GEN-LARAC: A Generalized Approach to the Constrained Shortest Path Problem under Multiple Additive Constraints*

Ying Xiao[1], Krishnaiyan Thulasiraman[1], and Guoliang Xue[2]

[1] School of Computer Science
University of Oklahoma, 200 Felgar Street, Norman, OK 73019, USA
{ying_xiao, thulsi}@ou.edu
[2] Arizona State University, Tempe, AZ 85287, USA
xue@asu.edu

Abstract. Given a network modeled as a graph G with each link associated with a cost and k weights, the Constrained Shortest Path (CSP(k)) problem asks for computing a minimum cost path from a source node s to a target node t satisfying pre-specified bounds on path weights. This problem is NP-hard. In this paper we propose a new approximation algorithm called GEN-LARAC for CSP(k) problem based on Lagrangian relaxation method. For $k = 1$, we show that the relaxed problem can be solved by a polynomial time algorithm with time complexity $O((m + n \log n)^2)$. Using this algorithm as a building block and combing it with ideas from mathematical programming, we propose an efficient algorithm for arbitrary k. We prove the convergence of our algorithm and compare it with previously known algorithms. We point out that our algorithm is also applicable to a more general class of constrained optimization problems.

1 Introduction

Recently there has been considerable interest in the design of communication protocols that deliver certain performance guarantees that are usually referred to as Quality of Service (QoS) guarantees. A problem of great interest in this context is the QoS routing problem that requires the determination of a minimum cost path from a source node to a destination node in a data network that satisfies a specified upper bound on the delay of the path. This problem is also known as the Constrained Shortest Path (CSP) problem. The CSP problem is known to be NP-hard [5]. So, in the literature, heuristic approaches and approximation algorithms have been proposed. Heuristics, in general, do not provide performance guarantees on the quality of the solution produced. On the other hand, ϵ-approximation algorithms deliver solutions within arbitrarily specified precision requirement but are not efficient in practice. References [6], [8], [13]

* The work of K. Thulasiraman has been supported by NSF ITR grant ANI-0312435
The work of G. Xue has been supported by NSF ITR grant ANI-0312635.

present ϵ-algorithms and contain several fundamental ideas of interest in developing such algorithms.

There are certain approximation algorithms based on mathematical programming techniques. These algorithms start with an integer linear programming (ILP) formulation and relax the integrality constraints. The relaxed problem is usually solved by Lagrangian dual (relaxation) method. The first such algorithm was reported in [7] by Handler and Zang. It is based on a geometric approach which is also called the hull approach by Mehlhorn and Ziegelmann [15]. More recently, in an independent work, Jüttner et al. [10] developed the Lagrangian Relaxation based Aggregated Cost (LARAC) algorithm which also uses the Lagrangian dual method. In contrast to the geometric method, they used an algebraic approach. In another work, Blokh and Gutin [2] defined a general class of combinatorial optimization problems of which the CSP problem is a special case. In a recent work, Xiao et al. [21] drew attention to the fact that the algorithms in [2], [7], [10] are equivalent. In view of this equivalence, we shall refer to these algorithms simply as the LARAC algorithm. Ziegelmann [24] provides a fairly complete list of references to the literature on the CSP and related problems.

The CSP(k) problem is more general than the CSP problem in that it allows more than one delay constraint. Given a communication network modeled as a graph G with each link in the graph associated with a cost and $k(\geq 1)$ weights, the CSP(k) problem asks for a minimum cost path from a source node s to a target node t satisfying multiple constraints on the path weights.

A variation of CSP(k) problem, Multi-Constrained Path (MCP) problem has also been a topic of extensive study. The difference between CSP(k) and MCP problems is that MCP problem only asks for a path satisfying all the constraints simultaneously without the requirement of minimizing the cost. For the MCP problem, a series of heuristics and approximation algorithms can be found in [4], [9],[11],[12], [16], [22], [23].

Two methods for the CSP(k) problem based on mathematical programming have been proposed by Beasley and Christofides [1], and Mehlhorn and Ziegelmann [15]. Reference [1] uses a subgradient procedure to compute the Lagrangian relaxation function of the ILP formulation. With geometrical interpretation of the algorithm of [7], the authors of [1] proposed an algorithm called hull approach which is a special case of cutting planes method [18].

In this paper we present a new approach to the CSP(k) problem using Lagrangian relaxation. We first show that for $k = 1$, an approximation solution can be computed in $O((m + n \log n)^2)$ time. Because this algorithm and LARAC are based on the same methodology and obtain the same solution, we also denote our algorithm as LARAC.

For arbitrary k, we use our LARAC algorithm as a building block and combine it with ideas from mathematical programming to achieve progressively higher values of the Lagrangian function. We present the GEN-LARAC algorithm and prove its correctness and convergence properties in Sect. 3. Simulation results comparing our algorithm with two other algorithms are presented in Sect. 4. We conclude in Sect. 5 pointing out that our approach is quite general and is applicable for the general class of combinatorial optimization problems studied in [2].

2 Problem Definition and Preliminaries

Consider a directed graph $G(N, E)$ where N is the set of nodes and E is the set of links in G. Each link (u, v) is associated with a set of $k + 1$ additive non-negative integer weights $C_{uv} = (c_{uv}, w^1_{uv}, w^2_{uv} \dots, w^k_{uv})$. Here c_{uv} is called the cost of link (u, v) and w^i_{uv} is called the ith delay of link (u, v). Given two nodes s and t, an s-t path in G is a directed simple path from s to t. Let P_{st} denote the set of all s-t paths in G. For path p define

$$c(p) = \sum_{(u,v)\in p} c_{uv} \text{ and } d_i(p) = \sum_{(u,v)\in p} w^i_{uv}, i = 1, 2 \dots, k.$$

The value $c(p)$ is called the cost of path p and $d_i(p)$ is called the ith delay of path p. Given k positive integer r_1, $r_2 \dots$, r_k, an s-t path is called feasible (resp. strictly feasible) if $d_i(p) \le r_i$ (resp. $d_i(p) < r_i$), for all $i = 1, 2 \dots, k$ (r_i is called the bound on the ith delay of a path).

The CSP(k) problem is to find a minimum cost feasible s-t path. An instance of the CSP(k) problem is strictly feasible if all the feasible paths are strictly feasible. Without loss of generality, we assume that the problem under consideration is always feasible. In order to guarantee strict feasibility, we do the following transformation.

For $i = 1, 2 \dots, k$, transform the delays of link (u, v) such that the new weight vector C'_{uv} is given by $C'_{uv} = (c_{uv}, 2w^1_{uv}, 2w^2_{uv} \dots, 2w^k_{uv})$.

Also transform the bounds r_i's so that the new bound vector R' is given by $R' = (2r_1 + 1, 2r_2 + 1 \dots, 2r_k + 1)$.

In the rest of the paper, we only consider the transformed problem. Thus all link delays are even integers, and delay bounds are odd integers. We shall use symbols with capital or bold letters to represent vectors and matrices. For the simplicity of presentation, we shall use C_{uv} and R instead of C'_{uv} and R' to denote the transformed weight vector and the vector of bounds. Two immediate consequences of this transformation are stated below.

Lemma 1. *$\forall p \in P_{st}, \forall i \in \{1, 2 \dots, k\}, d_i(p) \ne r_i$ in the transformed problem.*

Lemma 2. *A path in the original problem is feasible (resp. optimal) iff it is strictly feasible (resp. optimal) in the transformed problem.*

Starting with an ILP formulation of the CSP(k) problem and relaxing the integrality constraints we get the RELAX-CSP(k) problem below. In this formulation, for each s-t path p, we introduce a variable x_p.

$$\begin{aligned} &\text{RELAX - CSP}(k): \\ &\min \quad \sum_p c(p)x_p \end{aligned} \tag{1}$$

$$\text{s.t.} \quad \sum_p x_p = 1 \tag{2}$$

$$\sum_p d_i(p) x_p \leq r_i, i = 1 \ldots, k \tag{3}$$

$$x_p \geq 0, \forall p \in P_{st}. \tag{4}$$

The Lagrangian dual of RELAX-CSP(k) is given below.

$$\text{DUAL - RELAX-CSP}(k):$$

$$\max \quad w - \lambda_1 r_1 \cdots - \lambda_k r_k \tag{5}$$

$$\text{s.t.} \quad w - d_1(p)\lambda_1 \cdots - d_k(p)\lambda_k \leq c(p), \forall p \in P_{st} \tag{6}$$

$$\lambda_i \geq 0, i = 1 \ldots, k. \tag{7}$$

In the above dual problem λ_1,$\lambda_2 \ldots$,λ_k and w are the dual variables, with w corresponding to (2) and each λ_i corresponding to the ith constraint in (3).

It follows from (6) that $w \leq c(p) + d_1(p)\lambda_1 \cdots + d_k(p)\ \lambda_k, \forall p \in P_{st}$. Since we want to maximize (5), the value of w should be as large as possible, i.e.

$$w = \min_{p \in P_{st}} \{c(p) + d_1(p)\lambda_1 \cdots + d_k(p)\lambda_k\}.$$

With the vector Λ defined as $\Lambda = (\lambda_1, \lambda_2 \ldots, \lambda_k)$, define

$$L(\Lambda) = \min_{p \in P_{st}} \{c(p) + \sum_{i=1}^{k} \lambda_i (d_i(p) - r_i)\}. \tag{8}$$

Notice that $L(\Lambda)$ is called the Lagrangian function in literature and is a continuous concave function of Λ [3].

Then DUAL-RELAX-CSP(k) can be written as follows.

$$\text{DUAL- RELAX-CSP}(k)$$

$$\max\ L(\Lambda), \Lambda \geq \mathbf{0}. \tag{9}$$

The Λ^* that maximizes (9) is called the maximizing multiplier and is defined as

$$\Lambda^* = \arg\ \max_{\Lambda \geq 0} L(\Lambda). \tag{10}$$

Lemma 3. *If an instance of the CSP(k) problem is feasible and a path p_{opt} is an optimal path, then $\forall \Lambda \geq 0, L(\Lambda) \leq c(p_{opt})$.*

We shall use $L(\Lambda)$ as an lower bound of $c(p_{opt})$ to evaluate the approximation solution obtained by our algorithm. Given $p \in P_{st}$ and Λ, define

$$C(p) \equiv (c(p), d_1(p), d_2(p) \ldots, d_k(p)),$$
$$D(p) \equiv (d_1(p), d_2(p) \ldots, d_k(p)),$$
$$R \equiv (r_1, r_2 \ldots, r_k),$$
$$c_\Lambda(p) \equiv c(p) + d_1(p)\lambda_1 \cdots + d_k(p)\lambda_k, and$$
$$d_\Lambda(p) \equiv d_1(p)\lambda_1 \cdots + d_k(p)\lambda_k.$$

Here $c_\Lambda(p)$ and $d_\Lambda(p)$ are called the aggregated cost and the aggregated delay of path p, respectively. We shall use P_Λ to denote the set of s-t paths attaining the minimum aggregated cost w.r.t. to Λ. A path $p_\Lambda \in P_\Lambda$ is called a Λ-minimal path.

The key issue now is to search for the maximizing multipliers and termination conditions. For the case $k = 1$, we have the following theorem.

Theorem 1. *DUAL-RELAX-CSP(1) is solveable in $O((m + n \log n)^2)$ time.*

Proof. See Appendix. □

Because our algorithm and LARAC are based on the same methodology and obtain the same solution, we shall also call our algorithm LARAC. In the rest of the paper, we shall discuss how to extend it for $k > 1$. In particular we develop an approach that combines the LARAC algorithm as a building block with certain techniques in mathematical programming. We shall call this new approach as GEN-LARAC.

3 GEN-LARAC for the CSP(k) Problem

3.1 Optimality Conditions

Theorem 2. *Given an instance of a feasible CSP(k) problem, a vector $\Lambda \geq 0$ maximizes $L(\Lambda)$ iff the following problem in variables u_j's is feasible.*

$$\sum_{p_j \in P_\Lambda} u_j d_i(p_j) = r_i, \forall i, \lambda_i > 0 \tag{11}$$

$$\sum_{p_j \in P_\Lambda} u_j d_i(p_j) \leq r_i, \forall i, \lambda_i = 0 \tag{12}$$

$$\sum_{p_j \in P_\Lambda} u_j = 1 \tag{13}$$

$$u_j \geq 0, \forall j, p_j \in P_\Lambda. \tag{14}$$

Proof. **Sufficiency**: Let $\boldsymbol{x} = (u_1 \ldots, u_r, 0 \ldots, 0)$ be a vector of size $|P_{st}|$, where $r = |P_\Lambda|$.

Obviously, $\boldsymbol{x}$ is a feasible solution to RELAX-CSP(k). It suffices to show that $\boldsymbol{x}$ and Λ satisfy the complementary slackness conditions.

According to (6), $\forall p \in P_{st}, w \leq c(p) + d_1(p)\lambda_1 \cdots + d_k(p)\lambda_k$. Since we need to maximize (5), the optimal $w = c(p_\Lambda) + d_1(p_\Lambda)\lambda_1 \cdots + d_k(p_\Lambda)\lambda_k, \forall p_\Lambda \in P_\Lambda$. For all other paths $p, w - c(p) + d_1(p)\lambda_1 \cdots + d_k(p)\lambda_k < 0$. So $\boldsymbol{x}$ satisfies the complementary slackness conditions. By (11) and (12), Λ also satisfies complementary slackness conditions.

Necessary: Let $\boldsymbol{x}^*$ and (w, Λ) be the optimal solution to RELAX-CSP(k) and DUAL-RELAX-CSP(k), respectively. It suffices to show that we can obtain a feasible solution to (11)-(14) from $\boldsymbol{x}^*$.

We know that all the constraints in (6) corresponding to paths in $P_{st} - P_\Lambda$ are strict inequalities and $w = c(p_\Lambda) + d_1(p_\Lambda)\lambda_1 \cdots + d_k(p_\Lambda)\lambda_k, \forall p_\Lambda \in P_\Lambda$. So, from complementary slackness conditions we get

$$x_p = 0, \forall p \in P_{st} - P_\Lambda.$$

Now let us set u_j corresponding to path p in P_Λ equal to x_p, and set all other u_j's corresponding to paths not in P_Λ equal to zero. The u_i's so elected will satisfy (11) and (12) since these are complementary conditions satisfied by (w, Λ). Since x_i's satisfy (2), u_j's satisfy (13). Thus we have identified a solution satisfying (11)-(14). $\square$

3.2 GEN-LARAC: A Coordinate Ascent Method

Algorithm 1 GEN-LARAC: A Coordinate Ascent Algorithm

```
Step 1: $\Lambda^0 \leftarrow (0, \ldots, 0); flag \leftarrow true; B \leftarrow 0; t \leftarrow 0$
Step 2: {Coordinate Ascent Step}
while $flag$ do
  $flag \leftarrow false$
  for $i = 1$ to $k$ do
    $\gamma \leftarrow \arg\max_{\xi \geq 0} L(\lambda_1^t \ldots, \lambda_{i-1}^t, \xi, \lambda_{i+1}^t \ldots, \lambda_k^t)$
    if $\gamma \neq \lambda_i^t$ then
      $flag \leftarrow true$
      $\lambda_j^{t+1} = \begin{cases} \gamma & \text{if } j = i; \\ \lambda_j^t & \text{if } j \neq i. \end{cases}, j = 1, 2 \ldots, k$
      $t \leftarrow t + 1$
    end if
  end for
end while
Step 3: if $\Lambda^t$ is optimal then return $\Lambda^t$
Step 4: $B \leftarrow B+1$ and stop if $B > B_{max}$ ($B_{max}$ is the maximum number of iteration allowed)
Step 5: Compute a vector $\Lambda^+$ such that $L(\Lambda^+) > L(\Lambda^t)$.
Step 6: $t \leftarrow t + 1, \Lambda^t \leftarrow \Lambda^+$, and go to Step 2.
```

Our approach is based on the coordinate ascent method and proceeds as follows. Given a multiplier Λ, in each iteration we try to improve the value of $L(\Lambda)$ by updating one component of the multiplier vector. If the objective function is not differentiable, the coordinate ascent method may get stuck at a corner Λ_s not being able to make progress by changing only one component. We shall call Λ_s pseudo optimal point which requires updates of at least two components to achieve improvement in the solution. We shall discuss how to jump to a better solution from a pseudo optimal point in Sect. 3.3. Our simulations show that the objective values attained at pseudo optimal points are usually very close to the maximum value of $L(\Lambda)$.

3.3 Verification of Optimality of Λ

In Step 3 we need to verify if a given Λ is optimal. We show that this can be accomplished by solving the following LP problem, where $P_\Lambda = \{p_1, p_2 \dots, p_r\}$ is the set of Λ-minimal paths.

$$\begin{aligned}
\max \quad & 0 && (15)\\
\text{s.t.} \quad & \sum_{p_j \in P_\Lambda} u_j d_i(p_j) = r_i, \forall i, \lambda_i > 0 && (16)\\
& \sum_{p_j \in P_\Lambda} u_j d_i(p_j) \leq r_i, \forall i, \lambda_i = 0 && (17)\\
& \sum_{p_j \in P_\Lambda} u_j = 1 && (18)\\
& u_j \geq 0, \forall j, p_j \in P_\Lambda. && (19)
\end{aligned}$$

By Theorem 2, if the above linear program is feasible then the multiplier Λ is a maximizing multiplier.

Let $(y_1 \dots, y_k, \delta)$ be the dual variables corresponding to the above problem. Let $Y = (y_1, y_2 \dots, y_k)$. The dual of (15)-(19) can be written as follows

$$\begin{aligned}
\min \quad & RY^T + \delta && (20)\\
\text{s.t.} \quad & D(p_i)Y^T + \delta \geq 0, i = 1, 2 \dots, r && (21)\\
& y_i \geq 0, \forall i, \lambda_i > 0. && (22)
\end{aligned}$$

Evidently the LP problem (20)-(22) is feasible. From the relationship between primal and dual problems, it follows that if the linear program (15)-(19) is infeasible, then the objective of (20) is unbounded ($-\infty$). Thus, if the optimum objective of (20)-(22) is 0, then the linear program (15)-(19) is feasible and by Theorem 2 the corresponding multiplier Λ is optimal. In summary we have the following lemma.

Lemma 4. *If (15)-(19) is infeasible, then* $\exists Y = (y_1, y_2 \dots, y_k)$ *and* δ *satisfying (21)-(22) and* $RY^T + \delta < 0$.

The Y required in the above lemma can be identified by applying the simplex method on (20)-(22) and terminating it once the objective value becomes negative.

Let Λ be a non-optimal Lagrangian multiplier and denote $\Lambda(s, Y) = \Lambda + Y/s$ for $s > 0$.

Theorem 3. *If a multiplier* $\Lambda \geq 0$ *is not optimal, then*

$$\exists M > 0, \forall s > M, L(\Lambda(s, Y)) > L(\Lambda).$$

Proof. If M is big enough, $P_\Lambda \cap P_{\Lambda(s,Y)} \neq \emptyset$. Let $p_J \in P_\Lambda \cap P_{\Lambda(s,Y)}$.

$$\begin{aligned} L(\Lambda(s,Y)) &= c(p_J) + (D(p_J) - R)(\Lambda + Y/s)^T \\ &= c(p_J) + (D(p_J) - R)\Lambda^T + (D(p_J) - R)(Y/s)^T \\ &= L(\Lambda) + (D(p_J)Y^T - RY^T)/s. \end{aligned}$$

Since $D(p_J)Y^T + \delta \geq 0$ and $RY^T + \delta < 0, D(p_J)Y^T - RY^T > 0$.
Hence $L(\Lambda(s,Y)) > L(\Lambda)$. □

We can find the proper value of M by binary search after computing Y. The last issue is to compute P_Λ. It can be expected that the size of P_Λ is usually very small. So we adapted the k-shortest path algorithm to compute P_Λ.

3.4 Analysis of the Algorithm

In this section, we shall discuss the convergence properties of GEN-LARAC.

Lemma 5. *If there is a strictly feasible path, then for any given τ, the set $S_\tau = \{\Lambda | L(\Lambda) \geq \tau\} \subset R^k$ is bounded.*

Proof. Let p^* be a strictly feasible path. For any $\Lambda = (\lambda_1 \cdots, \lambda_k) \in S_\tau$, we have

$$c(p^*) + \lambda_1(d_1(p^*) - r_1) \cdots + \lambda_k(d_k(p^*) - r_k) \geq L(\Lambda) \geq \tau.$$

Since $d_i(p^*) - r_i < 0$ and $\lambda_i \geq 0$ for $i = 1, 2 \ldots, k, \Lambda$ must be bounded. □

If there is only one delay constraint, i.e., $k = 1$, we have the following property [10].

Lemma 6. *A value $\lambda > 0$ maximizes the function $L(\lambda)$ if and only if there are paths p_c and p_d which are both c_λ-minimal and for which $d(p_c) \geq r$ and $d(p_d) \leq r$. (p_c and p_d can be the same; in this case $d(p_d) = d(p_c) = r$).*

By Lemma 6, we have the following lemma.

Lemma 7. *A multiplier $\Lambda \geq 0$ is pseudo optimal iff $\forall i \exists p_c^i, p_d^i \in P_\Lambda, d_i(p_c^i) \geq r_i$ and $d_i(p_d^i) \leq r_i$.*

For an n-vector $V = (v_1, v_2 \ldots, v_n)$, let $|V|_1 = |v_1| + |v_2| \cdots + |v_n|$ denote the L^1-norm.

Lemma 8. *Let Λ and H be two multipliers obtained in the same while-loop of Step 2 in Algorithm 1. Then $|L(H) - L(\Lambda)| \geq |H - \Lambda|_1$.*

Proof. Let $\Lambda = \Lambda^1, \Lambda^2 \ldots, \Lambda^j = H$ be the consecutive sequence of multipliers obtained in Step 2. We first show that $|L(\Lambda^{i+1}) - L(\Lambda^i)| \geq |\Lambda^{i+1} - \Lambda^i|_1$.
Consider two cases.
Case 1: $\lambda_b^{i+1} > \lambda_b^i$. By Lemma 6 and Lemma 1, $\exists p_{\Lambda^{i+1}}$, $d_b(p_{\Lambda^{i+1}}) > r_b$.
By definition, we have:

$c(p_{\Lambda^{i+1}}) + \Lambda^{i+1} D^T(p_{\Lambda^{i+1}}) \leq c(p_{\Lambda^i}) + \Lambda^{i+1} D^T(p_{\Lambda^i})$, and
$c(p_{\Lambda^{i+1}}) + \Lambda^{i} D^T(p_{\Lambda^{i+1}}) \geq c(p_{\Lambda^i}) + \Lambda^{i} D^T(p_{\Lambda^i})$.

Then

$$\begin{aligned}
&L(\Lambda^{i+1}) - L(\Lambda^i) \\
&= c(p_{\Lambda^{i+1}}) + \Lambda^{i+1}(D(p_{\Lambda^{i+1}}) - R)^T - [c(p_{\Lambda^i}) + \Lambda^i(D(p_{\Lambda^i}) - R)^T] \\
&= c(p_{\Lambda^{i+1}}) + \Lambda^{i}(D(p_{\Lambda^{i+1}}) - R)^T + (\Lambda^{i+1} - \Lambda^i)(D(p_{\Lambda^{i+1}}) - R)^T \\
&\quad - [c(p_{\Lambda^i}) + \Lambda^i(D(p_{\Lambda^i}) - R)^T] \\
&\geq c(p_{\Lambda^i}) + \Lambda^i(D(p_{\Lambda^i}) - R)^T + (\Lambda^{i+1} - \Lambda^i)(D(p_{\Lambda^{i+1}}) - R)^T \\
&\quad - [c(p_{\Lambda^i}) + \Lambda^i(D(p_{\Lambda^i}) - R)^T] \\
&= (\Lambda^{i+1} - \Lambda^i)(D(p_{\Lambda^{i+1}}) - R)^T \\
&= (\lambda_b^{i+1} - \lambda_b^i)(d_b(p_{\Lambda^{i+1}}) - r_b) \geq |\lambda_b^{i+1} - \lambda_b^i|.
\end{aligned}$$

Case 2: $\lambda_b^{i+1} < \lambda_b^i$. We have $\exists p_{\Lambda^{i+1}}, \, d_b(p_{\Lambda^{i+1}}) < r_b$.
The rest of the proof is similar to Case 1.
Hence

$$\begin{aligned}
&|L(\Lambda^j) - L(\Lambda^1)| \\
&\quad = |L(\Lambda^2) - L(\Lambda^1) + L(\Lambda^3) - L(\Lambda^2) \cdots + L(\Lambda^j) - L(\Lambda^{j-1})| \\
&\quad = |L(\Lambda^2) - L(\Lambda^1)| + |L(\Lambda^3) - L(\Lambda^2)| \cdots + |L(\Lambda^j) - L(\Lambda^{j-1})| \\
&\quad \geq |\Lambda^2 - \Lambda^1|_1 + |\Lambda^3 - \Lambda^2|_1 \cdots + |\Lambda^j - \Lambda^{j-1}|_1 \geq |\Lambda^j - \Lambda^1|_1.
\end{aligned}$$

The second equality holds because $L(\Lambda^1) < L(\Lambda^2) < \cdots < L(\Lambda^j)$. □

Obviously, if the while-loop in Step 2 terminates in a finite number of steps, the multiplier is pseudo optimal by definition. If the while loop does not terminate in a finite number of steps (this occurs only when infinite machine precision is assumed), we have the following theorem.

Theorem 4. *Let $\{\Lambda^i\}$ be a consecutive sequence of multipliers generated in the same while-loop in Step 2 in Algorithm 1. Then the limit point of $\{L(\Lambda^i)\}$ is pseudo optimal.*

Proof. Since $L(\Lambda^1) < L(\Lambda^2) < \cdots$ and Λ^i is bounded from above, $\lim_{s\to\infty} L(\Lambda^s)$ exists and is denoted as L^*. We next show the vector $\lim_{s\to\infty} \Lambda^s$ also exists.

By Lemma 8, $\forall s, j > 0$, $|\Lambda^{s+j} - \Lambda^s|_1 \leq |L(\Lambda^{s+j}) - L(\Lambda^s)|$.

By Cauchy criterion, $\lim_{s\to\infty} \Lambda^s$ exists. We denote it as Λ^*.

We label all the paths in P_{st} as $p_1, p_2 \ldots, p_N$ such that $c_{\Lambda^*}(p_1) \leq c_{\Lambda^*}(p_2) \ldots \leq c_{\Lambda^*}(p_N)$. Obviously p_1 is a Λ^*-minimal path. Let

$$\begin{aligned}
&\theta = \min\{c_{\Lambda^*}(p_j) - c_{\Lambda^*}(p_i) | \forall p_i, p_j \in P_{st}, c_{\Lambda^*}(p_j) - c_{\Lambda^*}(p_i) > 0\}, and \\
&\pi = max\{d_w(p_j) - d_w(p_i) | \forall p_i, p_j \in P_{st}, w = 1, 2 \ldots, k\}.
\end{aligned}$$

Let M be a large number, such that $\forall t \geq M, |\Lambda^* - \Lambda^t|_1 < \theta/(2\pi)$.

Consider any component $j \in \{1, 2 \ldots, k\}$ of the multiplier after computing $\arg\max_{\xi\geq 0} L(\lambda_1 \ldots, \lambda_{j-1}, \xi, \lambda_{j+1} \ldots, \lambda_k)$ in iteration $t \geq M$.

By Lemma 6, $\exists p_c^t$ and $p_d^t \in P_{\Lambda^t}$ and $d_j(p_c^t) \geq r_j \geq d_j(p_d^t)$.

It suffices to show $P_{\Lambda^t} \subseteq P_{\Lambda^*}$. Given $p_{\Lambda^t} \in P_{\Lambda^t}$, we shall show $c_{\Lambda^*}(p_{\Lambda^t}) = c_{\Lambda^*}(p_1)$. We have

$$\begin{aligned} 0 &\leq c_{\Lambda^t}(p_1) - c_{\Lambda^t}(p_{\Lambda^t}) = [c(p_1) + d_{\Lambda^t}(p_1)] - [c(p_{\Lambda^t}) + d_{\Lambda^t}(p_{\Lambda^t})] \\ &= c(p_1) + d_{\Lambda^*}(p_1) - [c(p_{\Lambda^t}) + d_{\Lambda^*}(p_{\Lambda^t})] + (\Lambda^t - \Lambda^*)(D(p_1) - D(p_{\Lambda^t}))^T \\ &= c_{\Lambda^*}(p_1) - c_{\Lambda^*}(p_{\Lambda^t}) + (\Lambda^t - \Lambda^*)(D(p_1) - D(p_{\Lambda^t}))^T. \end{aligned}$$

Since $|(\Lambda^t - \Lambda^*)(D(p_i) - D(p_j))^T| \leq \pi|(\Lambda^t - \Lambda^*)|_1 \leq \theta/2$,

$$0 \leq c_{\Lambda^t}(p_1) - c_{\Lambda^t}(p_{\Lambda^t}) \leq c_{\Lambda^*}(p_1) - c_{\Lambda^*}(p_{\Lambda^t}) + \theta/2.$$

Then $c_{\Lambda^*}(p_{\Lambda^t}) - c_{\Lambda^*}(p_1) \leq \theta/2$, which implies that $c_{\Lambda^*}(p_{\Lambda^t}) = c_{\Lambda^*}(p_1)$.
Hence, $\forall j \in \{1, 2 \ldots, k\}, \exists p_c^*, p_d^* \in P_{\Lambda^*}, d_j(p_c^*) \geq r_j \geq d_j(p_d^*)$. □

4 Numerical Simulation

We use $COPT$, OPT, and $POPT$ to denote the cost of the optimal path to the CSP(k) problem, the optimal value, and the pseudo optimal value of the Lagrangian function, respectively. In our simulation, we first verify that the objectives at pseudo optimal points are very close to the optimal objectives. We use 3 types of graphs: Power-law out-degree graph (PLO) [17], Waxman's random graph (RAN) [20], and regular graph (REG) [19]. The number of weights is 4, 8 and 12, i.e., k = 4, 8, and 12. Link weights are random even integers uniformly distributed between 2 and 200. We use the following two metrics to measure the quality of path p in Table 1.

$$g(p) = c(p)/POPT \text{ and } f(p) = \max_{i=1,2\ldots,k} d_i(p)/r_i.$$

By Lemma 3, $g(p)$ is the upper bound of the ratio of the cost of p and $COPT$. The $f(p)$ indicates the degree of violation of p to the constraints on its delays.

In Fig. 1, the label LARAC-REG means the results obtained by running GEN-LARAC algorithm on regular graphs. Other labels can be interpreted similarly. We only report the results on regular graph and random graph for better visibility.

We conducted extensive experiments to compare our algorithm with the Hull approach [15], the subgradient method [1], and the general-purpose LP solver CPLEX. Because the four approaches share the same objective, i.e., maximizing the Lagrangian function, they always obtain similar results. Due to space limitation we only report the number of shortest path computation which dominate the running time of all the first three algorithms. Generally, GEN-LARAC algorithm and Hull approach are faster than the subgradient methods and CPLEX (See [24] for the comparison of Hull approach and CPLEX). But GEN-LARAC and Hull approach beat each other on different graphs. Figure 1 shows that on the regular graph, GEN-LARAC is the fastest. But for the random and Power-law out degree graphs, the Hull approach is the fastest. The probable reason

Table 1. Quality of Pseudo-optimal paths: Error = $(OPT - POPT)/OPT$
#SP = Number of invocations of shortest path algorithm

Type	k	OPT	$POPT$	**Error**	$g(p)$	$f(p)$	**#SP**	**Time(s)**
REG	4	1116.8	1109.9	0.006	1.01	1.07	15.2	0.14
REG	8	1078.3	1069.0	0.032	1.00	1.09	20.7	0.21
REG	12	1066.2	1057.8	0.008	1.00	1.08	28.2	0.32
PLO	4	401.75	382.71	0.047	1.00	1.25	9.6	0.07
PLO	8	328.95	320.77	0.025	1.01	1.15	7.2	0.04
PLO	12	368.43	342.43	0.071	1.02	1.24	18.3	0.21
RAN	4	1543.6	1531.5	0.008	1.01	1.08	13.4	0.13
RAN	8	1473.3	1456.5	0.011	1.00	1.09	20.0	0.25
RAN	12	1438.6	1423.7	0.010	1.00	1.01	17.9	0.45

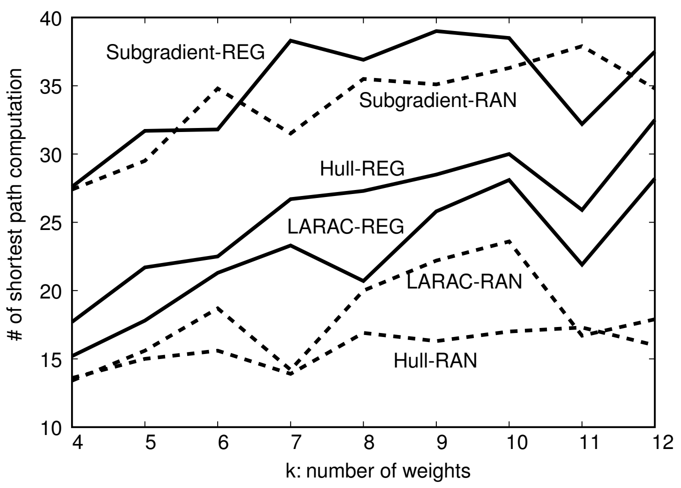

Fig. 1. The comparison of the number of shortest path computation among GEN-LARAC, Hull approach, and subgradient method. All algorithms terminate when have reached 99% of the OPT

is that the number of s-t paths is relatively small in these two types of graphs because the length (number of hops)of s-t paths is usually small even when the number of nodes is large. This will bias the results in favor of Hull approach which adds one s-t path into the linear system in each iteration [15]. So we choose the regular graph because we have a better control of the length of s-t paths.

5 Summary and Conclusion

In this paper we developed a new approach to the constrained shortest path problem involving multiple additive constraints. Our approach uses the LARAC algorithm as a building block and combines it with certain ideas from mathematical programming to design a method that progressively improves the value of the Lagrangian function until optimum is reached. The algorithm is analyzed and its convergence property has been established. Simulation results comparing our approach with two other approaches show that the new approach is quite competitive.

Since the LARAC algorithm is applicable for the general class of optimization problems (involving one additive delay constraint) studied in [2] our approach can also be extended for this class of problems to include multiple additive constraints, whenever an algorithm for the underlying optimization problem (such as Dijkstra's algorithm for the shortest path problem) is available.

References

1. J. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989.
2. B. Blokh and G. Gutin. An approximation algorithm for combinatorial optimization problems with two parameters. *Australasian Journal of Combinatorics*, 14:157–164, 1996.
3. S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, Cambridge, UK, 2003.
4. S. Chen and K. Nahrstedt. On finding multi-constrained path. In *ICC*, pages 874–879, 1998.
5. M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman, San Francisco, CA, USA, 1979.
6. Ashish Goel, K. G. Ramakrishnan, Deepak Kataria, and Dimitris Logothetis. Efficient computation of delay-sensitive routes from one source to all destinations. In *INFOCOM*, pages 854–858, 2001.
7. G. Handler and I. Zang. A dual algorithm for the constrained shortest path problem. *Networks*, 10:293–310, 1980.
8. R. Hassin. Approximation schemes for the restricted shortest path problem. *Math. of Oper. Res.*, 17(1):36–42, 1992.
9. J. M. Jaffe. Algorithms for finding paths with multiple constraints. *Networks*, 14:95–116, 1984.

10. Alpár Jüttner, Balázs Szviatovszki, Ildikó Mécs, and Zsolt Rajkó. Lagrange relaxation based method for the QoS routing problem. In *INFOCOM*, pages 859–868, 2001.
11. Turgay Korkmaz and Marwan Krunz. Multi-constrained optimal path selection. In *INFOCOM*, pages 834–843, 2001.
12. Gang Liu and K. G. Ramakrishnan. A*prune: An algorithm for finding k shortest paths subject to multiple constraints. In *INFOCOM*, pages 743–749, 2001.
13. D. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest paths problem. *Oper. Res. Letters*, 28:213–219, 2001.
14. Nimrod Megiddo. Combinatorial optimization with rational objective functions. In *STOC '78: Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 1–12, New York, NY, USA, 1978. ACM Press.
15. Kurt Mehlhorn and Mark Ziegelmann. Resource constrained shortest paths. In *ESA*, pages 326–337, 2000.
16. H. De Neve and P. Van Mieghem. Tamcra: A tunable accuracy multiple constraints routing algorithm. *Comput. Commun.*, 23:667–679, 2000.
17. C. R. Palmer and J. G. Steffan. Generating network topologies that obey power laws. In *IEEE GLOBECOM*, pages 434–438, 2000.
18. A. Schrijver. *Theory of linear and integer programming.* John Wiley, New York, 1986.
19. K. Thulasiraman and M. N. Swamy. *Graphs: Theory and algorithms.* Wiley Interscience, New York, 1992.
20. B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, Dec. 1988.
21. Y. Xiao, K. Thulasiraman, and G. Xue. Equivalence, unification and generality of two approaches to the constrained shortest path problem with extension. In *Allerton Conference on Control, Communication and Computing, University of Illinois*, pages 905–914, 2003.
22. G. Xue, A. Sen, and R. Banka. Routing with many additive QoS constraints. In *ICC*, pages 223–227, 2003.
23. Xin Yuan. Heuristic algorithms for multiconstrained quality-of-service routing. *IEEE/ACM Trans. Netw.*, 10(2):244–256, 2002.
24. M. Ziegelmann. *Constrained shortest paths and related problems.* PhD thesis, Max-Planck-Institut fr Informatik, 2001.

Appendix: Proof to Theorem 1

Lemma 9. *If $\lambda < \lambda^*(\lambda > \lambda^*)$, then $d(p_\lambda) \geq T(d(p_\lambda) \leq T)$ for each c_λ-minimal path, where T is the path delay constraint [10].*

We next give the proof to Theorem 1.

Proof. We prove this theorem by showing an algorithm with $O((m + n \log n)^2)$ time complexity. Assume vertex 1 (n) is the source (target). Algorithm 2 computes shortest path using lexicographic order on a pair of link weights (l_{uv}, c_{uv}), $\forall(u, v) \in E$ based on parametric search [14], where $l_{uv} = c_{uv} + \lambda^* d_{uv}$ and λ^* is unknown. The algorithm is the same as Bellman-Ford algorithm except for Step 4 which needs special cares (We use Bellman-Ford algorithm here because it is easy to explain. Actually the Dijkstra's algorithm is used to get the claimed time

Algorithm 2 Parametric Search Based Algorithm for CSP(1) Problem

```
Step 1: $M_v = (x_v, y_v) \leftarrow (+\infty, +\infty)$ for $v = 2, 3 \ldots, n$ and $M_1 = (0, 0)$
Step 2: $i \leftarrow 1$
Step 3: $u \leftarrow 1$
Step 4: $\forall v, (u, v) \in E$, if $(x_v + \lambda^* y_v > x_u + \lambda^* y_u + c_{uv} + \lambda^* d_{uv})$ or
              $(x_v + \lambda^* y_v = x_u + \lambda^* y_u + c_{uv} + \lambda^* d_{uv})$ and $(x_v > x_u + c_{uv}))$
                  $M_v \leftarrow (x_u + c_{uv}, y_u + d_{uv})$
Step 5: $u \leftarrow u + 1$ and if $u \leq n$, go to Step 4.
Step 6: $i \leftarrow i + 1$ and if $i < n$ , go to Step 3.
```

complexity). In Algorithm 2, we need extra steps to evaluate the Boolean expression in the if statement in Step 4 since $\lambda^* \geq 0$ is unknown. If $x_v = \infty, y_v = \infty$, then the inequality holds. Assume x_v and y_v are finite (non-negative) values. We need an Oracle test to tell whether the value of $p + q\lambda^*$ is less than, equal to, or larger than 0, where $p = x_u + c_{uv} - x_v$ and $q = y_u + d_{uv} - y_v$. If $p\ q \geq 0$, it is trivial to find the sign of $p + q\lambda^*$. WLOG, assume $p\ q < 0$, i.e., $-p/q > 0$. The Oracle test is presented as Algorithm 3.

The time complexity of the Oracle test is $O(m + n \log n)$. On the other hand, we can revise Algorithm 2 using Dijkstra's algorithm and the resulting algorithm has time complexity $O((m + n \log n)^2)$.

Next, we show how to compute the value of λ^* and $L(\lambda^*)$. Algorithm 2 computes a λ^*-minimal path p with minimal cost. Similarly, we can compute a λ^*-minimal path q with minimal delay. Then the value of λ^* is given by the following equation: $c(p) + \lambda^* d(p) = c(q) + \lambda^* d(q)$ and $L(\lambda^*) = c(p) + \lambda^*(d(p) - T)$. Notice that $d(q) < T < d(p)$. □

Algorithm 3 Oracle Test with Unknown λ^*

```
Step 1: Let $\lambda = -p/q > 0$ and $\forall (u, v) \in E$, define link length $l_{uv} = c_{uv} + \lambda d_{uv}$
Step 2: Compute two shortest paths $p_c$ and $p_d$ using the lexicographic order on
$(l_{uv}, c_{uv})$ and $(l_{uv}, d_{uv})$, respectively
Step 3: Obviously, $d(p_c) \geq d(p_d)$. Consider 4 cases:

  1. $d(p_c) > T$ and $d(p_d) > T$: By Lemma 6 and Lemma 9, $\lambda < \lambda^*$ and thus
     $p + q\lambda^* < 0$ if $q < 0$ and $p + q\lambda^* > 0$ otherwise
  2. $d(p_c) < T$ and $d(p_d) < T$: By Lemma 6 and Lemma 9, $\lambda > \lambda^*$ and thus
     $p + q\lambda^* > 0$ if $q < 0$ and $p + q\lambda^* < 0$ otherwise
  3. $d(p_c) > T$ and $d(p_d) < T$: By Lemma 6, $\lambda = \lambda^*$and thus $p + q\lambda^* = 0$
  4. $d(p_c) = T$ or $d(p_d) = T$: By Lemma 1, this is impossible
```