Charge Me If You Can: Charging Path Optimization and Scheduling in Mobile Networks

Lin Chen
LRI-CNRS UMR 8623
Univ. Paris-Sud
chen@lri.fr

Shan Lin
Dpt. Elec. Comp. Eng.
Stony Brook Univ.
shan.x.ling@stonybrook.edu

Hua Huang
Dpt. Elec. Comp. Eng.
Stony Brook Univ.
hua.huang@stonybrook.edu

ABSTRACT

We study a class of generic optimization problems on charger scheduling and charging path planing. These problems arise from emerging networking applications where mobile chargers are dispatched to deliver energy to mobile agents (e.g., robots, drones, and vehicles), which have specified tasks and mobility patterns. We instantiate our work by focusing on finding the charging path maximizing the number of nodes charged within a fixed time horizon. We prove that this problem is APX-hard. By recursively decomposing the problem into sub-problems of searching sub-paths, we design a quasi-polynomial time algorithm that achieves polylogarithmic approximation to the optimum charging path. Our approximation algorithm can be further adapted and extended to solve a variety of charging path optimization and scheduling problems with realistic constraints, such as limited time and energy budget.

CCS Concepts

•Networks → Network control algorithms; •Theory of computation → Scheduling algorithms;

Keywords

Mobile charger scheduling; Mobile wireless networks; Approximation algorithm

1. INTRODUCTION

In many emerging applications such as first responder, infrastructure monitoring, and scientific exploration, battery-powered mobile agents (e.g., sensors [1], robots [2], drones [3], and vehicles [4]) usually have specified tasks and mobility patterns [5]. To supply energy to these agents, mobile chargers are dispatched to visit these agents, which can significantly prolong the lifetime of mobile nodes. However, as the mobile charger only deliver energy to a target node when it encounters the node (or close to the node if wireless charging is used), inefficient path planning may incur long latency to pursue target nodes, result in dead nodes or even task failures. Therefore, optimizing the trajectory of the mobile charger is a primary concern for maintaining the operation of these systems.

In this paper, we focus on the path optimization and charger scheduling problems with mobile nodes in mobile charging applications. To make our analysis generic and widely applicable, we do not impose any constraint on the mobility patterns of nodes, i.e., the trajectory of any node can be a curve of any form. We formulate a class of generic path optimization problems and concentrate on the problem of maximizing the number of nodes charged within a fixed time horizon. Our framework allows a variety of path optimization problems to be formulated with realistic constraints, such as limited time and energy budget. We prove that these problems are either NP-hard or APX-hard. We design a quasi-polynomial time algorithm that achieves polylogarithmic approximation to the optimum charging path. The core technicality of our design is a recursive algorithm that decomposes the problem into sub-problems of searching sub-paths. We also demonstrate how our approximation algorithm can be adapted and extended to solve other charging path optimization and scheduling problems.

Despite our focus on mobile charging, the generic formulation of the problem makes our analysis methodology and obtained results applicable to a wide range of related problems such as data mule scheduling, package delivery, target monitoring, and security patrolling. These problems have a common generic objective of designing an optimum path and the corresponding scheduling that maximise the number of encountered targets under a given budget in terms of time or energy, or minimizes the cost of encountering a given minimum number of targets.

The rest of the paper is organized as follows. We review the literature in Section 2. We formulate the charging path optimization problem in Section 3 and establish its APX-hardness. In Section 4, we present our approximation algorithm and analyze its performance. Section 5 discusses relevant extensions and variants of our work. Section 6 presents the simulation results. Section 7 concludes the paper.

MobiHoc'16, July 04-08, 2016, Paderborn, Germany
© 2016 ACM. ISBN 978-1-4503-4184-4/16/07...$15.00
DOI: http://dx.doi.org/10.1145/2942358.2942364

2. RELATED WORK

The problem we address and the methodology we employ are related to the following research fields.

2.1 Path Optimization in Vehicular Routing and Data Ferry Assisted Data Harvesting

There have been a significant amount of research works on the Vehicular Routing Problem (VRP) [6]. In general, the VRP seeks to optimise the routing decisions of a single or fleet of vehicles to deliver goods to specified locations according to demand requirements and other specified constraints. The VRP has many variants based on the application scenarios and constraints. Some recent papers deal with optimizations with dynamism [7,8], uncertainty [9], and many real-life constraints [10, 11]. These works provide a rich foundation for related algorithmic research. However, it is commonly assumed in these works that the targets are stationary and the vehicle travels through a set of fixed locations.

Another related problem is path optimization in data ferry assisted data harvesting [12–15] (cf. [16] for a comprehensive survey), where a data ferry (e.g., robot, vehicle) travels across the sensor field and harvests data from sensor nodes. Again, existing works focus on finding the optimum path of minimum length in the setting where sensor nodes are stationary.

2.2 Mobile Charger Scheduling

The mobile charger scheduling problem [17–19] seeks paths for mobile chargers (e.g., mobile robots) to replenish batteries for sensor networks. Many variants are studied by introducing different optimization goals, application scenarios, and constraints. For example, the authors of [20] consider collaborate mobile charger scheduling, where different mobile chargers can recharge each other so that the chargers can cover a larger area. In [21], the authors assume that the charging time for sensor nodes is much longer than the mobile charger's traveling time. Their goal is to maximize the timespan that all sensor nodes are alive. In [22], the authors assume that the charger can recharge all the sensor nodes lying within a distance through wireless power transfer. Their goal is to find a charging path and stopping points to minimize the charging time. In a recent paper [23], the authors consider the power heterogeneity of sensor nodes. They divide sensor nodes into groups, and apply the TSP algorithms to recharge nodes within each group. In [24], the problem of ensuring monitoring quality for stochastic events is studied. When the traveling time of the mobile charger is ignored, this problem can be reformulated as a sub-modular problem. A polynomial time algorithm with constant time approximation ratio is then developed.

Existing works on mobile charger scheduling all assume that the nodes are fixed in locations. In contrast, we study a more generic and practical scenario of moving target nodes, and develop a quasi-polynomial time scheduling algorithm that achieves poly-logarithmic approximation.

2.3 Traveling Salesman Problem and Orienteering Problem

The Traveling Salesman Problem (TSP) is a class of combinatorial optimization problems which have been extensively studied. Many approximation algorithms have been proposed [25, 26]. A relevant extension to the TSP is the deadline-TSP or the TSP with time window, where each node can be visited only within a time interval [27–30].

In [28] and [29], the authors study the Max-Prize Path problem, also referred to as the Orienteering problem, where the goal is to visits as many nodes as possible, but only before a hard time deadline D. The APX-hardness of the Orienteering problem can be shown via reduction from the TSP on bounded metrics which is APX-hard [31]. For the Orienteering problem on undirected graphs, the best approximation ratio in the literature is $2+\epsilon$ [32]. For the Orienteering problem on directed graphs, the best approximation ratio is $O(\log^2 OPT)$ [32].

Our paper can be regarded as a non-trivial extension of the Orienteering problem in a more generic case where nodes are are mobile. In this regard, only [33] has considered the TSP with moving nodes by designing a $(1+\alpha)$-approximation algorithm, where α denotes the approximation ratio of the TSP heuristic. However, the algorithm developed in [33] only works when the number of moving targets is sufficiently small. In contrast, we address the generic case even when the network scales. We would like to emphasize that the analysis method of the classic TSP and the (directed) Orienteering problems where the heuristic path is formed by joining several trees cannot be applied in our problem as the resulting path may not be feasible when nodes are mobile. Therefore, an original study is called for which cannot draw on existing results.

3. SYSTEM MODEL AND PROBLEM FORMULATION

3.1 Network Model

We consider a network composed of n mobile nodes (e.g., robots, sensors, drones), denoted by the set $V \triangleq \{v_i\}_{i=1}^n$, deployed over a 2-D Euclidean plan. Nodes are battery-powered and thus need to be recharged periodically. To perform the charging task a mobile wireless charging vehicle, referred to as charger for short, travels from a starting point, denoted by a virtual node s, then visits a number of nodes to charge them before returning to a terminal point, denoted by t. If the charger needs to return to the starting point, t coincides with s, and the charging path becomes a tour. A node remains stationary when being charged.

The novel challenge we address, w.r.t. the state-of-the-art works, is that both the charger and the nodes are mobile. We denote $v_i(t)$ $(1 \le i \le n)$ the position of node v_i at time t. We assume the mobility pattern of the nodes (i.e., their moving trajectories $v_i(t)$) are known to the charger and leave the unknown or partially known mobility case for future research. We denote r_i the upper-bound of the moving speed of v_i and r_s the moving speed of the charger. We assume that $r_s > r_i, \forall i \in [1, n]$ which holds in typical mobile sensing applications. We denote $P \in \mathcal{P}$ the charging path followed by the charger, where $\mathcal{P}$ denotes the set of all possible paths. To make our analysis generic and widely applicable, we do not impose any constraint on the mobility patterns of nodes, i.e., the trajectory of any node can be a curve of any form. Table 1 lists the major notations used in the paper. The example in Figure 1 further illustrates our network model.

Example 1. *Consider the network illustrated in Figure 1 composed of three nodes v_1, v_2, v_3. v_1 is stationary. v_2 and v_3 are mobile with their trajectories illustrated in the figure. Charging is immediate at any node. By following the path P, the charger can charge v_1, v_2 and v_3 at time 1, 2 and 3.*

3.2 Optimum Charging Path Design

We consider a class of charging path optimization and scheduling problems including (but not limited to) the following:

Table 1: Notations

Section 3: System Model and Problem Formulation	
V	Set of nodes in the network
n	Number of nodes in the network
v_i	Node i
$v_i(t)$	Position of node i at time t
s	Starting point of the charger
t	Terminal point of the charger
r_s	Moving speed of the charger
r_i	Upper-bound of moving speed of v_i
$\mathcal{P}$	Possible charging path set
$d(P)$	Euclidean length of path P
$t_s(p)$	Sojourn time of the charger at p without charging
$\Lambda(P)$	Set of nodes charged on path P
$\Gamma(P)$	Timespan of path P
$f_i(t)$	Charging level of node v_i as a function of charging time t
$g_i(x)$	Inverse function of $f_i(t)$: $g_i(x) = f_i^{-1}(t)$
α	Required charging level
B	Maximum charging path timespan
Section 4: Approximation Algorithm Design	
Δt	Time stepsize
G_d	Discretized graph with node set $V(G_d)$ and edge set $E(G_d)$
n_d	Number of nodes in G_d: $n_d = \lvert V(G_d)\rvert$
$c_d(e)$	Cost of edge e in G_d
λ	Scaling factor in edge cost rounding
$\epsilon(\lambda)$	Relative error under rounding factor λ
$C(v)$	Set of nodes in the same clique of v
V_{v_-}	Set of nodes in G_d to which there exists an edge from v
V_{v_+}	Set of nodes in G_d from which there exists an edge to v
L	Maximum recursion level in Alg. 1

- The charger has a fixed time budget for the charging journey and aims at charging the maximum number of nodes in one journey;
- The charger has a battery reservoir and aims at charging the maximum number of nodes before returning to its service station to replenish itself;
- The charger has a fixed number of $M \leq n$ nodes to charge and aims at minimizing the total charging time;
- The charger needs to charge all nodes within a charging journey and seeks the charging path minimizing the energy consumption or the total time.

The above problems can be classified into two categories:

- The charger has a certain budget (e.g., in terms of time, energy) and it seeks a path to maximise the number of nodes it can charge within the given budget;
- The charger has a number of nodes to charge and it seeks a path of minimum cost (e.g. in terms of time, energy consumption) to accomplish the charging task.

Our work establishes a generic framework on the charging path optimization and scheduling problems. To instantiate our work, we focus on the first problem of maximizing the number of nodes charged within a fixed time horizon, as formulated below. We discuss in Section 5 how our framework can be applied to address other problems formulated above.

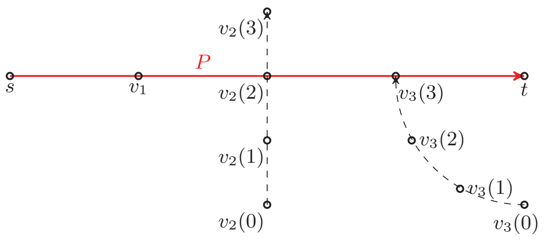


Figure 1: An example illustrating network model.

3.3 Problem Formulation

We start by modeling the charging process. Let $x \in [0, 1]$ denote the battery level (in percentage) of a node during the charging process. For each node v_i, x can be expressed as a function of the charging time t and the initial battery level x_0: $x = f_i(t, x_0)$. Figure 2 traces an example of charging curve for $x_0 = 0$. Throughout our analysis, we assume the battery level of any node remains constant during the charging journey unless it is charged by the charger. This assumption is reasonable as the duration of one charging journey is typically negligibly small compared to the lifetime of a node. This assumption also implies that each node is charged at most once during a charging journey. Under this assumption, f_i can be expressed as a function of t from the charger's perspective. Generically, the following property holds from elementary electrical circuit analysis:

- $f_i(t)$ is continuous, derivable and monotonously increasing in t;
- $f_i(t)$ is concave in t, meaning that the marginal charging utility decreases in t.

Define $g_i(x) \triangleq f_i^{-1}(t)$. The following properties on $g_i(x)$ directly follow from the properties of $f_i(t)$:

- $g_i(x)$ exists and is the time required to charge v_i to x;
- $g_i(x)$ is continuous and convex in x;
- The derivative of $g_i(x)$, $g_i'(x)$, is increasing in x.

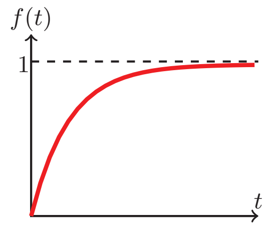


Figure 2: Example of charging curve.

Given a charging path P whose Euclidean length is denoted by $d(P)$, let $\Lambda(P) \subseteq P$ denote the set of nodes that the charger charges to battery level $\alpha \in [0, 1]$ (e.g., 90%) while traveling along P[1]. For any point $p \in P$, let $t_s(p)$ denote the sojourn time during which s stays at p without charging any node. The entire charging time along P, denoted by $\Gamma(P)$, can be established below

$$\Gamma(P) = \frac{d(P)}{r_s} + \sum_{v_i \in \Lambda(P)} g_i(\alpha) + \sum_{p \in P} t_s(p),$$

where $\frac{d(P)}{r_s}$ is the time required for the charger to travel

[1]To make our analysis clear, we assume nodes need to be charged to the same battery level α. The extension to the generic case with different charging levels is straightforward.

distance $d(P)$, $\sum_{v_i \in A(P)} g_i(\alpha)$ is the time to charge all nodes in $\Lambda(P)$ to α and $\sum_{p \in P} t_s(p)$ is the total sojourn time.

We refer to the total charging time along a path P as the *timespan* of P. Without introducing ambiguity, a charging path P also denotes the corresponding charging schedule.

Problem 1 (Charging Path Optimization)**.** *The charging path optimization problem is as follows:*

$$P^* = \underset{P \in \mathcal{P}, \Gamma(P) \leq B}{\text{argmax}} |\Lambda(P)|.$$

That is, given the required charging level α and the maximum timespan B, the charger seeks an optimum path that maximises the number of charged nodes. The solution P^* is termed as an optimum charging path.

In many practical scenarios, it is acceptable to have small marge on the charging performance, which motivates the following definition.

Definition 1 (ϵ-optimum Charging Path)**.** *A charging path P_ϵ is called an ϵ-optimum charging path ($0 \leq \epsilon \leq 1$) if the following conditions are satisfied:*

- *$\Gamma(P_\epsilon) \leq B$, i.e., the timespan of P_ϵ does not exceed B;*
- *By following P_ϵ, the charger can charge all the nodes in $\Lambda(P^*)$ to at least $(1-\epsilon)\alpha$.*

3.4 Problem Hardness

We prove the APX-hardness of Problem 1 below.

Theorem 1 (APX-hardness of Charging Path Optimization)**.** *Problem 1 is APX-hard, or NP-hard to approximate within an arbitrarily small constant factor.*

Proof. Consider the Orienteering problem which has been proved APX-hard.

Problem 2 (Orienteering Problem)**.** *Given an edge-weighted graph $G = (V(G), E(G))$ (directed or undirected), two nodes $s, t \in V(G)$ and a path length upper-bound B, the Orienteering problem seeks a path starting at s ending at t of total length at most B maximising the number of nodes visited.*

To prove the APX-hardness of Problem 1, we prove that it can be reduced to Problem 2 in polynomial time.

Given any graph $G = (V(G), E(G))$ on which we need to solve Problem 2, we instantiate Problem 1 as below. We set $V = V(G)$. All nodes are stationary. Charging is immediate at any node, i.e., it takes 0 time to charge a node. In this instance, it is straightforward to see that the solution of Problem 2 is also the solution of Problem 1 and vice versa. Since Problem 2 is APX-hard, Problem 1 is also APX-hard. □

The proof of Theorem 1 demonstrates that even the static version of Problem 1 is APX-hard. It becomes much more complex with nodes being mobile.

4. APPROXIMATION ALGORITHM DESIGN

In this section, we design an approximation algorithm to solve Problem 1.

4.1 Problem Discretization

We divide time into time instances $\{t_k, k = 0, 1, \cdots\}$ with stepsize $\Delta t \triangleq t_k - t_{k-1}$[2]. The trajectory of v_i can then be discretized into a vector $[v_i(t_0), v_i(t_1), \cdots]$ where $v_i(t_k)$ is the position of v_i at time t_k. By the discretization above, we construct a directed acyclic graph, denoted by G_d, whose vertices and edges are specified as follows.

Vertices. The vertex set $V(G_d)$ includes:

- the discretized points on the trajectories of all nodes in V, i.e., $v_i(k_i\Delta t)$ $(0 \leq k_i \leq K_i)$, $\forall v_i \in V$[3],
- the starting point s which is denoted as $v_0(0)$,
- the terminal point t denoted as $v_{n+1}(K_{n+1}\Delta t)$ with $K_{n+1} = \left\lfloor \frac{B}{\Delta t} \right\rfloor$.

Edges. For any two points $v_i(k_i\Delta t)$ and $v_j(k_j\Delta t)$ with $v_i \neq v_j$, there exists an edge $\overrightarrow{v_i(k_i\Delta t)v_j(k_j\Delta t)}$ in G_d iff the inequality below holds

$$(k_j - k_i)\Delta t \geq g_i[(1-\epsilon)\alpha] + \frac{|\overline{v_i(k_i\Delta t)v_j(k_j\Delta t)}|}{r_s}, \quad (1)$$

where $g_i[(1-\epsilon)\alpha]$ is the time to charge v_i to $(1-\epsilon)\alpha$ and $g_0[(1-\epsilon)\alpha] = g_{n+1}[(1-\epsilon)\alpha] = 0$ since the starting point v_0 and the terminal point v_{n+1} do not need to be charged. $|\overline{v_i(k_i\Delta t)v_j(k_j\Delta t)}| = d[v_i(k_i\Delta t), v_j(k_j\Delta t)]$ is the Euclidean distance between $v_i(k_i\Delta t)$ and $v_j(k_j\Delta t)$, i.e., the length of the line segment $\overline{v_i(k_i\Delta t)v_j(k_j\Delta t)}$. The inequality (1) implies that starting at $v_i(k_i\Delta t)$ at time $k_i\Delta t$, the charger can charge v_i to level $(1-\epsilon)\alpha$ and then arrive at $v_j(k_j\Delta t)$ no later than $k_j\Delta t$. For any edge $\overrightarrow{v_i(k_i\Delta t)v_j(k_j\Delta t)}$, we set its cost $c_d[v_i(k_i\Delta t), v_j(k_j\Delta t)]$ as follows:

$$c_d[v_i(k_i\Delta t), v_j(k_j\Delta t)] \triangleq (k_j - k_i)\Delta t.$$

Note that there is no edge between $v_i(k_i^1\Delta t)$ and $v_i(k_i^2\Delta t)$ for any $v_i \in V$ and $0 \leq k_i^1, k_i^2 \leq K_i$.

The index upper-bound K_i $(1 \leq i \leq n)$ is set as below:

$$\frac{c_d[v_0(0), v_i(K_i\Delta t)] + c_d[v_i(K_i\Delta t), v_{n+1}(K_{n+1}\Delta t)]}{r_s} + g_i[(1-\epsilon)\alpha] \leq B, \quad (2)$$

where the left hand side of (2) is the minimum time required for the charger to go from the starting point to $v_i(K_i\Delta t)$ and charge v_i to $(1-\epsilon)\alpha$ and then return to the terminal point. For any node $v_i(k_i\Delta t)$ with $k_i > K_i$, it is impossible for the charger to charge v_i at the position $v_i(k_i\Delta t)$ under the timespan budget B.

We next introduce two definitions that are useful in subsequent analysis.

Definition 2 (Clique)**.** *In G_d, a clique is defined as a set of nodes $v_i(k_i\Delta t)$ $(0 \leq k_i \leq K_i)$ belonging to the trajectory of the same physical node v_i.*

Definition 3 (Feasible Path)**.** *Let $\mathcal{P}_d$ denote the set of all paths in G_d, a path $P \in \mathcal{P}_d$ is called a feasible path if P starts from $v_0(0)$, terminates at $v_{n+1}(K_{n+1}\Delta t)$, passes exactly one node of each clique it traverses, and $\Gamma(P) \leq B$. An optimum feasible path is a feasible path passing maximum number of cliques.*

Example 2. *Figure 3 illustrates an example of G_d and a feasible charging path P. There are two nodes to be charged with v_1 being stationary and v_2 being mobile. The coordinates of nodes and the trajectory of v_2 are shown in the left subfigure. $\Delta t = 1$. $B = 3$. The speed of the charger and v_2*

[2]To make the analysis concise, we set the same stepsize for all the nodes. Nevertheless, our analysis can be slightly adapted to the case with different per-node stepsize Δt_i.

[3]How to set K_i will be analysed later.

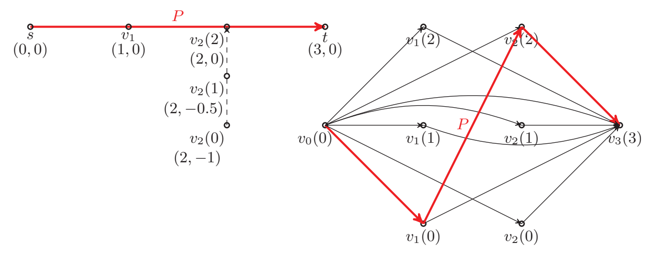


Figure 3: An example of G_d and a feasible path P.

are $r_s = 1$ unit length per unit time and $r_2 = 0.5$. Charging is immediate at any node. The corresponding discretized graph G_d is shown in the right subfigure with a feasible path P depicted in both subfigures.

Theorem 2. *Any feasible path passing m cliques corresponds to a charging path of maximum timespan B that can charge m nodes to $(1-\epsilon)\alpha$.*

Proof. We prove the theorem by constructing a feasible charging path and schedule from any feasible path passing m cliques. Let P denote a feasible path passing sequentially $v_0(0)$ (i.e., s) and then m nodes $v_i(k_i\Delta t)$ sequentially from $i = 1$ to m before terminating at $v_{n+1}(K_{n+1}\Delta t)$ (i.e., t). We construct the charging path and schedule where the charger starts charging node v_i $(1 \le i \le m)$ at time $k_i\Delta t$ for $g_i[(1-\epsilon)\alpha]$ time and then goes straightly to node v_{i+1}. It follows from (1) that under this charging schedule, the charger can charge nodes v_i $(1 \le i \le m)$ to $(1-\epsilon)\alpha$. Moreover, the timespan of the constructed charging path equals to $\Gamma(P)$ which is by definition upper-bounded by B. □

We next show that an optimum charging path can be approximated arbitrarily close by a feasible path in terms of the number of charged nodes.

Theorem 3. *Given any $\epsilon > 0$, under the condition that $\Delta t \le \frac{\alpha\epsilon}{3}\min_{1\le i\le n} g_i'[(1-\epsilon)\alpha]$, there exists a feasible path in G_d which is also an ϵ-optimum charging path.*

Proof Sketch. Without loss of generality, assume the optimum charging path P^* starts from s, then charges M nodes sequentially from v_1 to v_M before terminating at t. By slightly introducing ambiguity, we denote $v_{n+1}(K_{n+1}\Delta t)$ as $v_{M+1}(K_{M+1}\Delta t)$ requiring 0 time to be charged for notation and analysis convenience. Let T_i $(1 \le i \le M)$ denote the time when the charger starts charging v_i and let $k_i\Delta t$ denote the discretized time instance where $(k_i - 1)\Delta t < T_i \le k_i\Delta t$. The core part of the proof consists of (1) proving that the edges $\overrightarrow{v_i(k_i\Delta t)v_{i+1}(k_{i+1}\Delta t)}$ $(0 \le i \le M)$ exist in G_d, (2) showing that concatenating these edges yields a feasible path that is also an ϵ-optimum charging path. The detailed proof is given in Appendix. □

Theorem 3 demonstrates that the performance loss due to discretization can be controlled to arbitrarily small, at the price of increasing computation complexity in terms of the size of G_d. Given a tolerance level ϵ, Theorem 3 also quantifies the bound on the discretization granularity Δt to meet the performance requirement.

It follows from Theorem 2 and Theorem 3 that the problem of finding ϵ-optimum charging path can be transformed to the problem of finding the optimum feasible path, which is however APX-hard. The proof follows from the same deduction to the Orienteering problem as that in the proof of Theorem 1. Given its APX-hardness, we focus on approximation algorithm design in the next subsection.

4.2 Finding Optimum Feasible Path: Approximation Algorithm

This subsection presents our design of a quasi-polynomial time algorithm that achieves poly-logarithmic approximation to the optimum feasible path. We first state the following property of G_d which is useful in later analysis.

Lemma 1. *For any pair of nodes $v_s, v_t \in V(G_d)$, if there exists a path from v_s to v_t, then there must exist an edge from v_s to v_t, i.e., $c(v_s, v_t) \ne \infty$. Equivalently, if $c(v_s, v_t) = \infty$, then there does not exist a path from v_1 to v_2.*

Proof. We denote the path from v_s to v_t as the node sequence $\{v_1(k_1\Delta t), v_2(k_2\Delta t), \cdots, v_m(k_m\Delta t)\}$ where $v_s = v_1(k_1\Delta t)$ and $v_t = v_m(k_m\Delta t)$. As there exists a path from v_s to v_t, it holds that the edges $\overrightarrow{v_i(k_i\Delta t), v_{i+1}(k_{i+1}\Delta t)}$ $(1 \le i \le m-1)$ exist in G_d. It then follows from (1) that

$$(k_{i+1} - k_i)\Delta t \ge g_i[(1-\epsilon)\alpha] + \frac{|\overrightarrow{v_i(k_i\Delta t)v_{i+1}(k_{i+1}\Delta t)}|}{r_s}. \quad (3)$$

Summing (3) for all i from 1 to $m-1$ yields

$$\begin{aligned}(k_m - k_1)\Delta t &\ge \sum_{i=1}^{m-1} g_i[(1-\epsilon)\alpha] + \frac{|\overrightarrow{v_1(k_1\Delta t)v_m(k_m\Delta t)}|}{r_s} \\ &\ge g_1[(1-\epsilon)\alpha] + \frac{|\overrightarrow{v_1(k_1\Delta t)v_m(k_m\Delta t)}|}{r_s}.\end{aligned}$$

It then follows from (1) that the edge $\overrightarrow{v_1(k_1\Delta t)v_m(k_m\Delta t)}$ exists. That is, there exists an edge from v_s to v_t. □

To develop our algorithm, we assume that the edge costs of G_d are integers. If not, we can round them to integers by scaling each edge cost by a factor λ and rounding the scaled cost to its ceiling integer. The relative error incurred by the rounding process can be upper-bounded by λ, denoted by $\epsilon(\lambda)$, as follows:

$$\varepsilon(\lambda) = \frac{\lceil c_d(e)\lambda\rceil - c_d(e)\lambda}{c_d(e)\lambda} \le \frac{1}{c_d(e)\lambda} = O\left(\frac{1}{\lambda}\right).$$

The core idea of our algorithm, inspired by the idea of recursion in [34, 35], is summarized below:

- For each $m = [1..n_d]$ $(n_d \triangleq |V(G_d)|)$ and each node $v \in V(G_d) - \{s, t\}$:
  - Recursively search a path P_1 from s to v of minimum timespan that charges m nodes, denote the timespan of P_1 by b_1;
  - Recursively search another path P_2 from v to t of timespan at most $B - b_1$ that charges the maximum number of nodes;
- Output the concatenated path $P = (P_1, P_2)$ that charges the maximum number of nodes;

In the recursion process, we need to carefully choose P_1 and P_2 such that the resulting concatenated path does not visit any clique more than once.

Formally, the pseudo-code of the algorithm is illustrated in Algorithm 1. The core part of Algorithm 1 is the recursive procedure OPF, which has the following inputs:

Algorithm 1 Finding optimum feasible path (OFP)

```
Input: $G_d$, $s$, $t$, $B$, $L$
Output: $P_f$
 1: return $P_f$ := OPF($G_d$, $s$, $t$, $V(G_d)$, $L$, $B$)

 2: procedure OPF($G_d$, $v_s$, $v_t$, $V$, $l$, $b$)
 3:     if $b < c_d(v_s, v_t)$ or $c_d(v_s, v_t) = \infty$ then
 4:         return $P := \emptyset$
 5:     else
 6:         $P := (v_s, v_t)$
 7:     end if
 8:     if $l = 0$ then
 9:         return $P$
10:     end if
11:     for all $v \in V - C(v_s) - C(v_t)$ do
12:         for $m := 1$ to $n_d$ do
13:             $(P_1, b_1) :=$ BMIN($G_d$, $v_s$, $v$, $V_{v-}$, $l-1$, $b$, $m$)
14:             if $P_1 = \emptyset$ then
15:                 Break
16:             end if
17:             $P_2 :=$OPF($G_d$, $v$, $v_t$, $V_{v+} - C(P_1)$, $l-1$, $b-b_1$)
18:             if $P_2 = \emptyset$ then
19:                 Break
20:             end if
21:             if $|\Lambda(P_1)| + |\Lambda(P_2)| > |\Lambda(P)|$ then
22:                 $P := (P_1, P_2)$
23:             end if
24:         end for
25:     end for
26:     return $P$
27: end procedure

28: procedure BMIN($G_d$, $v_s$, $v$, $V$, $l$, $b$, $m$)
29:     $b_{min} := 1$, $P_{max} :=$ OPF($G_d$, $v_s$, $v$, $V$, $l$, $b$)
30:     if $|\Lambda(P_{max}|) < m$ then
31:         return $(\emptyset, 0)$
32:     end if
33:     $b_{max} := b$
34:     while $b_{max} - b_{min} > 1$ do
35:         $b_{mid} := \left\lceil \frac{b_{min}+b_{max}}{2} \right\rceil$
36:         $P :=$ OPF($G_d$, $v_s$, $v$, $V$, $l$, $b_{mid}$)
37:         if $|\Lambda(P)| \leq m$ then
38:             $b_{min} := b_{mid}$, $P_{min} := P$
39:         else
40:             $b_{max} := b_{mid}$, $P_{max} := P$
41:         end if
42:     end while
43:     return $(P, b_{mid})$
44: end procedure
```

- G_d: the discretised graph;
- v_s, v_t: the starting and terminating nodes;
- V: the set of nodes to be charged; initially $V = V(G_d)$;
- l: the recursion level, upper-bounded by L;
- b: the timespan budget of the charging journey.

OPF returns the optimum feasible path starting from v_s ending at v_t charging maximum number of nodes in V, whose timespan is upper-bounded by b, by invoking l recursions.

- OFP first checks the timespan budget b and returns $P = \emptyset$ if the budget is infeasible. OFP also returns $\emptyset$ if $c_d(v_s, v_t) = \infty$ as it follows from Lemma 1 that $c_d(v_s, v_t) = \infty$ implies there does not exist a path from v_s to v_t. Otherwise P is initialized to $c_d(v_s, v_t)$.
- If $l = 0$, meaning that the current instance is the last recursion, then OFP returns P.
- Otherwise, OFP iterates on each node $v \in V(G_d) - C(v_s) - C(v_t)$ and $m = [1..n_d]$ to recursively find: (1) P_1 with minimum timespan (denoted by b_1) starting from v_s ending at v that charges m nodes, each in a distinct clique, and (2) P_2 which starts from v and ends at v_t with timespan $B - b_1$ that charges maximum number of nodes, each in a distinct clique and different to the cliques in P_1. To find P_1, it follows from Lemma 1 that only nodes in V_{v-} need to be searched. Symmetrically, only nodes in $V_{v+} - C(P_1)$ need to be searched to find P_2.
- The output is the concatenation of P_1 and P_2 that charges the maximum number of nodes.

To find P_1 OFP calls the procedure BMIN, essentially a binary search function that returns the path with minimum timespan starting from v_s ending at v that charges m nodes, each in a distinct clique.

In the following three lemmas, we show that OFP indeed returns a feasible path (Lemma 2) and establish its time complexity (Lemma 3) and approximation ratio (Lemma 4).

Lemma 2 (Correctness of OFP)**.** *If $B \geq c_d(s, t)$, OFP returns a feasible path for any L, otherwise it returns $P = \emptyset$.*

PROOF. If $B < c_d(s, t)$, it follows from Line 3 of the pseudo-code of OFP that it returns $P = \emptyset$. We then prove the non-trivial case $B \geq c_d(s, t)$ by induction on L.

In the case where $L = 0$, it follows from Line 9 of the pseudo-code of OFP that it returns $P = (v_s, v_t)$ which is feasible.

Assume Lemma 2 holds for $L \leq L_0$, we now prove it holds for $L = L_0 + 1$. To that end, we consider the procedure OFP(G_d, v_s, v_t, V, $L_0 + 1$, b). From the pseudo-code of OFP and the induction result we have that P_1 and P_2 are either $\emptyset$ or feasible.

- If either $P_1 = \emptyset$ or $P_2 = \emptyset$, the current iteration is broken and P is not updated. Since P is initialised to (v_s, v_t) which is feasible, it holds that the path P returned at the end is feasible.
- If both P_1 and P_2 are feasible, it follows from the construction of P_1 and P_2 that their concatenation (P_1, P_2) is feasible. Since P is initialised to (v_s, v_t) which is feasible, it holds that the returned P at the end is feasible, no matter whether it is updated or not.

Lemma 2 thus holds for $L = L_0 + 1$. □

Lemma 3 (Time complexity of OFP)**.** *OFP terminates in $O\left((n_d \min(n_d, B) \log B)^L\right)$ time.*

PROOF. Let $\Upsilon(l)$ denote the time complexity of OFP under the recursion level l, let $m^* = |\Lambda(P_f^*)|$ where P_f^* denotes the optimum feasible path, we observe from the pseudo-code of OFP that

$$\Upsilon(l) = O([\min(n_d, m^*) \log b] \Upsilon(l-1)).$$

Since the edge costs are integers, i.e., $c_d(e) \geq 1$, it holds that $m^* \leq B$. Note that $\Upsilon(0) = O(1)$, by recursion we have $\Upsilon(l) = O\left((n_d \min(n_d, B) \log B)^L\right)$. □

Lemma 4 (Approximation ratio of OFP)**.** *Let P_f^* denote the optimum feasible path and let $m^* = |\Lambda(P_f^*)|$. Under the condition $L \geq \lceil \log m^* \rceil + 1$*[4]*, it holds that*

$$|\Lambda(P)| \geq \frac{|\Lambda(P_f^*)|}{1 + \lceil \log m^* \rceil}, \text{ i.e., } |\Lambda(P)| = \Omega\left(\frac{|\Lambda(P_f^*)|}{\log m^*}\right).$$

Proof. In the case $m^* = 1$, under the condition $L \geq 1$, it can be noted from the pseudo-code of OFP that it searches all nodes $v \in V(G_d) - C(s) - C(t)$ and returns a feasible path $P = (s, v, t)$. It holds that $|\Lambda(P)| = |\Lambda(P_f^*)| = 1$.

We now prove the lemma for $m^* \geq 2$. To that end, we decompose P_f^* to two sub-paths P_1^* and P_2^* where

$$|\Lambda(P_1^*)| = \left\lfloor \frac{m^*}{2} \right\rfloor, \quad |\Lambda(P_2^*)| = m^* - \left\lfloor \frac{m^*}{2} \right\rfloor = \left\lceil \frac{m^*}{2} \right\rceil.$$

Let $B_1^* = |\Lambda(P_1^*)|$ and $B_2^* = |\Lambda(P_2^*)|$. Let v_1^* denote the terminal node of P_1^*. Consider the procedure OPF(G_d, v_s, v_t, V, L, B) under the condition $L \geq \lceil \log m^* \rceil + 1$; we establish the relationship between $|\Lambda(P_i)|$ and $|\Lambda(P_i^*)|$ ($i = 1, 2$) as below:

- We first consider the call for BMin(G_d, v_s, v, V_{v_-}, $L-1$, b, m) with $m = \frac{\lfloor \frac{m^*}{2} \rfloor}{1 + \lceil \log(\frac{m^*}{2}) \rceil}$ and $v = v_1^*$. It follows from the condition $L \geq \lceil \log m^* \rceil + 1$ that

$$L - 1 \geq \lceil \log m^* \rceil \geq \left\lfloor \log \frac{m^*}{2} \right\rfloor + 1.$$

  It then follows from the induction that BMin returns a path P_1 with timespan $b_1 \leq B_1^*$ such that

$$|\Lambda(P_1)| \geq \frac{|\Lambda(P_1^*)|}{1 + \lceil \log \frac{m^*}{2} \rceil} = \frac{|\Lambda(P_1^*)|}{\lceil \log m^* \rceil}. \quad (4)$$

- On the other hand, it follows from the condition $L \geq \lceil \log m^* \rceil + 1$ that

$$L - 1 \geq \lceil \log m^* \rceil = \left\lceil \log \frac{m^*}{2} \right\rceil + 1.$$

  Denote $C(P_1) = \bigcup_{v \in P_1} C(v)$. We construct a graph G' whose node set is $V(G') = V(G_d) - C(P_1)$ and whose edges are the edges in G_d with end points in $V(G')$. We denote by $\Lambda'(P)$ the set of nodes in G' on path P. Define $P_2' \triangleq \arg \max_{\Gamma(P) \leq B - b_1} |\Lambda'(P)|$. We have

$$\begin{aligned} |\Lambda(P_2)| &\geq \frac{|\Lambda'(P_2')|}{\lceil \log \frac{m^*}{2} \rceil + 1} \geq \frac{|\Lambda'(P_2')|}{\lceil \log m^* \rceil} && (5) \\ &\geq \frac{|\Lambda'(P_2^*)|}{\lceil \log m^* \rceil} && (6) \\ &\geq \frac{|\Lambda(P_2^*)| - |\Lambda(P_1)|}{\lceil \log m^* \rceil} && (7) \\ &\geq \frac{|\Lambda(P_2^*)| - |\Lambda(P)|}{\lceil \log m^* \rceil}. && (8) \end{aligned}$$

  In the above inequalities, (5) follows from the induction by noticing that $B - b_1 \geq B - B_1^* = B_2^*$; (6) follows from the definition of P_2'; (7) follows from $|\Lambda'(P_2')| \geq |\Lambda(P_2^*)| - |\Lambda(P_1)|$ based on the definition of Λ'; (8) follows from $|\Lambda(P_1)| \leq |\Lambda(P_f)|$.

Since P_1 and P_2 pass distinct cliques, it follows from (4) and (8) that

$$|\Lambda(P)| = |\Lambda(P_1)| + |\Lambda(P_2)| \geq \frac{|\Lambda(P_1^*)| + |\Lambda(P_2^*)| - |\Lambda(P)|}{\lceil \log m^* \rceil},$$

which yields $|\Lambda(P)| \geq \frac{|\Lambda(P_f^*)|}{1 + \lceil \log m^* \rceil}$. Lemma 4 is proven. □

Lemma 2, Lemma 3 and Lemma 4 together lead to the main theorem below on the performance of OFP.

Theorem 4. *By setting $L = 1 + \lceil \log m^* \rceil$,* OFP *finds an $O(\log m^*)$-approximate optimum feasible path in quasi-polynomial time.*

In practice, to set L without knowing m^*, we need to estimate the upper-bound of m^*. Since the edge costs are integers, it holds that $m^* \leq B$. L can thus be set to B to ensure the $O(\log B)$ approximation.

5. DISCUSSION ON VARIANTS AND EXTENSIONS

We now discuss how our approximation algorithm can be adapted and extended to solve other charging path optimization and scheduling problems formulated in Section 3.

For the problem where the charger has a battery reservoir and aims at charging the maximum number of nodes before returning to its service station to replenish itself, we can set the edge cost between v_i and v_j ($v_i, v_j \in V(G_d)$) to the energy required to charge v_i to $(1-\epsilon)\alpha$ plus the energy consumption to move from v_i to v_j. Our algorithm OFP can then be invoked to find the $O(\log m^*)$-optimum solution. In a broader sense, our approach can be adopted to solve the first category of the charging path optimization and scheduling problems formulated in Section 3.

We next focus on the second class of charging path optimization problems where the charger has a number of nodes M to charge and it seeks a path with minimum cost (e.g. in terms of time, energy consumption) to accomplish the charging task. This class of problems are NP-hard because when nodes are stationary and charging is immediate and $M = n$, the problems degenerate to the classical TSP problem which is NP-hard. To solve these problems, we devise a recursive algorithm similar to OFP, summarized below:

- For each $m = [1..M]$ and each node $v \in V(G_d) - \{s, t\}$:
  - Recursively search a path P_1 from s to v of minimum timespan (or any form of budget) charging m nodes, denote the timespan of P_1 by b_1;
  - Recursively search another path P_2 from v to t of minimum timespan charging $M - m$ nodes;
- Output the concatenated path $P = (P_1, P_2)$ of minimum timespan;

Using the similar analysis as OFP, we can establish the logarithmic approximation ratio of the above algorithm.

In many practical scenarios, some nodes are more critical than others. Instead of seeking $P^* = \text{argmin}_{P \in \mathcal{P}} |\Lambda(P)|$, it makes more sense to solve $P^* = \text{argmin}_{P \in \mathcal{P}} \sum_{v_i \in \Lambda(P)} w_i$, where w_i is a weight for node v_i. Our algorithm OFP can be readily applied to solve such weighted version by attributing a reward w_i to v_i and by adjusting the objective to finding the path maximizing the collected reward.

[4]All logarithms in our analysis are to base 2. In the case where $m^* = 0$, meaning that the optimum feasible path cannot pass any node other than s and t, it holds that $|\Lambda(P_f^*)| = 0$, Lemma 4 holds trivially for any L.

Our algorithm can also be adapted to solve the charging path optimization with time windows where node v_i needs to be charged within a time window. Specifically, this can be done by redefining the feasible charging path such that only nodes charged within its time window is counted in $\Lambda(P)$.

6. NUMERICAL ANALYSIS

In this section, we conduct numerical analysis to evaluate the performance of our approximation algorithm. We evaluate our algorithm w.r.t. the following two algorithms:

- The random algorithm where the charger randomly targets a node to charge if the remaining budget allows it to arrive at t after charging;
- The greedy algorithm where the charger targets the nearest node to charge if the remaining budget allows it to arrive at t after charging.

Specifically, we set up a simulation area of an Euclidean square $[0, 10000]^2$ (in meters) and randomly deploy a number of n nodes in the square, where n varies from 20 to 100. The starting and terminating points of the charger are located at $(5000, 5000)$ and $(10000, 10000)$. The traveling speed of the charger is set to $r_s = 5$ m/s. The network nodes follow the random way point model with maximum speed r_i $(1 \leq i \leq n)$ randomly chosen in $[0, 2]$ m/s with no pause.

For the battery model, we choose a regular NiMH battery with the nominal cell voltage and the quantity of electricity being 1.2 V/2.5 Ah [36]. We have $E_{max} = 1.2\text{V}\times 2.5\text{A}\times 3600\text{s}=$ 10.8 kJ. We set the initial battery level of v_i to $e_i E_{max}$ with e_i randomly chosen from $[0, 1]$. We set $\alpha = 90\%$ et $\epsilon = 0.1$. We simulate two charger profiles where the first one corresponds to a more powerful charger with the average energy transfer rate being 50 W and the second one is less powerful with the average energy transfer rate being 20 W. We set $\Delta t = 0.1$s and $\lambda = 100$. The budget B is set to 6 hours.

We trace the following metric to evaluate the performance of our algorithm compared to the random algorithm:

$$\Upsilon = \frac{\text{Number of nodes charged by our algorithm}}{\text{Number of nodes charged by random algorithm}}.$$

Similarly, we define the same metric Υ' over the greedy algorithm. For each set of chosen parameters, we run a number of independent simulations where the nodes' positions are randomly chosen and the required number of simulation runs is calculated using "independent replications" [37].

The simulation results are shown in Figure 4 with an confidence interval 90%. The simulation results demonstrate that our algorithm consistently outperforms both the random and the greedy algorithms under both charger profiles. The performance gain is more significant when the charger is more powerful. This is because a more powerful charger implicates less charging time. Hence the performance gain on charging path optimization becomes more visible.

We then study the impact of the algorithm parameters, particularly the number of recursion levels L and the time stepsize Δt, on the performance of our approximation algorithm. We thus simulate the same network as the previous set of simulations and trace the number of charged nodes $\Lambda(P)$ as functions of L and Δt where P is the charging path output by our approximation algorithm. The results, illustrated in Figure 5, demonstrate that by increasing L and/or decreasing Δt, we get better performance at the price of increased computation complexity. The performance improvement is less significant when L is sufficiently large and Δt is sufficiently small. In this regard, our algorithm allows to tradeoff between the performance gain and the computation complexity by tuning L and Δt. We observe from Figure 5 that in the simulated scenario our algorithm is able to achieve reasonably good performance with relatively light computation complexity with a mild setting of L and Δt.

7. CONCLUSION AND FUTURE WORK

In this paper, we have studied a class of generic charging path optimization problems arising from emerging networking applications where mobile chargers are dispatched to delivery energy to mobile agents, which have specified tasks and mobility patterns. We have instantiated our analysis by focusing on finding the charging path maximizing the number of nodes charged within a fixed time horizon. We established the APX-hardness of the problem. We designed a quasi-polynomial time algorithm that achieves polylogarithmic approximation to the optimum charging path. Our algorithm can be further adapted and extended to solve a variety of path optimization problems with realistic constraints, such as limited time and energy budget.

There are a number of research directions we plan to investigate following our work in this paper. The first is to study the case where the trajectories of nodes are partially known, or even unknown to the charger. In this context, the charger seeks a charging path that maximizes the expected number of nodes charged. The second direction is to use the methodology in the paper to study more sophisticated variants of the problem, e.g., the case of multiple chargers with heterogeneous moving speed.

8. ACKNOWLEDGEMENTS

The work of S. Lin and H. Huang is supported by NSF awards: NSF CNS 1536086, CNS 1463722, and IIS 1460370.

Appendix: Proof of Theorem 3

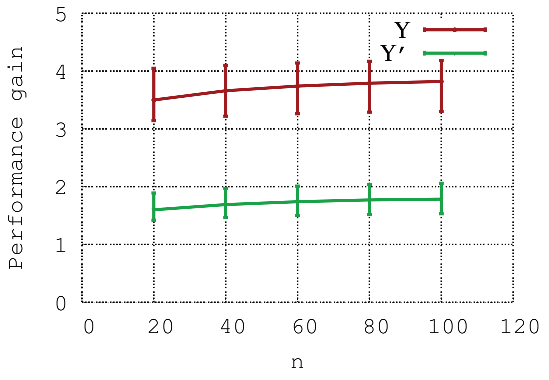
Without loss of generality, assume the optimum charging path P^* starts from s, then charges M nodes sequentially from v_1 to v_M before terminating at t. By slightly introducing ambiguity, we denote $v_{n+1}(k_{n+1}\Delta t)$ as $v_{M+1}(k_{M+1}\Delta t)$ requiring 0 time to be charged for notation and analysis convenience. We also denote s and t as two stationary nodes v_0 and v_{M+1} requiring 0 time to be charged. Let T_i $(1 \leq i \leq M)$ denote the time when the charger starts charging v_i and let $k_i\Delta t$ denote the discretized time instance where $(k_i - 1)\Delta t < T_i \leq k_i\Delta t$. The core part of the proof consists of (1) proving that the edges $\overrightarrow{v_i(k_i\Delta t)v_{i+1}(k_{i+1}\Delta t)}$ $(0 \leq i \leq M)$ exist in G_d, (2) showing that concatenating these edges yields a feasible path that is also an ϵ-optimum charging path.

We first prove the existence of $\overrightarrow{v_i(k_i\Delta t)v_{i+1}(k_{i+1}\Delta t)}$ $(0 \leq i \leq M)$. Recall that (1) T_i and T_{i+1} are the time when the charger starts charging v_i and v_{i+1} and (2) v_i is charged to at least α, we have

$$T_{i+1} - T_i \geq g_i(\alpha) + \frac{|\overline{v_i(T_i)v_{i+1}(T_{i+1})}|}{r_s}. \tag{9}$$

Recall that $(k_i - 1)\Delta t < T_i \leq k_i\Delta t$ for all i, it holds that

$$T_{i+1} - T_i \leq (k_{i+1} - k_i + 1)\Delta t. \tag{10}$$



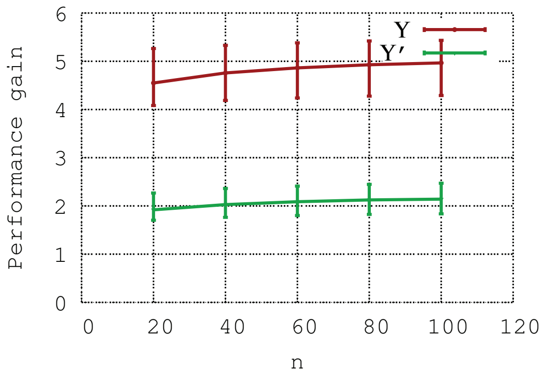
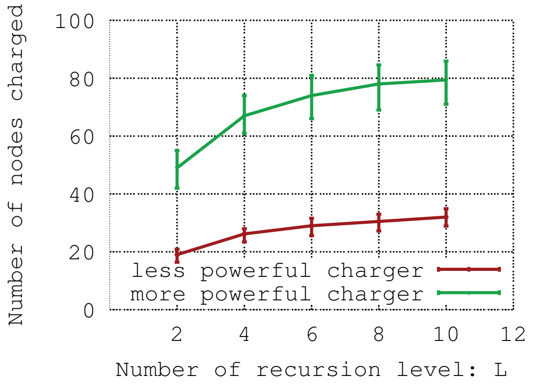


Figure 4: Performance gain of our algorithm over the random algorithm and the greedy algorithm as functions of n with less powerful charger (left) and more powerful charger (right).



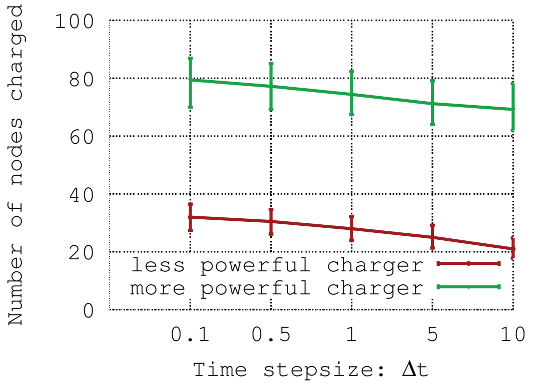


Figure 5: Performance of our algorithm as functions of the recursion level L (left) and time stepsize Δt (right).

On the other hand, applying the triangle inequality, we have:

$$|\overline{v_i(k_i\Delta t)v_{i+1}(k_{i+1}\Delta t)}| \le |\overline{v_i(k_i\Delta t)v_i(T_i)}| + |\overline{v_i(T_i)v_{i+1}(T_{i+1})}| + |\overline{v_{i+1}(T_{i+1})v_{i+1}(k_{i+1}\Delta t)}|. \quad (11)$$

Noticing that r_i is the speed upper-bound of v_i and $r_i < r_s$, we have

$$\begin{cases} |\overline{v_i(k_i\Delta t)v_i(T_i)}| \le r_i(k_i\Delta t - T_i) \le r_s\Delta t, \\ |\overline{v_i(T_i)v_{i+1}(T_{i+1})}| \le r_i(k_{i+1}\Delta t - T_i) \le r_s\Delta t. \end{cases}$$

Injecting the above inequalities into (11) yields

$$|\overline{v_i(T_i)v_{i+1}(T_{i+1})}| \ge |\overline{v_i(k_i\Delta t)v_{i+1}(k_{i+1}\Delta t)}| - 2r_s\Delta t. \quad (12)$$

By injecting (9) and (10) into (12), we have

$$(k_{i+1}-k_i)\Delta t \ge g_i(\alpha) + \frac{|\overline{v_i(k_i\Delta t)v_{i+1}(k_{i+1}\Delta t)}|}{r_s} - 3\Delta t. \quad (13)$$

On the other hand, it follows from the properties of $f_i(t)$ and $g_i(x)$ that for any ϵ, it holds that

$$g_i(\alpha) - g_i[(1-\epsilon)\alpha] \ge g_i'[(1-\epsilon)\alpha]\epsilon\alpha. \quad (14)$$

By setting $\Delta t \le \frac{\alpha\epsilon}{3}\min_{1\le i\le n} g_i'[(1-\epsilon)\alpha]$ and recall that $g_i(x)$ is increasing and convex in x, we have

$$3\Delta t \le g_i(\alpha) - g_i[(1-\epsilon)\alpha], \ \forall i.$$

We can then rewrite (13) as follows:

$$(k_{i+1}-k_i)\Delta t \ge g_i[(1-\epsilon)\alpha] + \frac{|\overline{v_i(k_i\Delta t)v_{i+1}(k_{i+1}\Delta t)}|}{r_s}.$$

It follows from (1) that $\overrightarrow{v_i(k_i\Delta t)v_{i+1}(k_{i+1}\Delta t)}$ exists in G_d.

We then prove that by concatenating the following edges $\overrightarrow{v_i(k_i\Delta t)v_{i+1}(k_{i+1}\Delta t)}$ $(0 \le i \le M)$, we can construct a feasible path that is also an ϵ-optimum charging path. To prove this, we construct the charging schedule of the concatenated charging path as follows: the charger starts from $v_0(0)$ (i.e., s) and then passes nodes $v_i(k_i\Delta t)$ sequentially from $i = 1$ to M before terminating at $v_{M+1}(k_{M+1}\Delta t)$ (i.e., t), during which the charger starts charging node v_i $(1 \le i \le M)$ at time $k_i\Delta t$ for $g_i[(1-\epsilon)\alpha]$ time and then goes straightly to node v_{i+1}. It follows from (1) that under this charging schedule, the charger can charge nodes v_i $(1 \le i \le M)$ to $(1-\epsilon)\alpha$. Moreover, it follows from the path construction that the charger arrives at $v_{M+1}(k_{M+1}\Delta t)$ (i.e., $v_{n+1}(k_{n+1}\Delta t)$ with our notation) at time $k_{n+1}\Delta t$. Recall $K_{n+1} = \lfloor \frac{B}{\Delta t} \rfloor$, it holds that $k_{n+1}\Delta t \le B$. It follows from Definition 1 that the constructed path is an ϵ-optimum charging path.

9. REFERENCES

[1] L. Xie, Y. Shi, Y. T. Hou, W. Lou, H. D. Sherali, H. Zhou, and S. F. Midkiff, "A mobile platform for wireless charging and data collection in sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 8, pp. 1521–1533, 2015.

[2] N. Atanasov, J. Le Ny, and G. J. Pappas, "Distributed Algorithms for Stochastic Source Seeking with Mobile Robot Networks," *ASME J. Dynamic Syst., Measurement, Control*, 2014.

[3] L. Di Puglia Pugliese, F. Guerriero, D. Zorbas, and T. M. Razafindralambo, “Modelling the mobile target covering problem using flying drones,” *Optimization Lett.*, Aug. 2015.

[4] H. Qin and W. Zhang, “Charging scheduling with minimal waiting in a network of electric vehicles and charging stations,” in *Proc. ACM VANET 2011*.

[5] B. Coltin and M. Veloso, “Mobile robot task allocation in hybrid wireless sensor networks,” in *Proc. IEEE/RSJ Intl. Conf. Intelligent Robots Syst. (IROS)*, 2010.

[6] P. Toth and D. Vigo, eds., *The Vehicle Routing Problem.* MOS/SIAM Series on Optimization, 2001.

[7] A. Haghani and S. Jung, “A dynamic vehicle routing problem with time-dependent travel times,” *Comput. Oper. Res.*, 2005.

[8] P. Ni, H. T. Vo, D. Dahlmeier, W. Cai, J. Ivanchev, and H. Aydt, “DEPART: Dynamic route planning in stochastic time-dependent public transit networks,” in *Proc. IEEE Conf. Intelligent Transportation Syst. (ITSC)*, 2015.

[9] F. Miao, S. Han, S. Lin, and G. J. Pappas, “Taxi dispatch under model uncertainties.,” in *Proc. IEEE CDC*, 2015.

[10] J. Yang, P. Jaillet, and H. Mahmassani, “Real-time multivehicle truckload pickup and delivery problems,” *Transportation Sci.*, vol. 38, pp. 135–148, 2004.

[11] D. Zhang, T. He, S. Lin, S. Munir, and J. Stankovic, “Online cruising mile reduction in large-scale taxicab networks,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 11, pp. 3122–3135, 2015.

[12] R. Sugihara and R. K. Gupta, “Speed control and scheduling of data mules in sensor networks,” *ACM Trans. Sen. Netw.*, vol. 7, Aug. 2010.

[13] D. Ciullo, G. Celik, and E. Modiano, “Minimizing transmission energy in sensor networks via trajectory control,” in *Proc. IEEE WiOpt*, 2010.

[14] G. Xing, T. Wang, W. Jia, and M. Li, “Rendezvous design algorithms for wireless sensor networks with a mobile base station,” in *Proc. ACM MobiHoc*, 2008.

[15] L. Chen, W. Wang, H. Huang, and S. Lin in *Proc. IEEE INFOCOM, title=Time-constrained Data Harvesting in WSNs: Theoretical Foundation and Algorithm Design, year=2015.*

[16] E. Ekici, Y. Gu, and D. Bozdag, “Mobility-based communication in wireless sensor networks,” *IEEE Comm. Mag.*, vol. 44, pp. 56–62, July 2006.

[17] J. Reich, V. Misra, D. Rubenstein, and G. Zussman, “Connectivity maintenance in mobile wireless networks via constrained mobility,” *IEEE J. Sel. Areas Commun.*, vol. 30, no. 5, pp. 935–950, 2012.

[18] Y. Hu, X. Wang, and X. Gan, “Critical sensing range for mobile heterogeneous camera sensor networks,” in *Proc. IEEE INFOCOM*, pp. 970–978, 2014.

[19] H. Huang, S. Lin, L. Chen, J. Gao, A. Mamat, and J. Wu, “Dynamic mobile charger scheduling in heterogeneous wireless sensor networks,” in *Proc. IEEE MASS*, 2015.

[20] S. Zhang, J. Wu, and S. Lu, “Collaborative mobile charging,” *IEEE Trans. Computers*, vol. 64, no. 3, pp. 654–667, 2015.

[21] Y. Peng, Z. Li, W. Zhang, and D. Qiao, “Prolonging sensor network lifetime through wireless charging,” in *Proc. IEEE RTSS*, 2010.

[22] L. Xie, Y. Shi, Y. T. Hou, W. Lou, and H. D. Sherali, “On traveling path and related problems for a mobile station in a rechargeable sensor network,” in *Proc. ACM MobiHoc*, 2013.

[23] L. He, L. Fu, L. Zheng, Y. Gu, P. Cheng, J. Chen, and J. Pan, “Esync: An energy synchronized charging protocol for rechargeable wireless sensor networks,” in *Proc. ACM MobiHoc*, ACM, 2014.

[24] H. Dai, G. Chen, C. Wang, S. Wang, X. Wu, and F. Wu, “Quality of energy provisioning for wireless power transfer,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 2, pp. 527–537, 2015.

[25] G. Laporte, “The traveling salesman problem: An overview of exact and approximate algorithms,” *European J. Oper. Res.*, 1992.

[26] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study: A Computational Study.* Princeton Univ. Press, 2011.

[27] R. Bar-Yehuda, G. Even, and S. M. Shahar, “On approximating a geometric prize-collecting traveling salesman problem with time windows,” *J. Algorithms*, vol. 55, no. 1, pp. 76–92, 2005.

[28] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff, “Approximation algorithms for orienteering and discounted-reward tsp,” *SIAM J. Comp.*, vol. 37, no. 2, pp. 653–670, 2007.

[29] C. Chekuri, N. Korula, and M. Pál, “Improved algorithms for orienteering and related problems,” *ACM Trans. Algorithms*, vol. 8, no. 3, p. 23, 2012.

[30] N. Bansal, A. Blum, S. Chawla, and A. Meyerson, “Approximation algorithms for deadline-tsp and vehicle routing with time-windows,” in *Proc. ACM STOC*, pp. 166–174, 2004.

[31] A. Blum, S. Chawla, D. Karger, T. Lane, A. Meyerson, and M. Minkoff, “Approximation algorithms for orienteering and discounted-reward tsp,” *SIAM J. Comp.*, vol. 37, no. 2, pp. 653–670, 2007.

[32] C. Chekuri, N. Korula, and M. Pál, “Improved algorithms for orienteering and related problems,” *ACM Trans. Algorithms*, vol. 8, pp. 1–27, July 2012.

[33] C. Helvig, G. Robins, and A. Zelikovsky, “The moving-target traveling salesman problem,” *J. Algorithms*, vol. 49, no. 1, pp. 153 – 174, 2003.

[34] C. Chekuri and M. Pal, “A recursive greedy algorithm for walks in directed graphs,” in *Proc. IEEE FOCS*, 2005.

[35] “A greedy approximation algorithm for the group steiner problem,” *Discrete Applied Mathematics*, vol. 154, no. 1, pp. 15 – 34, 2006.

[36] D. Linden and T.B.Reddy, *Handbook of Batteries.* McGraw-Hill, 2002.

[37] W. Whitt, “The efficiency of one long run versus independent replications in steady-state simulation,” *Management Sci.*, vol. 37, no. 6, pp. 645–666, 1991.