Truthful Least-Priced-Path Routing in Opportunistic Spectrum Access Networks

Tao Shu and Marwan Krunz
Department of Electrical and Computer Engineering
The University of Arizona
Technical Report
TR-UA-ECE-2009-3
July 2009

Abstract

We study the problem of finding the least-priced path (LPP) between a source and a destination in opportunistic spectrum access (OSA) networks. This problem is motivated by economic considerations, whereby spectrum opportunities are *sold/leased* to secondary radios (SRs). This incurs a cost for the communication services, e.g., for traffic relaying, provided by these SRs. As the beneficiary of these services, the end user must compensate the service-providing SRs for their spectrum cost. To give an incentive (i.e., profit) for SRs to report their true cost, typically the payment to a SR should be higher than the actual cost. However, from an end user's perspective, unnecessary overpayment should be avoided. So we are interested in the optimal route selection and payment determination mechanism that minimizes the price tag of the selected route and at the same time guarantees truthful cost reports from SRs. This is in sharp contrast to the conventional truthful least-cost path (LCP) problem, where the interest is to find the minimum-cost route. Our investigation of the LPP problem is divided into two parts, in which we solve the problem without and with capacity constraints at individual SRs, respectively. For both cases, our algorithmic solutions can be executed in polynomial time. The effectiveness of our algorithms on price saving is then verified through extensive simulations.

1 Introduction

Spectrum is the most critical yet scarce resource in wireless communications. Recently, it was revealed that under the current fixed-spectrum-allocation paradigm, spectrum is being significantly underutilized. This finding motivated extensive research on *opportunistic spectrum access* (OSA) [19], the premise of which is to allow for secondary re-use of the spectrum to improve its utilization. Under OSA, a secondary radio (SR) is allowed access to a channel

that is not currently being used by the primary radios (PRs) of the channel, given that such an opportunity-based access does not intervene with PRs' normal communication.

Inevitably, economic considerations are critical driving forces behind the realization of OSA. Under the assumption of economically rational players, a primary owner of the spectrum has interest in opening its idle spectrum for secondary re-use only if such action is profitable. As a result, the acquisition and access of the spectrum by SRs incur a certain cost, which generally takes the form of monetary payment made to the primary owner of the spectrum. Such an economic consideration has been reflected extensively in recent studies on OSA. For example, spectrum auction works in [22] and [9] considered the situation where SRs acquire spectrum through a bidding process. The spectrum-leasing architecture studied in [13] requires SRs to subscribe to (and pay for) the spectrum-status information broadcasted by a spectrum server. In addition, the current IEEE 802.22 WRAN standard is based on an infrastructure-type architecture, which by design is suitable and intended for implementing fee-based services for SRs.

Different from most of the existing works that study the cost aspects of acquiring spectrum, in this paper we study its "end system" perspective, focusing on the implication of such a cost. In particular, we limit our interest to the economical routing problem under OSA. We consider a situation where a source SR in an OSA network wishes to purchase a route to some destination. Intermediate SRs along this route relay the traffic from this source, until it finally reaches its destination. Such traffic relaying is not free for the source, since intermediate SRs must have paid to access their channels (and thus are able to provide relay service for the source). They do not have the incentive to relay the traffic unless the source compensates them for their spectrum cost. The problem for the source is to decide the cheapest route, in the sense that its total out-of-pocket payment to the relaying nodes, or equivalently the price of the route, is the minimum among all feasible routes from the source to the destination. We refer to this problem as the least-priced-path (LPP) problem. Note that even though we here assume that the source pays for the route, the problem does not lose its generality when the destination makes the payment.

At a first glance, the LPP problem is seemingly trivial and can be solved by the following naive method. The source would ask all SRs to report their costs. The source would then choose the shortest path to the destination with the reported cost used as a node's weight. Each SR along the selected path would be paid with the equivalent amount of its reported cost. The problem with this method is that intermediate SRs may exaggerate their claimed costs for the purpose of getting higher profits (in here, profit is defined as the difference between the payment a SR receives and its true spectrum cost). This is especially true when the intermediate node belongs to a different administrative domain than the source, and thus is opt to act selfishly to maximize its own interests. As a result, the source could end up paying an unnecessarily high price for the route it chooses.

We note that the above cheating behavior has been well addressed in the literature under a different setup, namely, finding the least-cost path (LCP) (e.g., see [12, 8, 7, 16]). The basic idea in the LCP algorithm is to design a *truthful* payment mechanism, e.g., the VCG mechanism in [12], which guarantees that cheating in the claimed cost cannot increase the relaying node's profit. As a result, such a node has an incentive to report its true cost in order to maximize its profit. The LCP can be subsequently constructed based on the reported costs, since they are guaranteed by the payment mechanism to be the true ones.

In contrast to the LCP problem, the LPP problem studied in this paper takes into account the following three new aspects presented by the OSA paradigm. First, instead of minimizing the *cost* of the route, we aim at minimizing the *price* of the route. This is more attractive to an end user, because price is the actual out-of-pocket expense the end user has to pay. As will become clear shortly, under the truthfulness requirement, there is no straightforward conversion between these two objectives. A new formulation is required for our LPP problem.

Second, different from all existing LCP formulations, where the cost of a node is a constant, here cost is modeled as a random variable. The node cost in OSA represents the monetary rate the SR node has to pay in order to acquire the spectrum. Such a rate changes with the supply-and-demand dynamics of the available spectrum in the vicinity of the node.

When the spectrum's supply-and-demand is tight, a SR may pay higher in order to access the spectrum. On the contrary, the cost drops when more idle spectrum becomes available for secondary re-use. In line with this randomized node cost, we are interested in finding a randomized routing strategy, which can adapt to the dynamics of the node cost, with the goal of minimizing the expected price of the route.

Third, when constructing a route, we consider the capacity limit of each node. Being restricted to a secondary role, a SR cannot guarantee it can always acquire the amount of spectrum it needs. Therefore, an end-to-end flow may have to be split into multiple sub-flows in order to get accommodated by the capacity-limited intermediate nodes, leading to multi-path routing. This is in contrast to the single-path situation considered in the LCP problem, where no capacity constraint is imposed on relaying nodes.

In this paper, we model truthful LPP routing as a mechanism design problem. Our investigation is divided into two parts. In the first part, we find the LPP without imposing a node-capacity constraint. This simplified formulation applies to the scenario where the rate demand of the source is extremely low, such that intermediate nodes can always support the required demand. In the second part, we solve the full version of the problem by considering a given source rate demand and given capacity constraints at intermediate nodes. Because this version requires capacity information to be collected, a payment materialization algorithm is also developed to guarantee truthful capacity reporting by relaying nodes. Our work can be considered as an extension to the Myerson's optimal auction theory [11], whereby the bidders in the Myerson's auction problem correspond to paths in the LPP formulation. The major difference is that in Myerson's problem, the bidders are independent individuals. No matter how a bidder changes its bid, it cannot change other bidders' bids. However, in our LPP problem, paths need not be node-disjoint. As a result, when a node that is shared by multiple paths changes its claimed cost, the claimed cost of all the involved paths will also be changed. In other words, the players in our problem are no longer independent. Therefore, the results in Myerson's work cannot be directly applied to our problem.

The remainder of this paper is organized as follows. We review the related work in

Section 2. We formulate the LPP problem in Section 3. The problem is addressed without and with a node-capacity constraint in Sections 4 and 5, respectively. Simulation results are presented in Section 6. We conclude our work in Section 7.

2 Related Works

Reference [6] is probably the most relevant work to the first part of our work, i.e., LPP without a node capacity constraint. In [6], the authors studied the minimization of the expected price of a *single-path* route. Our results in Section 4 is compatible with theirs. The major difference is that, by definition, their optimization only targets single-path routes, but our formulation starts from a more general setting that allows for multi-path routing. This change in formulation is nontrivial, because now we need to explicitly account for the inter-dependence between paths that traverse through common nodes. We then rigidly prove that in the absence of capacity constraints, the LPP route only contains a single path. Our major contribution is to prove that the algorithm under development is not only optimal among the set of single paths, but also optimal among all possible combinations of paths. As to the second part of our work, i.e., LPP with a node capacity constraint, to the best of our knowledge, ours is the first that formulates and addresses this problem.

Aside from [6], other related works on truthful routing have focused on the LCP problem. This problem was first introduced by Nisan and Ronen in their seminal paper [12], where they solved the truthful unicast LCP routing by applying the celebrated VCG mechanism. In [12], the cost of an agent (a node or an edge) is used as the agent's weight in the graph. The LCP is the shortest path from the source to the destination. The payment p^e to an agent e is 0 if e is not in the LCP and $p^e = d_{G|e=\infty} - d_{G|e=0}$ if it is. Here $d_{G|e=\infty}$ is the length of the shortest path that does not contain e, and $d_{G|e=0}$ is the length of the shortest path when the cost of e is zero.

Based on the model developed in [12], a large body of follow-up works extended the basic VCG algorithm into various networking environments. This includes the ad hoc-VCG in [1], the VCG-based BGP (Border Gateway Protocol) in [7], the multicast version of VCG

in [17], and more recently for opportunistic routing [18]. The work in [10] formulated the multi-path LCP problem but did not provide a solution. The work in [14] gave initial results to this problem by only considering the special case that all paths in the graph are disjoint. Some works, e.g., [8], focused on the complexity issues of the VCG payment calculation. Other works, e.g., [21][15][16], went beyond the routing layer and encompassed a cross-layer methodology in studying the LCP problem.

The high overpayment issue in VCG-based LCP routing was first noticed by Archer and Tardos [2]. They investigated the *frugal path* problem (FPP), which aims at designing a mechanism that selects a path and induces truthful cost revelation, but without paying high price. The work in [2] primarily contributed negative results for the FPP by showing that no reasonable mechanism can always avoid paying a high premium to induce truthtelling. Subsequent works on FPP, e.g., [7] [5], focused on characterizing the bounds for the price of general truthful routing mechanisms. Rather than studying the bounds, the LPP problem in this paper differs from FPP in that it explicitly minimizes the price of the route in a given graph.

3 System Model and Problem Formulation

3.1 Preliminaries

The LPP problem is well suited for analysis by means of *mechanism design*, a branch of game theory. First, we briefly review a few definitions and concepts from mechanism design. We then describe our model and formulate the problem using a mechanism design's terminology.

A mechanism design problem considers a game where there are n agents, each with its own strategy set. For each agent i, $1 \leq i \leq n$, there is some private information t_i (only known to agent i), called its *type*. We consider the *direct revelation* strategy set in this study, i.e., agents simultaneously report their types to the mechanism. Denote the reported type of agent i by $\tilde{t}_i$ (this may not be the same as the true type t_i) and the vector of reported types from all agents as $\tilde{t} = (\tilde{t}_1, \ldots, \tilde{t}_n)$. The mechanism takes as input $\tilde{t}$, and then computes an output $X(\tilde{t})$ and a payment to agents $p(\tilde{t}) = (p_1(\tilde{t}), \ldots, p_n(\tilde{t}))$. Given the output X and

the type t_i, agent i's *valuation* is decided by a real-valued function $v_i(t_i, X)$. Accordingly, agent i's profit for reporting $\tilde{t}_i$ is given by $u_i \stackrel{\text{def}}{=} p_i(\tilde{t}) - v_i(t_i, X)$. An agent is defined as being rational if it always reports a $\tilde{t}_i$ that maximizes its profit.

The main task of a mechanism design problem is to design the functions $X(\tilde{t})$ and $p(\tilde{t})$ that maximize some social interests (e.g., social welfare) of the game. Usually the following properties are desired in the mechanism:

1. Incentive compatible (IC): A mechanism is IC if each rational agent maximizes its profit by reporting its true type t_i.

2. Individual rational (IR): A mechanism is IR if each agent's profit for participating in the game is nonnegative.

When a mechanism is both IC and IR, we say it is truthful (or equivalently, strategy-proof).

3.2 Network Model and Problem Formulation

We consider an OSA network whose topology is defined by a directional graph $G = (V, E)$, where V and E are the set of SR nodes and the set of directional links between SRs, respectively. For node j in V, it can access a certain amount of spectrum, say b_j, by paying a cost c_j to the primary spectrum owner for each packet it transmits over the spectrum. This cost is due to, for example, a winning bid in a spectrum auction or an agreed rent in a spectrum lease, and thus is considered a private information (i.e., the type) only known to node j. Because this cost depends on the supply-demand dynamics of available spectrum around node j, c_j is modeled as a random variable. Here we stick to the conventional economics approach of Bayesian optimal mechanism design by assuming that the probability density function (p.d.f.) of c_j, say f_j, is known to the mechanism. We denote the domain of the p.d.f. f_j as $D_j \stackrel{\text{def}}{=} [v_j, w_j]$, where $v_j \geq 0$, $w_j \geq 0$, and $v_j \leq w_j$. In practice, for a truthful mechanism, f_j and D_j can be pragmatically constructed based on historical reports of the cost c_j. In our analysis, we assume perfect knowledge of f_j, but we relax this condition in

our simulations to study the sensitivity issue of our algorithm. We assume that c_j does not change during a session, but may change from one session to another. Here, a session represents the continuous transmission of a burst of packets. For two nodes j and k, f_j and f_k are independent but not necessarily identical. This assumption captures the basic fact that in an OSA system, different nodes may experience heterogeneous spectrum availabilities, and thus their costs are stochastically non-identical. For example, a node in a "hot" zone (a zone of tight spectrum supply-and-demand) has higher mean cost than nodes outside that zone. We also assume that the interference issue has been taken care of by the spectrum allocation mechanism employed in the network, such that interfering nodes are getting different channels in the spectrum. This assumption is true for nearly all spectrum auction and leasing mechanisms, e.g., the algorithm in [22] takes as input an interference-graph of the nodes to avoid selling a channel to two nodes that interferes with each other. As a result, interference is not an issue at the routing layer. In addition, even though a node is only allowed to transmit over channels it has acquired, it can tune to any channel for reception.

Now consider a traffic flow that originates from source node s and terminates at destination node d. The flow lasts for multiple sessions. Let $n \stackrel{\text{def}}{=} |V - \{s, d\}|$ ($|.|$ is the cardinality of the set), $m \stackrel{\text{def}}{=} |E|$, and label the nodes in $V - \{s, d\}$ as $j = 1, \ldots, n$. Let the set of all paths from s to d be $\mathbf{R}_{sd}$, where each path consists of nodes and links defined in G. Let $N \stackrel{\text{def}}{=} |\mathbf{R}_{sd}|$. Note that the enumeration of paths from s to d is only required for the problem formulation. Our final algorithms do not have such a requirement. The routing mechanism operates on a session-by-session basis. At the beginning of a session, each node j in V reports its cost to the routing mechanism as $\tilde{c}_j$. A truthful mechanism should guarantee that each node reports its true cost $\tilde{c}_j = c_j$, i.e., the mechanism must satisfy the IC and IR constraints. We will formulate these constraints after we finish describing the general operation of the mechanism.

The mechanism takes as input the costs $c = (c_1, \ldots, c_n)$ from the nodes in $V - \{s, d\}$, and computes an outcome consisting of a route selection vector $X = (x_1, \ldots, x_N)$ and a payment vector $p = (p_1, \ldots, p_n)$. In general, we allow multi-path routing. So an element in X, say x_i,

$1 \leq i \leq N$, represents the fraction of traffic that will be carried over path i in $\mathbf{R}_{sd}$ during the current session, and p_j, $1 \leq j \leq n$, is the payment to node j for every packet delivered by paths in $\mathbf{R}_{sd}$. Because x_i and p_j are outputs of the mechanism, we write them as functions of the costs, i.e., $x_i \stackrel{\text{def}}{=} x_i(c)$ and $p_j \stackrel{\text{def}}{=} p_j(c)$. To simplify the presentation, in our subsequent formulation we use the following convenient notations: $c_{-j} \stackrel{\text{def}}{=} (c_1, \ldots, c_{j-1}, c_{j+1}, \ldots, c_n)$, $c \stackrel{\text{def}}{=} (c_j, c_{-j})$, $D_{-j} \stackrel{\text{def}}{=} \bigcup_{1 \leq k \leq n, k \neq j} D_k$, $D \stackrel{\text{def}}{=} D_j \bigcup D_{-j}$, $f_{-j}(c_{-j}) \stackrel{\text{def}}{=} \prod_{1 \leq k \leq n, k \neq j} f_k(c_k)$, and $f(c) \stackrel{\text{def}}{=} f_j(c_j) f_{-j}(c_{-j}) \stackrel{\text{def}}{=} \prod_{1 \leq k \leq n} f_k(c_k)$.

The truthful LPP routing mechanism is formulated as follows. The optimization objective is to minimize the expected price that s needs to pay for each packet it sends in a session, i.e.,

$$\text{minimize}_{(X,p)} \quad U_s(X,p) = \int_D \sum_{j \in V} p_j(c) f(c) dc. \tag{1}$$

This objective is also in line with the minimization of the total payment of s over all sessions for a long-lasting flow. Our truthful routing mechanism is subject to the following constraints:

Incentive Compatibility Constraint: For each packet delivered by the route in the underlying session, the expected profit of node j for reporting cost $\tilde{c}_j$ is given by

$$\int_{D_{-j}} \left(p_j(\tilde{c}_j, c_{-j}) - c_j \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(\tilde{c}_j, c_{-j}) \right) f_{-j}(c_{-j}) dc_{-j} \tag{2}$$

where $\mathbf{R}_{sd}^{(j)}$ is the subset of paths in $\mathbf{R}_{sd}$ that traverse node j. At the same time, the expected profit of node j for reporting its true cost c_j is given by

$$U_j(X,p,c_j) = \int_{D_{-j}} \left(p_j(c) - c_j \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c) \right) f_{-j}(c_{-j}) dc_{-j} \tag{3}$$

The IC constraint requires that

$$U_j(X,p,c_j) \geq \int_{D_{-j}} \left(p_j(\tilde{c}_j, c_{-j}) - c_j \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(\tilde{c}_j, c_{-j}) \right) f_{-j}(c_{-j}) dc_{-j}, \quad \forall \tilde{c}_j \geq 0 \tag{4}$$

Individual Rationality Constraint: This constraint requires that node j participates in the relay only when its expected profit is nonnegative, i.e.,

$$U_j(X, p, c_j) \geq 0 \tag{5}$$

Multi-path Constraint: This constraint says that the sum of the fractions of traffic carried over various paths must equal to the total traffic volume, i.e.,

$$\sum_{i=1}^{N} x_i(c) = 1. \tag{6}$$

Node Capacity Constraint: The aggregate traffic rate of the sub-flows that traverse the same node should not exceed the node's capacity, i.e.,

$$\sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c) R \leq b_j, \forall j \in V \tag{7}$$

where R is the flow rate demand. Note that in our formulation, the node cost is defined for each packet relayed by the node, whereas the payment, the price, and the profit are with respect to each packet delivered over the (multi-path) route. Hereafter, we use the notation i and j to refer to a path and a node, respectively. We follow these norms in all our subsequent sections, unless indicated otherwise.

4 LPP Without a Node-Capacity Constraint

The main difficulty in solving the LPP problem is that the feasible paths between nodes s and d are not node-disjoint. Therefore, the traffic carried by various nodes is not independent. This inter-dependence appears in the IC (4) and the node-capacity constraint (7), preventing us from directly using the Myerson's optimal auction theory [11]. In this section, we first consider a simplified version of the LPP formulation, by ignoring the node-capacity constraint in (7). The resulting mechanism will still be truthful, because the IC and IR constraints are still being accounted for.

Our analysis of the simplified problem proceeds as follows. We first try to simplify the IC constraint (4). For node j, the fraction of traffic it expects to relay given that its cost is

c_j can be calculated as

$$Q_j(c_j) \stackrel{\text{def}}{=} \int_{D_{-j}} \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c_j, c_{-j}) f_{-j}(c_{-j}) dc_{-j} \tag{8}$$

Based on of $Q_j(c_j)$, we have the following lemma.

Lemma 1: The IC constraint in (4) is equivalent to the following two conditions:

(1) Monotonicity: If $c_j^{(1)} \geq c_j^{(2)}$, then $Q_j(c_j^{(1)}) \leq Q_j(c_j^{(2)})$;

(2) $U_j(X, p, c_j) = \int_{c_j}^{w_j} Q_j(\tau_j) d\tau_j + U_j(X, p, w_j)$

Proof: Let's first assume that for node j, $c_j^{(1)}$ is its true cost, but node j reports $c_j^{(2)}$. Then the expected profit for this situation is calculated as follows

$$\begin{aligned}
& \int_{D_{-j}} \left[p_j(c_j^{(2)}, c_{-j}) - c_j^{(1)} \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c_j^{(2)}, c_{-j}) \right] f_{-j}(c_{-j}) dc_{-j} \\
= & \int_{D_{-j}} \left[p_j(c_j^{(2)}, c_{-j}) - (c_j^{(1)} + c_j^{(2)} - c_j^{(2)}) \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c_j^{(2)}, c_{-j}) \right] f_{-j}(c_{-j}) dc_{-j} \\
= & \int_{D_{-j}} \left[p_j(c_j^{(2)}, c_{-j}) - c_j^{(2)} \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c_j^{(2)}, c_{-j}) \right] f_{-j}(c_{-j}) dc_{-j} \\
& - \int_{D_{-j}} (c_j^{(1)} - c_j^{(2)}) \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c_j^{(2)}, c_{-j}) f_{-j}(c_{-j}) dc_{-j} \\
= & U_j(X, p, c_j^{(2)}) - (c_j^{(1)} - c_j^{(2)}) Q_j(c_j^{(2)})
\end{aligned} \tag{9}$$

The incentive compatibility constraint dictates that

$$U_j(X, p, c_j^{(1)}) \geq U_j(X, p, c_j^{(2)}) - (c_j^{(1)} - c_j^{(2)}) Q_j(c_j^{(2)}) \tag{10}$$

Follow the same treatment, for the situation that $c_j^{(2)}$ is true cost of node j, but $c_j^{(1)}$ is reported, we get

$$U_j(X, p, c_j^{(2)}) \geq U_j(X, p, c_j^{(1)}) - (c_j^{(2)} - c_j^{(1)}) Q_j(c_j^{(1)}) \tag{11}$$

Combining (10) and (11), we derive

$$(c_j^{(2)} - c_j^{(1)}) Q_j(c_j^{(2)}) \leq U_j(X, p, c_j^{(1)}) - U_j(X, p, c_j^{(2)}) \leq (c_j^{(2)} - c_j^{(1)}) Q_j(c_j^{(1)}) \tag{12}$$

This means $(c_j^{(2)} - c_j^{(1)})Q_j(c_j^{(2)}) \leq (c_j^{(2)} - c_j^{(1)})Q_j(c_j^{(1)})$. So condition (1) of Lemma 1 follows.

Note that the inequalities in (12) can be rewritten for any $\delta > 0$ as

$$\delta Q_j(c_j^{(2)}) \leq U_j(X, p, c_j^{(2)} - \delta) - U_j(X, p, c_j^{(2)}) \leq \delta Q_j(c_j^{(2)} - \delta) \tag{13}$$

Since $Q_j(c_j)$ is decreasing in c_j, it is Riemann integrable []. So

$$\int_{c_j}^{w_j} Q_j(\tau_j)d\tau_j = U_j(X, p, c_j) - U_j(X, p, w_j) \tag{14}$$

This proves condition 2 of Lemma 1. ■

Based on Lemma 1, the simplified LPP problem can be re-formulated as follows.

Lemma 2: The following formulation is equivalent to the simplified LPP problem defined in Section 3: the optimal route selection vector X is the solution to the following optimization problem

$$\begin{array}{rl} \text{minimize} & \sum_{1\leq j\leq n} \int_D \left[c_j + \frac{F_j(c_j)}{f_j(c_j)}\right] \sum_{i\in\mathbf{R}_{sd}^{(j)}} x_i(c) f(c) dc \\ s.t. & \\ & \sum_{i=1}^{N} x_i(c) = 1 \end{array} \tag{15}$$

where $F_j(c_j) = \int_{v_j}^{c_j} f_j(\tau_j)d\tau_j$ is the c.d.f. of c_j. In addition, the truthful payment to each node is given by $p_j(c) = c_j \sum_{i\in\mathbf{R}_{sd}^{(j)}} x_i(c) + \int_{c_j}^{w_j} \sum_{i\in\mathbf{R}_{sd}^{(j)}} x_i(\tau_j, c_{-j})d\tau_j$, for $1 \leq j \leq n$.

Proof: The objective function in (1) can be rewritten as follows

$$U_s(X, p) = \sum_{1\leq j\leq n} \int_D c_j \sum_{i\in\mathbf{R}_{sd}^{(j)}} x_i(c) f(c) dc + \sum_{1\leq j\leq n} \int_D \left[p_j(c) - c_j \sum_{i\in\mathbf{R}_{sd}^{(j)}} x_i(c)\right] f(c) dc \tag{16}$$

The second term in (16) can be written as

$$\begin{aligned} & \int_D \left[p_j(c) - c_j \sum_{i\in\mathbf{R}_{sd}^{(j)}} x_i(c)\right] f(c) dc \\ = & \int_{v_j}^{w_j} U_j(X, p, c_j) f_j(c_j) dc_j \end{aligned} \tag{17}$$

Substituting condition (2) of Lemma 1 into (17), we get

$$\begin{aligned}
&\int_D \left[p_j(c) - c_j \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c) \right] f(c) dc \\
= \;& U_j(X, p, w_j) + \int_{v_j}^{w_j} \int_{c_j}^{w_j} Q_j(\tau_j) d\tau_j f_j(c_j) dc_j \\
= \;& U_j(X, p, w_j) + \int_{v_j}^{w_j} \int_{v_j}^{\tau_j} f_j(c_j) dc_j Q_j(\tau_j) d\tau_j \\
= \;& U_j(X, p, w_j) + \int_{v_j}^{w_j} F_j(c_j) Q_j(c_j) dc_j \\
= \;& U_j(X, p, w_j) + \int_D F_j(c_j) \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c) f_{-j}(c_{-j}) dc \qquad (18)
\end{aligned}$$

Substituting (18) into (16), after some mathematical manipulation, the objective function of (1) can be rewritten as

$$U_s(X, p) = \sum_{1 \le j \le n} \int_D \left[c_j + \frac{F_j(c_j)}{f_j(c_j)} \right] \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c) f(c) dc + \sum_{1 \le j \le n} U_j(X, p, w_j) \qquad (19)$$

It is easy to show that when the payment is $p_j(c) = c_j \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c) + \int_{c_j}^{w_j} \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(\tau_j, c_{-j}) d\tau_j$, for $1 \le j \le n$, the second term in (19), i.e., $U_j(X, p, w_j)$, equals 0. According to the Individual Rationality constraint, $U_j \ge 0$. Therefore, the payment method in Lemma 2 minimizes $U_j(X, p, w_j)$. As a result, the objective of minimizing $U_s(X, p)$ is reduced to minimizing the new objective function $\sum_{1 \le j \le n} \int_D \left[c_j + \frac{F_j(c_j)}{f_j(t_j)} \right] \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c) f(c) dc$. Note that the IU and IR constraints have been reflected in the derivation of the new objective function. It is easy to verify that the payment function always satisfies the IR constraint, because $p_j(c) \ge c_j \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c)$. The only constraint which has not been reflected is the multi-path constraint, which we append to the new optimization formulation. This proves Lemma 2.

■

According to Lemma 2, we can resort to the new formulation in (15) to find the solution to the simplified LPP problem. Denote $h(c) \stackrel{\text{def}}{=} \sum_{1 \le j \le n} \left[c_j + \frac{F_j(c_j)}{f_j(c_j)} \right] \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c)$. An inspection of (15) shows that this formulation actually minimizes the expected value of $h(c)$ given the distribution of $f(c)$. A simple probabilistic argument says that the expected value

will be minimized if $h(c)$ is minimized over every c. This gives rise to the following lemma, which reduces our problem from a randomized setup to a deterministic one.

Lemma 3: Given the cost vector c, the LPP route selection is given by the following optimization problem:

$$\begin{array}{rl} \text{minimize} & h(c) = \sum_{1\le j\le n}\left[c_j + \frac{F_j(c_j)}{f_j(c_j)}\right]\sum_{i\in\mathbf{R}_{sd}^{(j)}} x_i(c) \\ s.t. & \\ & \sum_{i=1}^{N} x_i(c) = 1 \end{array} \tag{20}$$

and the truthful payment is given by

$$p_j(c) = c_j \sum_{i\in\mathbf{R}_{sd}^{(j)}} x_i(c) + \int_{c_j}^{w_j} \sum_{i\in\mathbf{R}_{sd}^{(j)}} x_i(\tau_j, c_{-j})d\tau_j, \quad 1\le j\le n. \tag{21}$$

Regarding the solution to (20), we have the following results.

Theorem 1: For the simplified LPP problem, for each c, the optimal route only contains one path and is given by $R_{sd}^o(c) = \arg\min_{\forall R_{sd}^{(i)}\in\mathbf{R}_{sd}}(\sum_{j\in R_{sd}^{(i)}}\left[c_j + \frac{F_j(c_j)}{f_j(c_j)}\right])$, where $R_{sd}^{(i)}$ denotes the ith path in $\mathbf{R}_{sd}$, $1\le i\le N$. The optimal route selection vector $X^o(c) = (0,\ldots,0,1,0,\ldots,0)$, where the non-zero element corresponds to the optimal path R_{sd}^o.

Proof: We rewrite $h(c)$ according to the routes in $\mathbf{R}_{sd}$, i.e.,

$$h(c) = \sum_{1\le i\le N}\left(\sum_{j\in R_{sd}^{(i)}}\left[c_j + \frac{F_j(c_j)}{f_j(c_j)}\right]\right)x_i(c). \tag{22}$$

Denote $\xi_j \stackrel{\text{def}}{=} \left[c_j + \frac{F_j(c_j)}{f_j(c_j)}\right]$. Note that for a given c, ξ_j is a constant for all $1\le j\le n$. Accordingly, for each path i, the term $W_i \stackrel{\text{def}}{=} \left(\sum_{j\in R_{sd}^{(i)}}\xi_j\right)$ is also a constant. Therefore, for a given c, problem (20) becomes a linear program (LP) of the form:

$$\begin{array}{rl} \text{minimize} & \sum_{i=1}^{N} W_i x_i \\ s.t. & \\ & \sum_{i=1}^{N} x_i = 1 \end{array} \tag{23}$$

It is straightforward to see that the optimal solution to the above LP is $X^o = (0,\ldots,0,1,0,\ldots,0)$, where the index of the non-zero element is $i^o = \arg\min_{1\le i\le N}(W_i)$. This proves Theorem 1.

■

Theorem 1 suggests the following algorithm for computing the LPP and the related payments: For a given c, we use $\xi_j(c_j)$ as the weight of node j. The LPP is simply the

shortest path from s to d w.r.t. weight ξ_j's. Since ξ_j is a function of the cost c_j, it may also be considered as the *virtual cost* of node j. Because now the optimal route contains only one path, the payment can be largely simplified: Equation (21) shows that $p_j(c) = 0$ if node j is not included in the shortest path. Otherwise, $p_j(c) = \min(w_j, \bar{c}_j)$, where $\bar{c}_j$ is the cutoff cost of node j, beyond which node j will not be included in the LPP. This cutoff cost can be computed in the virtual cost domain. Specifically, in the virtual cost domain, denote the difference of length between the shortest path that traverses node j and the shortest path that does not traverse node j as ζ. If the virtual cost of node j grows from $\xi_j(c_j)$ to $\xi_j(c_j)+\zeta$, node j will not be included in the LPP. So $\bar{c}_j = \xi_j^{-1}(\zeta + \xi_j(c_j))$, where ξ_j^{-1} denotes the inverse of function ξ_j. The pseudo-code description of the above process is given in Algorithm 1.

Algorithm 1 Computing LPP and Payments without a Node-Capacity Constraint

Require: $c = (c_1, \ldots, c_n)$
Ensure: LPP and *payments* to nodes
1: **for** $j = 1$ to n **do**
2: $\quad \xi_j \Leftarrow c_j + \frac{F_j(c_j)}{f_j(c_j)}$
3: **end for**
4: $LPP \Leftarrow$shortest path from s to d w.r.t. weight ξ_j's
5: $minlength \Leftarrow$ length of(LPP)
6: $payments(j) \Leftarrow 0, \quad \forall j \notin LPP$
7: **for all** nodes j in LPP **do**
8: $\quad length\_runnerup(j) \Leftarrow$ length of the shortest path that does not traverse node j
9: $\quad virtual\_cutoff\_cost(j) \Leftarrow length\_runnerup(j) - minlength$
10: $\quad payments(j) \Leftarrow \min(\xi_j^{-1}(virtual\_cutoff\_cost(j)), w_j)$
11: **end for**

Theorem 2: The route selection and payment mechanism given in Algorithm 1 is truthful and minimizes the expected price of the resulting route.

Proof: The proof is straightforward based on our previous discussion. ∎

In Algorithm 1, the computation of LPP (line 4) involves finding a shortest path. Because all node weights are non-negative, this can be done using the Dijkstra's algorithm in time $\mathcal{O}(m + n \log n)$. It is easy to see that overall, the running time of Algorithm 1 is $\mathcal{O}(mn + n^2 \log n)$.

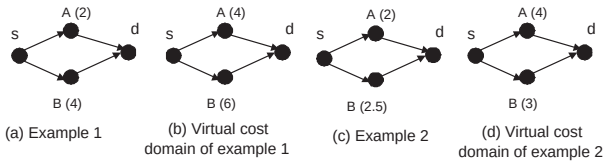


Figure 1: Toy examples of LPP without node-capacity constraint (the cost and virtual cost of a node are indicated in the bracket beside the node).

4.1 Examples

To better understand how Algorithm 1 operates, consider the example in Figure 1. Two alternative paths are available from s to d, via either node A or B. Suppose the true costs c_A and c_B are uniformly distributed over the domains $[0, 10]$ and $[2, 10]$, respectively. So their virtual costs are given by $\xi_A(c_A) = 2c_A$ and $\xi_B(c_B) = 2c_B - 2$, respectively. We first consider the situation in sub-figure (a), where $c_A = 2$ and $c_B = 4$ in the underlying session. Their virtual costs are then computed as $\xi_A = 4$ and $\xi_B = 6$, as shown in sub-figure (b). Algorithm 1 chooses the path $s \to A \to d$ as the LPP, and computes the virtual-cutoff-cost of node A as 6, which is then mapped to the actual cutoff-cost of node A according to the inverse function $\xi_A^{-1}(y) = 0.5y$, giving $\bar{c}_A = 3$. This is smaller than $w_A = 10$, so node A is paid 3 and the price for the LPP $s \to A \to d$ is 3. In contrast, even though the LCP computed according to the VCG algorithm is also $s \to A \to d$, its price is 4, which is 33% higher than the price under LPP.

Note that the LCP may not always be a LPP. Consider the situation in sub-figure (c), where $c_A = 2$ and $c_B = 2.5$. Following the same procedure of the previous example, Algorithm 1 chooses $s \to B \to d$ as the LPP, with a price of 3.25. The VCG algorithm will choose $s \to A \to d$ as the LCP, with a price of 2.5. So here LPP is different from LCP. Note that the higher price paid by the LPP in this example does not contradict with the optimization objective that the LPP's expected (or average) price is minimized. The higher price paid by LPP in some situations is to maintain its truthfulness in expectation.

5 LPP with A Node-capacity Constraint

In this section, we study the full-fledged LPP problem where node-capacity constraint is included in the formulation. Such a constraint is needed when the source rate demand is greater than the minimum node capacity in the OSA network.

5.1 Optimal Route Selection

Based on Lemma 3 and Theorem 1, the inclusion of the node-capacity constraint leads to the following formulation for the optimal route selection:

$$\begin{array}{rl} \text{minimize}_{(x_1,\ldots,x_N)} & \sum_{i=1}^{N} W_i x_i \\ s.t. & \\ & \sum_{i=1}^{N} x_i = 1 \\ & R\sum_{i\in\mathbf{R}_{sd}^{(j)}} x_i \le b_j, \forall j \in V \end{array} \tag{24}$$

An inspection of (24) shows that this is a LP. However, the main challenge in solving this problem using a conventional LP solver is that, in practice it is difficult, if not impossible, to enumerate all the paths from s to d. So it is difficult to explicitly express the formulation (24). This fact prevents any existing LP solver from working, because an explicit formulation of the problem is a pre-requisite for these solvers to construct and navigate through the space of feasible solutions. Here, we follow a different method to solve (24). We first investigate the structure of the optimal solution to (24), given the existence of such a solution. We then design algorithms that directly construct the optimal solution, even though in practice the explicit expression of the problem is hard to write down.

Theorem 3: The following greedy algorithm finds an optimal solution to (24), given the existence of such a solution.

Proof: Note that (24) belongs to the category of min-cost flow problem, but with non-negative nodal weights/capacity constraints, and two sub-flows in oppositive directions do not cancel. Algorithm 2 is actually a modification of the Ford-Fulkerson algorithm [4], in which the search for an augmenting path is done by finding the shortest path in terms of the nodal weight ξ_j's. The optimality of the algorithm can be proved as follows.

Label the paths in $\mathbf{R}_{sd}$ according to the ascending order of their length W_i's. For path

Algorithm 2 Computing LPP Under a Node-Capacity Constraint

Require: $c = (c_1, \ldots, c_n)$
Ensure: LPP
1: **for** $j = 1$ to n **do**
2: $\quad \xi_j \Leftarrow c_j + \frac{F_j(c_j)}{f_j(c_j)}$
3: **end for**
4: STEP 1: Find the shortest-path from s to d in graph G w.r.t. weight ξ_j's.
5: STEP 2: Find the bottleneck node j^* on this path that has the minimum capacity. Allocate a fraction of traffic, $b(j^*)/R$, from the flow to this path. Update the capacity of each node on this path by subtracting $b(j^*)$, the portion that has been used by the underlying shortest path, from its original capacity. Delete from G the bottleneck node j^* along with other possible bottleneck nodes whose capacities become 0.
6: STEP 3: Repeat STEPs 1 and 2 until all traffic of the flow has been allocated or no additional shortest-path in STEP 1 can be found. In the former case, the sequence of paths found by the procedure along with the associated traffic allocation is an optimal solution to (24). If the latter case happens, it means that an optimal solution to (24) does not exist.

1, assign $x_1^o = \frac{b(j^*)}{R}$, where j^* is the bottleneck node in path 1. Update the residual capacity of the nodes in path 1 by decrementing by $b(j^*)$. For path 2, assign $x_2^o = \frac{b(j'^*)}{R}$, where j'^* is the bottleneck node (the one with the minimum residual capacity) in path 2. Update the residual capacity of the nodes in path 2 by decrementing by $b(j'^*)$, and so on, until all traffic of the source has been allocated. After that, $x_i^o = 0$. Denote the resulting traffic allocation by $X^o = (x_1^o, \ldots, x_N^o)$. Note that X^o is a feasible, since it satisfies both constraints in (24).

Now, we need to prove that for any different feasible traffic allocation $X' = (x'_1, \ldots, x'_N)$, $\sum_{i=1}^N W_i x_i^o \leq \sum_{i=1}^N W_i x'_i$ is true. This can be proved as follows. Because $X' \neq X^o$, denote the index of their first different elements by k, i.e., $x'_k \neq x_k^o$, but $x'_i = x_i^o$ for $1 \leq i \leq k-1$. Also denote the bottleneck node for path k in X^o as node $j_{(k)^*}$. Because x_k^o is the largest feasible traffic assignment given the previous assignments of x_1^o to x_{k-1}^o, we must have $x_k^o > x'_k$. Denote the difference by $\epsilon = x_k^o - x'_k$. Due to the multi-path constraint, we must have $\sum_{i=k+1}^N x'_i - \sum_{i=k+1}^N x_i^o = \epsilon$. Now, consider the following re-assignment of traffic from x'_i's ($i \neq k+1$) to x'_k. The total volume of re-assigned traffic is ϵ, making the resulting $x''_k = x'_k + \epsilon = x_k^o$. This traffic reassignment involves cutting the traffic for x'_i's ($i \neq k+1$) and transfer the traffic being cut to x'_k. The first operation, i.e., cutting the traffic, only leads to the decrease of traffic carried by each involved node, thus it does not

violate the capacity constraint at any node. The second operation, i.e., adding the traffic to path k, may lead to the violation of the capacity constraint at some nodes in path k. This is resolved by carefully selecting those paths whose traffic will be cut. Specifically, if the node whose capacity is being violated, say node q, is not node $j_{(k)^*}$, there must be some paths l, where $l \geq k+1$, that also traverse node q and $x'_q > 0$. In addition, the total traffic volume carried by such paths must be greater than ϵ, otherwise node q has to be bottleneck node for path k in X^o, which contradicts the premise that node q is not the bottleneck node. As a result, we can cut the traffic of these paths and move these traffic to x'_k. Because these paths share node q with path k, such a traffic re-assignment does not change the traffic volume carried by q, and thus eliminate the possibility of violating the capacity constraint of node q. When the above traffic re-assignment finishes, node $j_{(k)^*}$ is the only node in path k whose capacity constraint is being reached. Denote the traffic allocation after the above adjustment by $X'' = (x''_1, \ldots, x''_N) = (x'_1, \ldots, x'_{k-1}, x'_k + \epsilon, x''_{k+1}, \ldots, x''_N)$, where $\sum_{i=k+1}^{N} x'_i - \sum_{i=k+1}^{N} x''_i = \epsilon$. Note that X'' has eliminated the difference between the elements x^o_k and x'_k. In addition, it is also a feasible solution to (24) because it satisfies both constraints. Clearly, $\sum_{i=1}^{N} W_i x''_i \leq \sum_{i=1}^{N} W_i x'_i$. We now repeat the above traffic-adjustment process over X'' to eliminate its first different element from X^o, which must be after element k. Note that the objective function under the new allocation is mono-decreasing. Finally, the above process must leads to X^o, yielding $\sum_{i=1}^{N} W_i x'i \geq \sum_{i=1}^{N} W_i x''_i \geq \ldots \geq \sum_{i=1}^{N} W_i x^o_i$. So $\sum_{i=1}^{N} W_i x^o_i \leq \sum_{i=1}^{N} W_i x'_i$. This proves the optimality of X^o.

Algorithm 2 gives the same route allocation to the process described in this proof. The only difference is that Algorithm 2 directly precludes those paths that have contained at least one bottleneck node from the subsequent traffic allocation, and thus is more computationally efficient. ■

The execution of Algorithm 2 involves repetitive computation of several shortest paths. Because all weights are non-negative, Dijkstra's algorithm can be used to find the shortest path in each iteration in $\mathcal{O}(m + n \log n)$ time. It is straightforward to see that the total running time of Algorithm 2 is $\mathcal{O}(mn + n^2 \log n)$.

5.2 Payment Calculation

For the LPP with a node-capacity constraint, the general form of the optimal payment defined in (21) still applies. This is because this general equation is solely decided by the IC and IR constraints, and is not related to whether the node-capacity constraint is present in the formulation. The key challenge in calculating this payment is in computing the integration term in the equation. Note that in Algorithm 2, x_i^o is given only as an implicit function of c_j, so it is not suitable to be used directly in the integration. To calculate the payment, we need to take a closer look at the relationship between node j's traffic and its cost c_j.

A re-examination of Algorithm 2 and its proof indicates that the traffic carried by a node j presents a multi-threshold structure in relation to c_j. More specifically, for a node j that is included in the LPP route, the traffic it carries is the sum of the traffic carried by the paths that are part of this route and that also traverse node j. Denote these set of paths by $\mathbf{R}_{sd}^{(j)o}$. Algorithm 2 dictates that, given the capacity of each node, the optimal traffic allocation across various paths, i.e., $(x_1^o, \ldots, x_N^o)$, is fully decided by the ranking of the paths according to their weights. As a result, when c_j is increased, the traffic allocation to the paths in $\mathbf{R}_{sd}^{(j)o}$ will change only when the increment of c_j is significant enough such that the ranking of the paths change. It is not difficult to see that this is a mono-decreasing process, i.e., with the increase in c_j, the amount of traffic allocated to the paths in $\mathbf{R}_{sd}^{(j)o}$ always decreases after each change in ranking, because their ranking only drops with a larger c_j. Finally, no traffic will be allocated to the paths in $\mathbf{R}_{sd}^{(j)o}$, since all traffic has been allocated to the paths ranked before them. At this point, c_j corresponds to the cutoff cost we defined in the simplified LPP problem. The major difference here is that, before c_j reaches the cutoff cost, there exist multiple threshold costs, at which a change in the ranking of the paths happens. Such a multi-threshold structure largely simplifies our calculation of the payment, because the traffic allocation does not change between two consecutive threshold costs. So, to compute the integration in (21), we only need to find out the threshold costs and evaluate the traffic allocation at these particular points.

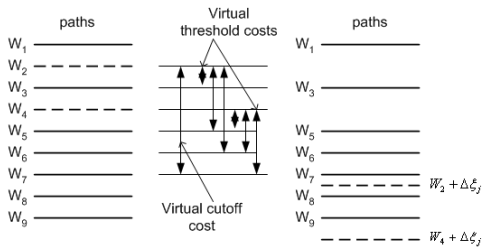


Figure 2: Calculation of the virtual cutoff cost and threshold costs.

We use an example to illustrate the basic idea of computing the threshold costs and cutoff cost for a node j that is included in the LPP. Consider the example shown on the left in Figure 2, where paths in $\mathbf{R}_{sd}$ are labeled in the increasing order of their length W_i, defined based on the nodes' virtual cost. Suppose paths 1 to 5 constitute the multi-path LPP constructed according to Algorithm 2. Suppose paths 2 and 4 traverse through node j. So, with the increase in ξ_j, the rankings of paths 2 and 4 drop simultaneously. Now consider the situation shown to the right, where path 2 has dropped to a position that no traffic will be allocated to it, but non-zero traffic is still allocated to the path ranked before it, say path 7 in this example. Note that at this time no traffic is allocated to path 4, either. The underlying increment in ξ_j, i.e., $\Delta\xi_j = W_7 - W_2$, is the critical cutoff point for node j. If $\Delta\xi_j < W_7 - W_2$, traffic will still be allocated to path 2, so the traffic carried by node j will be non-zero. But if $\Delta\xi_j > W_7 - W_2$, no traffic passes through node j. So, the virtual cost of node j at this point, i.e., $\xi_j + (W_7 - W_2)$, is the node's cutoff virtual cost. A virtual threshold cost is simply the increment in ξ_j that leads to a change in the rankings of paths 2 or 4. As shown in the middle of Figure 2, in total there are 7 such increments: $W_5 - W_4, W_6 - W_4, W_7 - W_4, W_3 - W_2, W_5 - W_2, W_6 - W_2$, and $W_7 - W_2$ (the last corresponds to the cutoff virtual cost). The actual costs of node j can then be obtained from the corresponding virtual costs by the inverse mapping $\xi_j^{(-1)}(.)$.

In the above process, the critical path 7 can be found by excluding node j from the graph G and calling Algorithm 2 to construct another LPP that does not include node j. The longest path in this LPP that has non-zero traffic allocation corresponds to the critical path 7 in this example. Note that there is no need to consider the paths after this critical path, because they will never receive any traffic allocation no matter how big ξ_j

becomes. After getting the threshold costs and the cutoff cost, Algorithm 2 can be called to evaluate the traffic allocations at these particular costs. The above procedure for calculating the payments is formalized in Algorithm 3, which is executed after an LPP is found by Algorithm 2.

Algorithm 3 Payment Calculation Under a Node-capacity Constraint

Require: LPP
Ensure: *payments* to nodes
for $j = 1$ to n **do**
  $\xi_j \Leftarrow c_j + \frac{F_j(c_j)}{f_j(c_j)}$
end for
for all $j \notin LPP$ **do**
  $payments(j) \Leftarrow 0$
end for
for all $j \in LPP$ **do**
  Construct $\mathbf{R}_{sd}^{(j)o}$ from LPP
  Excluding node j from graph G, call Algorithm 2 to find the LPP that does not include node j. $l_j^* \Leftarrow$ the length of the longest path in this LPP that has non-zero traffic allocation
  $\mathbf{R}_{sd}(l_j^*) \Leftarrow \{$paths from s to d whose length is not longer than $l_j^*\} - \mathbf{R}_{sd}^{(j)o}$
  $k \Leftarrow 1$
  for each path $r \in \mathbf{R}_{sd}^{(j)o}$ and each path $u \in \mathbf{R}_{sd}(l_j^*)$ **do**
    threshold virtual cost $\xi_j^{(k)*} \Leftarrow \max\{$length of path $u -$ length of path $r, 0\} + \xi_j$
    threshold cost $c_j^{(k)*} \Leftarrow \min(\xi_j^{(-1)}(\xi_j^{(k)*}), w_j)$
    Call Algorithm 2 to evaluate the optimal traffic allocation $X(c_j^{(k)*}, c_{-j})$
    $k \Leftarrow k + 1$
  end for
  $K \Leftarrow$ number of threshold costs, including the cutoff cost
  Sort $c_j^{(k)*}$'s according to the increasing order
  $payments(j) \Leftarrow c_j \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c_j, c_{-j}) + \sum_{k=1}^{K-1} \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(c_j^{(k)*}, c_{-j})(c_j^{(k+1)*} - c_j^{(k)*})$
end for

The computation complexity of Algorithm 3 can be calculated as follows. For each node j that is included in the LPP, the algorithm needs to call Algorithm 2 K times to evaluate the traffic allocation under the K threshold costs of node j. So the computation time for calculating the payment for one node is $\mathcal{O}(K(mn + n^2\ logn))$. The computation time for calculating the payments for the entire LPP is $\mathcal{O}(Kmn^2 + Kn^3 \log n)$. In practice, the value of K has a big impact on the overall computational complexity of the algorithm. For each node j, K is at most the product of the number of paths in $\mathbf{R}_{sd}^{(j)o}$ and in $\mathbf{R}_{sd}(l_j^*)$ (defined in

line 10 of Algorithm 3). Intuitively, this implies that if the number of paths included in the LPP is large, then K will be very large. This scenario happens when the rate demand of the source is much larger than the capacity of the nodes, such that the flow has to be split among many sub-flows to get accommodated by each path. When the rate demand of the source is low, our simulation shows that Algorithm 3 can be quickly executed, because of the small number of threshold costs at each node.

5.3 Examples

We illustrate the operation of Algorithms 2 and 3 using the example in Figure 3 (a), where s wants to send a flow of rate 1 to d. The underlying cost and capacity of a node are given next to the node in the form (c_j, b_j), respectively. We assume that the cost of each node is uniformly distributed over $[0, 5]$. So the virtual cost is given by $\xi_j(c_j) = 2c_j$. The virtual cost of each node is shown in sub-figure (b). Algorithm 2 first finds path $s \rightarrow A \rightarrow B \rightarrow C \rightarrow d$ (length=3) as the shortest path. Because B and C are bottleneck nodes, 0.5 of the original flow is allocated to this path. Accordingly, the residual capacity of A is changed to $1 - 0.5 = 0.5$. Excluding nodes B and C, Algorithm 2 next finds path $s \rightarrow A \rightarrow E \rightarrow F \rightarrow d$ (length=4) as the shortest path. Now, A, E, and F are all bottleneck nodes. So 0.5 of the original flow is allocated to this path. Since all traffic of the original flow has been allocated, Algorithm 2 terminates. So the LPP contains two paths with the flow equally distributed among them.

Now consider how Algorithm 3 calculates the payments. We choose the payment to A as an example, because both paths in LPP traverse A, making it the most complicated case among all other nodes. To calculate the payment to A, Algorithm 3 first excludes A from the graph and calls Algorithm 2 to find an alternate LPP that does not traverse A. This leads to the alternate LPP: $s \rightarrow G \rightarrow E \rightarrow F \rightarrow d$ (weight=5, traffic allocated =0.5) and path $s \rightarrow G \rightarrow H \rightarrow I \rightarrow d$ (weight=6, traffic allocated=0.5). The latter is the critical cutoff path. So there are 3 critical increments of ξ_A, namely $5 - 3 = 2$, $6 - 3 = 3$, $5 - 4 = 1$, and $6 - 4 = 2$. For $\Delta\xi_A = 1$, the new virtual cost of A is $1 + 1 = 2$. Beyond

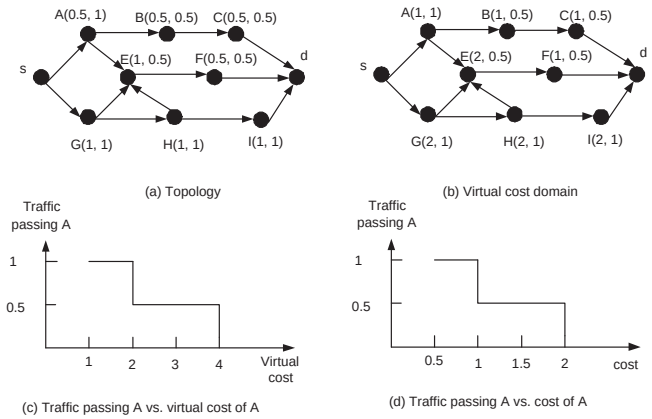


Figure 3: Toy example of LPP with a node-capacity constraint.

this new virtual cost, Algorithm 2 finds the new LPP as path $s \to A \to B \to C \to d$ that carries 0.5 of the flow, and path $s \to G \to E \to F \to d$ that carries 0.5 of the flow. So overall A carries 0.5 of the flow. We repeat the above process and find that when $\Delta\xi_A = 2$, the LPP is path $s \to G \to E \to F \to d$ (traffic=0.5) and path $s \to A \to B \to C \to d$ (traffic=0.5); when $\Delta\xi_A = 3$, the LPP is path $s \to G \to E \to F \to d$ (traffic=0.5) and path $s \to G \to H \to I \to d$ (traffic=0.5). So the traffic of A is 0.5 and 0, respectively, when ξ_A goes beyond 3 and 4. The traffic of A vs. ξ_A is plotted in sub-figure (c). ξ_A is then inversely converted to the actual cost according to $\xi_A^{-1}(y) = 0.5y$, leading to the sub-figure (d). So the payment to A is $1 \times 0.5 + 1 \times (1 - 0.5) + 0.5 \times (2 - 1) = 1.5$.

5.4 Truthful Capacity Reporting

The above LPP route selection and payment calculation procedure is heavily dependent on the capacity constraint b_j at each node. Just like the nodal cost c_j, b_j should also be considered as private information that is only known to node j. As a result, node j may not report its true b_j when reporting its capacity to the routing mechanism, if such cheating makes more profit for node j. For example, node j may exaggerate its capacity by reporting a larger value than b_j. This may lead to a larger fraction of traffic being passed through node j and a higher payment to node j, leading to a possibly higher profit for node j. To maintain the overall truthfulness of the mechanism, the cheating in capacity reporting should be eliminated by design.

We propose the following payment materialization algorithm to guarantee truthful capacity reporting from every node, and thus maintain the overall truthfulness of the routing mechanism. Specifically, at the beginning of the underlying session, based on the reported node cost and capacity, the mechanism computes the LPP route X^o and the payment p^o by using Algorithms 2 and 3, respectively. After this calculation, s starts its transmission by following the route X^o. The actual payment to intermediate nodes, a.k.a. payment materialization, will be deferred until the end of the underlying session. During the transmission, each intermediate node records the traffic volume it has carried during the session. This traffic-recording is either based on the tamper-proof hardware proposed in [3] or on the cryptographic-receipt-based software proposed in [20]. Either way, it ensures that the actual traffic volume carried by the node is honestly recorded. After the session, the mechanism starts the payment materialization process: For node j, we compare the volume of traffic that should be carried by j according to X^o with the actual volume of traffic that has been passed through j. The former quantity is calculated according to X^o as $s_j^o = \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i^o$. The latter one, denoted by s_j^*, comes from the traffic recording of node j. If $s_j^* < s_j^o$, then zero payment will be materialized to node j. Otherwise, the calculated payment p_j^o is paid to node j for every packet sent by s.

Theorem 4: The above payment materialization method is truthful for capacity reporting, i.e., a cheating about node's capacity does not result in more profit for that node.

Proof: Denote the reported capacity of node j by $\tilde{b}_j$. Node j cheats by either reporting $\tilde{b}_j > b_j$ or $\tilde{b}_j < b_j$. Either way, note that the ranking of the paths from s to d in terms of their length does not change with $\tilde{b}_j$. As a result, according to Algorithm 2, the sequence by which the paths are allocated with traffic does not change, but the volume of traffic a path receives may change with $\tilde{b}_j$.

First note that node j does not have an incentive to under-report its capacity, i.e., claiming $\tilde{b}_j < b_j$. The reason is as follows: By Algorithm 2, a smaller $\tilde{b}_j$ can only reduce or maintain, but never increase, the traffic carried by those paths that traverse through node j, i.e., $\sum_{i \in \mathbf{R}_{sd}^{(}j)} x_i(c)$ is non-decreasing with $\tilde{b}_j$. Substituting the payment (21) into (3), the

profit of node j is given by

$$U_j(X, p, c_j) = \int_{D_{-j}} \int_{c_j}^{w_j} \sum_{i \in \mathbf{R}_{sd}^{(j)}} x_i(\tau_j, c_{-j}) d\tau_j f_{-j}(c_{-j}) dc_{-j} \tag{25}$$

Therefore, a smaller volume of traffic carried by node j only leads to smaller profit of node j. Thus node j does not have incentive to under-report its capacity.

Now consider the situation of claiming $\tilde{b}_j > b_j$. Node j falls into one of the following four cases under this situation: (1) Node j is not included in the LPP when reporting b_j, and is not included when reporting $\tilde{b}_j$; (2) Node j is not included in the LPP when reporting b_j, and is included by reporting $\tilde{b}_j$; (3) Node j is included in the LPP when reporting b_j and is not included when reporting $\tilde{b}_j$; (4) node j is included in the LPP when reporting b_j and is included when reporting $\tilde{b}_j$.

For case (1), the profit for node j is zero whether it reports b_j or $\tilde{b}_j$, so a cheating does not lead to better profit.

Case (2) cannot happen. This is because the fact that node j is not included in the LPP when reporting b_j implies that the ranking of the shortest path from s to d that traverses through node j, denoted as path δ, is behind those paths that receive non-zero traffic allocation. This also means that the paths that receive non-zero traffic allocation does not traverse node j. Therefore, increasing the reported capacity of node j does not change the sequence and the volume by which these paths are allocated with traffic. So node j cannot be included if it is not included when reporting b_j.

In case (3), the profit of node j is greater than 0 when it reports b_j. When it reports $\tilde{b}_j$, it is not included in the route, so its profit becomes 0. So its profit is reduced when node j cheats on capacity.

Case (4) contains two sub-cases. Sub-case (1), when b_j is reported, node j is not a bottleneck node in any paths in the LPP and (2) when b_j is reported, node j is a bottleneck node in some path included in the LPP. For sub-case (1), reporting a larger capacity of node j does not change the traffic allocation among routes, because such allocation is solely decided by those bottleneck nodes. Monotonically increasing the capacity of node j does not

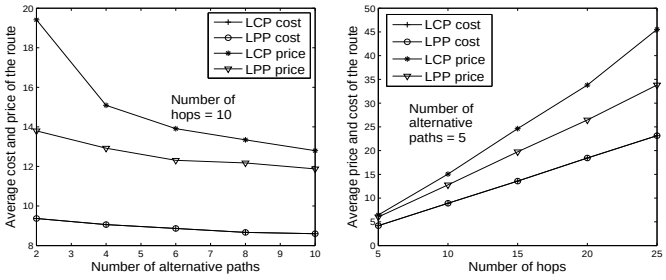
change these bottleneck nodes, so the traffic allocation does not change. Similarly, reporting $\tilde{b}_j$ also does not change the calculated payment to node j, because the traffic allocation in the integration term of (21) does not change, either. So the profit for node j remains the same when the reported capacity is changed from b_j to $\tilde{b}_j > b_j$. For sub-case (2), because node j is a bottleneck node when b_j is reported, note that the traffic passed through node j has reached its true capacity limit b_j at this point. When $\tilde{b}_j$ is reported, more traffic will be allocated to node j according to Algorithm 2. This makes the traffic that is assigned to node j by Algorithm 2 exceeds the node's actual capacity limit. As a result, during the transmission, some traffic must be dropped at node j, making the actual relayed traffic volume smaller than the planned volume. According to the proposed payment materialization method, the actual payment to node j is dropped to 0. Remembering that the profit of node j is greater than 0 when b_j is reported, the node's profit is reduced when a higher (faked) capacity is reported.

Combining all the above cases, Theorem 4 is proved. ■

Combining the truthful route selection and payment calculation algorithms described in Sections 5.1 and 5.2 with the above truthful payment materialization algorithm, the complete LPP routing mechanism is truthful, because no cheating on the node's cost and capacity can lead to a better profit for the node.

6 Performance Evaluation

In this section, we evaluate the performance of LPP using simulations. The VCG-based LCP algorithm is also simulated for comparison. For an end-to-end flow, the computed cost and price are averaged over sessions to obtain their expected value. We simulate a flow of 500 sessions to ensure the average value is based on sufficiently large samples. Our simulations show that in general, similar performance trends are observed under different node cost distributions. So we only show the results under uniformly distributed node costs. Our results presented below are averaged over 10 independent runs.



(a) Impact of no. of alternative paths

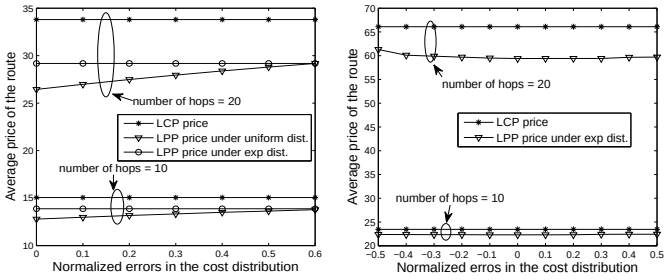
(b) Impact of path length

Figure 4: Impact of topology.

6.1 LPP Without a Node-capacity Constraint

In Figure 4, we study the impact of network topology on performance. In this set of simulations, all paths from the source to the destination are node-disjoint and contain the same number of hops. Nodes' costs are independently and uniformly distributed over $[0, 2]$. Albeit highly idealized, this topology allows us to directly control two important topological parameters: the number of alternate paths and their lengths. This enables us to gain insight to the algorithm's performance in relation to basic topological parameters.

In general, Figure 4 shows that a significant saving in price is achieved by LPP when compared with LCP. This is not surprising, because of the different objectives targeted by LPP and LCP. At the same time, the figure shows that the cost difference between LPP and LCP is trivial. This phenomenon indicates that LPP sacrifices little system-wide (social) efficiency in order to ensure a lower out-of-pocket payment for the end user. Two additional observations can be made on the two sub-figures. First, the saving in price is more significant when the number of alternative paths is small (sub-figure(a)). This is because under the VCG algorithm, the price of the LCP is the cost of the second-shortest path. When the total number of alternative paths is small, the monopoly effect of the second-shortest path becomes more significant, leading to a higher price for the LCP. Second, the saving by LPP is more significant when the path length is larger (sub-figure(b)). This is due to the aggregate saving over various intermediate nodes along the path, which becomes more and more significant with the increase in the number of hops.



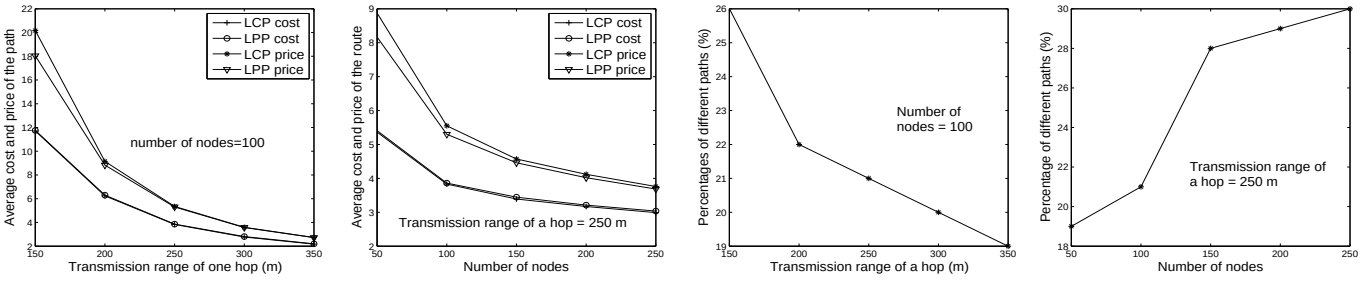
(a) Uniform (b) Exponential

Figure 5: Sensitivity analysis under various cost distributions.

In Figure 5, we study the sensitivity of our algorithm to the errors in estimating the distribution of the node cost. We are interested in two types of estimation errors: error in parameter and error in (distribution) type. Specifically, we generate the node's cost according to a distribution $f^{(1)}$, but we assume the mechanism uses an estimated version, say $f^{(2)}$, to compute the virtual cost. For an error in parameter, distributions $f^{(1)}$ and $f^{(2)}$ are of the same type (function), but differ slightly in the value of their parameters. To make our results more general, we show the results for both uniform and exponential node cost. For uniformly distributed costs, $f^{(1)}$ has a domain of $[v, w]$ and $f^{(2)}$ has a domain of $[v + 0.5\epsilon(w - v), w - 0.5\epsilon(w - v)]$, where $\epsilon > 0$ is the normalized error (this is because by observing the historical samples, the domain of $f^{(2)}$ should be a subset of that of $f^{(1)}$). For exponentially distributed costs, the rate parameters of $f^{(1)}$ and $f^{(2)}$ are given by $\lambda^{(2)} = (1 + \epsilon)\lambda^{(1)}$, where ϵ represents the normalized error. For an error in the type of distribution, $f^{(2)}$ and $f^{(1)}$ are taken as different functions. In particular, when $f^{(1)}$ is uniform, we assume its estimated version, $f^{(2)}$, follows an exponential distribution of the same mean. The price of the LPP under both estimation errors are plotted in Figure 5. From this figure, it is clear that as long as the means of $f^{(1)}$ and $f^{(2)}$ are close, both types of errors have only minor impact on the price of the LPP. This phenomenon indicates that our algorithm is insensitive to estimation errors in the node-cost distribution.

In Figure 6, we simulate LPP under random topology . We consider a 1000×1000 (meter^2) area. The source and the destination are located at the middle of two opposite sides of the square. Other nodes are uniformly distributed. Heterogeneous spectrum situation is

simulated. Specifically, we assume that in the middle of this square, there is a 200-meter-radius circular "hot" zone, where the node cost is uniformly distributed between $[0.5, 5]$. For a node outside the hot zone, its cost is between $[0.5, 2]$. Figure 6 shows that the price and cost of LPP and LCP decrease with the node's transmission range and with the number of nodes in the network. The savings in price by LPP is more significant at small transmission ranges and small numbers of nodes. This can be explained by noting that a smaller transmission range per hop corresponds to a longer path, and a smaller number of nodes means smaller number of alternative paths from the source to the destination. So these trends are in line with our results in Figure 4. Sub-figures (c) and (d) show the percentage of sessions for which LPP≠LCP. In general, in more than 10% sessions, the LPP differs from the LCP. However, even with this magnitude of difference, sub-figures (a) and (b) show that the cost of the LPP is only slightly higher than the LCP. This phenomenon indicates that the LPP in general tends to employ a node of relatively low cost, but not necessarily the node of the lowest cost.



(a) cost/price vs. tx range

(b) cost/price vs. no. of nodes

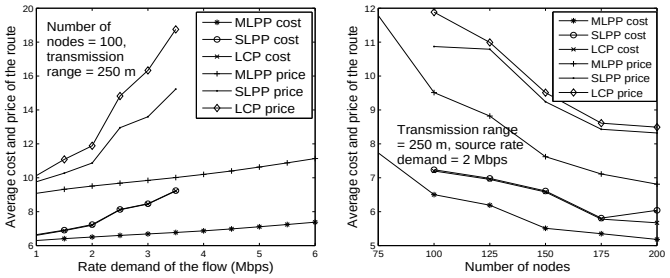
(c) Percentage of LPP≠LCP vs. tx range

(d) Percentage of LPP≠LCP vs. no. of nodes

Figure 6: LPP under random topologies.

6.2 LPP with a Node-capacity Constraint

This set of simulation is based on the same random topologies generated for Figure 6, but now each node is associated with a capacity limit (in Mbits/s). For a node in the hot zone, its capacity is uniformly drawn between 0 and 2 Mbits/s. Otherwise, its capacity is uniformly drawn between 0 and 5 Mbits/s. Once the node's capacity is randomly selected,



(a) Impact of rate demand of the source

(b) Impact of number of nodes

Figure 7: Impact of rate demand and number of nodes on path selection.

it does not change throughout the simulation. Due to the lack of counterpart multi-path LCP algorithms, we compare the multi-path route found by LPP, denoted as MLPP in our results, with two single-path routing algorithms. These two algorithms are straightforward extensions of the VCG-LCP and the simplified LPP (the one that does not consider the node capacity constraint) into the scenario where a node-capacity constraint is imposed. To make these extensions possible, we prune out from the topology those nodes whose capacities are smaller than the flow's rate demand. The LCP and the simplified LPP algorithms can then be directly applied to the residual topology. To distinguish it from the MLPP, the extension of the simplified LPP is denoted as SLPP.

Our simulation results are plotted in Figure 7. It is clear that the cost and price of MLPP are significantly smaller than its single-path counterparts. In addition, the figure shows that one problem of these single-path routing algorithms is that the destination may become unreachable in the residual topology after pruning out nodes of low capacity. As a result, MLPP supports a much higher rate demand by bonding the capacities of multiple paths.

7 Conclusions

We have developed polynomial-time algorithms in this paper to find the truthful LPP route in an OSA network. Compared with LCP, the adoption of LPP can lower the price paid by end users for their route purchase. The saving in payment by LPP largely depends on the number of alternative paths between the source and the destination and the length (in

number of hops) of these paths. In general, the more alternative paths and the longer these paths are, the more saving the LPP can make. Although the LPP has a lower price tag for the end user, its cost is only slightly higher than that of the LCP. This indicates that the LPP mechanism only needs to sacrifice a trivial amount of social efficiency in exchange for a lower price tag. Some open topics remain for future work. In this paper, we have focused on the mathematical structures of the LPP mechanism. Protocols that take into account practical issues for the implementation are yet to be developed. In addition, collusion between nodes is not considered in this work. This will be studied in a future work.

References

[1] L. Anderegg and S. Eidenbenz. Ad hoc-VCG: A truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In *Proceedings of the ACM MobiCom Conference*, 2003.

[2] A. Archer and E. Tardos. Frugal path mechanisms. In *Proceedings of the Anunal ACM SIAM Symposium on Discrete Algorithms (SODA)*, 2002.

[3] L. Buttyan and J. P. Hubaux. Enforcing service availability in mobile ad-hoc WANs. In *Proceedings of the ACM MobiHoc Conference*, 2000.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (2nd Ed.)*. MIT Press and McGraw-Hill, 2001.

[5] E. Elkind, L. A. Goldberg, and P. W. Goldberg. Frugality ratios and improved truthful mechanism for vertex cover. *Technical Report, University of Southampton, available at http://arxiv.org/abs/cs/0606044v4*, 2008.

[6] E. Elkind, A. Sahai, and K. Steiglitz. Frugality in path auctions. In *Proceedings of the Anunal ACM SIAM Symposium on Discrete Algorithms (SODA)*, 2004.

[7] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. *Distributed Computing*, 18:61–72, 2005.

[8] J. Hershberger and S. Suri. Vickrey prices and shortest paths: what is an edge worth? In *Proceedings of IEEE Symposium on Foundations of Computer Science*, pages 252–259, 2001.

[9] J. Jia, Q. Zhang, Q. Zhang, and M. Liu. Revenue generation for truthful spectrum auction in dynamic spectrum access. In *Proceedings of the ACM MobiHoc Conference*, 2009.

[10] X. Y. Li, Y. Wu, P. Xu, G. Chen, and M. Li. Hidden information and actions in multi-hop wireless ad hoc networks. In *Proceedings of the ACM MobiHoc Conference*, pages 283–292, 2008.

[11] R. B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, Feb. 1981.

[12] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[13] T. Shu and M. Krunz. Coordinated channel access in cognitive radio networks: A multi-level spectrum opportunity perspective. In *Proceedings of the IEEE INFOCOM (mini-conference)*, 2009.

[14] X. Su, S. Chan, and G. Peng. Auction in multi-path multi-hop routing. *IEEE Communication Letter*, 13(2):154–156, Feb. 2009.

[15] W. Wang, S. Eidenbenz, Y. Wang, and X. Y. Li. OURS: optimal unicast routing systems in non-cooperative wireless networks. In *Proceedings of the ACM MobiCom Conference*, 2006.

[16] W. Wang and X. Y. Li. Low-cost routing in selfish and rational wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(5):596–607, May 2006.

[17] W. Wang, X. Y. Li, and Y. Wang. Truthful multicast routing in selfish wireless networks. In *Proceedings of the ACM MobiCom Conference*, pages 245–259, 2004.

[18] F. Wu, T. Chen, S. Zhong, L. Li, and Y. R. Yang. Incentive-compatible opportunistic routing for wireless networks. In *Proceedings of the ACM MobiCom Conference*, pages 303–314, 2008.

[19] Q. Zhao and B. M. Sadler. A survey of dynamic spectrum access: signal processing, networking, and regulatory policy. *IEEE Signal Processing Magazine*, 24(3):79–89, 2007.

[20] S. Zhong, J. Chen, and Y. R. Yang. Sprite: a simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *Proceedings of the IEEE INFOCOM Conference*, 2003.

[21] S. Zhong, L. Li, Y. G. Liu, and Y. R. Yang. On designing incentive-compatible routing and forwarding protocols in wireless ad-hoc networks–an integrated approach using game theoretical and cryptographic techniques. In *Proceedings of the ACM MobiCom Conference*, 2005.

[22] X. Zhou, S. Gandhi, S. Suri, and H. Zheng. eBay in the sky: strategy-proof wireless spectrum auctions. In *Proceedings of the ACM MobiCom Conference*, 2008.