IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier

Coverage Path Planning for Decomposition Reconfigurable Grid-Maps Using Deep Reinforcement Learning based Travelling Salesman Problem

PHONE THIHA KYAW[1,2], AUNG PAING[2], THEINT THEINT THU[2], RAJESH ELARA MOHAN[1], ANH VU LE[3], AND PRABAKARAN VEERAJAGADHESWAR[1]

[1]ROAR Lab, Engineering Product Development, Singapore University of Technology and Design, Singapore 487372, Singapore (e-mail: mlsdphonethk@gmail.com, rajeshelara@sutd.edu.sg, prabakaran@sutd.edu.sg)
[2]Department of Mechatronic Engineering, Yangon Technological University, Insein, Myanmar (e-mail: aungpaingcha@gmail.com, theintthu@ytu.edu.mm)
[3]Optoelectronics Research Group, Faculty of Electrical and Electronics Engineering, Ton Duc Thang University, Ho Chi Minh City 700000, Vietnam

Corresponding author: Anh Vu Le (e-mail: leanhvu@tdtu.edu.vn).

This research is supported by the National Robotics Programme under its Robotics Enabling Capabilities and Technologies (Funding Agency Project No. 192 25 00051), National Robotics Programme under its Robot Domain Specific (Funding Agency Project No. 192 22 00058) and administered by the Agency for Science, Technology and Research.

ABSTRACT Optimizing the coverage path planning (CPP) in robotics has become essential to accomplish efficient coverage applications. This work presents a novel approach to solve the CPP problem in large complex environments based on the Travelling Salesman Problem (TSP) and Deep Reinforcement Learning (DRL) leveraging the grid-based maps. The proposed algorithm applies the cellular decomposition methods to decompose the environment and generate the coverage path by recursively solving each decomposed cell formulated as TSP. A solution to TSP is determined by training Recurrent Neural Network (RNN) with Long Short Term Memory (LSTM) layers using Reinforcement Learning (RL). We validated the proposed method by systematically benchmarked with other conventional methods in terms of path length, execution time, and overlapping rate under four different map layouts with various obstacle density. The results depict that the proposed method outperforms all considered parameters than the conventional schemes. Moreover, simulation experiments demonstrate that the proposed approach is scalable to the larger grid-maps and guarantees complete coverage with efficiently generated coverage paths.

INDEX TERMS Coverage Path Planning, Cellular Reconfigurable Decomposition, Deep Reinforcement Learning, Recurrent Neural Network, Travelling Salesman Problem

I. INTRODUCTION

COVERAGE Path Planning (CPP) algorithms have been applied to several robotics applications in real-world such as floor-cleaning robots [1], [2], underwater operations [3], agricultural robots [4], tiling robotics [5]–[7], ship hull cleaning [8], benchmarking for inspection [9] and pavement sweeping [10], [11] to name a few. CPP describes an approach designed to determine a path that passes through all the workspace points while avoiding obstacles. A CPP algorithm is complete if the robot covers the workplace such that the union of all the sub-trajectories completely covers the workplace with finite time. The algorithm is non-overlapping if the robot does not cover the previously covered area. The total operational energy and time required for coverage, computation cost, and completeness determine the robustness of a CPP algorithm [12]. An efficient and optimal path generation algorithm needs to satisfy all the mentioned criteria simultaneously.

Generally, CPP algorithms can be classified as either offline or online [13]. Offline algorithms use fixed information, and the environment is assumed to be known in advance. In contrast, online coverage algorithms utilize real-time sen-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045027, IEEE Access

sor measurements and decisions to sweep the entire target area. Thus, these later algorithms are also called sensor-based coverage algorithms. In some scenarios, the method for solving the problem is to use the randomization approach. This method randomly sweeps the floor for long enough to cover the area thoroughly. Examples of commercial floor-cleaning robots based wholly or partially on this approach are the RC3000 by Karcher, Trilobite by Electrolux, and Roomba by iRobot [14]. There are advantages to this approach since no sophisticated sensors for localization nor expensive computational resources are needed. However, relying mostly on the randomized approach could not be beneficial, as the cost of operating the vehicle (in terms of energy and time) would be unaffordable.

The majority of CPP methods use approaches based on the grid information obtained from the initial map creation process. Since it is easier to create a grid map, coverage algorithms widely use these grid-based map representations [13]. However, grid maps suffer the exponential growth of memory usage because the resolution remains constant despite the environment complexity. As the map size increases, grid-based CPP methods usually utilize simple cost functions and demand high computational time leading to inefficient and costly paths, making them not usable in real-world scenarios. To efficiently address these limitations, the CPP methods need to consider the complexity and the size of the environment to generate scalable coverage paths with lesser path length and computation time.

In this work, we propose a new CPP method to determine an efficient and effective coverage path in large and complex environments by utilizing grid-based map representations. The main contributions are summarized as follows:

- A Recursive Approach for TSP-based CPP: We develop a CPP method, which uses a line-sweeping technique to decompose the grid-based environment into cells and solves each decomposed cell recursively formulating as an NP-hard Travelling Salesman Problem (TSP).
- A Deep Reinforcement Learning Approach for TSP: The proposed algorithm uses deep reinforcement learning to find the solution of TSP by training a Recurrent Neural Network (RNN), which consists of Long Short-Term Memory (LSTM) cells. REINFORCE algorithm, a policy gradient method for reinforcement learning is utilized to optimize the proposed neural network parameters. With the help of the proposed reward function designed explicitly for TSP in grid-based environments, this RL approach for TSP yields near-optimal tours with faster computation time.
- Simulation Studies: We conduct simulation studies to evaluate the performance of the proposed CPP approach from different perspectives in comparison with conventional methods. The experimental results in large complex environments demonstrate the efficiency and scalability of the proposed approach.

The rest of the article is organized as follows. We discuss related work in Section II and the proposed recursive TSP-based CPP algorithm in Section III. Section IV presents the reinforcement learning approach for finding the solution of TSP. We then discuss and analyze the RL training details and compare the algorithm's performance through extensive simulation studies in Section V and conclude in Section VI.

II. RELATED WORK

The problem of Coverage Path Planning (CPP) utilizing grid representation maps has been addressed by several authors in the literature. This grid representation was initially proposed by Moravec, and Elfes [15], where the authors mapped an indoor environment using a sonar sensor mounted on a mobile robot. Each grid cell in this representation occupies a value denoting the cell's availability, stating whether an obstacle is present or free. Since these grid representations only approximate the shape of the environment and its obstacles, grid-based methods were classified as approximate cellular decomposition methods [16]. These methods decompose the environment into specific grid shapes such as rectangles [17], triangles [18], or tetromino [19], rhombus [20]. Usually, each grid's size is square in shape and set to the same size as the robot to reduce unexplored areas. Utilizing grid-based methods for coverage path planning opens numerous opportunities for research and development in the field of robot navigation. There are numerous grid-based algorithms proposed for efficient CPP for mobile robots.

The first offline grid-based method for coverage path planning, called the Wavefront Algorithm, was proposed by Zelinsky et al. [21]. The algorithm defines a starting and a goal cell and uses a distance transform that propagates a wavefront from the goal to start to assign a specific number to each grid cell. A coverage path can be obtained by starting on the start cell and selecting the adjacent cell with the highest label that is unexplored. Another method by utilizing grid representation is the Spiral-STC (Spanning Tree Coverage) algorithm, proposed by [22]. This online approach generates the systematic spiral path by following a spanning tree of the partial grid map that the robot incrementally constructs using its onboard sensors. The authors validate the proposed method in simulation. Extended Spiral-STC, and selective path planning, modified A star, have been proposed in several works [23]–[25]. In other approaches, [26]–[28] proposed a novel neural network approach for CPP with obstacle avoidance of cleaning robots in non-stationary environments. This approach generates the robot path from the dynamic activity landscape of the neural network and the previous robot location. The authors claim that the proposed method is computationally simple, and they validate the approach in simulation. Nevertheless, most CPP algorithms designed for the grid representation still have several challenges to find the least optimal cost path in larger environments considering both total path length and computation time required.

Bormann et al. [29] proposed the heuristic approach for solving CPP called Grid-based Local Energy Minimization. The algorithm determines the potential next unvisited grid

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045027, IEEE Access

cell within the current neighborhood by minimizing the energy function. This approach never visits grid cells twice and terminates when all grid cells have been covered. Other well-known grid-based CPP methods are the Grid-based Travelling Salesman Problem (Grid-based TSP) algorithm [30] and Graph theory-based technique [31]. The planner decomposes the map into grid cells, and the solution to CPP is determined by finding the shortest path that visits all the grid cells as TSP. The authors approximated the travel distances between grid cells using A*. Since TSPs are NP-hard optimization problems, finding an optimal solution considering optimality and scalability has several limitations when dealing with larger environments.

III. PROPOSED RECURSIVE TSP-BASED CPP

This section explains the proposed recursive TSP-based approach for solving CPP. The environment map is decomposed by utilizing the cellular decomposition methods, and the algorithm recursively solves each decomposed cell that contains gridpoints, formulating as an NP-hard TSP optimization.

A. ENVIRONMENT MAP DECOMPOSITION METHOD

The final goal of resolving area coverage problems is to cover the free space areas defined in the environment map by navigating autonomously and following the CPP algorithm's predefined path. In this work, we utilize the ideas of approximate cellular decomposition methods and grid-based representation maps to achieve complete coverage tasks. First, the environment map is decomposed into finitely many robot-sized grids $m = \sum_i m_i$ where m_i denotes the gridpoint with index $i \in \mathbb{R}^2$. Each m_i has attached an associated binary value, which specifies whether an obstacle is present or if it is instead free space. Figure 1 illustrates the grid map decomposition with free and approximated occupied grids where each grid has the same size as the robot. The CPP solution can then be determined by finding a globally shortest path that visits all the gridpoint locations. This becomes TSP, and solving TSP on all the gridpoints offers a valid solution to the original CPP problem. However, since TSPs are NP-hard, this approach is undesirable when dealing with larger environments consisting of several gridpoint locations.

Thus, we propose to utilize the exact cell decomposition technique, also known as trapezoidal decomposition [32], to decompose the grid map, as illustrated in Figure 2. The grid map's free space is broken down into simple, non-overlapping regions called cells by sweeping a line through space from left to right. The cell boundaries are formed when the sweep line intersects with obstacle boundaries, as indicated in Figure 2a. Figure 2b represents the decomposed cells where each cell contains the corresponding centroid information and gridpoint locations. Figure 2c indicates the graph cell structure where a node represents a cell, and an edge represents the euclidean distance between each node. Instead of using graph traversal algorithms, we propose to determine the visit order of the cell graph by utilizing the

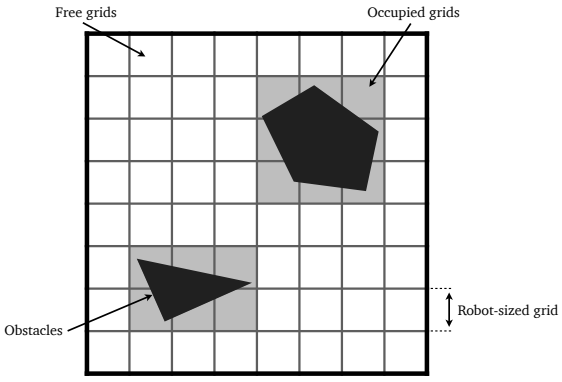

FIGURE 1: Decomposition of the environment into free and occupied robot-sized grids.

TSP approach, i.e., finding a permutation of all the cells to get the minimum total cost length that visits each cell exactly once, as indicated in Figure 2d.

B. CPP PROBLEM FORMULATION IN DECOMPOSED CELLS

We formalize the problem of CPP in the decomposed grid cells as a standard 2D TSP. Given a list of gridpoints and the distances between each pair of points, the TSP algorithm has to determine the shortest possible route that visits each gridpoint exactly once. It is a well-known NP-hard combinational optimization problem. Compared to Euclidean Distance, also known as the L2 norm in standard TSP, we use the Manhattan Distance or L1 norm to find the distance between each grid point m_i, where distance d can be calculated using the following equation:

$$d(x, y) = \sum_{i=1}^{m} \mid x_i - y_i \mid, \tag{1}$$

Then, the total tour length or the solution to TSP can be defined as given an input state graph s, represented as a sequence of gridpoint locations in a two-dimensional space $s = \{m_i\}_{i=1}^{n}$ where each $m_i = (x_i, y_i) \in \mathbb{R}^2$, we are concerned with finding a permutation of the nodes, i.e. a tour π, that visits each node once and has the minimum total length. The cost of a tour noted by a permutation π can be defined as:

$$L(\pi|s) = \sum_{i=1}^{n-1} \|m_{\pi(i)} - m_{\pi(i+1)}\|_1, \tag{2}$$

where $\|.\|_1$ denotes the L1 norm.

C. OVERVIEW OF THE PROPOSED ALGORITHM

By utilizing the proposed grid cell decomposition, and the computed cell visit order, the coverage path of the environment can be obtained by recursively solving each decomposed cell. Algorithm 1 describes the proposed recursive

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045027, IEEE Access

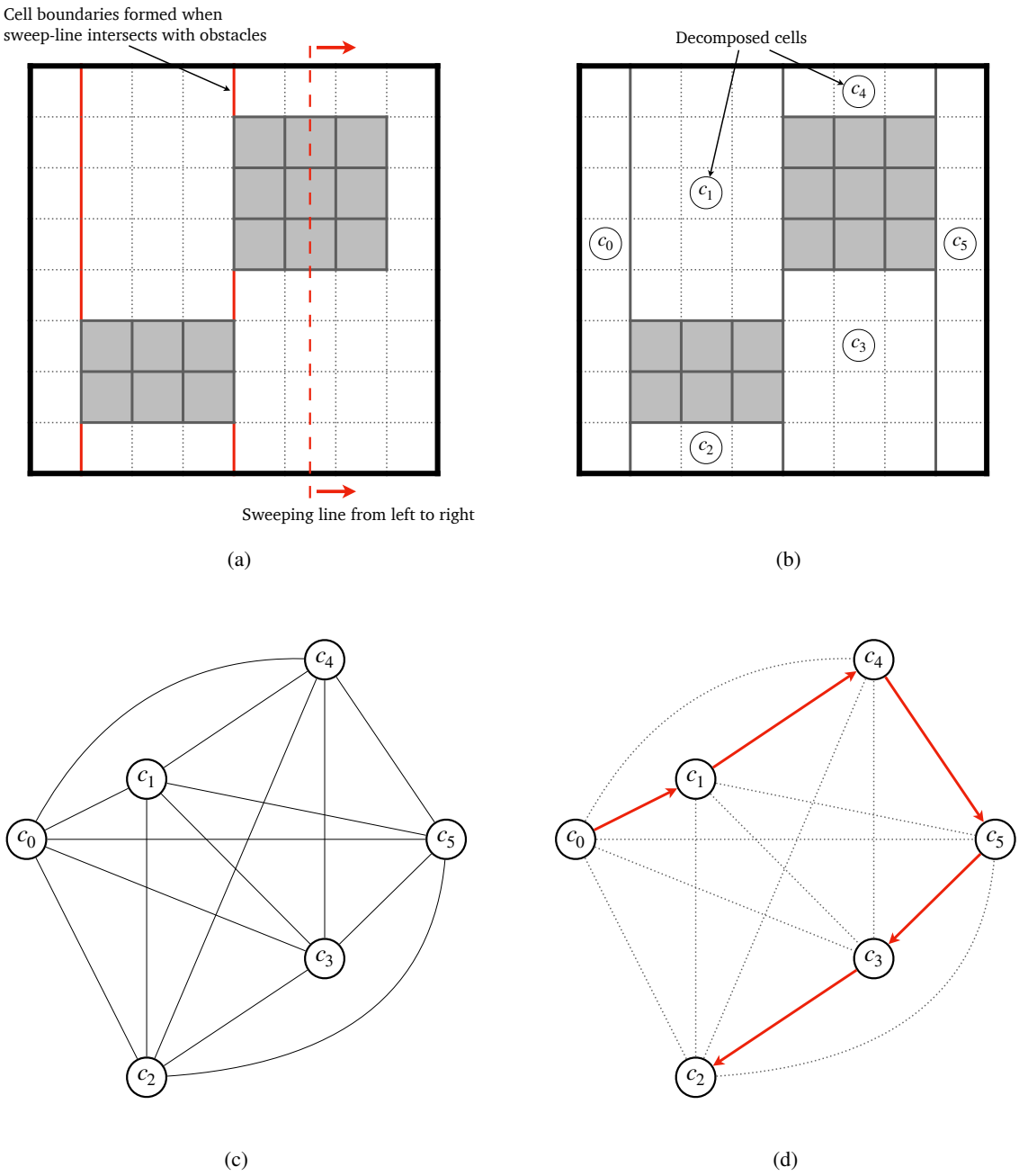

FIGURE 2: Decomposition of the grid map into cells and determining the cell visit order. (a) Sweeping from left-to-right. (b) Decomposed cells holding their corresponding centroid and gridpoint locations. (c) Graph cell structure with euclidean metric edges. (d) TSP approach to finding the minimal cell visit order.

algorithm. The input to the algorithm is the order list of cells C, in which each cell has the corresponding centroid and gridpoint locations, empty vector G to store all the solved gridpoints, and the starting index, which is used as an initial gridpoint to find the optimal path of each cell. Figure 3 represents the proposed recursive approach. All the points of the current cell are stored in the gridpoint vector g, and instead of solving only those points, we add the centroid of the upcoming cell into the gridpoint vector g as an endpoint of TSP to make the path more guidance towards the next cell, thereby reducing sub-optimal paths, as indicated in Figure 3a. Once the TSP is solved for a particular cell, the next cell's centroid is removed from the solution. In Figure 3b and c, the starting index for the subsequent TSP is determined by iterating over the next cell and finding the closest point from the endpoint of the solved TSP gridpoint list. The algorithm

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045027, IEEE Access

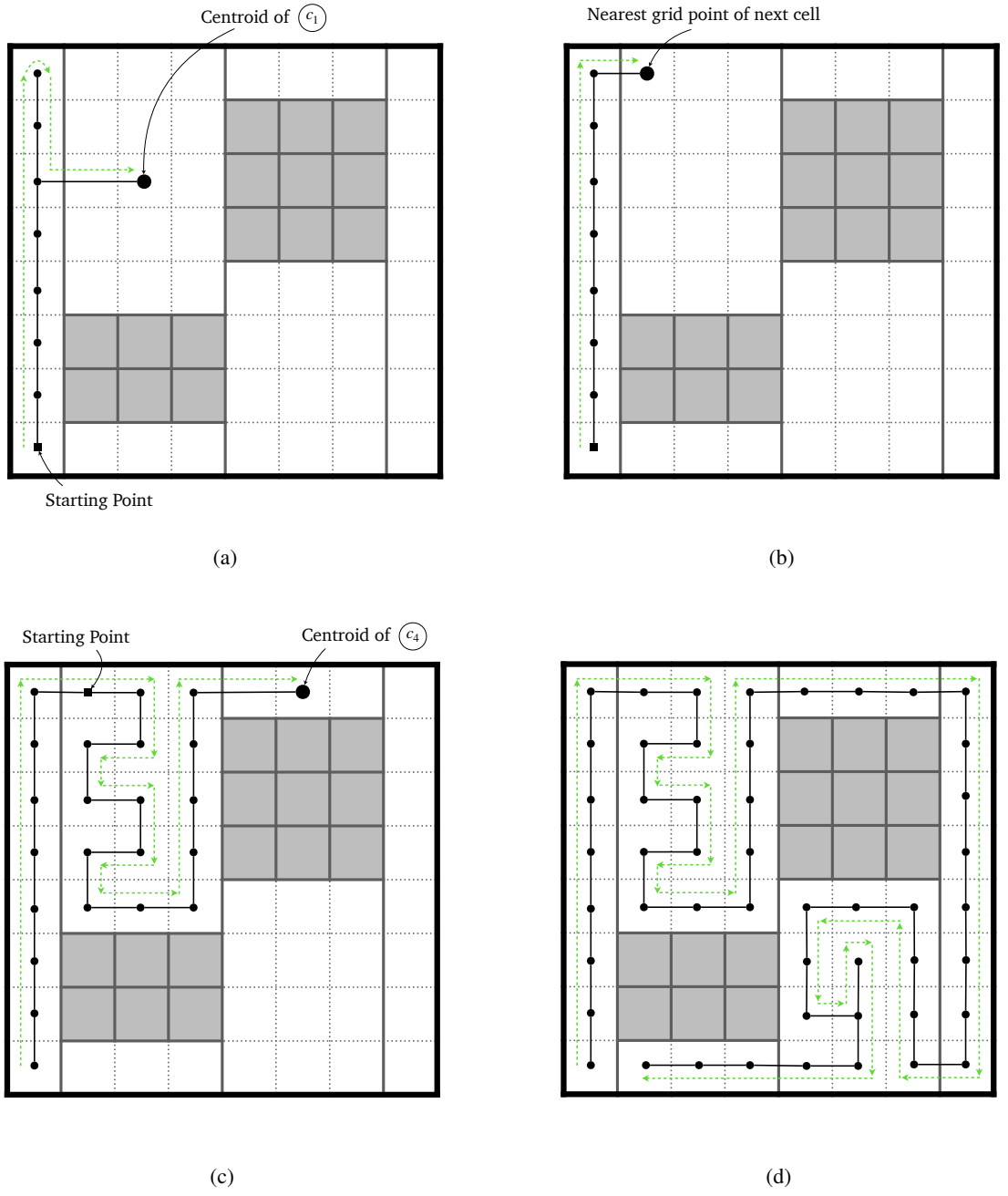

FIGURE 3: Recursive TSP-based CPP Steps: (a) centroid of cell-1 is added as an endpoint for TSP of cell-0, (b) starting point for TSP of cell-1 is determined by finding the closest point from the endpoint of cell-0, (c) solving TSP of cell-1 by appending centroid of cell-4 as an endpoint, and (d) final resulted coverage path.

then recursively resolves the solution of TSP by following the visit order of cells and remove the cell from the list once it is solved. This process continues until all the cells are solved (or the list is empty), which is the base case for our recursion. Figure 3d illustrates the resulted coverage path of the environment.

IV. REINFORCEMENT LEARNING APPROACH FOR TSP-CPP

In this work, we propose to obtain the coverage path in each decomposed cell by formulating as 2D Manhattan TSP. The solution of TSP is determined by training recurrent neural networks using deep reinforcement learning.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045027, IEEE Access

Algorithm 1: Recursive TSP-based CPP Pseudocode

```
Function RecursiveCPP(C, G, index):
    if cells list C is empty then
        return gridpoints G
    end if
    else
        if length of cells list C > 1 then
            nextCentroid ← centroid of next cell c_{i+1}
        end if
        else
            nextCentroid ← centroid of initial cell
        end if
        g ← all gridpoints of current cell c_i ∈ C
        g ← g + nextCentroid
        endIndex ← last index of g
        solveList ← SolveTSP(g, index, endIndex)
        remove nextCentroid from the solveList
        add all the gridpoints from solveList into G
        lastPoint ← last gridpoint of solveList
        loop through each gridpoint of next cell
          c_{i+1} ∈ C to get the nearest gridpoint index
          from lastPoint
        remove the current cell c_i from C
        return RecursiveCPP(C, G, index)
    end if
```

A. RL PROBLEM FORMULATION

We formalize the TSP in the decomposed grid cells as a standard reinforcement learning setting. Let S be the state space, where each state $s_t \in S$ is defined as the set of all previously visited gridpoints, i.e., $s_t = \{m_{\pi(i)}\}_{i=1}^{t}$ and A be the action space, where each action $a_t \in A$ is the next selected grid point, i.e., $a_t = m_{\pi(t+1)}$. Given a set of visited gridpoints s_t, we denote the policy as $\pi_\theta(a_t \mid s_t)$, which will return a probability distribution over the next candidate gridpoints a_t that have not been chosen. The policy is represented by a neural network and parameterized by θ, which are the trainable parameters of the network. In this work, we define the negative of the tour length described in Equation 2 as the total expected reward $r(\pi|s)$, which we seek to maximize:

$$\begin{aligned} r(\pi|s) &= -L(\pi|s) \\ &= -\sum_{i=1}^{n-1} \|m_{\pi(i)} - m_{\pi(i+1)}\|_1 \end{aligned} \tag{3}$$

B. NEURAL NETWORK ARCHITECTURE

Our neural network architecture, utilizing the actor-critic architecture is depicted in Figure 4. Following [33], the proposed neural network architecture uses the chain rule to factorize the probability of tour π in Equation 2 as:

$$p(\pi|s) = \prod_{i=1}^{n} p(\pi(i) \mid \pi(<i), s), \tag{4}$$

Furthermore, each component on the right side of Equation 4 is processed sequentially by the softmax modules. Similar to [33], we employ the pointer network architecture [34], illustrated in Figure 5, as our actor policy model, which consists of two RNN modules, encoders and decoders, each includes LSTM cells [35] to parameterize $p(\pi|s)$. The encoder network examines the input states, one gridpoint location at a time, and converts it into a series of latent memory states $\{enc_i\}_{i=1}^{n}$ where $enc_i \in \mathbb{R}^d$. The input to the encoder network at timestep i is a d-dimensional embedding of 2D gridpoints m_i, obtained via a linear transformation of m_i, shared across all input steps. The decoder network also maintains its Latent memory states $\{dec_i\}_{i=1}^{n}$ where $dec_i \in \mathbb{R}^d$ and utilizes the pointing mechanism to predict a distribution over the upcoming gridpoints to yield the optimal tour length. The pointing mechanism is parameterized by two learned attention matrices $K \in \mathbb{R}^{d\times d}$ and $Q \in \mathbb{R}^{d\times d}$ and an attention vector $v \in \mathbb{R}^d$ as:

$$u_i = \begin{cases} v^T \tanh(Kr_i + Qq), & \text{if } i \neq \pi(j) \text{ for all } j < i \\ -\infty, & \text{otherwise} \end{cases} \tag{5}$$

where q is a query vector from the hidden variable of the LSTM, and r_i is a reference vector containing the information of the context of all gridpoints. Then the probability distribution policy over all the candidate gridpoints is given by:

$$p = \pi_\theta(a_i \mid s_i) = \text{softmax}(u_i) \tag{6}$$

where we predict the next visited gridpoint $a_i = m_{\pi(i+1)}$, by sampling or choosing greedily from the policy as: $a_i = \text{argmax}(p)$.

C. OPTIMIZATION WITH REINFORCEMENT LEARNING

Solving NP-hard problems such as TSP and its variations using supervised learning is challenging since the performance of the trained model highly depends on the quality of the supervised labels. Furthermore, getting the training data with labels is costly and infeasible. On the contrary, RL provides a proper and simplistic paradigm for training neural networks, where an RL agent traverses different tours and observes the corresponding rewards to maximize the expected reward objective. Hence, in this work, we propose to optimize the policy of the actor pointer network parameters by utilizing the REINFORCE algorithm [36], which is a policy gradient method for RL. The reward function described in Equation 3 is utilized to optimize the parameters θ of the policy pointer network concerning the training objective, i.e., the expected tour length given an input state graph s as Equation 7:

$$J(\theta|s) = \mathbb{E}_{\pi\sim p_\theta(.|s)} r(\pi|s), \tag{7}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045027, IEEE Access

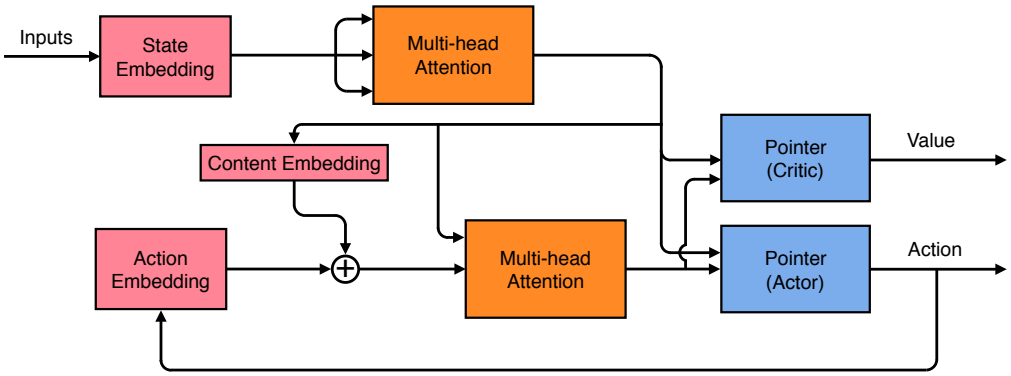

FIGURE 4: Proposed actor-critic architecture.

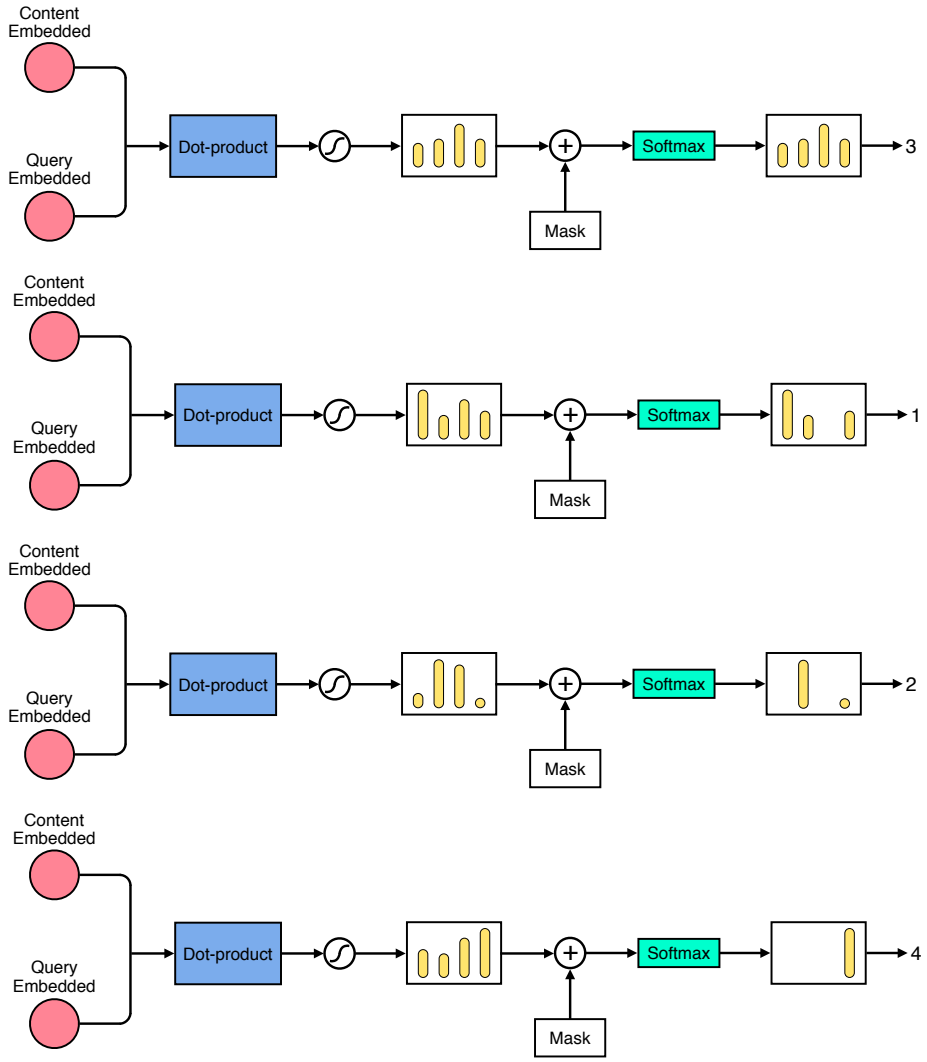

FIGURE 5: A pointer network architecture introduced by [34].

then we formulate the policy gradient of the objective using REINFORCE as in Equation 8, which controls stable updates during the optimization step.

$$\nabla_\theta J(\theta|s) = \hat{\mathbb{E}}_{\pi\sim p_\theta(.|s)}\left[\hat{A}_t \nabla_\theta \log p_\theta(\pi|s)\right], \qquad (8)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045027, IEEE Access

where $\hat{A}_t = r(\pi|s) - b(s)$ is an estimator of the advantage function at timestep t, where b(s) being the baseline that does not depend on the policy π and estimates the expected tour length to reduce the variance of the gradients.

The proposed baseline $b(s)$, which is the estimated tour length value, is obtained from the same pointer network without the final softmax layer, called a critic and parameterized by θ_v, where the value estimate or the baseline is predicted based on the final state input. The critic network parameters θ_v are trained using the stochastic gradient descent on a mean squared error objective between its predictions $b(s)$ and the reward tour length $r(\pi|s)$, which we obtain from the most recent episode:

$$J(\theta_v) = \frac{1}{B}\sum_{i=1}^{B}\left(b(s_i) - r(\pi_i|s_i)\right)^2 \tag{9}$$

V. RESULTS AND DISCUSSION

A. RL TRAINING DETAILS

For training the proposed network architecture, the Pytorch framework was used. Mini-batches of size 64 were applied with 6400 instances for one epoch, and the number of the epoch was set to be the same as the TSP length, i.e., for each n TSP, the model was trained for n epoch. All the training instances were generated randomly on the fly. After training for each TSP length, the trained model was then used as the pre-trained weight for the next TSP training. The Adam optimizer [37] was used for regularising the model with the initial learning rate of 3×10^{-4}. During the training, the model was compared with Lin–Kernighan heuristic (LKH) [38] as the baseline in each epoch to observe the model performance. The range of pointer value u_i is clipped between $[-C, C]$ with the value of $C = 10$ similar to [33]. The reward function described in Equation 3 was used, and the training was done for TSP20, TSP30, and TSP50. Figure 6 plots the proposed model's training log compared to the optimal LKH solver. The trained model of each TSP length converges fast to the near-optimal solution and generalizes well in the later training steps.

The trained model was validated by comparing it with the optimal LKH solver, and the vehicle routing solver from Google OR Tools [39]. We utilized the two solution search strategies from OR Tools; the PATH CHEAPEST ARC algorithm and the GUIDED LOCAL SEARCH as the most efficient metaheuristic for TSP optimization, where the search time limit for each TSP instance was set to 5s, and the testing was done for TSP20, TSP50, TSP100, and TSP250. Table 1 describes the average comparison results of each trained model and the baselines, where 1000 TSP instances were used for each evaluation tour length. Overall the results in Table 1 indicate that our proposed trained models could generalize well to larger TSP instances by merely training on smaller TSP sizes. Furthermore, as the TSP size increases, the inference time needed to generate the optimal results for methods like LKH increases exponentially. In contrast, the proposed RL solution achieves the lowest inference time with 3 times and 20 times less than the two OR Tools strategies and 12 times lower than the LKH.

B. SIMULATION STUDIES

This section presents the simulation studies to illustrate the scalability of the proposed recursive TSP-based CPP approach using RL. Experiments were conducted on four random map layouts of $25\,\mathrm{m} \times 25\,\mathrm{m}$ size displayed in Figure 7. Robot of dimensions $0.5\ \mathrm{meter} \times 0.5\ \mathrm{meter}$ was simulated and used as the grid size in cell decomposition. In this study, a Grid-based TSP approach was utilized to benchmark the performance of the proposed method. Conventional coverage planning techniques such as zigzag and spiral were also adopted for the comparison. Note that Google OR Tools was utilized to find the optimal solution of the Grid TSP method. All methods were implemented using C++, and the experiments were conducted on computing nodes with the following specifications: Intel Core i7-9750H processor and 16GB Memory. Three metrics were used for the performance evaluation as follows: 1) path length, 2) execution time, and 3) overlapping rate. The path length measures the final tour of the path generated by the CPP algorithm, where Manhattan Distance was used as a distance metric to measure the distance between each gridpoint. The execution time measures the running time of the CPP algorithm required to obtain the final coverage path. The overlapping rate measures the ratio between the revisited gridpoints and the total coverable gridpoints. In this work, we designed two experiments to validate the results and demonstrate the impact of the previously mentioned metrics on the proposed approach.

In the first experiment, diverse shapes and a different number of obstacles were added to the maps to study the efficiency of the proposed method in complex environments. Table 2 describes the experiment's comparison results on four test map layouts with several percent of obstacle quantity. Figure 8 illustrates the coverage path results of the proposed method and benchmark methods on Map-0 layout with 5% obstacles. Overall, the data in Table 2 indicates that all the validated techniques have considerably different values in terms of CPP criteria, including path length, execution time, and overlap rate and better from the top row to bottom row. All the performance metrics of each method are proportional to the complexity of obstacles inside the tested map layouts. Specifically, if the map is more complicated with a higher percentage of obstacles, the outperform between the proposed RL-TSP-based method and other tested methods is more obvious. Although the fastest execution time is achieved, the simple zigzag and spiral techniques linking the pair by straight lines and outer-wise order produce higher cost weights and lead to larger overlapped areas. The evolutionary Grid-based TSP strategy achieves the second-best method with lower path lengths and overlapping rate. However, determining the near-optimal solution utilizing the TSP strategy has several drawbacks in path execution time when dealing with larger map layouts consisting of many gridpoints. This data confirms that the proposed method

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045027, IEEE Access

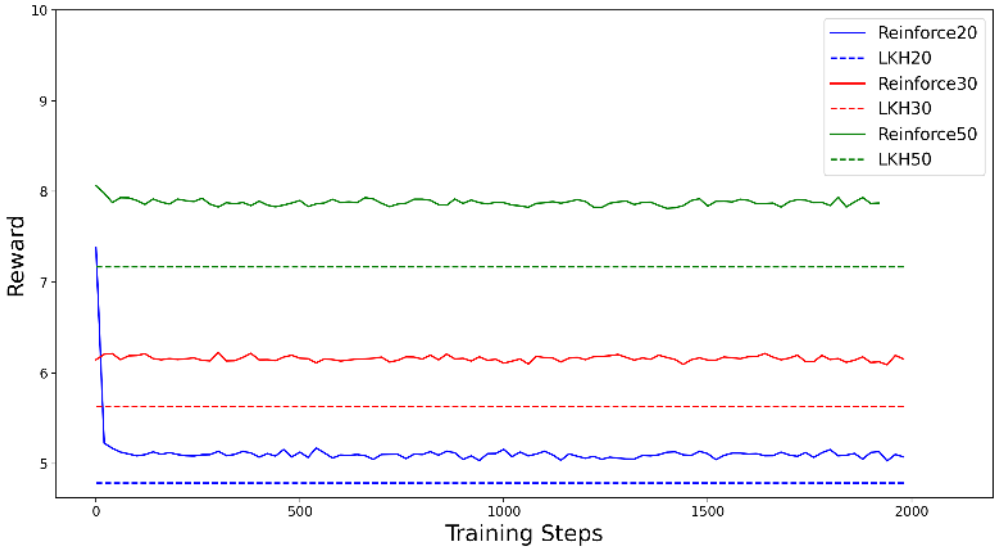

FIGURE 6: Training log for TSP20, TSP30, TSP50 and their respective baselines.

TABLE 1: Average comparison results of the trained models and the baselines obtained by running on 1000 TSP instances.

Method	TSP 20		TSP 50		TSP 100		TSP 250	
	Tour Length	Time (s)	Tour Length	Time (s)	Tour Length	Time (s)	Tour Length	Time (s)
LKH	4.506	23.04	6.948	244.02	9.757	1284.35	14.877	10703.73
OR-Tools (Cheapest)	4.787	8.08	7.178	153.48	10.005	574.89	15.532	2203.86
OR-Tools (Guided)	4.587	5002.57	7.167	5006.14	9.772	5018.81	15.450	5106.12
TSP@Reinforce20	5.632	36.851	9.063	104.563	12.143	220.928	26.808	654.055
TSP@Reinforce30	5.650	38.947	8.857	100.551	12.367	219.295	22.104	655.677
TSP@Reinforce50	5.990	37.106	8.780	100.514	11.770	219.741	18.958	653.779

TABLE 2: Comparison results of the proposed method and benchmark methods on four different map layouts with different number of obstacles for the performance evaluation metrics.

		Number of Obstacles (%)								
		0			5			10		
	Method	Path Length (m)	Overlapping Rate (%)	Execution Time (s)	Path Length (m)	Overlapping Rate (%)	Execution Time (s)	Path Length (m)	Overlapping Rate (%)	Execution Time (s)
Map-0	ZigZag	1259	8.44	**0.174**	1254	12.67	0.281	1258	18.91	**0.427**
	Spiral	1294	11.46	0.198	1217	9.34	**0.265**	1169	10.64	0.572
	Grid-TSP	1216	3.75	73.65	1175	5.59	87.54	1135	7.32	100.9
	Proposed	**1193**	**1.94**	5.284	**1158**	**4.04**	9.56	**1112**	**5.16**	12.84
Map-1	ZigZag	1232	6.68	**0.165**	1228	10.83	**0.247**	1219	14.52	**0.391**
	Spiral	1250	8.24	0.217	1243	12.16	0.362	1231	15.73	0.478
	Grid-TSP	1204	4.25	197.4	1176	6.09	173.6	1134	6.54	201.1
	Proposed	**1185**	**2.59**	6.472	**1170**	**4.62**	11.93	**1128**	**5.98**	15.29
Map-2	ZigZag	1264	7.09	**0.168**	1256	10.95	**0.353**	1231	13.15	**0.436**
	Spiral	1253	6.31	0.272	1287	13.57	0.418	1271	16.84	0.605
	Grid-TSP	1219	3.42	125.3	1192	5.16	163.6	1145	5.27	159.5
	Proposed	**1213**	**2.96**	7.805	**1182**	**4.28**	10.94	**1138**	**4.61**	14.36
Map-3	ZigZag	1270	9.38	0.196	1277	14.61	0.398	1274	19.06	**0.475**
	Spiral	1241	6.85	**0.182**	1235	10.86	**0.307**	1267	18.29	0.538
	Grid-TSP	1208	4.03	97.50	1177	5.64	125.9	1142	6.75	140.8
	Proposed	**1202**	**3.51**	6.937	**1163**	**4.39**	10.65	**1125**	**5.13**	15.94

is suitable for dynamic and cluttered workspaces. Further analyzing the numerical data, we can notice as follows. The execution time of the proposed method grows linearly as the percent of obstacles increases. This is because the proposed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045027, IEEE Access

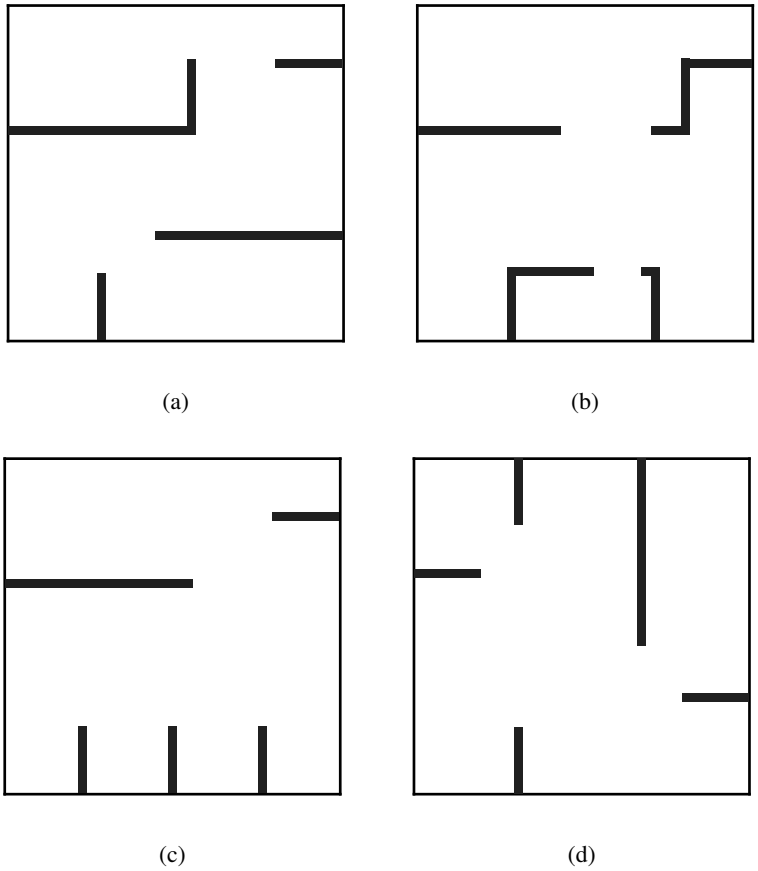

FIGURE 7: Four randomly simulated test map layouts: (a) Map-0, (b) Map-1, (c) Map-2, and (d) Map-3.

TABLE 3: Average comparison results of the proposed method and benchmark methods on four different map layouts with different number of obstacles for the scalability experiment.

		Number of Obstacles (%)								
		0			5			10		
	Method	Path Length (m)	Overlapping Rate (%)	Execution Time (s)	Path Length (m)	Overlapping Rate (%)	Execution Time (s)	Path Length (m)	Overlapping Rate (%)	Execution Time (s)
Map-0	ZigZag	6985	10.04	7.436	6968	13.80	**8.036**	7014	20.35	**10.09**
	Spiral	7201	12.73	**6.482**	6804	11.10	12.54	6536	12.11	11.67
	Grid-TSP	6751	5.74	2.55e3	6514	6.67	2.93e3	6305	8.34	3.29e3
	Proposed	**6648**	**4.23**	9.63	**6452**	**5.62**	14.32	**6210**	**6.57**	18.13
Map-1	ZigZag	6880	8.26	**9.648**	6839	12.31	**8.84**	6792	15.99	**9.58**
	Spiral	6950	9.59	10.79	6904	13.37	9.11	6860	17.29	11.40
	Grid-TSP	6720	5.67	3.91e3	6553	7.25	4.38e3	6309	7.79	4.10e3
	Proposed	**6599**	**3.81**	12.49	**6518**	**6.84**	16.46	**6295**	**7.58**	20.38
Map-2	ZigZag	7016	8.23	10.19	7004	12.45	11.50	6832	14.29	**8.52**
	Spiral	6972	7.49	**8.20**	7179	15.11	**9.35**	7101	18.56	9.82
	Grid-TSP	6772	4.41	3.98e3	6652	6.56	5.08e3	6367	6.50	4.93e3
	Proposed	**6754**	**4.29**	11.94	**6593**	**5.87**	14.13	**6355**	**6.26**	17.90
Map-3	ZigZag	7080	10.93	**9.11**	7108	15.97	**9.34**	7076	20.49	**8.96**
	Spiral	6920	8.56	10.16	6896	12.56	10.46	7057	19.78	9.55
	Grid-TSP	6731	5.25	3.32e3	6556	6.91	3.97e3	6366	8.06	3.55e3
	Proposed	**6701**	**5.05**	11.16	**6496**	**6.08**	14.79	**6279**	**6.76**	20.17

algorithm's running time is proportional to the number of decomposed cells, while the Grid-TSP method depends on the number of grid points. Nevertheless, the proposed RL-based approach achieves both outperform in numerical values

of path length and overlapping rate. The proposed method's path length and overlapping rate are slightly about 1.1% and 1.2%, respectively, less than the Grid-TSP strategy as the second-best approach. Moreover, the proposed method yields a significantly smaller value in execution time, with 12 times lower than the second-best approach and 30 times higher than the conventional coverage techniques.

In the second experiment, we studied the proposed method's scalability by increasing the map layouts $1 \sim 10$ times the original size while maintaining their obstacle shapes and numbers as the same in the first experiment. Table 3 presents the average comparison results of the second experiment on the scalability of the proposed approach. From experimental data in the given Table, the significantly outperformed results of the proposed method in comparison with other methods still present for the terms of path length and overlapping rate despite the average numerical achievements are slightly reduced to numerical values of 0.8% and 0.9%, respectively. Moreover, the Grid-TSP method's execution time grows exponentially when the map layout size increases while the running time of the proposed approach remains the same as the first experiment, slightly increased by only 1.4 times.

VI. CONCLUSION

In this paper, we have presented the recursive CPP framework, which generates an efficient trajectory to cover the predefined grid-based workspace leveraging cell decomposition and state of the art RL-TSP techniques. The proposed method was validated in simulation environments with the three metrics. Although the proposed approach's execution time does not achieve the lowest value compared to conventional methods, our approach outperforms the benchmark methods in terms of path length and overlapping rate, which are the essential metrics in offline coverage path planning. The numerical results achieved by the proposed recursive TSP-based CPP are consistent for all tested map layouts with different sizes and obstacle settings and prove that it is feasible to deploy in real environments.

Our future work will focus on 1) redesigning the reward strategy to reduce the number of turns in the coverage path, 2) extending to online CPP with the integration of SLAM, 3) expand the proposed approach into a multi-robot coverage system, 4) explore better cell decomposition techniques to reduce the number of decomposed cells in the proposed method, 5) implementing the latest RL optimization algorithms to advance the training results and 6) testing the method in real environments.

ACKNOWLEDGMENT

The authors would like to thank Thein Than Tun for insightful comments and discussion.

REFERENCES

[1] A. V. Le, N. H. K. Nhan, and R. E. Mohan, "Evolutionary algorithm-based complete coverage path planning for tetriamond tiling robots," Sensors, vol. 20, no. 2, p. 445, 2020.

[2] A. V. Le, R. Parween, R. Elara Mohan, N. H. Khanh Nhan, and R. Enjikalayil, "Optimization complete area coverage by reconfigurable htrihex tiling robot," Sensors, vol. 20, no. 11, p. 3170, 2020.

[3] E. Galceran and M. Carreras, "Efficient seabed coverage path planning for asvs and auvs," in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2012, pp. 88–93.

[4] I. A. Hameed, D. Bochtis, and C. A. Sørensen, "An optimized field coverage planning approach for navigation of agricultural robots in fields involving obstacle areas," International journal of advanced robotic systems, vol. 10, no. 5, p. 231, 2013.

[5] A. V. Le, P.-C. Ku, T. Than Tun, N. Huu Khanh Nhan, Y. Shi, and R. E. Mohan, "Realization energy optimization of complete path planning in differential drive based self-reconfigurable floor cleaning robot," Energies, vol. 12, no. 6, p. 1136, 2019.

[6] P. Veerajagadheswar, K. Ping-Cheng, M. R. Elara, A. V. Le, and M. Iwase, "Motion planner for a tetris-inspired reconfigurable floor cleaning robot," International Journal of Advanced Robotic Systems, vol. 17, no. 2, p. 1729881420914441, 2020.

[7] Y. Shi, M. R. Elara, A. V. Le, V. Prabakaran, and K. L. Wood, "Path tracking control of self-reconfigurable robot htetro with four differential drive units," IEEE Robotics and Automation Letters, vol. 5, no. 3, pp. 3998–4005, 2020.

[8] M. Muthugala, A. V. Le, E. Sanchez Cruz, M. Rajesh Elara, P. Veerajagadheswar, and M. Kumar, "A self-organizing fuzzy logic classifier for benchmarking robot-aided blasting of ship hulls," Sensors, vol. 20, no. 11, p. 3215, 2020.

[9] V. Prabakaran, A. V. Le, P. T. Kyaw, R. E. Mohan, P. Kandasamy, T. N. Nguyen, and M. Kannan, "Hornbill: A self-evaluating hydro-blasting reconfigurable robot for ship hull maintenance," IEEE Access, vol. 8, pp. 193 790–193 800, 2020.

[10] A. V. Le, A. A. Hayat, M. R. Elara, N. H. K. Nhan, and K. Prathap, "Reconfigurable pavement sweeping robot and pedestrian cohabitant framework by vision techniques," IEEE Access, vol. 7, pp. 159 402–159 414, 2019.

[11] L. Yi, A. V. Le, A. A. Hayat, C. S. C. S. Borusu, R. E. Mohan, N. H. K. Nhan, and P. Kandasamy, "Reconfiguration during locomotion by pavement sweeping robot with feedback control from vision system," IEEE Access, vol. 8, pp. 113 355–113 370, 2020.

[12] K. P. Cheng, R. E. Mohan, N. H. K. Nhan, and A. V. Le, "Multi-objective genetic algorithm-based autonomous path planning for hinged-tetro reconfigurable tiling robot," IEEE Access, vol. 8, pp. 121 267–121 284, 2020.

[13] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," Robotics and Autonomous systems, vol. 61, no. 12, pp. 1258–1276, 2013.

[14] J. Palacin, T. Palleja, I. Valganón, R. Pernia, and J. Roca, "Measuring coverage performances of a floor cleaning mobile robot using a vision system," in Proceedings of the 2005 IEEE International Conference on Robotics and Automation. IEEE, 2005, pp. 4236–4241.

[15] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in Proceedings. 1985 IEEE international conference on robotics and automation, vol. 2. IEEE, 1985, pp. 116–121.

[16] H. Choset, "Coverage for robotics–a survey of recent results," Annals of mathematics and artificial intelligence, vol. 31, no. 1-4, pp. 113–126, 2001.

[17] A. V. Le, M. Arunmozhi, P. Veerajagadheswar, P.-C. Ku, T. H. Q. Minh, V. Sivanantham, and R. E. Mohan, "Complete path planning for a tetris-inspired self-reconfigurable robot by the genetic algorithm of the traveling salesman problem," Electronics, vol. 7, no. 12, p. 344, 2018.

[18] R. Parween, A. V. Le, Y. Shi, and M. R. Elara, "System level modeling and control design of htetrakis–a polyiamond inspired self-reconfigurable floor tiling robot," IEEE Access, vol. 8, pp. 88 177–88 187, 2020.

[19] B. Ramalingam, A. K. Lakshmanan, M. Ilyas, A. V. Le, and M. R. Elara, "Cascaded machine-learning technique for debris classification in floor-cleaning robot application," Applied Sciences, vol. 8, no. 12, p. 2649, 2018.

[20] A. V. Le, R. Parween, P. T. Kyaw, R. E. Mohan, T. H. Q. Minh, and C. S. C. S. Borusu, "Reinforcement learning-based energy-aware area coverage for reconfigurable hrombo tiling robot," IEEE Access, 2020.

[21] A. Zelinsky, R. A. Jarvis, J. Byrne, S. Yuta et al., "Planning paths of complete coverage of an unstructured environment by a mobile robot," in Proceedings of international conference on advanced robotics, vol. 13, 1993, pp. 533–538.

[22] Y. Gabriely and E. Rimon, "Spiral-stc: An on-line coverage algorithm of grid environments by a mobile robot," in Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292), vol. 1. IEEE, 2002, pp. 954–960.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3045027, IEEE Access

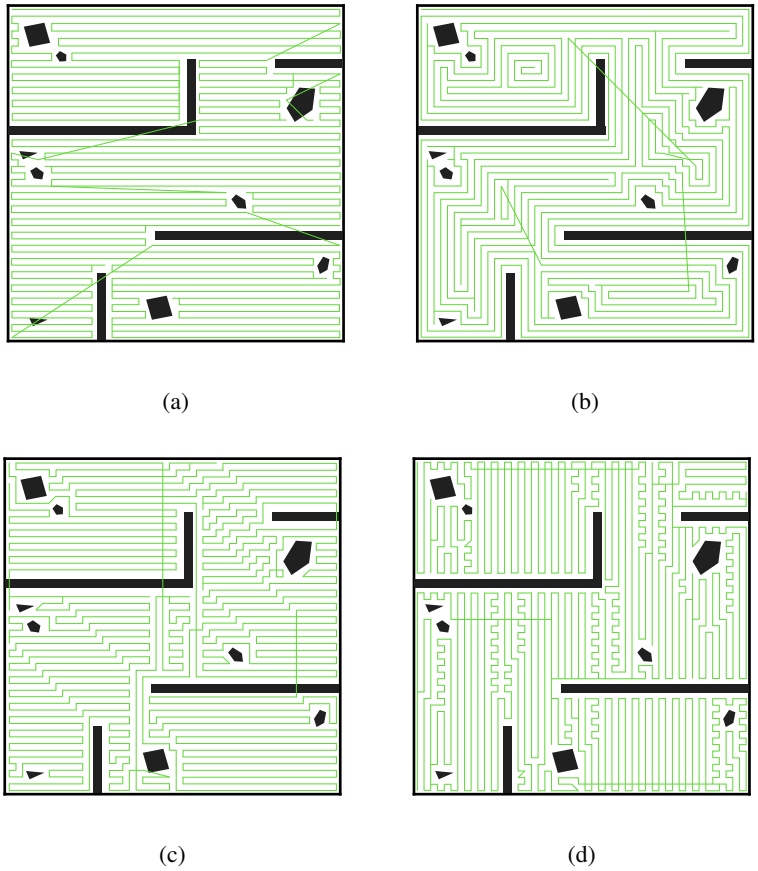

FIGURE 8: Coverage results on test map layout with 5% obstacles. (a), (b), (c), and (d) are the Map-0 results of ZigZag, Spiral, Grid-TSP and Proposed methods respectively.

[23] E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra, and C. Bustacara, "Bsa: a complete coverage algorithm," in Proceedings of the 2005 IEEE International Conference on Robotics and Automation. IEEE, 2005, pp. 2040–2044.

[24] A. V. Le, V. Prabakaran, V. Sivanantham, and R. E. Mohan, "Modified a-star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor," Sensors, vol. 18, no. 8, p. 2585, 2018.

[25] J. Yin, K. G. S. Apuroop, Y. K. Tamilselvam, R. E. Mohan, B. Ramalingam, and A. V. Le, "Table cleaning task by human support robot using deep learning technique," Sensors, vol. 20, no. 6, p. 1698, 2020.

[26] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 34, no. 1, pp. 718–724, 2004.

[27] A. Manimuthu, A. V. Le, R. E. Mohan, P. Veerajagadeshwar, N. Huu Khanh Nhan, and K. Ping Cheng, "Energy consumption estimation model for complete coverage of a tetromino inspired reconfigurable surface tiling robot," Energies, vol. 12, no. 12, p. 2257, 2019.

[28] A. K. Lakshmanan, R. E. Mohan, B. Ramalingam, A. V. Le, P. Veerajagadeshwar, K. Tiwari, and M. Ilyas, "Complete coverage path planning using reinforcement learning for tetromino based cleaning and maintenance robot," Automation in Construction, vol. 112, p. 103078, 2020.

[29] R. Bormann, J. Hampp, and M. Hägele, "New brooms sweep clean-an autonomous robotic cleaning assistant for professional office cleaning," in 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2015, pp. 4470–4477.

[30] R. Bormann, F. Jordan, J. Hampp, and M. Hägele, "Indoor coverage path planning: Survey, implementation, analysis," in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 1718–1725.

[31] K. P. Cheng, R. E. Mohan, N. H. K. Nhan, and A. V. Le, "Graph theory-based approach to accomplish complete coverage path planning tasks for reconfigurable robots," IEEE Access, vol. 7, pp. 94 642–94 657, 2019.

[32] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, Principles of robot motion: theory, algorithms, and implementation. MIT press, 2005.

[33] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," arXiv preprint arXiv:1611.09940, 2016.

[34] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in Advances in neural information processing systems, 2015, pp. 2692–2700.

[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[36] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," Machine learning, vol. 8, no. 3-4, pp. 229–256, 1992.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[38] K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," European Journal of Operational Research, vol. 126, no. 1, pp. 106–130, 2000.

[39] L. Perron and V. Furno, "Or-tools google version 7.2," Available online: https://developers.google.com/optimization/ Access date: 2019-7-19.

PHONE THIHA KYAW is currently working as a Visit Fellow in the Robotics and Automation Research Laboratory (ROAR) at the Singapore University of Technology and Design. He is also a final year student in B.E. Mechatronics from Yangon Technological University. His research interests include autonomous robots, sensor fusion systems, control engineering, and computer vision applications. He participated in many different robotic competitions, including First Global Challenge 2017, which was held in Washington DC, and his Team Myanmar achieved rank 6 out of 163 teams.

AUNG PAING is currently pursuing his Bachelor Degree in Mechatronic Engineering at Yangon Technological University (YTU). He is also an assistant researcher in Computer Vision and Machine Learning Lab at YTU. His research interests are in computer vision applications, applied artificial intelligence and Industry 4.0.

THEINT THEINT THU received B.E. degree in Mechatronic Engineering from Mandalay Technological University and M.E. degree from Yangon Technological University, Myanmar, in 2004 and 2006, respectively. She also received Ph.D. degree in Mechatronic Engineering from Mandalay Technological University, in 2008 and Dr. Eng., degree in Computer and Information Sciences from Nagasaki University, Japan, in 2018. Currently, she is a professor and deputy head of department at Department of Mechatronic Engineering, Yangon Technological University. Her research interests include computer vision, data science, artificial intelligence, reconfigurable systems and parallel processing.

MOHAN RAJESH ELARA received the B.E. degree from the Bharathiar University, India, in 2003, and the M.Sc. and Ph.D. degrees from Nanyang Technological University in 2005 and 2012, respectively. He is currently an Assistant Professor with the Engineering Product Development Pillar, Singapore University of Technology and Design. He is also a Visiting Faculty Member with the International Design Institute, Zhejiang University, China. He has published over 80 papers in leading journals, books, and conferences. His research interests are in robotics with an emphasis on self-reconfigurable platforms as well as research problems related to robot ergonomics and autonomous systems. He was a recipient of the SG Mark Design Award in 2016 and 2017, the ASEE Best of Design in Engineering Award in 2012, and the Tan Kah Kee Young Inventors' Award in 2010.

ANH VU LE received the B.S. degree in electronics and telecommunications from the Hanoi University of Technology, Vietnam, in 2007, and the Ph.D. degree in electronics and electrical from Dongguk University, South Korea, in 2015. He is currently with the Opto-electronics Research Group, Faculty of Electrical and Electronics Engineering, Ton Duc Thang University, Ho Chi Minh City, Vietnam. He is also a Postdoctoral Research Fellow with the ROAR Laboratory, Singapore University of Technology and Design. His current research interests include robotics vision, robot navigation, human detection, action recognition, feature matching, and 3D video processing

PRABAKARAN VEERAJAGADHESWAR received his Bachelor degree in Electronics and Instrumentation Engineering from Sathyabama University, India in 2013, and Ph.D. degree in Advanced engineering form Tokyo Denki University in 2019. Currently, he is working as a Research Fellow in ROARS LAB at Singapore University of Technology and Design. He won SG Mark Design Award in 2017 for the designing of h-Tetro, a self-reconfigurable cleaning robot. His research interest includes the development of complete coverage path planning, SLAM framework and embedded control for reconfigurable, and climbing robots. He is also a visiting instructor for a design course of International Design Institute at Zhejiang University, China.