multiple QoS requirements. Recently heuristic algorithms based on Lagrange relaxation techniques have been proposed to resolve the contradiction between the time complexity and the quality of solution. In this paper, we investigate the performance of two heuristic algorithms, LR_DCLC and NR_DCLC, for the delay-constrained least-cost (DCLC) routing problem. Algorithm LR_DCLC is based on linear relaxation, while algorithm NR_DCLC, which is proposed in this paper, is based on nonlinear relaxation. A large number of simulations demonstrate that even though both algorithms have very good performance, NR_DCLC can obtain much better solutions than LR_DCLC by running Dijkstra's algorithm on average a few more times, especially in the case when the optimal solutions are hard to find.**

## I. INTRODUCTION

The need for high-speed networks to support a wide range of services (video, audio, interactive multimedia, teleconferencing, etc.) with divergent and stringent service requirements poses a particular difficulty for the quality of service (QoS) routing problem [2], [14], [15]. On one hand, a path selection algorithm for the QoS routing must be very fast so that the corresponding routing protocol can achieve a very low set-up time, while on the other hand it should also provide solutions of high quality to ensure the best utilization of network resources [12]. In recent years great progress has been made for providing efficient algorithms, thanks to the incessant efforts of several researchers.

QoS requirements are generally represented by constraints imposed upon the corresponding performance metrics such as delay, jitter, cost, etc. In [2], Chen *et al.* classified these QoS constraints into *link constraint*, which specifies a bound on the minimal (or maximal) value of a metric, and *path constraint*, which specifies a bound on the summation of a metric. Based on these concepts, the QoS routing problems are divided into several categories. Among the unicast routing problems, the *path-constrained path-optimization* (PCPO) problem and the *multi-path constrained* (MCP) problem have been proved to be NP-complete [7].

A PCPO problem needs to find a path with the minimal cost subject to *one* or more path constraints, while a MCP problem needs to find a path subject to *two* or more path constraints without necessarily finding an "optimal" solution. Regarding the two classes of problems, much work has been done when the number of constraints is small. For instance, for the MCP problem with two constraints, i.e., the delay-cost-constrained (DCC) problem [1], [5], previous works include the two heuristic algorithms proposed by Jaffe [9], and the "extended Dijkstra's shortest path" (EDSP) and the "extended Bellman-Ford" (EBF) algorithms proposed by Chen *et al.* [1]. It was proved that EDSP and EBF are much more efficient than Jaffe's algorithms either in time complexity or in quality of solution. Meanwhile, the PCPO problem with one constraint, which is also called the delay-constrained least-cost (DCLC) path problem [14] or restricted shortest path problem [8], have also attracted much attention. The earliest work probably should be accredited to Hassin's two $\epsilon$-optimal approximation algorithms [8] which can produce solutions with their costs less than $1+\epsilon$ times the cost of the optimal solution.[1] These two algorithms, albeit very efficient in finding feasible solutions, have very high time complexities [11]. In 1994, Widyono [17] proposed a constrained Bellman-Forman (CBF) algorithm that can solve the DCLC problem exactly. Since the DCLC problem is NP-complete, it is not surprising that in the worst case the time complexity of CBF grows exponentially with the network size.

Despite the good performance of some of the prior works, in the past two or three years a few authors [4], [10], [11], [13] found that the heuristic algorithms based on Lagrange relaxation techniques are much more attractive. The basic idea is to first construct an aggregate weight, and then use Dijkstra's algorithm [3] to find the corresponding shortest path. It has been proved through extensive simulations that these algorithms not only have very low time complexities, but also can achieve very high probabilities of finding feasible or optimal solutions. For example, Jüttner *et al.* [10] and Feng *et al.* [4] proposed similar iterative algorithms for the DCLC problem, which are based on linear Lagrange relaxation. Large numbers of experiments indicate that even for 200-node networks these algorithms can obtain the optimal solution with a very high probability by running Dijkstra's algorithm on average three or four times [4]. Feng *et al.* further extended the idea behind these algorithms to solve the DCC problem [5], showing that in this case it outperforms most existing algorithms as well.

In the mean time, Korkmaz *et al.* proposed a heuristic H_MCOP which only needs to run Dijkstra's algorithms (with slight modifications) twice to solve the PCPO problem with multiple constraints based on nonlinear Lagrange relaxation. Simulations indicated that H_MCOP can outperform almost all

This work was partially supported by NSF under grants No. 0196557 and No. 0123950, and by University of Piraeus Research Center.

[1] For most PCPO problems there exist multiple optimal solutions, thus the expression "the optimal" in this paper does not necessarily mean that there is only one optimal solution.

0-7803-7400-2/02/$17.00 © 2002 IEEE

known heuristic algorithms such as those proposed by Jaffe, Hassin and Chen *et al.* in terms of the success ratio of finding feasible solutions. Furthermore, our recent work [6] demonstrated that the success ratio of H_MCOP is actually very close to that of an exact algorithm. Nevertheless, we also found that in certain cases the solution of H_MCOP has a much higher cost than the optimal solution. In addition, it is inappropriate to solve the DCLC problem since the construction of an aggregate weight does not make sense if there is only one constraint.

In this paper, we propose a modified algorithm based on H_MCOP to solve the DCLC problem. The basic idea is to first find an initial feasible solution, and then convert the DCLC problem to a DCC problem, which is solved by a heuristic algorithm derived from H_MCOP. The proposed algorithm achieves a very high success ratio of finding the optimal solution, as will be verified through numerical experiments. Even though the basic idea in this paper is very similar to the one in our recent work [6] for solving the PCPO problem subject to multiple constraints, a number of issues need to be discussed at great length due to the particularity of the DCLC problem. Moreover, the performance comparison between the up-to-date heuristic algorithms based on linear and nonlinear relaxation techniques might also be of interest to many researchers and the professionals in the area.

The remainder of the paper is organized as follows. The DCLC problem is first formally defined in Section II. In Section III, the algorithm based on linear Lagrange relaxation is described. The modified algorithm based on H_MCOP for the DCLC problem is described in Section IV. In Section V, we analyze the performance of the two heuristic algorithms by comparing their solutions with the optimal solution. Section VI concludes the paper.

## II. Notations and Definitions

A *network* is represented by a directed graph $G(V, E)$, where $V$ is the set of nodes, and $E$ is the set of links. Associated with each link $e$ there are two non-negative weights $w_1(e)$ and $w_2(e)$, which are called *delay* and *cost*, respectively. The upper bound of the delay constraint is denoted by $C_1$.

A path is a sequence of non-repeated nodes $\mathbf{p} = (v_1, v_2, \cdots, v_k)$ such that for a given $1 \leq i < k$ there exists a link from $v_i$ to $v_{i+1}$. The notation $e \in \mathbf{p}$ means that path $\mathbf{p}$ passes through link $e$. The $w$-weight of path $\mathbf{p}$ is given by

$$w(\mathbf{p}) = \sum_{e \in \mathbf{p}} w(e).$$

Apparently, the $w_1$-weight and $w_2$-weight of a path are its delay and cost, respectively.

By means of the above notations, the DCLC problem can be defined as follows.

*Definition 1:* Given a routing request between a source $s$ and a destination $t$, the DCLC problem is to find a path $\mathbf{p}$ between $s$ and $t$ such that

(i) $w_1(\mathbf{p}) \leq C_1$,

(ii) $w_2(\mathbf{p}) \leq w_2(\mathbf{q})$ for any path $\mathbf{q}$ that satisfies (i).

A path satisfying (i) is called a *feasible path* (or *feasible solution*), and otherwise an *infeasible path* (or *infeasible solution*). If a path satisfies both (i) and (ii), it is called an optimal solution.

If part (ii) in the above definition is replaced by $w_2(\mathbf{p}) \leq C_2$, where $C_2$ is the upper bound of the cost constraint, then we get the definition of the DCC problem. A solution to the DCC problem is either feasible or infeasible.

In order to describe the proposed algorithm, we define the notation Dijk$(w)$ to be the shortest path w.r.t. $w$ between $s$ and $t$ found by Dijkstra's algorithm. Apparently, Dijk$(w_1)$ and Dijk$(w_2)$ are the least delay (LD) path and the least cost (LC) path, respectively.

## III. Heuristic algorithm LR_DCLC based on linear Lagrange Relaxation

In this Section, we describe the heuristic algorithm for the DCLC problem based on linear Lagrange relaxation proposed in [4], and [10], which is called LR_DCLC in this paper (LR stands for linear relaxation).

The basic idea of algorithm LR_DCLC is to first combine the delay and cost in terms of a parameter $\alpha$ to form an aggreagate weight $w = w_1 + \alpha w_2$ for each link, and then use Dijkstra's algorithm to find the shortest path w.r.t. $w$. As long as an appropriate parameter value is obtained, the resulting shortest path could be a feasible solution of high quality. The fundamental relationship between the parameter value and the cost and delay of the resulting shortest path can be illustrated by the following proposition.

*Proposition 1:* If $\mathbf{p} = \text{Dijk}(w_1 + \alpha w_2)$, $\mathbf{q} = \text{Dijk}(w_1 + \beta w_2)$, $\alpha$ and $\beta$ are non-negative with $\alpha \geq \beta$, then $w_1(\mathbf{p}) \geq w_1(\mathbf{q})$, $w_2(\mathbf{p}) \leq w_2(\mathbf{q})$.

*Proof:* See [4]. ■

From the above proposition we can see that the impact of the parameter on the delay and cost of the resulting shortest path can be simply stated as follows: The larger the parameter, the larger the delay, while the smaller the cost. This indicates that as long as the resulting shortest path does not violate the delay constraint, a larger parameter will definitely give rise to a better solution. In order to find the largest parameter with the corresponding shortest path not violating the delay constraint, the following proposition is provided as the theoretical basis of algorithm LR_DCLC.

*Proposition 2:* If $\mathbf{r} = \text{Dijk}(w_1 + \alpha w_2)$, $\mathbf{p} = \text{Dijk}(w_1 + \beta w_2)$, $\mathbf{q} = \text{Dijk}(w_1 + \gamma w_2)$, where $\beta < \gamma$, $w_2(\mathbf{p}) \neq w_2(\mathbf{q})$, $\alpha = \frac{w_1(\mathbf{q}) - w_1(\mathbf{p})}{w_2(\mathbf{p}) - w_2(\mathbf{q})}$, then $w_1(\mathbf{p}) \leq w_1(\mathbf{r}) \leq w_1(\mathbf{q})$, $w_2(\mathbf{p}) \geq w_2(\mathbf{r}) \geq w_2(\mathbf{q})$.

*Proof:* See [4]. ■

Proposition 2 indicates that with $\alpha = \frac{w_1(\mathbf{q}) - w_1(\mathbf{p})}{w_2(\mathbf{p}) - w_2(\mathbf{q})}$, the resulting shortest path must have a delay between the delays of paths $\mathbf{p}$ and $\mathbf{q}$, and a cost between the costs of the two paths. As shown in Fig. 1, the heuristic LR_DCLC is actually based on the above proposition. The LC path is first obtained, and if it is feasible then it must be an optimal solution. Otherwise the LD path is found, and if it violates the delay constraint then no feasible path can be found. If none of the above conditions is true, the algorithm enters an iterative procedure. In each iteration, either $\mathbf{p}$ is updated with a "better" feasible solution $\mathbf{r}$ in the sense that $\mathbf{r}$ has a lower cost, or $\mathbf{q}$ is updated with a better

```
LR_DCLC (G, s, t, w1, w2, C1)
1    q ← Dijk(w2)
2    if (w1(q) ≤ C1) then
3        return q
4    p ← Dijk(w1)
5    if (w1(p) > C1) then
6        return NULL
7    if (w2(p) ≠ w2(q)) then
8        continue = TRUE
9        while continue do
10           α ← (w1(q) − w1(p)) / (w2(p) − w2(q))
11           r ← Dijk(w1 + αw2)
12           if (w2(r) = w2(q) or w2(r) = w2(p)) then
13               continue = FALSE
14           else if (w1(r) ≤ C1) then
15                   p ← r
16               else
17                   q ← r
18   return p
```

Fig. 1. The heuristic algorithm LR_DCLC

infeasible solution $\mathbf{r}$ in the sense that $\mathbf{r}$ has a lower delay. The algorithm terminates if no better solution can be found.

By comparing LR_DCLC with the algorithm LARAC proposed in [10], one may notice two overlooks in LARAC. First, line 7 of LR_DCLC was not taken into account in LARAC. However, this line is necessary since even if the LC path is infeasible and the LD path is feasible, the two paths might have the same cost because Dijkstra's algorithm breaks ties arbitrarily. Second, the condition $w_2(\mathbf{r}) = w_2(\mathbf{p})$ in line 12 of LR_DCLC was also not considered in LARAC. However, if this condition is true, the algorithm should also stop. Otherwise, if the newly obtained path $\mathbf{r}$ is path $\mathbf{p}$ of the previous iteration, the algorithm will fall into a deadlock.

LR_DCLC can find the optimal solution with a very high probability while the time complexity is satisfactorily low [4], [10]. In spite of this, we can possibly achieve better performance by using a nonlinear Lagrange relaxation technique, as discussed in the next Section.

## IV. Heuristic Algorithm NR_DCLC Based on Nonlinear Lagrange Relaxation

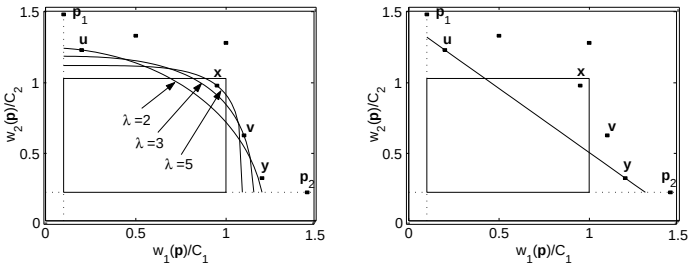
In this Section, we first briefly review the basic principle for the application of nonlinear Lagrange relaxation in QoS routing, and then discuss in detail how to use this technique to solve the DCLC problem.

### A. The nonlinear Lagrange relaxation

The basic principle of the nonlinear Lagrange relaxation technique can be illustrated by considering the solving of the DCC problem. Recall that for such a problem we need to find a path $\mathbf{p}$ such that its delay and cost do not exceed the upper bounds $C_1$ and $C_2$, respectively. Using the notations defined in Section II, we consider the following cost function:

$$g_\lambda(\mathbf{p}) = \left(\frac{w_1(\mathbf{p})}{C_1}\right)^\lambda + \left(\frac{w_2(\mathbf{p})}{C_2}\right)^\lambda \qquad (1)$$

where $\lambda \geq 1$. If there exists an algorithm that can find a path $\mathbf{p}$ that exactly minimizes the above cost function, then by setting




(a) nonlinear relaxation (b) linear relaxation

Fig. 2. Illustration of the linear and nonlinear Lagrange relaxation

$\lambda$ to a large value we must be able to find a feasible path if such a path exists. An example is shown in Fig. 2(a), where each small black square represents a path with its normalized delay and cost being the horizontal and vertical coordinates, respectively. $\mathbf{p}_1$ and $\mathbf{p}_2$ are the LD and the LC paths, respectively. Apparently, the rectangular area represents the feasibility region. Each curve denotes the set of paths that have the same value for the cost function $g_\lambda(\cdot)$ when $\lambda$ takes a specific value. In this example, when $\lambda = 2$ and $\lambda = 3$, the obtained shortest paths are $\mathbf{u}$ and $\mathbf{v}$, respectively, which are infeasible. However, by increasing $\lambda$ to 5 we can find the feasible path $\mathbf{x}$.

The major difficulty of this idea lies in that when $\lambda > 1$ there does not exist a polynomial-time algorithm that can find a path exactly minimizing cost function $g_\lambda(\cdot)$. This is because $g_\lambda(\cdot)$ is not an additive weight when $\lambda > 1$, i.e., $g_\lambda(\mathbf{p}) \neq \sum_{e\in\mathbf{p}} g_\lambda(e)$. In contrast, in case of linear relaxation where $w = w_1 + \alpha w_2$, the relation $w(\mathbf{p}) = \sum_{e\in\mathbf{p}} w(e)$ always holds, ensuring that Dijkstra's algorithm can be used to find the shortest path w.r.t. $w$.

For the above reason, the heuristic H_MCOP proposed in [11] runs Dijkstra's algorithm (with modifications) twice, one in reverse direction with linear relaxation and the other in forward direction with nonlinear relaxation. It has been proved through a large number of simulations that H_MCOP can achieve a very high success ratio of finding feasible paths in various situations even though the cost of its solution could be high. In the remainder of this Section, we describe a heuristic algorithm for the DCLC problem which can make use of the advantage of H_MCOP. Before doing that, we need to present the heuristic algorithm H_DCC for the DCC problem, which is a variant of the heuristic H_MCP described in our recent work [6].

### B. Heuristic algorithm H_DCC

Heuristic algorithm H_DCC is based on H_MCOP. Even though H_MCOP can be directly used to solve a DCC problem by considering only two constraints and skipping all codes regarding the cost, it does not achieve the best performance. Given a DCC problem, we are only concerned with whether there exists a feasible solution or not. Thus, if an algorithm for the DCC problem finds a feasible path, it may stop immediately with the path returned.

For this reason, we may make slight modifications on H_MCOP and thus describe a heuristic algorithm for the DCC problem. As shown in Fig. 3, algorithm H_DCC is very similar to H_MCOP when considering only two constraints

```
H_DCC(G(V,E), s, t, w_j, C_j, j = 1,2)
1    Reverse_Dijkstra (G(V,E), t, w_j, j = 1,2)
2    if (r[s] > 2) then
3        return NULL /* no feasible path available */
4    if (R_j[s] ≤ C_j, ∀j = 1,2) then
5        return the path found by Reverse_Dijkstra
6    Look_Ahead_Dijkstra(G(V,E), s, w_j, C_j, j = 1,2)
7    if (G_j[t] ≤ C_j, ∀j = 1,2) then
8        return the path found by Look_Ahead_Dijkstra
9    return NULL
```

Fig. 3. The heuristic algorithm H_DCC for the DCC problem

```
Prefer_the_best (a, b)
1    if (∀j = 1,2, G_j[a] + R_j[a] ≤ C_j) then return (a)
2    if (∀j = 1,2, G_j[b] + R_j[b] ≤ C_j) then return (b)
3    if (g[a] < g[b]) then return (a)
4    return (b)
```

Fig. 4. The preference rule used in H_DCC

(please see [11] for details of H_MCOP). The major difference lies in that if a feasible path is found after calling the Reverse_Dijkstra, H_DCC will stop and return this path (lines 4-5 in Fig. 3). Subroutine Reverse_Dijkstra including its relaxation procedure in H_DCC is the same as that in H_MCOP. The relaxation procedure Look_Ahead_Dijkstra_Relax for subroutine Look_Ahead_Dijkstra in H_DCC can be obtained from *Fig. 4* in [11] by neglecting all codes for updating the cost (lines 2 and 7 of *Fig. 4* in [11]). Correspondingly, procedure Prefer_the_best for H_DCC, as shown in Fig. 4, can be obtained from *Fig. 5* in [11] by eliminating the codes related to the cost. Note that only two constraints are considered in all of these subroutines.

Apparently, H_DCC can achieve the same success ratio of finding feasible solutions as H_MCOP does. However, unlike H_MCOP which always needs to run Dijkstra's algorithm twice, H_DCC could run only once if a feasible path is found by subroutine Reverse_Dijkstra.

### C. Heuristic Algorithm NR_DCLC

The pseudocode of heuristic NR_DCLC is shown in Fig. 5. As we see, the first few lines are the same as in LR_DCLC. In case that the LC path $\mathbf{q}$ is infeasible, while the LD path $\mathbf{p}$ is feasible, the DCLC problem is converted to a DCC problem by setting $w_2(\mathbf{p})-\epsilon$ to the upper bound $C_2$ of the cost constraint. $\epsilon$ is a small positive number such that no path between $s$ and $t$ has a cost in $[w_2(\mathbf{p})-\epsilon, w_2(\mathbf{p})]$. H_DCC is hence employed to solve this DCC problem. Obviously if a feasible solution $\mathbf{r}$ is found by H_DCC, $\mathbf{r}$ must have a lower cost than $\mathbf{p}$. To further search for better solutions, we can replace $\mathbf{p}$ by $\mathbf{r}$, update the upper bound $C_2$, and repeat this procedure until no feasible solution is found by H_DCC.

Since the heuristic algorithm NR_DCLC employs H_DCC to search for the best solution, it can possibly find a better solution than algorithm LR_DCLC. For instance, if we take the problem shown in Fig. 2(a) as a DCLC problem, then NR_DCLC can possibly find the best solution $\mathbf{x}$. In contrast, if algorithm LR_DCLC is used to solve this problem, the final solution must be $\mathbf{u}$. This is illustrated in Fig. 2(b). With linear relaxation where $w = w_1 + \alpha w_2$, the set of paths that have the same value for the aggregate weight $w$ must be on a straight line, rather than a curve as in the case of nonlinear relaxation. Using LR_DCLC

```
NR_DCLC(G(V,E), s, t, w_1, w_2, C_1)
1    q ← Dijk(w_2)
2    if (w_1(q) ≤ C_1) then
3        return q
4    p ← Dijk(w_1)
5    if (w_1(p) > C_1) then
6        return NULL
7    if (w_2(p) ≠ w_2(q)) then
8        set C_2 = w_2(p) − ε
9        repeat
10           r ← H_DCC(G(V,E), s, t, w_j, C_j, j = 1,2)
             /*solved as a DCC problem */
11           if (r ≠ NULL) then
12               p ← r
13               set C_2 = w_2(p) − ε
14       until r = NULL
15   return p
```

Fig. 5. The heuristic algorithm NR_DCLC for the DCLC problem

TABLE I
THREE SETS OF LINK-WEIGHT INTERVALS

| Set number | 1 | 2 | 3 |
|---|---|---|---|
| $w_1$ | $[1, 500]$ | $[1, 500]$ | $[1, 500]$ |
| $w_2$ | $[500, 1000]$ | $[1, 500]$ | $[1, 10000]$ |

to solve this problem, the aggregate weights of $\mathbf{u}$ and $\mathbf{y}$ must be equal at the final iteration, and no matter what parameter is used the best solution $\mathbf{x}$ can not have the smallest aggregate weight.

## V. PERFORMANCE EVALUATION

In this Section we investigate the performance of the two heuristic algorithms LR_DCLC and NR_DCLC through computer simulations. Waxman's method [16] is employed to generate three types of networks, i.e., 50-, 100-, and 200-node, each of which includes 10 instances. The link weights, i.e., delay and cost, are uniformly distributed on specific intervals, and three sets of distributions are investigated, as shown in Table I. For a specific set of link-weight intervals, 10 instances of link weights are generated. Given a network instance and a link-weight instance, 1000 routing requests are generated. To generate a routing request, a source and a destination are first randomly selected, then the LD path $\mathbf{p}_1$ and the LC path $\mathbf{p}_2$ are obtained, and finally the upper bound $C_1$ of the delay constraint is given by

$$C_1 = w_1(\mathbf{p}_1) + \Delta \cdot (w_1(\mathbf{p}_2) - w_1(\mathbf{p}_1))$$

where $\Delta$ is called *constraint factor*. Obviously, if $\Delta < 0$, then there is no feasible path. On the other hand, if $\Delta \geq 1$, then the LC path must be the optimal solution. In either case, the two heuristic algorithms can make a conclusion before entering the loop. For this reason, in the subsequent experiments we let $\Delta$ take values between 0 and 1 so that the optimal solutions are much more difficult to be found.

For each type of network with a specific value for the constraint factor, the following performance measures are computed based on 100,000 experimental results (10 network instances × 10 link-weight instances × 1000 routing requests):

- *Optimality*, which is the percentage of times that the optimal solutions have been found,

TABLE II

THE MAXIMUM NUMBER OF EXECUTIONS FOR LR_DCLC/NR_DCLC WITH LINK-WEIGHT SET 1

| $\Delta$ | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|---|
| 50-node | 7/10 | 7/12 | 7/12 | 7/12 | 7/16 |
| 100-node | 7/10 | 8/12 | 7/14 | 7/12 | 7/14 |
| 200-node | 8/12 | 8/14 | 8/14 | 8/14 | 7/14 |

TABLE III

THE MAXIMUM NUMBER OF EXECUTIONS FOR LR_DCLC/NR_DCLC WITH LINK-WEIGHT SET 2

| $\Delta$ | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|---|
| 50-node | 7/12 | 7/14 | 8/14 | 7/16 | 7/14 |
| 100-node | 8/12 | 8/14 | 8/14 | 8/14 | 8/16 |
| 200-node | 8/14 | 8/16 | 8/16 | 8/16 | 8/16 |

- *Average cost deviation* (AvgDeviation), which is the average deviation in percentage of the solution of a heuristic from the optimal solution,
- *Average number of the executions* of Dijkstra's algorithm,
- *Maximum number of the executions* of Dijkstra's algorithm.

Since Dijkstra's algorithm has a fixed time complexity, the third and fourth performance measures can be used to evaluate the time complexities of the heuristics. For simplicity, the executions of standard and modified Dijkstra's algorithm are counted together. Besides, 95% confidence intervals are also computed for the first three performance measures.

Fig. 6 shows the first three performance measures when the link weights are distributed on the first set of intervals in Table I, while the maximum number of executions in such case is shown in Table II. Corresponding to link-weight-interval sets 2 and 3, the performance measures are shown in Fig. 7 and Table III, and Fig. 8 and Table IV, respectively.

From Fig. 6(a), we may see that with the increase of $\Delta$ the optimality decreases. This is because the larger the value of $\Delta$, the more the feasible paths, and the harder to find the optimal solution. One may also notice that for a given heuristic the optimality decreases with the increase of the network size. On the other hand, the difference between the performance of the two heuristic algorithms becomes more and more conspicuous with the increase of $\Delta$. For example, for 200-node networks with $\Delta = 0.7$ and $\Delta = 0.9$, the differences between the optimality of the two algorithms are approximately 0.13 and 0.18, respectively. Opposed to the change of the optimality, the average cost deviation increases with $\Delta$, as illustrated in Fig. 6(b). However, it should be noted that the AvgDeviation of NR_DCLC changes much more steadily than that of LR_DCLC. The price paid by NR_DCLC for achieving the higher quality solutions is the higher time complexity. From Fig. 6(c), we can see that on average NR_DCLC needs to run Dijkstra's algorithm once or twice more than LR_DCLC. In the worst case, NR_DCLC needs to run 16 times, while LR_DCLC only 8 times, as shown in Table II.

By investigating Figs. 7 and 8 and Tables II and III, we can obtain very similar observations when link weights are distributed on the other two sets of intervals. The only difference is that in such cases the average cost deviation and the average number of executions are slightly higher, which indicates that it is slightly more difficult to find the optimal solution.

TABLE IV

THE MAXIMUM NUMBER OF EXECUTIONS FOR LR_DCLC/NR_DCLC WITH LINK-WEIGHT SET 3

| $\Delta$ | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|---|
| 50-node | 8/12 | 8/15 | 7/14 | 8/14 | 8/15 |
| 100-node | 7/12 | 8/15 | 7/14 | 8/14 | 7/16 |
| 200-node | 8/14 | 8/14 | 8/16 | 8/15 | 8/15 |

## VI. CONCLUSIONS

A QoS routing algorithm that can find solutions of high quality yet has a low time complexity is very important for high-speed networks to provide services with stringent and diverse QoS requirements. In this paper, we investigated the up-to-date heuristic algorithms for the DCLC problem based on linear and nonlinear Lagrange relaxation techniques. Algorithm NR_DCLC proposed in this paper is based on the nonlinear relaxation, while algorithm LR_DCLC published in prior works is based on the linear relaxation. A large number of simulations indicate that NR_DCLC can achieve better solutions than LR_DCLC by running Dijksra's algorithm a few more times, and its superiority becomes more appreciable in case that the optimal solution is more difficult to be found. One should notice that the experiments designed in this paper are the most difficult cases for searching for optimal solutions. In actual applications, both of the two algorithms might achieve better performance.

## REFERENCES


[1] S. Chen and K. Nahrstedt, "On finding multi-constrained paths," *ICC'98*, pp.874-879, Atlanta, Georgia, June 7-11, 1998.

[2] S. Chen and K. Nahrstedt, "An overview of quality of service routing for next-generation high-speed networks: problems and solutions," *IEEE Network*, pp.64-79, Nov./Dec. 1998.

[3] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol.1, pp.269-271, 1959.

[4] G. Feng and C. Doulgeris, "Fast algorithms for delay-constrained least-cost unicast routing," shortened version presented on *INFORMS'2001*, Miami Beach, Nov. 2001, available at http://www.students.miami.edu/∼gfeng/.

[5] G. Feng, K. Makki, N. Pissinou and C. Doulgeris, "An efficient approximate algorithm for delay-cost-constrained QoS routing," *ICCCN'2001*, pp. 395-400, Phoenix, Arizona, Oct 15-17, 2001.

[6] G. Feng, C. Doulgeris, K. Makki and N. Pissinou, "Heuristic and Exact Algorithms for QoS Routing with Multiple Constraints," submitted to *IEICE Trans. Communications*, November 2001, available at http://www.students.miami.edu/∼gfeng/.

[7] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[8] R. Hassin, "Approximation scheme for the restricted shortest path problem," *Mathematics of Operation Research*, 17(1): 36-42, Feb. 1992.

[9] J.M. Jaffe, "Algorithms for finding paths with multiple constraints," *Networks*, 14:95-116, 1984.

[10] A. Jüttner, B. Szviatovszki,I. Mëcs and Z. Rajkö, "Lagrange relaxation based method for the QoS routing problem", *INFOCOM'2001*, Alaska, 2001.

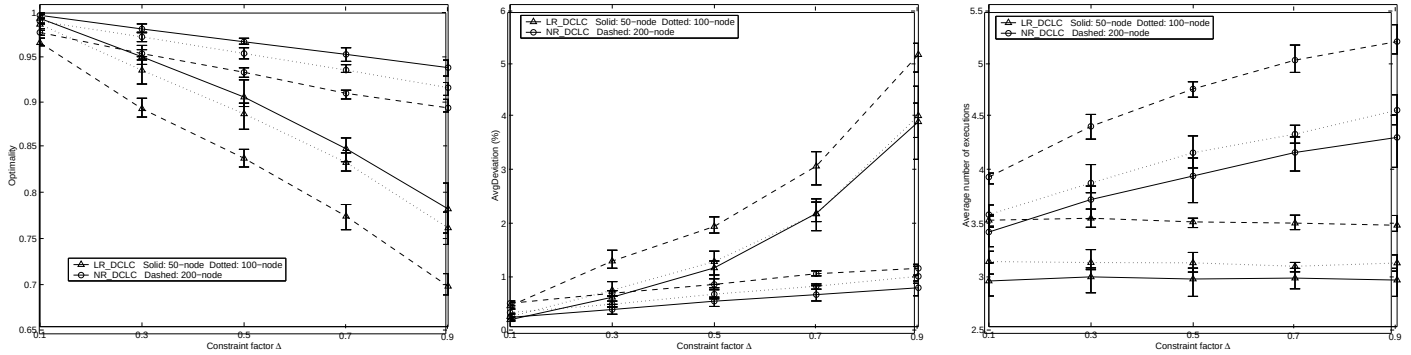
[11] T. Korkmaz and M. Krunz, "Multi-constrained optimal path selection," *INFOCOM'2001*, Alaska, 2001.

[12] K. Makki, N. Pissinou and O. Frieder, "Efficient solutions to multicast routing in communication networks," *Mobile networks and Applications*, vol. 1, pp. 221-232, 1996.

[13] H. Neve and P. Mieghem, "A multiple quality of service routing algorithm for PNNI," *Proc. Of the ATM Workshop*, pp. 324-328, May 1998.

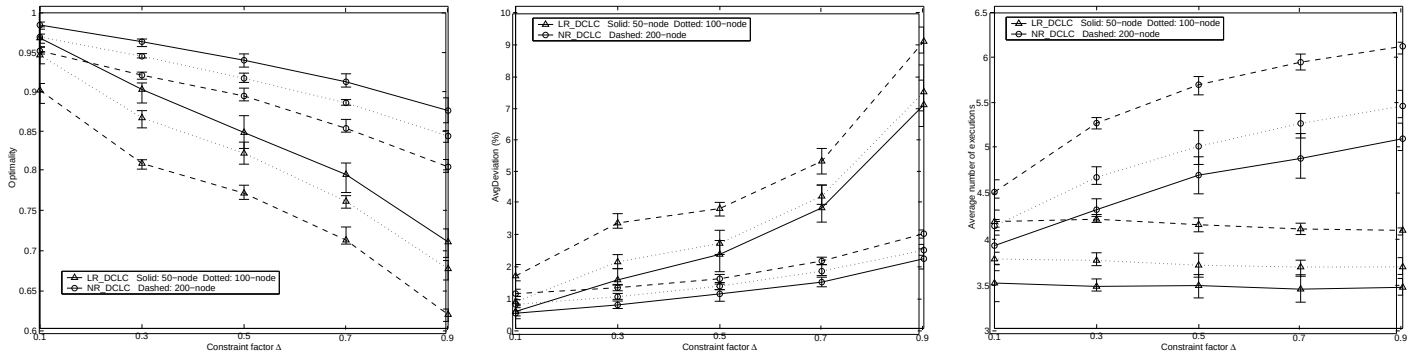
[14] A. Orda and A. Sprintson, "QoS routing: The precomputation perspective," *INFOCOM'2000*, vol.1, pp.128-13, Israel, March 26-30, 2000.

[15] Z. Wang and J. Crowcroft, "Quality-of-Service routing for supporting multimedia applications," *IEEE JSAC*, 14(7): 1288-1234, Sept. 1996.



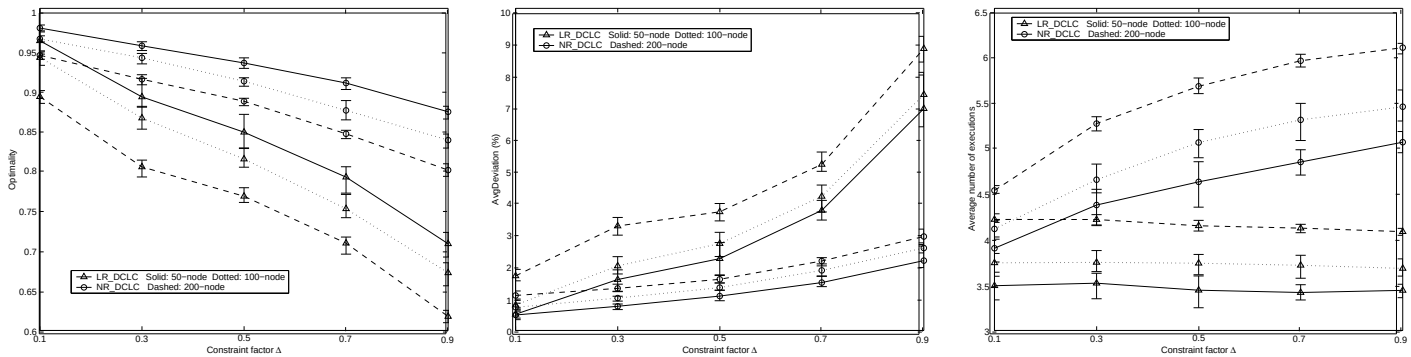

(a) Optimality (b) Average cost deviation (c) Average number of executions

Fig. 6. Performance comparison between algorithms LR_DCLC and NR_DCLC with link-weight set 1




(a) Optimality (b) Average cost deviation (c) Average number of executions

Fig. 7. Performance comparison between algorithms LR_DCLC and NR_DCLC with link-weight set 2




(a) Optimality (b) Average cost deviation (c) Average number of executions

Fig. 8. Performance comparison between algorithms LR_DCLC and NR_DCLC with link-weight set 3

[16] B.M. Waxman, "Routing of multipoint connections," *IEEE JSAC*, 6(9): 1617-1622, Dec. 1988.

[17] R. Widyono, "The design and evaluation of routing algorithms for real-time channels," Tech. Rep. ICSI TR-94-024, University of California at Berkley, International Computer Science Institute, June 1994.