SMT-Based Optimal Deployment of Mobile Rechargers

Tanmoy Kundu[1] and Indranil Saha[2]

***Abstract*— Efficient recharging is an essential requirement for autonomous mobile robots. In an indoor robotic application, charging stations can be installed offline. However, frequent trips to the charging stations cause inefficiency in the performance of the mobile robots. In an outdoor environment, a charging station cannot even be installed easily. We propose a framework and algorithms for enabling a group of mobile wireless rechargers to fulfill the energy requirement of autonomous mobile robots in a workspace efficiently. Our algorithm finds the optimal trajectories for the mobile rechargers in such a way that once there is a need for a recharge, the robots do not need to spend significant time and energy to get access to a recharger. Our algorithm is based on a reduction of the problems to Satisfiability Modulo Theory (SMT) solving problems. We present extensive experimental results to show that the optimal trajectories for mobile rechargers can be generated successfully for different types of robots and workspaces within a reasonable time. Moreover, a comparison with the performance of static charging stations establishes that mobile rechargers are more effective in terms of allowing the autonomous robot to continue their work for a longer time.**

I. Introduction

Autonomous mobile robots have the potential to be useful for a wide range of indoor and outdoor applications such as persistent surveillance [1], assembly planning [2], evacuation [3], search and rescue [4], and object transportation [5]. These robots are generally battery-powered. Their batteries need to be recharged at a regular interval to keep the robots operational for a long time. The efficiency of the robotic system depends significantly on how effective the power management mechanism is for the robots.

Three major approaches are employed to deal with the recharging problem of mobile robots. In the first approach, the robot needs to visit a static charging station to get its battery recharged, which is known as docking based autonomous recharging [6], [7], [8], [9], [10], [11], [12], [13], [14]. In the second approach, the robot is kept connected with a charging station using a long tether to provide it with an uninterrupted power supply [15], [16], [17]. In the third approach, natural energy resources like the sun and the slugs are exploited as energy resources for the robots [7], [18], [19]. These mechanisms either require pre-installed infrastructure or can operate in selective environments and thus not suitable for quick and unplanned deployment.

In this paper, we explore an alternative approach to address the energy supply problem for mobile robots. Our approach is based on employing mobile rechargers that move towards the energy-deficient worker robots whenever required and supply them with the needed energy. This approach is motivated by several mobile charging solutions proposed in recent times [20], [21]. Though the mobile rechargers have been developed keeping electric vehicles in mind, there is a vast potential for this technology to be useful for various multi-robot applications where worker robots need to reduce time and energy to travel towards charging stations or in scenarios where static recharger installation is difficult.

A few recent works have addressed the planning problem for a set of mobile recharger robots to support a specific set of worker robots carrying out some pre-determined tasks [8], [22], [23], [24], [25]. However, for many practical applications, we need to deploy mobile recharger robots to cover a workspace in such a way that the worker robots can access them easily irrespective of their working trajectories and tasks. Deploying mobile rechargers in a workspace has the following major challenge: how to determine the coverage area and trajectory loops of the mobile rechargers so that they can cover the workspace completely, and thereby require the worker robots to spend an insignificant amount of energy to move to a recharger? The dynamic constraints of the robots (both workers and rechargers) and the presence of the obstacles in the workspace add significant challenges to the problem.

Being motivated by the recent successes of the SMT solvers to solve robot motion planning problems [26], [27], [28], [29], [30], [31], [32], we design SMT based solutions to the above-mentioned motion planning problem for mobile rechargers. We present a naive SMT encoding and a global and a local optimization method over the naive algorithm. Our experimental results show that our algorithms can successfully generate optimal trajectories for the rechargers for different types of worker robots and workspaces in a reasonable time. The optimization techniques help in decreasing the trajectory lengths of the mobile rechargers. Moreover, the local optimization approach is significantly faster than the global optimization approach while providing results at par with the global optimization technique. To the best of our knowledge, this paper is the first to address the optimal path planning problem for a group of mobile rechargers serving an arbitrary set of mobile worker robots in a workspace.

II. Problem

A. Preliminaries

Workspace ($\mathcal{W}$). In this work, we assume that the robots operate in a 2-D workspace, which we represent as a 2-D occupancy grid map. The grid decomposes the workspace into square-shaped blocks, which are assigned unique identifiers to represent their locations in the workspace. We denote the set of locations in the workspace by $\mathcal{W}$ and the set of locations covered by obstacles by $\mathcal{O}$. The set of obstacle-free locations in the workspace is denoted by $\mathcal{F} = \mathcal{W} \setminus \mathcal{O}$. For an obstacle-free grid location $p \in \mathcal{F}$, its *neighbourhood* is

*Tanmoy Kundu is supported by Visvesvaraya Ph.D. Fellowship by the Ministry of Electronics and Information Technology, Government of India.

[1]Tanmoy Kundu and [2]Indranil Saha are with the Department of Computer Science and Engineering, Indian Institute of Technology Kanpur {tanmoy, isaha}@cse.iitk.ac.in .

defined as any obstacle-free location p' which is one unit distance (in any direction) away from p, i.e., $\mathcal{N}(p) = \{p' \mid p' \in \mathcal{F} \ \wedge \ |p'.x - p.x| \leq 1 \ \wedge \ |p'.y - p.y| \leq 1\}$, where $p.x$ and $p.y$ denote the x and y coordinates of grid location p.

Robot State (σ). The *state* of a robot σ consists of (1) its position in the workspace, $\sigma.p$, which determines a unique block in the occupancy grid, and (2) its velocity configuration, $\sigma.v$, which represents the current magnitude and direction of the velocity of the robot. We denote the set of all velocity configurations by V and assume that it contains a value v_0 denoting that the robot is stationary.

Motion Primitive (γ). We capture the motion of a robot using a set of *motion primitives* Γ. We assume that the robot moves in an occupancy grid in discrete steps of τ time units. A motion primitive is a short controllable action that the robot can perform in any time step. A robot can move from its current location to a destination location by executing a sequence of motion primitives.

With each motion primitive $\gamma \in \Gamma$, we associate a *pre-condition* $pre(\gamma)$, which is some propositional formulas over the states specifying the condition under which a motion can be executed. We write $post(\sigma, \gamma)$ for the state of a robot after the motion primitive γ is applied to a state σ satisfying $pre(\gamma)$. We use $intermediate(\sigma, \gamma)$ to denote the set of grid blocks through which the robot may traverse when γ is applied at state σ, including the beginning and the end blocks. Each motion primitive γ is associated with an energy cost as denoted by $cost(\gamma)$. To simplify the exposition, we assume that the execution of any motion primitive for a robot leads to the same amount of energy expenditure.

We assume that in Γ, there exists a motion primitive that can be applied when the robot is at the velocity configuration v_0, and it keeps the robot in the same state. This special primitive is called the *rest primitive*.

Motion Plan (ρ) and Trajectory (ξ). The run-time behavior of a robot is described by a discrete-time transition system $\mathcal{T}$. Let σ_1 and σ_2 be two states of the robot and γ be the motion primitive applied to the robot in state σ_1. We define a transition $\sigma_1 \xrightarrow{\gamma} \sigma_2$ iff

- $\sigma_1 \models pre(\gamma)$, $\sigma_2 = post(\sigma_1, \gamma)$, and
- the trajectory of the robot between the states σ_1 and σ_2 does not pass through a block occupied by an obstacle, i.e., $intermediate(\sigma_1, \gamma) \cap O = \emptyset$.

A *motion plan* for a robot is defined as a sequence of motion primitives to be applied to the robot to move it from a location $l_i \in F$ to another location $l_f \in F$. A motion plan is denoted by $\rho = (\gamma_1 \ldots \gamma_k)$, where $\gamma_i \in \Gamma$ for all $i \in \{1, \ldots, k\}$.

Given the current location of the robot l_0 and a motion plan $\rho = (\gamma_1 \ldots \gamma_k)$, the *trajectory* of the robot is given by $\xi = (\sigma_0 \sigma_1 \ldots \sigma_k)$ such that for all $i \in \{1, \ldots, k\}$, $\sigma_{i-1} \xrightarrow{\gamma_i} \sigma_i$. If any transition is invalid then the motion plan does not lead to a valid trajectory. In the rest of the paper, we use the word "step" to denote a transition governed by a motion primitive.

B. System Model

In our framework, the robots are categorized as *worker robots* (*workers*) and *recharger robots* (*rechargers*) based on their functionalities. Workers traverse the workspace to perform some tasks, while the sole purpose of the rechargers is to recharge the batteries of the workers. A recharger's trajectory (derived by our algorithm) essentially makes a loop that it traverses repeatedly. Without loss of generality, let us assume that the length of the trajectory loops of all the rechargers are the same and is denoted by ncp (number of charging points).

We assume that the workers have access to a motion planner $\mathcal{M}$ that has the knowledge of the trajectory loops of the rechargers. While carrying out its task, when the battery charge comes down to a threshold level (d), the robot aborts its task and invokes $\mathcal{M}$ with its current location. This event may happen at any obstacle-free location in the workspace. The motion planner computes the shortest trajectory from the current location of the workspace to some neighbouring location on the trajectory of a mobile recharger. This motion planning problem is simple and can be solved using any graph-search based planning algorithm like A* [33]. We assume that each motion primitive (transition) of every worker requires a unit amount of energy. We assume a simple battery model, and by the threshold d, we represent the maximum number of transitions the robot will be able to take before getting devoid of energy. To recharge its battery, the robot should be able to reach one of the neighbouring grid locations on the trajectory of a recharger within a pre-decided threshold (d) number of transitions. Our goal is to design the trajectories of the recharger robots in such a way that a worker robot, from any location, is capable of reaching at least one of the recharger robots within d steps.

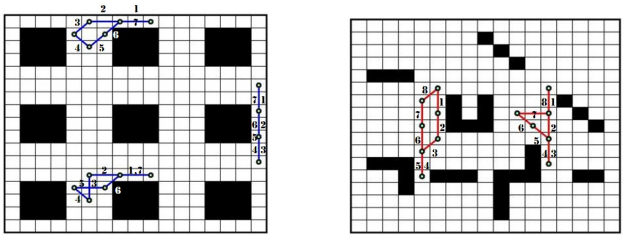
To perform the recharge operation, a worker computes its trajectory and communicates its end location to the recharger. From this message, the recharger gets to know the exact trajectory point where it can meet the worker. Now, both the worker and the recharger start moving simultaneously to their designated destinations. We assume that the execution of a motion primitive by both worker and recharger takes an equal amount of time. We also assume that the trajectory generation for the worker and the communication with the recharger incur negligible energy and time. With these assumptions, we define the following two performance measures of the proposed recharge mechanism:

- *Maximum energy consumption* for the worker robot to complete a recharge operation, which is given by d.
- *Maximum wait time*, denoted by w, which is given by $\max(d, ncp - 1)$. This expression can be derived from the fact that the worker and the recharger start moving simultaneously, and the recharger needs to move at most $(ncp - 1)$ steps, assuming conservatively that it can move only in one direction following its trajectory loop.

Note that, for static rechargers ($ncp = 1$), both *maximum energy consumption* and *maximum wait time* are same — d.

C. Plan Synthesis Problem

The planning problem for the mobile rechargers involves three parameters: the threshold on the number of transitions (d), the length of the trajectory loops of the rechargers (ncp, for the number of charging points), and the number of mobile rechargers ($nrech$). While it may be possible to solve the problem by co-optimizing these three parameters, it



(a) Warehouse (b) Artificial floor

Fig. 1. Trajectories of multiple rechargers deployed to recharge worker robots in Warehouse and Artificial floor workspaces (dimension 17×17).

is computationally challenging. It also requires deciding the weights of these three parameters appropriately to formulate the multi-objective optimization problem. We rather assume that two of the three parameters are given, and our goal is to optimize the third parameter, along with finding the trajectories of the rechargers. This leads to three different optimization problems. Below, we formally define the problem for the case where we aim to find the optimal value of ncp for a given d and $nrech$. The other two problems can be defined similarly.

Problem 2.1: Given a workspace $\langle \mathcal{F}, \mathcal{O} \rangle$, a set of recharger robots $R = \{r_1, \ldots, r_{nrech}\}$, and the threshold on the number of transitions d (which is also the battery charge threshold), minimize the number of points ncp on the trajectory of the recharger robots, and compute each r_i's trajectory loop $\xi^{r_i} = (\sigma_0^{r_i} \sigma_1^{r_i} \sigma_2^{r_i} \ldots \sigma_{ncp}^{r_i})$, such that a worker can reach the neighbourhood of the trajectory of at least one of the rechargers from any location $l_f \in \mathcal{F}$ in the workspace by executing at most d number of transitions.

In the above-mentioned problems, we consider the length of the trajectories for all the rechargers to be the same. As it will be evident later, the assumption of a fixed length for the trajectories of all the rechargers leads to the design of a simple algorithm. However, if we allow different rechargers to have trajectories of different lengths, it may be possible that we can obtain shorter trajectory loops for some of the rechargers. We will present two algorithms to achieve this goal.

D. Example

Figure 1 shows two different workspaces (Warehouse and Artificial floor) of dimension 17×17. Consider that a worker robot is a Turtlebot [34] which can move to one of the eight adjacent cells from its current cell in one step. In this example, we set the charging threshold d to 5, and use a fixed number of recharger robots (for Warehouse, the number is 3, and for Artificial floor, it is 2). Our aim is to find the optimal length trajectory loops of the rechargers such that whenever a worker needs to recharge its battery, it can reach at least one of the rechargers' trajectories within $d = 5$ steps.

Consider that a recharger robot has similar dynamics as any worker robot, plus four primitives (in N, S, E, W directions and two steps forward). The trajectories of the rechargers, as generated by our algorithm, are shown in Figure 1. Note that for a recharger, any two adjacent trajectory points (black circles) are connected by a single motion primitive of the recharger. Figure 1(a) and Figure 1(b) show the trajectories when we consider their lengths to be equal. Note that in generating the trajectory loops for the rechargers, the planner may use the rest primitives as is the case for the right-most recharger in the Warehouse example (transition 5). The transition corresponding to the rest primitive can be safely removed from the trajectory.

In the next section, we present a basic algorithm for generic inputs, followed by two more algorithms to further optimize the trajectory loop lengths by forcing the use of rest primitives in the trajectory loops wherever possible.

Algorithm III.1: Compute the optimal value of decision parameter ($dparam$) and rechargers' trajectories

Input: $\mathcal{F}$: the set of obstacle-free blocks, $\mathcal{O}$: the set of obstacles,
Γ^r: the set of motion primitives for the recharger,
Γ^w: the set of motion primitives for any worker,
$fixed$: the values of the fixed parameters.

Output: $dparam$: the value of the decision parameter
$rtraj$: the vector containing the rechargers' trajectories,
$wtraj$: the vector containing the workers' trajectories for each grid cell in $\mathcal{F}$.

```
function generate_trajectories (F, O, Γ^r, Γ^w, fixed)
begin
    dparam := 1;
    while true do
        C :=
          generate_constraints (F, O, Γ^r, Γ^w, fixed, dparam);
        [result, model] := solve_constraints(C);
        if result = SAT then
            [rtraj, wtraj] :=
              extract_trajectories(model);
            return [dparam, rtraj, wtraj];
        else
            dparam := dparam + 1;
        end
    end
end
```

III. Algorithm

In this section, we provide mechanisms to solve the problems introduced in Section II algorithmically by reducing them to a sequence of SMT solving problems.

A. Basic Algorithm for Finding Rechargers' Trajectories

We introduce a basic and *generic* algorithm (Algorithm III.1) for generating the trajectories for the rechargers, where any two of the three parameters d, ncp, and $nrech$ are given. The other goal of Algorithm III.1 is to find the optimal value of the third parameter.

The function `generate_trajectories` takes as input the set of obstacle-free cells $\mathcal{F}$, the set of obstacle occupied cells $\mathcal{O}$, the set of motion primitives for the recharger robots Γ^r and for the worker robots Γ^w, and the fixed parameter values $fixed$. The goal of the algorithm is to find the minimum value of the third parameter $dparam$ such that the trajectories for the rechargers exist to cover all the obstacle-free locations in the workspace. To find the minimum value of $dparam$, we start with $dparam$=1 and keep on increasing it until we get a solution. For each value of $dparam$, we formulate an SMT-solving problem C using `generate_constraints` function. The constraint C is then fed to an SMT solver. If the solver fails (UNSAT) to produce a solution, it indicates that there do not exist trajectories

for the recharger robots for the given values of the fixed parameters and the current value of the decision parameter. So, we increase the value of the decision parameter $dparam$ by 1 and repeat the same procedure until it achieves satisfiability (SAT). In an iteration, if the solver produces a $model$ ($result$ is SAT), using the model, we extract the rechargers' trajectories ($rtraj$) and the workers' trajectories ($wtraj$) from any obstacle-free location in $\mathcal{F}$ to one of the rechargers.

Below we present the constrains C for a *specific* input instance of Algorithm III.1 where d and $nrech$ are fixed parameters and ncp is the decision parameter $dparam$. Other input instances can be handled in a similar way.

$$\forall r_i \in R.\ \xi^{r_i} = (\sigma_0^{r_i}\sigma_1^{r_i}\ldots\sigma_{ncp}^{r_i})\ \texttt{with}\ \sigma_{ncp}^{r_i} = \sigma_0^{r_i}\ \wedge \quad \text{(III.1a)}$$

$$\forall l_f \in \mathcal{F}.\ \exists\ \xi^w = (\sigma_0^w\sigma_1^w\ldots\sigma_d^w)$$
$$\texttt{with}\ \big(\sigma_0^w.p = l_f\ \wedge\ \sigma_d^w.v = v_0\ \wedge$$
$$\bigvee_{r_i\in R}(\sigma_d^w.p \in \mathcal{N}(\sigma_1^{r_i}.p) \vee \ldots \vee \sigma_d^w.p \in \mathcal{N}(\sigma_{ncp}^{r_i}.p))\big). \quad \text{(III.1b)}$$

In Constraint III.1a, we introduce a trajectory loop of length ncp for each recharger robot. In constraint III.1b, we ensure that for each obstacle-free location $l_f \in \mathcal{F}$, there exists a trajectory of length d for a worker robot which starts at l_f and terminates with a zero velocity at a grid location which is in the neighbourhood of the trajectory of one of the rechargers. Note that the requirement of a trajectory of length exactly d is not too restrictive, as any trajectory of length less than d with zero velocity at the end can be extended to a valid trajectory of length d by applying a number of rest primitives.

The trajectory of a robot s (recherger or worker), denoted by $\xi^s = (\sigma_0^s, \ldots, \sigma_{len}^s)$, can be associated with its motion plan $\rho^s = (\gamma_1^s, \ldots, \gamma_{len}^s)$, where $\gamma_i^s \in \Gamma_s$ for all $i \in \{1, \ldots, len\}$, and the following constraints:

$$\bigwedge_{t\in\{0,\ldots,len-1\}} \Big(\sigma_t^s \models pre(\gamma_{t+1}^s)\ \wedge\ \sigma_{t+1}^s = post(\sigma_t^s, \gamma_{t+1}^s)$$
$$\wedge\ intermediate(\sigma_t^s, \gamma_{t+1}^s) \notin \mathcal{O}\Big) \quad \text{(III.2)}$$

Constraint III.2 ensures that for robot s, the state σ_t^s at time t satisfies the pre-condition of the motion primitive γ_{t+1}^s at time $t+1$; and applying the motion primitive γ_{t+1}^s at state σ_t^s takes the robot to state σ_{t+1}^s; and the intermediate blocks through which the robot passes are obstacle-free.

B. Optimizing Rechargers' Trajectories

The above-mentioned Algorithm III.1 considers the length of the trajectories for all the rechargers to be the same and is denoted by ncp. However, there is a possibility that the lengths of these trajectory loops can be further reduced if we allow different rechargers to have trajectories with different lengths. In what follows, we present two algorithms to optimize the trajectory lengths of the rechargers.

Global Optimization. In this approach (Algorithm III.2), we attempt to further reduce the length of the individual trajectory loops by minimizing the cumulative sum of the trajectory lengths of the rechargers. For a given input instance, first we obtain the specific values for the three parameters ncp, d, and $nrech$ from Algorithm III.1. These parameter values are passed to this algorithm (Algorithm III.2). Now, we run a loop to find the minimum value for $total_ncp$ (sum of the lengths of all rechargers' trajectories). The loop is similar to the loop in Algorithm III.1 with a few differences. The loop starts with a feasible value of $total_ncp$ and decreases it by 1 in each iteration until the constraints become unsatisfiable. All the parameters passed from Algorithm III.1 are kept fixed in this algorithm to find the minimum value of $total_ncp$.

Algorithm III.2: Global minimization of the total length of the trajectory loops of the rechargers

Input: $\mathcal{F}$: the set of obstacle-free blocks, $\mathcal{O}$: the set of obstacles
Γ^r: the set of motion primitives for the recharger,
Γ^w: the set of motion primitives for any worker,
$fixed : [ncp, d, nrech]$: the parameter values passed from Algorithm III.1 and kept fixed in this algorithm.

Output: $rtraj_{new}$: the vector containing re-optimized trajectories for the rechargers,
$wtraj_{new}$: the vector containing re-optimized trajectories for the workers.

```
function global_optimization(F, O, Γ^r, Γ^w, fixed)
begin
    total_ncp := nrech × ncp;
    while true do
        C :=
          generate_constraints(F, O, Γ^r, Γ^w, fixed, total_ncp);
        [result, model] := solve_constraints(C);
        if result = SAT then
            model' := model;
            total_ncp := total_ncp − 1;
        else
            [rtraj_new, wtraj_new] :=
              extract_trajectories(model');
            return [rtraj_new, wtraj_new];
        end
    end
end
```

Note that in the representation of the trajectory loops, we still need to keep ncp number of motion primitives for the trajectory loops for each recharger. However, our goal is to generate trajectory loops with as many rest primitives as possible so that the shortest trajectory loops can be generated by removing those rest primitives. To achieve this, we introduce the following constraint that equates $total_ncp$ to the number of only 'active' transitions which do not correspond to rest primitives.

$$total_ncp = \sum_{r\in R} |active_prim_r|,$$
$$active_prim_r = \{\gamma \mid \gamma \text{ is used in } \rho^r = (\gamma_0^r, \ldots, \gamma_{ncp}^r)\ \wedge$$
$$\gamma \text{ is not a } \textit{rest primitive}\}.$$

For a lower value of $total_ncp$, the solver attempts to generate recharger trajectories with more number of *rest primitives*. When the constraints become unsatisfiable, meaning that it is no more possible to reduce $total_ncp$ by introducing *rest primitives* in the trajectories, the algorithm uses the function `extract_trajectories` to extract the recharger trajectories ($rtraj_{new}$) by eliminating the transitions with rest primitives from the last feasible model. Worker trajectories ($wtraj_{new}$) are also derived from the model.

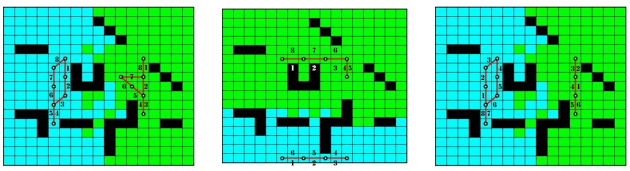
Local Optimization. Though global optimization finds the trajectories of the rechargers by minimizing their total

Algorithm III.3: Local minimization of the trajectory lengths of the rechargers

Input: $\mathcal{F}$: the set of obstacle-free blocks, $\mathcal{O}$: Set of obstacles, Γ^r: the set of motion primitives for the recharger, Γ^w: the set of motion primitives for any worker, $fixed : [ncp, d, nrech, rtraj, wtraj]$: passed from Algorithm III.1.

Output: $rtraj_{new}$, $wtraj_{new}$.

```
function local_optimization(F, O, Γ^r, Γ^w, fixed)
begin
    F_part := partition_workspace(rtraj, wtraj);
    r := 1;
    while r ≤ nrech do
        [ncp^r_new, rtraj^r_new, wtraj^r_new] :=
          generate_trajectories(F^r_part, O, Γ^r, Γ^w, d, nrech=1);
        r := r + 1;
    end
    return [rtraj_new, wtraj_new];
end
```



(a) Sub-optimal trajectories (b) Globally optimized trajectories (c) Locally optimized trajectories

Fig. 2. Deployment of mobile rechargers in an Artificial floor (17×17). Covered locations by different rechargers are indicated by different colors.

(cumulative) length, the algorithm suffers from lack of scalability. In Algorithm III.3, we present a scalable *local optimization* method to improve the trajectory lengths for the rechargers without compromising the optimality of their total cost (as in global optimization) significantly.

Given two fixed parameters and the decision parameter, we first invoke our basic algorithm (Algorith III.1) to find the recharger trajectories (denoted by $rtraj$) and the worker trajectories from any obstacle-free location $l \in \mathcal{F}$ (denoted by $wtraj$). We pass $rtraj$ and $wtraj$ to this algorithm (Algorithm III.3). Now, by using the trajectories in $rtraj$ and $wtraj$, for each mobile recharger $r \in R$ with trajectory $\xi^r = (\sigma^r_0 \ldots \sigma^r_{ncp})$, we find the subset $\mathcal{F}^r_{part}$ (of the entire obstacle-free cells $\mathcal{F}$) which is served by recharger r by using `partition_workspace` function. Mathematically, $\mathcal{F}^r_{part}$ can be captured as:

$$\mathcal{F}^r_{part} = \{l \mid \exists\ \xi^w = (\sigma^w_0 \ldots \sigma^w_d) \in wtraj \text{ such that } \sigma^w_0.p = l \wedge \big(\sigma^w_d.p \in \mathcal{N}(\sigma^r_1.p) \vee \ldots \vee \sigma^w_d.p \in \mathcal{N}(\sigma^r_{ncp}.p)\big)\}$$

For each recharger r, we now find the optimal trajectory $rtraj^r_{new}$ (hence ncp^r_{new}) by invoking the `generate_trajectories` method in Algorithm III.1 with parameter $\mathcal{F}^r_{part}$ instead of $\mathcal{F}$, the fixed d-value and $nrech$=1, and ncp^r_{new} as the decision variable. Finally, it returns the trajectories for the rechargers and workers.

C. Optimization Example

Figure 2(a) shows the trajectories computed by the basic Algorithm III.1 where we consider lengths of recharger trajectories to be equal. In the figure, we show the partition of the workspace into two regions (shown in green and cyan color) corresponding to the coverage areas of the two rechargers.

Figure 2(b) shows the optimal trajectories of the rechargers as computed by the global optimization approach (Algorithm III.2). For the Artificial floor, sub-optimal trajectories (Figure 2(a)) are generated with the maximum trajectory length of each recharger to be 8. It can be seen in Figure 2(a) that for both the rechargers, the trajectory length is 8 (some edges may be traversed more than once). However, when optimized globally, we obtain two different trajectory loops – one with length 8 and the other with length 6.

Figure 2(c) shows the trajectories obtained by applying local optimization (Algorithm III.3). The algorithm starts with the sub-optimally covered regions obtained from the basic algorithm and then finds the optimal trajectory for each region individually. As shown in Figure 2(c), the recharger trajectory length for one of the regions reduces (to 6 from 8) when compared to sub-optimal trajectories (Figure 2(a)).

In terms of total trajectory length, local optimization produces a result that is at par with the one produced by the global optimization. However, in terms of computation time, the local optimization (49min) beats its counterpart (162min) by a good margin.

IV. Evaluation

A. Experimental Setup

We run our algorithms on two different workspaces (Warehouse and Artificial floor, as shown in Figure 1) with dimensions 12×12 and 17×17, and two types of worker robots – Turtlebot [34] and Quadcopter [35]. All the workers are the same type of robots: either Turtlebot (9 primitives) or Quadcopter (57 primitives). The recharger robots are of Turtlebot type with 13 primitives, as mentioned before. We assume that the Quadcopters fly at a specific height, and they require a negligible amount of energy to come down to the ground from their flying plane.

Conventions like Ware-12 and Art-17 are for Warehouse (12×12) and Artificial floor (17×17) respectively. All the experiments have been carried out on a system with i7-6500U CPU @ 2.50 GHz, and 16 GB RAM. We have used SMT solver Z3 [36] in our experiments.

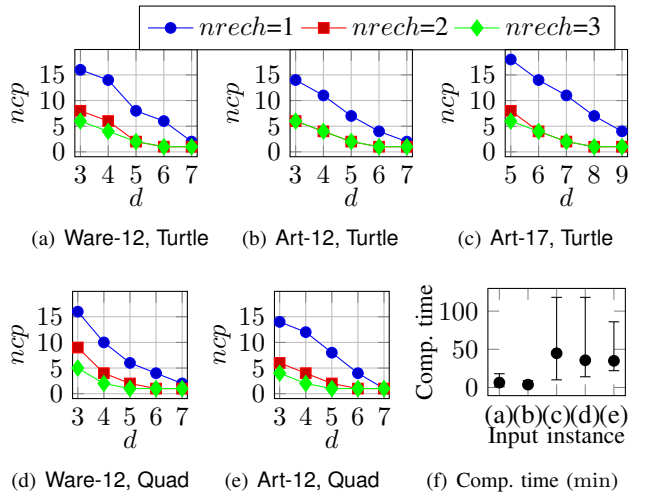
B. Results

We present our experimental results in detail here.

ncp vs d: Figures 3(a) to 3(e) show the trajectory lengths (ncp) of the rechargers against different d-values. Results are obtained by running Algorithm III.1 for five different input instances ($\langle workspace, robot\rangle$ pairs) and up to three rechargers ($nrech$=1,2,3).

Workspace dimension plays a key role in deciding the ncp-value. See Figures 3(b) and 3(c) where the same type of workspaces with different dimensions are considered. As the Artificial floor workspace is inflated to a larger size, ncp-value increases. Here, d-value and $nrech$ are kept unchanged.

Figures 3(b) and 3(e) compare two different robots in the same workspace (Artf-12). A Quadcopter has a richer set of motion primitives compared to a Turtlebot, resulting in a



(a) Ware-12, Turtle (b) Art-12, Turtle (c) Art-17, Turtle

(d) Ware-12, Quad (e) Art-12, Quad (f) Comp. time (min)

Fig. 3. Figures (a) to (e): ncp vs d with fixed parameter $nrech$ (=1, 2, 3). Results are shown for different robots and workspaces of varying dimensions. Figure (f): Computation time (in minutes) for input instances in figures (a) to (e). Ranges of computation time and their average is shown.

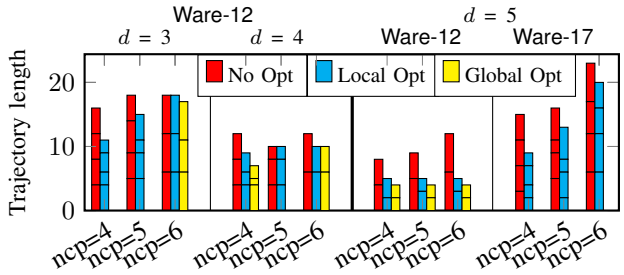


Fig. 4. Trajectory lengths of the rechargers for – *Sub-optimal* (red), *Locally optimized* (cyan) and *Globally optimized* (yellow) solutions. The number of subdivisions in each bar indicates the number of rechargers. Length of a subdivision indicates the trajectory length of some recharger. Results are shown for Turtlebot worker robots. No bar indicates a timeout (3 hours).

smaller ncp-value for a Quadcopter compared to a Turtlebot. Figures 3(a) and 3(d) show similar results.

Table I gives the computation time of Algorithm III.1 running on the input instances in Figures 3(a) to 3(e), while Figure 3(f) shows the minimum, maximum and average computation time for those input instances. With less obstacle density, Artificial floor takes less computation time which is evident for input instances (a) and (b) in Figure 3(f). With richer motion primitives, a Quadcopter requires more computation time in (a) and (d) instances and so on.

Optimization of recharger trajectories: In Figure 4, we compare the results obtained from the three algorithms (Algorithm III.1, Algorithm III.1 + global optimization, Algorithm III.1 + local optimization) for different dimensions of the Warehouse workspace and different values for d. The figure shows that local optimization does not always find the minimum cumulative or individual trajectory lengths, unlike global optimization. However, in many cases, the global optimization algorithm simply faces timeout (3 hours) (Figure 4). In the cases when the global optimization algorithm does not face timeout, the execution time of the local optimization algorithm is $\approx$ 3x $-$ 10x faster compared to the global optimization algorithm. However, for most of the instances, local optimization provides a result close to the globally optimal solution.

TABLE I

COMPUTATION TIME (IN MINUTES) OF ALGORITHM III.1 FOR DIFFERENT INPUT INSTANCES SHOWN IN FIGURE 3.

Turtlebot											Quadcopter					
	Ware-12			Art-12				Art-17				Ware-12			Art-12	
	nrech			*nrech*				*nrech*				*nrech*			*nrech*	
d	1	2	3	1	2	3	d	1	2	3	d	1	2	3	1	2	3
3	18	12	18	8	3	9	5	118	35	47	3	71	118	53	31	23	32
4	15	7	6	4	2	6	6	51	10	65	4	18	15	25	36	24	32
5	3	1	3	2	2	4	7	45	14	46	5	19	35	14	37	30	32
6	3	1	2	2	2	3	8	17	16	41	6	28	19	18	28	23	28
7	2	2	3	2	2	4	9	29	52	86	7	40	33	27	22	86	52

TABLE II

COMPARISON BETWEEN STATIC AND MOBILE RECHARGERS IN TERMS OF OBTAINED d-VALUES. MAXIMUM WAITING TIME HAS BEEN SHOWN FOR MOBILE RECHARGERS. RESULTS SHOWN FOR $nrech$ =1,2 AND 3.

	Turtlebot									Quadcopter					
	Ware-12			Art-12			Art-17			Ware-12			Art-12		
	S	M		S	M		S	M		S	M		S	M	
$nrech$	d_s	d_m	w_m	d_s	d_m	w_m	d_s	d_m	w_m	d_s	d_m	w_m	d_s	d_m	w_m
1	8	6	6	8	5	6	11	8	8	8	5	5	7	6	6
2	6	4	5	6	4	4	8	6	6	6	4	4	6	4	4
3	6	4	4	6	4	4	8	5	5	5	3	4	5	3	3

Comparison with the Static Charging Stations: We compare our proposed mobile recharger based power management framework (Algorithm III.1) with a framework that employs only static charging stations installed at strategic locations (see Table II). In Table II, d_s denotes, in the *static* charging station scenario, the threshold for the worker, which is a measure of both the maximum energy consumption and maximum wait time required to complete a recharge operation. On the other hand, these two performance measures for the *mobile* rechargers are denoted by d_m and w_m, respectively. For a given workspace and the number of charging stations ($nrech$), the value of d_s is computed by the algorithm presented in [13]. To compute the value of d_m and w_m, we identify the $\langle d, ncp \rangle$ pair for which the wait time, given by $w = \max(d, ncp - 1)$, is minimized and the corresponding d and w are denoted by d_m and w_m. As evident in Table II, using our mobile recharging framework, we can improve both the performance criteria significantly.

V. CONCLUSION

We have presented the algorithms for determining the trajectories of the mobile rechargers to fulfill the energy requirements of mobile robots in a static environment. Our experimental results show that our SMT-based algorithm can synthesize trajectories for the mobile rechargers serving worker robots with different dynamics and acting in different kinds of workspaces. Our plan synthesis framework is not only useful in scenarios where static charging stations are challenging to install but also for the applications where static charging stations can be deployed. In the latter case, our mobile recharger based solution outperforms the static charging station based solution both in terms of the required time and energy to recharge batteries of the worker robots. In the future, we would address the problem for heterogeneous worker robots and workspaces having dynamic obstacles.

REFERENCES

[1] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *IJRR*, vol. 32, no. 8, pp. 889–911, 2013.

[2] D. Halperin, J.-C. Latombe, and R. H. Wilson, "A general framework for assembly planning: The motion space approach," in *Annual Symposium on Computational Geometry*, 1998, pp. 9–18.

[3] S. Rodríguez and N. M. Amato, "Behavior-based evacuation planning," in *ICRA*, 2010, pp. 350–355.

[4] J. Jennings, G. Whelan, and W. Evans, "Cooperative search and rescue with a team of mobile robots," in *ICRA*, 1997, pp. 193–200.

[5] D. Rus, B. Donald, and J. Jennings, "Moving furniture with teams of autonomous robots," in *IROS*, 1995, pp. 235–242.

[6] A. Drenner and N. Papanikolopoulos, "Docking station relocation for maximizing longevity of distributed robotic teams," in *ICRA*, 2006, pp. 2436–2441.

[7] J. Wawerla and R. T. Vaughan, "Near-optimal mobile robot recharging with the rate-maximizing forager," in *Advances in Artificial Life*, 2007, pp. 776–785.

[8] B. Kannan, V. Marmol, J. Bourne, and M. B. Dias, "The autonomous recharging problem: Formulation and a market-based solution," in *ICRA*, 2013, pp. 3503–3510.

[9] G. P. Strimel and M. M. Veloso, "Coverage planning with finite resources," in *IROS*, 2014, pp. 2950–2956.

[10] J. Yu, J. Aslam, S. Karaman, and D. Rus, "Anytime planning of optimal schedules for a mobile sensing robot," in *IROS*, 2015, pp. 5279–5286.

[11] S. Mishra, S. Rodriguez, M. Morales, and N. M. Amato, "Battery-constrained coverage," in *CASE*, 2016, pp. 695–700.

[12] I. Shnaps and E. Rimon, "Online coverage of planar environments by a battery powered autonomous mobile robot," *T-ASE*, vol. 13, no. 2, pp. 425–436, 2016.

[13] T. Kundu and I. Saha, "Charging station placement for indoor robotic applications," in *ICRA*, 2018, pp. 3029–3036.

[14] ——, "Energy-aware temporal logic motion planning for mobile robots," in *ICRA*, 2019, pp. 8599–8605.

[15] P. H. Borgstrom, A. Singh, B. L. Jordan, G. S. Sukhatme, M. A. Batalin, and W. J. Kaiser, "Energy based path planning for a novel cabled robotic system," in *IROS*, 2008, pp. 1745–1751.

[16] S. Kim, S. Bhattacharya, and V. Kumar, "Path planning for a tethered mobile robot," in *ICRA*, 2014, pp. 1132–1139.

[17] S. McCammon and G. A. Hollinger, "Planning and executing optimal non-entangling paths for tethered underwater vehicles," in *ICRA*, 2017, pp. 3040–3046.

[18] I. Kelly, O. Holland, and C. Melhuish, "Slugbot: A robotic predator in the natural world," in *In Proc. 5th International Symposium on Artificial Life and Robotics*, 2000, pp. 470–475.

[19] D. Wettergreen, P. Tompkins, C. Urmson, M. Wagner, and W. Whittaker, "Sun-synchronous robotic exploration: Technical description and field experimentation," *IJRR*, vol. 24, no. 1, pp. 3–30, 2005.

[20] A. J. Hawkins, "Volkswagen's mobile charging station will help solve a key problem with EVs." [Online]. Available: https://www.theverge.com/2019/1/2/18165265/volkswagen-ev-mobile-charging-station-battery

[21] SparkCharge, "A portable and ultrafast charging unit for electric vehicles." [Online]. Available: https://sparkcharge.io

[22] Y. Litus, R. T. Vaughan, and P. Zebrowski, "The frugal feeding problem: Energy-efficient, multi-robot, multi-place rendezvous," in *ICRA*, 2007, pp. 27–32.

[23] N. Kamra and N. Ayanian, "A mixed integer programming model for timed deliveries in multirobot systems," in *CASE*, 2015, pp. 612–617.

[24] N. Mathew, S. L. Smith, and S. L. Waslander, "Multirobot rendezvous planning for recharging in persistent tasks," *T-RO*, vol. 31, no. 1, pp. 128–142, 2015.

[25] T. Gao and S. Bhattacharya, "Multirobot charging strategies: A game-theoretic approach," *RA-L*, vol. 4, no. 3, pp. 2823–2830, 2019.

[26] W. N. N. Hung, X. Song, J. Tan, X. Li, J. Zhang, R. Wang, and P. Gao, "Motion planning with Satisfiability Modulo Theroes," in *ICRA*, 2014, pp. 113–118.

[27] S. Nedunuri, S. Prabhu, M. Moll, S. Chaudhuri, and L. E. Kavraki, "SMT-based synthesis of integrated task and motion plans from plan outlines," in *ICRA*, 2014, pp. 655–662.

[28] Y. Wang, N. T. Dantam, S. Chaudhuri, and L. E. Kavraki, "Task and motion policy synthesis as liveness games," in *ICAPS*, 2016, pp. 536–540.

[29] I. Saha, R. Ramaithitima, V. Kumar, G. J. Pappas, and S. A. Seshia, "Automated composition of motion primitives for multi-robot systems from safe LTL specifications," in *IROS*, 2014, pp. 1525–1532.

[30] ——, "Implan: Scalable incremental motion planning for multi-robot systems," in *ICCPS*, 2016, pp. 43:1–43:10.

[31] A. Desai, I. Saha, J. Yang, S. Qadeer, and S. A. Seshia, "DRONA: a framework for safe distributed mobile robotics," in *ICCPS*, 2017, pp. 239–248.

[32] I. Gavran, R. Majumdar, and I. Saha, "ANTLAB: a multi-robot task server," in *EMSOFT*, 2017.

[33] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[34] "TurtleBot," http://www.turtlebot.com.

[35] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *ICRA*, 2011, pp. 2520–2525.

[36] L. M. de Moura and N. Bjørner, "Z3: An efficient SMT solver," in *TACAS*, 2008, pp. 337–340.