Artificial Life and Robotics (2021) 26:52–60
https://doi.org/10.1007/s10015-020-00612-8

ORIGINAL ARTICLE

Multi-robot path planning for smart access of distributed charging points in map

Abhijeet Ravankar[1] · Ankit A. Ravankar[2] · Michiko Watanabe[1] · Yohei Hoshino[1] · Arpit Rawankar[3]

Received: 20 February 2020 / Accepted: 15 June 2020 / Published online: 9 July 2020

Abstract
Autonomous mobile robots are being used to automate many tasks such as cleaning, delivering items, and surveillance. Such tasks often require uninterrupted and continuous service. However, robots have limited battery power and must be recharged frequently. Since manually charging each robot is not always feasible, automatic charging (docking) stations have been developed to automate the process of charging. In a multi-robot system, the tasks are generally distributed between the robots, and different robots have different amounts of remaining battery power. Since the charging stations are expensive, a limited number of charging points are generally available. Hence, an intelligent planner to manage a limited number of charging points for a large number of robots is essential. In this work, we propose a multi-robot path planner for intelligently accessing a limited number of charging points distributed on the map. Unlike traditional path planners, which mainly consider the shortest path criterion to generate paths, the proposed planner also considers the remaining battery power of the robots, task priority, and robot's location in the map. It allocates the most appropriate charging station to the robots, which require recharging. Simulation results show that the proposed planner can reduce trajectory re-planning, and plan efficient paths to the available charging points.

Keywords Autonomous mobile robots · Service robots · Path planning · Robot charging

1 Introduction

Service robots in the real world need a continuous operation to perform several tasks. Such tasks include cleaning at hospitals and public places, patrolling and surveillance of infrastructures, and moving items in a warehouse. These tasks often require uninterrupted and continuous service. Service robots generally operate in a multi-robot environment and navigate using various path planning algorithms. The robots need to recharge frequently for continuous operation. Manually charging each robot is cumbersome and inefficient; hence, automatic charging stations have been developed to automate charging. Autonomous robots can localize themselves in the map, and navigate to the charging dock to recharge when the remaining battery power falls below a certain threshold. However, charging stations are expensive, and a limited number of charging points might be available. There may be cases when all of the charging stations are occupied. Hence, an intelligent planner to manage a limited number of charging points with a large number of robots is essential. In this work, we propose a multi-robot path planner to manage a limited number of charging points. The proposed planner considers the remaining battery power of the robots, task priority, robot's location in the map, and deadlock avoidance. On the other hand, traditional path planners mainly consider the shortest distance between the start and goal location for path planning, while completely ignoring the remaining battery power of the robots. This is a limitation as

This work was presented in part at the 25th International Symposium on Artificial Life and Robotics (Beppu, Oita, January 22–24, 2020).

✉ Abhijeet Ravankar
aravankar@mail.kitami-it.ac.jp

✉ Ankit A. Ravankar
ankit@eng.hokudai.ac.jp

[1] School of Regional Innovation and Social Design Engineering, Kitami Institute of Technology, Hokkaido, Japan

[2] Division of Human Mechanical System and Design, Hokkaido University, Sapporo, Japan

[3] Department of Electronics and Telecommunication, Vidyalankar Institute of Technology, Mumbai, India

inefficient planning could lead to wastage of battery power. This also leads to interruption of service. The proposed method allocates the most appropriate charging station to the robots, which requires recharging depending on the robot's position in the map, remaining battery power, and task priority. Collision avoidance is essential in multi-robot systems. Collision avoidance in complex scenarios often consumes a lot of battery power and time as the robots need to plan collision-free trajectories. Multi-robot collision avoidance is in-built in the proposed method. In addition, the proposed planner intelligently selects the next task, which the robot can finish with the remaining battery power. This is done by generating a profile of each task and prioritizing the tasks. We present the results of the proposed algorithm in various scenarios and discuss the merits compared to traditional path planning algorithms.

The work presented in this paper focuses on the software-based control of a limited number of distributed charging points in a multi-robot scenario. Previous works have mainly focussed on the hardware design of charging point. This includes, infra-red sensor based [20], multi-sensor-based [5], vision-based [17], WiFi-based [18], wireless charging-based [4], k-mediods algorithm-based [22], and RFID-based [3] design of charging stations. Other approaches include battery exchange [21], and communication-based control [8].

2 Robot configuration and priority score estimation

The proposed planner considers the remaining battery power, and task priority of the robots in path planning. To achieve this, parameters like the power required to complete each task, and task priority needs to be configured which is explained below.

2.1 Localization and communication setup

It is assumed that all the robots have a map of the environment. It is also assumed that the robots can localize themselves in the map. Mapping and localization can be achieved using any of the SLAM (simultaneous localization and mapping) [9, 15, 16] modules available in the literature. A navigation module is also assumed to be available, which could comprise of any of the traditional algorithms [12]. More information about path planning and path smoothing algorithms can be found in [10, 11, 14]. It is also important to configure the network module so that all the robots can communicate with each other. In addition, there is a command executor module which generates commands to the robot to follow a particular path or leave a charging point.

2.2 Priority estimation

First, we estimate the battery power required to complete each task. This information is maintained in the robot's database. Each robot tracks its battery power status before the start of ith task (P^i_s), and at end of that task (P^i_e). The total power (P^i_t) required to complete the ith task is given as,

$$P^i_t = P^i_s - P^i_e. \tag{1}$$

The robot's database is shown in Table 1. The power required to complete a particular task may vary for different (j) runs of the same task. If the ith task is repeated for total m times, the effective power (P^i_E) required to complete the ith task is calculated as,

$$\begin{aligned} P^i_E &= \frac{\langle P^i_s - P^i_e\rangle^{(j=1)} + \langle P^i_s - P^i_e\rangle^{(j=2)} + \cdots + \langle P^i_s - P^i_e\rangle^{(j=m)}}{m} \\ &= \frac{1}{m}\sum_{j=1}^{m}\langle P^i_s - P^i_e\rangle^{(j)}. \end{aligned} \tag{2}$$

Different tasks might be assigned to a robot. The same task might also be assigned multiple times. In all cases, the total effective power required to complete all the (N) tasks is given as,

$$P_{\text{total}} = \sum_{i=1}^{N} P^i_E. \tag{3}$$

Different tasks are assigned different task-priority scores (T_{tp}). Critical tasks such as surveillance and escorting people are assigned higher task-priority scores. On the other hand, tasks like cleaning are assigned a lower task-priority score. This is shown in Table 1. An overall priority score is calculated using the task-priority score and remaining battery power. The task-priority score represents the execution priority of that task. The overall priority score determines which robot needs to be given priority to access a charging station in case of multiple requests. A robot which has a high task-priority score, and lower remaining battery power (P_{rbp}) is prioritized, and the overall priority score is calculated as,

Table 1 Mapping tasks, task priority, and power consumed

Task	Task-ID (T_i)	Priority (T_p)	Power (P_i)
Patrolling	T_1	10	P_1
Item delivery	T_2	8	P_2
Cleaning	T_3	3	P_3
Escort	T_4	9	P_4
⋮	⋮	⋮	⋮
Others	T_N	5	P_N

$$\text{Priority score} \propto \left(\frac{1}{\text{Remaining battery power}}\right) \times \text{Task priority} = \left(\alpha_p \cdot \frac{1}{P_{\text{rbp}}}\right) \times \alpha_t \cdot T_{\text{tp}} \quad (4)$$

where α_p and α_t are the weighing coefficients for the power, and the task priority, respectively. The coefficients are used to tune the influence of the remaining battery power, and the task-priority on the overall priority score. A higher α_p value could be set for robots with small batteries. On the other hand, a higher α_t value could be set for scenarios where tasks of higher execution priorities must be given preference.

In real-world scenarios, the robot's battery may be about to discharge completely. This is a critical case because if the robot's battery is completely discharged while the robot is in operation, the robot will stop in the middle of the passage blocking it. In such scenarios, the robot must be moved manually from the area. Such critical situations are handled by assigning the robot with the highest priority score. If P_{TH} is the critical battery power threshold, the priority score is calculated as,

$$\text{Score, } S = \begin{cases} \left(\alpha_p \cdot \frac{1}{P_{\text{rbp}}}\right) \times \alpha_t \cdot T_{\text{tp}}, & \text{if } P_{\text{rbp}} > P_{\text{TH}} \\ \infty, & \text{otherwise} \end{cases} \quad (5)$$

2.3 Priority queue-based access to charging station

Each robot is programmed to broadcast a request to access a charging station. The request comprises of a key-value pair of the robot-id (R_{id}), and a corresponding parameter (Ψ_i). The parameter Ψ_i comprises of the task-priority score (T_{tp}), remaining battery power (P_{rbp}), and current x and y coordinates ($X_{\text{rid}}, Y_{\text{rid}}$) of the robot in the map. Hence, a request message is a dictionary key-value pair of $\{R_i : \Psi_i\}$, where,

$$\Psi_i = \{T_{\text{tp}}, P_{\text{rbp}}, X_{\text{rid}}, Y_{\text{rid}}\}. \quad (6)$$

All the broadcasted messages are maintained in a priority queue. A priority queue is an extension of a normal queue in which every item has a priority associated with it, and the item with the highest priority is dequeued before items of lower priority. If multiple items have the same priority, they are dequeued by their order. The status of the charging station is also maintained in a database that includes information such as the empty and occupied charging points, the current battery power level of the docked robots, and the next task priorities of the docked robots. The database is updated every time there is a change. The database is shown in Table 2.

As shown in Table 2, the first four columns of the station database shows the charging station's state i.e. charging station ID (D_i), coordinates of D_i (X_{di}, Y_{di}), and a flag to represent if D_i is occupied (Y) or not (N). The next five columns show the charging robot's specifications. The fifth column contains the charging robot's ID (R_{id}) or ID of the robot, which has been granted access to D_i. The last four columns comprise of the value of the key Ψ_i given by Eq. 6.

If a robot requests access to the charging station, an empty charging dock D_i is searched by checking the occupancy flag in the fourth column of the station database. If an empty charging point exists, the robot is granted a token to access that dock. A shortest path is calculated from the robot's current location $X_{\text{rid}}, Y_{\text{rid}}$ to the empty dock location X_{di}, Y_{di} using any of the path planning algorithms like A-star [1], or D-star algorithm [19]. If multiple empty charging points are found, the robot is assigned the nearest charging point. In both the cases, the station database is updated with the R_{id}, Ψ_i, and occupancy flag is set to 'Y'.

However, the charging station could be fully occupied. If the charging station is full, requesting robots must wait until a charging point is available. In the case of multiple requests, access is granted depending on the priority score. In case of 'critical' conditions in which the requesting robot's battery power is about to drain completely, an empty dock is created by forcefully removing a robot which is currently charging in the station. First, a charging robot with minimum task priority T_{tp} and maximum remaining battery power P_{rbp} is removed from the dock. If such criteria are not satisfied, then a robot with medium task priority and adequate battery power is

Table 2 Charging station database

Charging station specification				Docked robot specification				
Station id.	X & Y Coord. (station)		Free	Robot id	Task priority	Power	X & Y Coord. (robot)	
D_1	X_{d1}	Y_{d1}	N	Rid$_1$	T_{p1}	P_{l1}	X_{r1}	Y_{r1}
D_2	X_{d2}	Y_{d2}	N	Rid$_2$	T_{p2}	P_{l2}	X_{r2}	Y_{r2}
D_3	X_{d3}	Y_{d3}	Y	–	–	–	–	–
D_4	X_{d4}	Y_{d4}	N	Rid$_4$	T_{p4}	P_{l4}	X_{r4}	Y_{r4}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
D_n	X_n	Y_n	N	Rid$_n$	T_{pn}	P_{ln}	X_{rn}	Y_{rn}

removed. In other words, a robot that has sufficient battery power to finish the next task is removed from the charging station. The robot is instructed to relinquish the dock access token and continue with the next job. If it is not possible to remove a robot from the charging point, the requesting robots are denied access. They must wait until a charging point is available.

If a robot is forcefully exited, a check is performed to ensure that the paths of the requesting robot and the dock relinquishing robot do not overlap to avoid a collision. Collision avoidance is particularly important if the common path of the two robots is too narrow. Let $H_{a\rightarrow di}$ represent the path from the requesting robot's location (say a) to the allotted dock (di), and $H_{di\rightarrow b}$ represent the path of the dock relinquishing robot from di to its goal location (say b). Collision avoidance is accomplished by calculating and assigning the shortest path to the requesting robot as it has low battery power. The path of the relinquishing robot ($H_{di\rightarrow b}$) is generated in such a way that it does not overlap with $H_{a\rightarrow di}$. In doing so, it is permitted that the path of the relinquishing robot is relatively long as it is sufficiently charged.

$$\begin{aligned} H_{a\rightarrow di} &= \min_{\forall i}\{H_{i_{a\rightarrow di}}\} \\ H_{di\rightarrow b} &= \min_{\forall j}\{H_{j_{di\rightarrow b}} \cap H_{a\rightarrow di}\} \end{aligned} \tag{7}$$

To generate collision-free trajectory, first a virtual obstacle [13] is placed on the shortest path ($H_{a\rightarrow di}$) generated by the requesting robot. A virtual obstacle does not physically exist in the environment but is numerically represented on the map. Details of the virtual obstacle mechanism can be found in [6, 13]. The dock relinquishing robot then plans a path considering this virtual obstacle. This automatically avoids the shortest path and generates an alternate path ($H_{di\rightarrow b}$).

3 Experiments and results

Four different experiments were performed to test the algorithm. Experiments 1 and 2 were performed to test the algorithm when the charging points are located in the same location. Experiments 3 and 4 were performed to test the algorithm when the charging points are distributed. In all cases, A* algorithm [1] was used for path planning. However, any other path planning algorithm like D-star algorithm [19] or Probabilistic Roadmap planner [2] can also be used. The cost of movement in the perpendicular direction and diagonal direction were set to 1 unit, and $\sqrt{2}$ units, respectively. The experiments were performed in a multi-robot scenario with 12 robots in which the robots could communicate with each other. In real-world scenarios, such configuration can easily be set up using ROS [7] (robot operating system) which provides features for inter-node communication between robots.

3.1 Experiment 1

Experiments 1 and 2 were performed in a simulated 100×100 grid environment shown in Fig. 1. There are 6 charging points marked as 1, 2, …, 6. The first experiment was performed to test a scenario in which a robot has to recharge, and an empty charging point is available. As shown in Fig. 1a, the initial position of the robot (shown in green color) was set to $x = 90$, and $y = 80$. Charging point number 5 was unoccupied. The robot planned a path form its start position to the empty slot shown by the green path in Fig. 1a. Compared to the traditional approach, the proposed algorithm has advantages. The information about an empty charging point is immediately available to the robot. Moreover, the exact location of the empty charging point is also available to the robot. Thus, the robot can easily plan a path from its current location to the empty charging point. In the

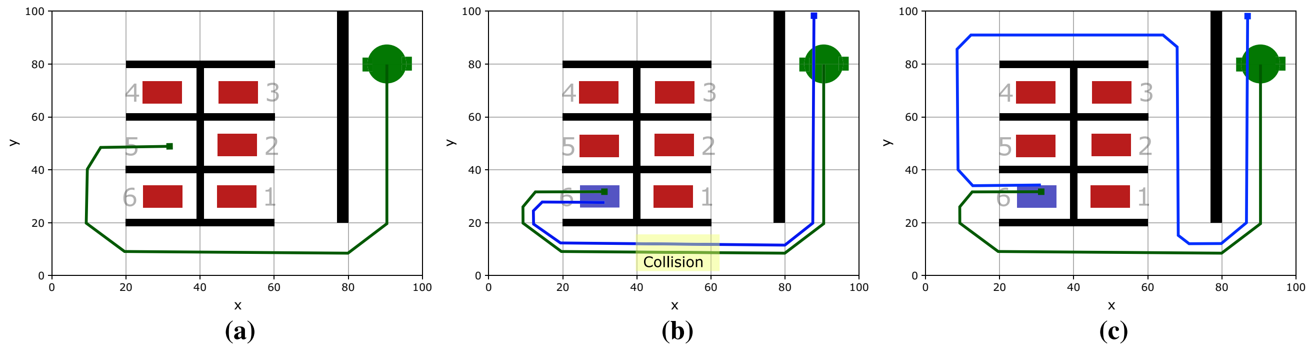

Fig. 1 Experiment results. **a** Exp-1: the empty charging point 5 is immediately accessible to the robot. **b** Exp-2: traditional planning with shortest path criterion generates colliding paths. **c** Exp-2: proposed method generates non-colliding paths. The requesting robot has priority over the shortest path. The dock-relinquishing robot can take a longer path as it is well charged

traditional approach, the robot would have to check each slot if it is empty. This checking could be done using visual sensors or Lidar. Successively checking each charging point is time-consuming and inefficient for robots with low remaining battery power. In the simulation, the time to check if a slot is empty or occupied was set to 15 s. Moreover, checking was programmed to be done sequentially from the nearest charging point 1 to the last until an empty slot is found. The total time (path planning and navigation) consumed in accessing the empty charging point in the traditional and the proposed algorithm is given in Fig. 2a.

3.2 Experiment 2

The second experiment was performed to test a scenario in which all the charging points are occupied, and a robot has to recharge. The experiment setup was the same as that of experiment 1, and the robot's initial position was set to $x = 90$, and $y = 80$. Since all the charging points are occupied, the station database is referred, and information about the current battery power levels of the charging robots is obtained which is shown in Fig. 2b. It can be seen that the robot occupying the charging point 6 is almost fully charged (92%), while robots charging at other points are still not recharged enough. The priority score of the requesting robot was set to be higher than the robot charging at point 6. Hence, the robot at charging point 6 shown in blue color in Fig. 1b was requested to relinquish the charging point.

Shortest path criterion is considered in many traditional path planning algorithms. Traditional algorithms generate the shortest path for both the requesting and the dock relinquishing robots, which overlap, as shown in Fig. 1b.

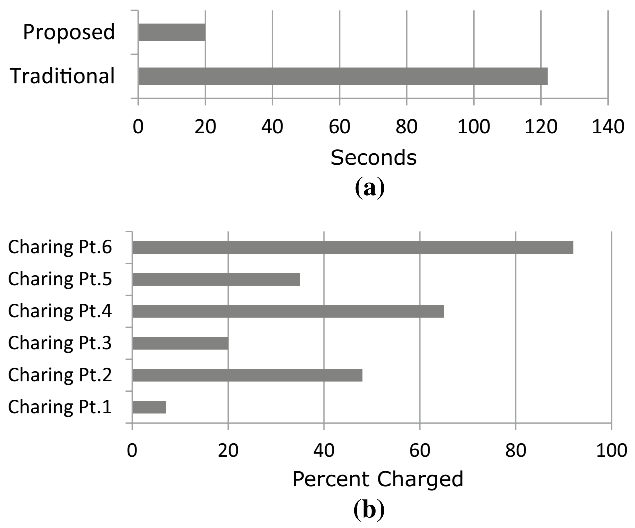

Fig. 2 **a** Exp-1: time to access empty charging point in traditional vs proposed method. **b** Exp-2: robot's battery power levels at various points

In Fig. 1b, the path of the requesting robot and the dock-relinquishing robot is shown in green, and blue color, respectively. It can be seen that robot collision is inevitable in this case. This is not favorable to the requesting robot as its remaining battery power is low, and considerable battery power is consumed in collision avoidance and computing a collision-free trajectory. However, the proposed method considers the power levels of the robots in path planning. Therefore, using Eq. 7, the robot relinquishing the charging point planned an alternate path that does not collide with the requesting robot's path. The requesting robot has access to the shortest path. The result is shown in Fig. 1c. It can be seen that the path planned by the relinquishing robot (shown in blue color in Fig. 1c) is longer than the path planned using the traditional approach shown in Fig. 1b. However, since the dock relinquishing robot is sufficiently charged it is not a significant limitation.

3.3 Experiment 3

The third experiment was performed to test the scenario of distributed charging stations in a large and complex map. Figure 3 shows a 500×500 map in which the obstacles are shown in black color, and the white color represents free navigational space. The obstacles were expanded so that the planned paths do not come too close to the obstacles. This expansion is shown in green color. There are 12 charging stations in Fig. 3 which have been marked as: $\{A1, A2, A3\}$, $\{B1, B2, B3\}$, $\{C1, C2, C3\}$, and $\{D1, D2, D3\}$. The charging stations are distributed in the map, and their locations are summarized in Table 3.

In the simulation, 11 of the 12 charging stations were set to be occupied by robots leaving only a single non-occupied charging station. In this case, the traditional approach would require either a random search or a sequential search

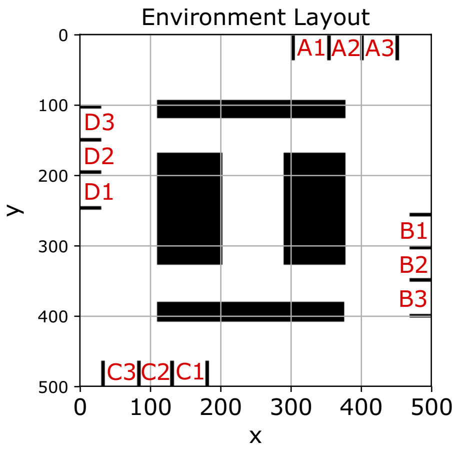

Fig. 3 Map with distributed charging points $A1, A2, \ldots, D3$ for experiment 3 and experiment 4

Table 3 Location of distributed stations in map

Charging station	Position-X	Position-Y
'A1'	20	327
'A2'	20	376
'A3'	20	425
'B1'	280	485
'B2'	327	485
'B3'	375	485
'C1'	482	156
'C2'	482	107
'C3'	482	57
'D1'	223	15
'D2'	173	15
'D3'	127	15

to find the empty charging station. The results of such random and sequential search are shown in Fig. 4. Figure 4 shows the paths of the robot from the starting position of $x = 250, y = 250$ which is the center of the map. The start location is marked as a blue circle, the random goal location is marked as a blue square, and the path is shown in red color. Figure 4 shows the worst-case scenario of random search in which the empty station is found at last. Results of random search approach are shown in Fig. 4a–e. Result of sequential search of the empty station is shown in Fig. 4f.

Table 4 summarizes the total distance navigated by the robot to find the empty station. It also shows the sequence in which the goal is reached using a random search and sequential search. It is evident that although sequential search outperforms random search, the robot still has to navigate a large distance before it finds an empty charging station.

The results of path planning with the proposed method are discussed. In the proposed method, robots have access to the charging station database, and can immediately acquire information about the unoccupied charging points. This experiment used the same setup as the previous experiment. All the stations except one were set to the occupied state by setting the free flag to '*N*'. The results of path planning with the proposed method are shown in Fig. 5. Figure 5a–l shows the shortest paths planned from the start location $x = 250, y = 250$ to the random empty stations. Figure 6 summarizes the total distance navigated by the robot to the different empty stations. It also shows the total execution time of path planning in the proposed method. It can be seen

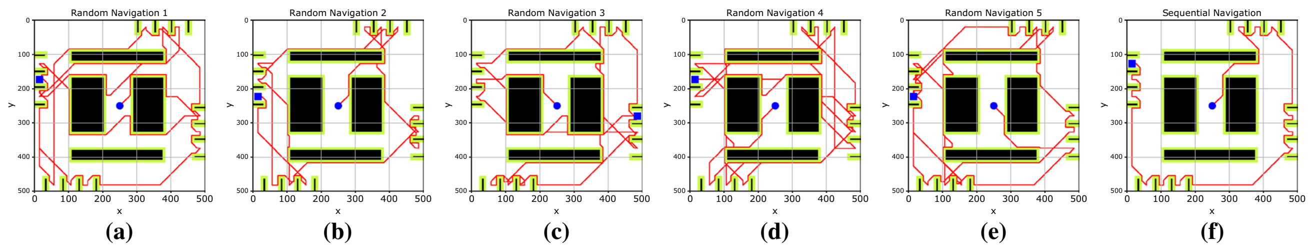

Fig. 4 Path planning to empty station from x=250,y=250. **a**–**e** Random search to empty station. **f** Sequential search

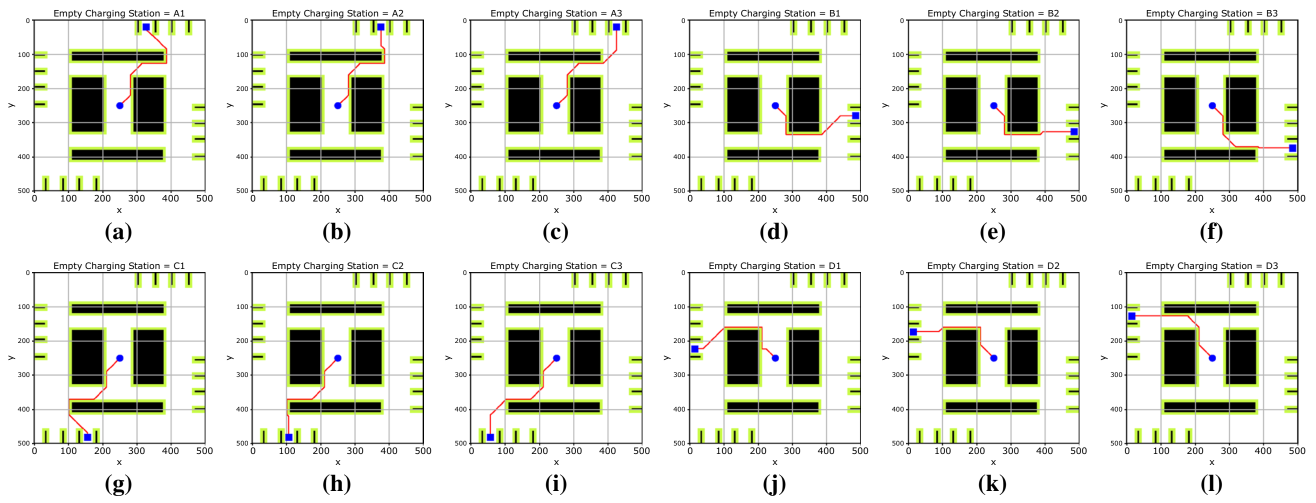

Fig. 5 Path planning to empty station with proposed method. **a**–**l** Empty station directly accessed through shortest path

Table 4 Path planning to empty charging station using random search and sequential search

Algorithm	Empty doc	Path. worst case scenario with empty doc at last (S: Start [$x = 250, y = 250$])	Distance
Random-1	$D2$	$S \rightarrow A3 \rightarrow A1 \rightarrow C3 \rightarrow D3 \rightarrow B1 \rightarrow D1 \rightarrow A2 \rightarrow B2 \rightarrow B3 \rightarrow C1 \rightarrow C2 \rightarrow D2$	5004.25
Random-2	$D1$	$S \rightarrow A2 \rightarrow D2 \rightarrow C2 \rightarrow B1 \rightarrow B3 \rightarrow B2 \rightarrow A1 \rightarrow A3 \rightarrow D3 \rightarrow C3 \rightarrow C1 \rightarrow D1$	3840.39
Random-3	$B1$	$S \rightarrow D2 \rightarrow D1 \rightarrow B2 \rightarrow C3 \rightarrow C1 \rightarrow C2 \rightarrow A2 \rightarrow A3 \rightarrow B3 \rightarrow A1 \rightarrow D3 \rightarrow B1$	4656.09
Random-4	$D2$	$S \rightarrow C3 \rightarrow A3 \rightarrow C2 \rightarrow B1 \rightarrow D1 \rightarrow A2 \rightarrow A1 \rightarrow B2 \rightarrow D3 \rightarrow C1 \rightarrow B3 \rightarrow D2$	6221.19
Random-5	$D1$	$S \rightarrow B3 \rightarrow C3 \rightarrow D3 \rightarrow C2 \rightarrow B2 \rightarrow B1 \rightarrow A2 \rightarrow C1 \rightarrow D2 \rightarrow A1 \rightarrow A3 \rightarrow D1$	5018.92
Sequential	$D3$	$S \rightarrow A1 \rightarrow A2 \rightarrow A3 \rightarrow B1 \rightarrow B2 \rightarrow B3 \rightarrow C1 \rightarrow C2 \rightarrow C3 \rightarrow D1 \rightarrow D2 \rightarrow D3$	2131.0

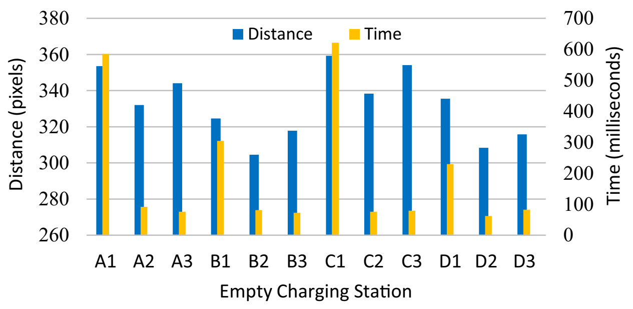

Fig. 6 Distance to empty stations from $x = 250, y = 250$. Execution time for path planning to stations is also shown

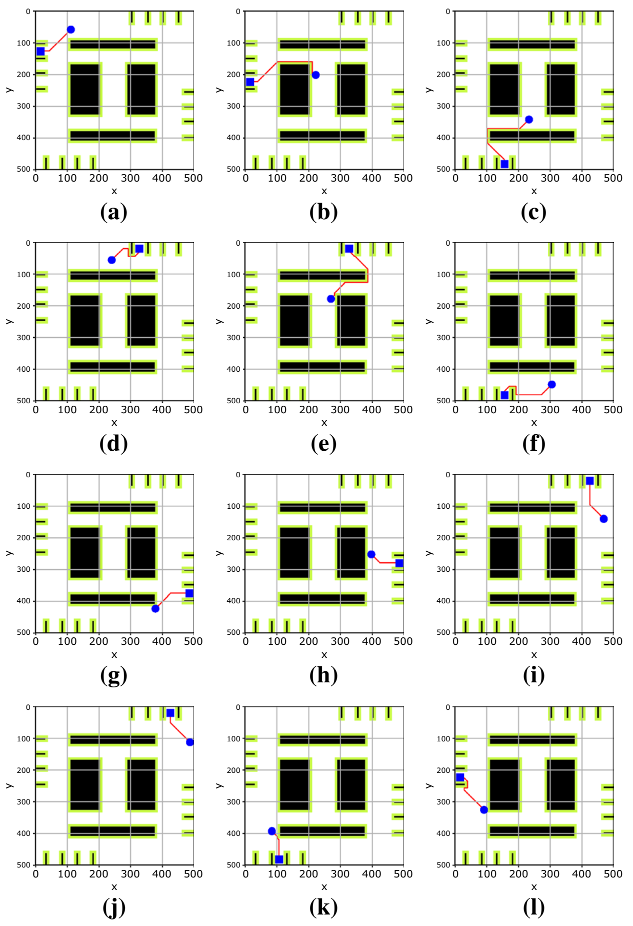

Fig. 7 Direct access to the nearest empty charging station from different random starting points through shortest path

that the paths navigated by the robot are the shortest path from the start to the empty charging point. The proposed method can immediately find any empty charging point and plans the shortest path.

The simulation software was developed on Ubuntu Linux OS with Intel Core-i7 2.80 GHz processor and 16 GB RAM. Programming was done using Python 3.6, NumPy, and Matplotlib library. The average path planning time for random search of empty charging station shown in Fig. 4a–e was 2.73 s. The average time of sequential search of the empty charging station shown in Fig. 4f was 1.28 s. In a random and sequential search, path planning had to be done multiple times until an empty charging station was found. However, the average path planning time to an empty charging station using the proposed method was 196.73 ms. Robot's navigation time depends on the speed of the robot and is not included in all cases.

3.4 Experiment 4

The fourth experiment was performed to test if the robot could access the nearest empty charging point when multiple empty charging points are available. In the experiment, multiple charging points were set to empty, and the starting location of the robot was set randomly. The robot accessed the charging station database to get a list of all the free charging points. Then, distances between the robot's current location and all the free charging points were calculated. Finally, the robot selected the charging point with the minimum distance. Figure 7 shows the results of the nearest charging points searched by the robot, and the path planned to it from a random starting location. Table 5 shows the starting coordinates of the robot generated randomly, the nearest empty charging point searched, and the distance from the starting position to the empty charging point. In all the cases, it was confirmed that the robot accessed the nearest available empty charging point.

Table 5 Path planning to nearest empty station

Figure	Start-X	Start-Y	Nearest dock	Distance
Fig. 7a	111	58	D3	124.581
Fig. 7b	222	201	D1	277.167
Fig. 7c	233	342	C1	277.794
Fig. 7d	240	56	A1	142.539
Fig. 7e	270	178	A1	273.25
Fig. 7f	305	448	C1	207.711
Fig. 7g	378	423	B3	126.882
Fig. 7h	397	252	B1	99.598
Fig. 7i	469	140	A3	138.225
Fig. 7j	487	113	A3	118.681
Fig. 7k	84	393	C2	98.527
Fig. 7l	91	326	D1	155.309

4 Conclusion and discussion

Robots need frequent recharging to provide continuous service. We proposed an intelligent path planner for smart access of distributed charging points on the map. The algorithm first calculates an overall priority score based on the tasks undertaken and the remaining battery power. The total power required to complete each task is calculated. The algorithm also handles critical conditions when the remaining battery power of the robot falls below a threshold. Complete information about the free charging points is available to the robots, and they can immediately plan a path to the nearest empty charging point. If all the charging points are occupied, the algorithm can forcefully empty a charging point occupied by a robot which is sufficiently charged to do the next task. In this case, collision-free paths are planned, and the requesting robot is given the shortest path. The charging stations could be distributed on the map. Simulation results confirm the merits of the proposed algorithm compared to the traditional approaches. Testing in real environments is considered as future work.

References

1. Hart P, Nilsson N, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. IEEE Trans Syst Sci Cybern 4(2):100–107. https://doi.org/10.1109/TSSC.1968.300136
2. Kavraki L, Svestka P, Latombe JC, Overmars M (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Robot Autom 12(4):566–580. https://doi.org/10.1109/70.508439
3. Kim M, Kim HW, Chong NY (2007) Automated robot docking using direction sensing rfid. In: 2007 IEEE international conference on robotics and automation, pp 4588–4593. https://doi.org/10.1109/ROBOT.2007.364186
4. Li Z (2017) Wireless charging system based on substation inspection robot. In: 2017 9th international conference on modelling, identification and control (ICMIC), pp 919–923. https://doi.org/10.1109/ICMIC.2017.8321587
5. Luo R, Su K (2003) A multiagent multisensor based real-time sensory control system for intelligent security robot. In: IEEE international conference on robotics and automation, 2003. Proceedings. ICRA '03, vol 2, pp 2394–2399. https://doi.org/10.1109/ROBOT.2003.1241951
6. Masehian E, Mohamadnejad N (2015) Path planning of nonholonomic flying robots using a new virtual obstacle method. In: 2015 3rd RSI international conference on robotics and mechatronics (ICROM), pp 612–617
7. Quigley M, Conley K, Gerkey BP, Faust J, Foote T, Leibs J, Wheeler R, Ng AY (2009) Ros: an open-source robot operating system. In: ICRA workshop on open source software
8. Ravankar A, Ravankar AA, Kobayashi Y, Jixin L, Emaru T, Hoshino Y (2015) An intelligent docking station manager for multiple mobile service robots. In: 2015 15th international conference on control, automation and systems (ICCAS), pp 72–78. https://doi.org/10.1109/ICCAS.2015.7364881
9. Ravankar A, Ravankar AA, Hoshino Y, Emaru T, Kobayashi Y (2016a) On a hopping-points svd and hough transform based line detection algorithm for robot localization and mapping. Int J Adv Robot Syst 13(3):98. https://doi.org/10.5772/63540
10. Ravankar A, Ravankar AA, Kobayashi Y, Emaru T (2016b) Avoiding blind leading the blind. Int J Adv Robot Syst 13(6):1729881416666,088. https://doi.org/10.1177/1729881416666088
11. Ravankar A, Ravankar A, Kobayashi Y, Hoshino Y, Peng CC (2018a) Path smoothing techniques in robot navigation: state-of-the-art, current and future challenges. Sensors 18(9):3170. https://doi.org/10.3390/s18093170
12. Ravankar A, Ravankar A, Kobayashi Y, Hoshino Y, Peng CC, Watanabe M (2018b) Hitchhiking based symbiotic multi-robot navigation in sensor networks. Robotics 7(3):37. https://doi.org/10.3390/robotics7030037
13. Ravankar A, Ravankar A, Hoshino Y, Kobayashi Y (2019) Virtual obstacles for safe mobile robot navigation. In: 2019 8th international congress on advanced applied informatics (IIAI-AAI), pp 552–555
14. Ravankar A, Ravankar A, Rawankar A, Hoshino Y, Kobayashi Y (2019a) Itc: infused tangential curves for smooth 2d and 3d navigation of mobile robots. Sensors 19(20):4384. https://doi.org/10.3390/s19204384
15. Ravankar A, Ravankar AA, Hoshino Y, Kobayashi Y (2019b) On sharing spatial data with uncertainty integration amongst multiple robots having different maps. Appl Sci 9(13):2753. https://doi.org/10.3390/app9132753
16. Ravankar AA, Ravankar A, Emaru T, Kobayashi Y (2020) Line segment extraction and polyline mapping for mobile robots in indoor structured environments using range sensors. SICE J Control Meas Syst Integr 13(3):138–147. https://doi.org/10.9746/jcmsi.13.138
17. Silverman MC, Nies D, Jung B, Sukhatme GS (2002) Staying alive: a docking station for autonomous robot recharging. In: ICRA, IEEE, pp 1050–1055
18. Song G, Wang H, Zhang J, Meng T (2011) Automatic docking system for recharging home surveillance robots. IEEE Trans Consum Electron 57(2):428–435. https://doi.org/10.1109/TCE.2011.5955176
19. Stentz A, Mellon IC (1993) Optimal and efficient path planning for unknown and dynamic environments. Int J Robot Autom 10:89–100
20. Vaz P, Ferreira R, Grossmann V, Ribeiro M (1997) Docking of a mobile platform based on infrared sensors. In: Proceedings

of the IEEE international symposium on industrial electronics, 1997. ISIE '97, vol 2, pp 735–740. https://doi.org/10.1109/ISIE.1997.649089

21. Wu YC, Teng MC, Tsai YJ (2009) Robot docking station for automatic battery exchanging and charging. In: IEEE international conference on robotics and biomimetics, 2008. ROBIO 2008, pp 1043–1046. https://doi.org/10.1109/ROBIO.2009.4913144
22. Yuan Q, Liu J (2017) Application of improved k-medoids algorithm in charging station planning for mobile robot. In: 2017 7th IEEE international conference on electronics information and emergency communication (ICEIEC), pp 322–325. https://doi.org/10.1109/ICEIEC.2017.8076573