Augmenting Multi-Objective Genetic Algorithm and Dynamic Programming for Online Coverage Path Planning

Mina G. Sadek
Electronics, Communication and Computer Engineering dept.
Faculty of Engineering
Helwan University
Cairo, Egypt
sadek.mina@h-eng.helwan.edu.eg

Amr E. Mohamed
Electronics, Communication and Computer Engineering dept.
Faculty of Engineering
Helwan University
Cairo, Egypt
amr_mohamed@h-eng.helwan.edu.eg

Ahmed M. El-Garhy
Electronics, Communication and Computer Engineering dept.
Faculty of Engineering
Helwan University
Cairo, Egypt
agarhy2003@yahoo.co.in

***Abstract*—This paper introduces a sensor-based approach for finding an optimized solution for online coverage path planning problem. Compared to traditional approaches we can augment. Multi-objective optimization genetic algorithm (GA) with Dynamic Programming (DP) for finding a short path with complete coverage; while using on-board sensors data only. Simulation results prove the effectiveness of the proposed approach compared to current adapted approaches.**

Keywords—Online CPP, Multi-Objective Optimization, Genetic Algorithm, Dynamic Programming.

I. Introduction

The online coverage path planning (CPP) is the problem of finding a path that passes through all points of a given area; such area can have a known or unknown surroundings. Based on the available information the problem hardness is defined. However, the online CPP problem is classified as NP-hard problem [1]. Many applications embraces CPP algorithms such vacuum cleaning applications [2], automated harvesters [3], and complex structures inspection [4]. CPP problem solution is planning a path which ensures visiting all points in a given area while avoiding obstacles and guaranteeing differential constraints [5].

CPP algorithms are divided into two classes: online and offline classes [6]. The difference between these two classes is the surroundings information availability. On the one hand, offline CPP algorithms work only with stationary information and require the environment to be known in advance; which such scenario tends to be unrealistic. On the other hand, online CPP algorithms depend on real-time sensors measurements. CPP algorithms can be implemented using randomized or cellular decomposition-based techniques.

Nevertheless, the difference between these two techniques is that the randomized approach does not rely on any maps and do not guarantee complete coverage, unlike cellular decomposition-based approach are mostly complete. Finally, CPP optimization objectives are user-dependent; which can be optimized regarding minimum time, energy, path repetitions, and some turns. In this paper, we present a sensor-based approach to finding a near optimal solution for unknown environments through Merging GA and DP techniques. First, the proposed approach breaks the main problem into a set of simpler sub-problems; then Multi-objective GA and DP find an optimal solution for each sub-problem. Therefore, these sub-problems solutions are the main problem solution. This paper is organized as follows. Section II provides CPP related work. In section III the notation, problem formulation, and assumptions are provided. Our proposed approach is presented in section IV. Simulations and discussion are provided in section V and VI. Finally, section VII provides our conclusion.

II. Related work

A. Related work on Cellular Decomposition-Based Approach

In cellular decomposition based algorithms, the environment is decomposed into simpler non-overlapping sub-regions; using some decomposition techniques. Decomposition can be exact, such as trapezoidal [7], boustrophedon [8], or morse-based [9] decomposition techniques. Afterward, each sub-region can be covered using simple motion planning techniques such as back and forth boustrophedon motion (zigzag) [8], spiral or spanning tree coverage (STC) [10] algorithms.

Martin Waanders used boustrophedon decomposition as an exact cellular decomposition to divide the environment into multiple smaller regions (cells) [11]. Then inside each region, a simple motion path is generated to cover it fully. He optimized the order of visiting the regions by formulating the problem as a traveling salesperson problem (TSP) and finding the best order. However, this approach only considered known environments. E. Gonzalez et al. provided an online approach called Backtracking Spiral Algorithm (BSA) [12]. It can be considered a better technique from the normal spiral coverage algorithm. In BSA they used spiral filling paths based on squares of robots tool size. Then they extended it to cover partially occupied cells to achieve complete coverage [13]. They focused on completeness, avoiding obstacles and being online, however, did not consider time or energy consumed.
S. Bochkarev and S. L. Smith proposed an approach that finds a coverage path with a minimal number of robot turns [14]. They first use convex decomposition to decompose the area into simpler sub-areas, then produce straight (parallel) lines

in the decomposed sub-areas, then try to minimize the total number of turns. However, they do not guarantee complete area coverage and dont consider unknown environments. I. A. Hameed et al. provided an intelligent coverage approach for agricultural robots based on GA [15]. A complete coverage path is generated using back and forth motion. Then, the area is clustered into blocks based on obstacles. Finally, the GA is used to find the best order for block visits. However, they did not consider unknown environments.

B. Related work on Randomized CPP Approach

Evolutionary algorithms have been widely used in planning the best paths for the CPP. One of the successful and easy to implement optimization algorithms is the Genetic Algorithm (GA) [16]. It has been used in many CPP approaches, mostly offline. To find the optimum path for CPP, full knowledge about the environment is required. This makes it hard to apply one of the optimization algorithms in unknown environments. Some approaches combined cellular decomposition with evolutionary algorithms, they simplified the problem by decomposing the environment into sub-regions, then optimizing the sequence of visiting sub-regions, like in traveling salesperson problem (TSP), and when in a sub-region the robot uses one of the simple motion planning techniques to cover it. They differ in the used decomposition technique and how the problem is encoded and formulated in the GA form.

P. Zhou et al. proposed an optimization approach using dynamic programming (DP) [17]. At first, the environment is modeled and divided into sub-regions using boustrophedon decomposition. Then, define a matrix of the connected regions. Moreover, then, use dynamic programming to optimize this matrix and find the shortest path. However, this method requires full prior knowledge about the environment.

T. R. Schafle et. al. divided the environment into small squared cells which diagonal is the size of the robot tool, four simple motions are used (i.e. straight, right turn, left turn and U-turn) [18]. Genetic algorithm (GA) is used to find the near optimal coverage path in terms of minimizing energy consumption by minimizing turns. The coverage path is represented as a chromosome of a set of consecutive simple motions. But, they didn't consider number of revisits and the environment must be known a priori to the robot.

M. A. Yakoubi and M. T. Laskri introduced a novel approach to solve the CPP problem in unknown environments for a vacuum cleaning robot (PPCR) [19]. They followed a sensor-based approach and used the GA to optimize the coverage path. They used the mini-path idea to simplify the problem, and used the GA to optimize that mini-path. The GA's chromosomes contained cells order as genes. However, they only considered number of revisits, total path length and number of turns in their approach. Also, the algorithm responds far from optimal in complex environments.

III. Notation, Assumptions, and Problem Formulation

A. Preliminaries

Let $\mathcal{Q}$ denote an environment, where $\mathcal{Q} \in \mathbb{R}^d$ and d holds the problem dimension. The robot's position is described by $\mathcal{P}$; which defines the robot's current configuration regarding the current present cell and orientation. $\mathcal{M}$ and $\mathcal{V}$ presents obstacles cells and visited cells respectively, such that $\mathcal{M}$ and $\mathcal{V} \in \mathbb{R}^d$. On-board sensor diameter range is defined by $\mathcal{R}$ while describing the footprint by $\mathcal{F}$ and the allowed directions $\mathcal{D}$, where $\mathcal{D} \in \{4, 8\}$.

B. Assumptions

First, we assume that for a given environment $\mathcal{Q}$ is discretized into a set of cells q. Generated cells must have a square size. Additionally, the environment is obstacle-free as an initial condition. For each sub-cell division, size is equal to the robot's footprint.

C. Problem Formulation - Coverage Path Planning

Given 5-tuple $\langle \mathcal{Q}, \mathcal{M}, \mathcal{V}, \mathcal{P}_{initial}, \mathcal{R} \rangle$, find a complete coverage path $\mathcal{S}$ represented as a set of robot's positions, $\mathcal{S} = \{\mathcal{P}_{initial}, \mathcal{P}_1, \ldots\}$ that fully covers the given environment $\mathcal{Q}$.

IV. Proposed Approach

The proposed approach divides the CPP problem into sub-problems using DP techniques, then performs a Multi-objective GA on each sub-problem individually to find its optimal solution. Algorithm.1 describes the main flow. After it receives an environment $\mathcal{Q}$ and starts position $\mathcal{P}_{initial}$, coverage path is populated with start position and both obstacles and visited cells are cleared (lines:1-2). Afterward, it performs the divide-and-conquer policy on free cells, by dividing the present cell into small cells equal to the robot's footprint using DP techniques; and using MOGA techniques for CPP solution generation (lines:4-6). Then visited and obstacles cells sets are updated (lines:5-8). Finally, the robot executes the sub-path to cover it from its current position and updates the return solution (lines:7-9). The robot keeps operating until the defined termination condition occurs. At last, algorithm.1 returns a solution if environment $\mathcal{Q}$ is covered. Otherwise, it returns $failure$. For a visual explanation, fig.1 shows the life-cycle of the proposed approach. Fig.1(b) embraces the set of CPP sub-problems, the blue box denotes the robot's start position. For each cell the robot scans its surrounding using on-board sensors (fig.1(c)), after scanning it solves the CPP sub-problem by generating a sub-path (fig.1(d)) and executes this solution (figs.1(e)-1(f)). Afterwards the robot starts solving a new CPP sub-problem until termination condition occurs (fig.1(g)).

Algorithm 1 Augmented-CPP

Input: $\mathcal{Q}, \mathcal{P}_{initial}$
Output: $coverage - path$ or $failure$
1: $\mathcal{S} \leftarrow \{\mathcal{P}_{initial}\}$
2: $\mathcal{M} \leftarrow \emptyset; \mathcal{V} \leftarrow \emptyset$
3: **while** $\neg$ **termination condition do**
4: $\quad \mathcal{M}_{surrounding} \leftarrow ScanSurroundings(\mathcal{P}, \mathcal{M}, \mathcal{R}, \mathcal{D})$
5: $\quad \mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}_{surrounding}$
6: $\quad \mathcal{S}_{sub} \leftarrow MOGA(\mathcal{P}, \mathcal{M}, \mathcal{V})$
7: $\quad \mathcal{P}, \mathcal{V}_{updated} \leftarrow ExecutePath(\mathcal{S}_{sub})$
8: $\quad \mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{V}_{updated}$
9: $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{sub}$
10: **end while**
11: **if** $|\mathcal{M}| + |\mathcal{V}| = |\mathcal{Q}|$ **then**
12: $\quad$ **return** $\mathcal{S}$
13: **end if**
14: **return** $failure$

A. Multi-Objective Optimization GA (MOGA)

Multi-objective genetic algorithm extends the traditional genetic algorithms through satisfying more than one attribute. Each attribute is weighted based on it's priority. We present the gene as a tuple of position and orientation ($\mathcal{P} = \langle \mathbb{R}^d, \theta \rangle$) and the genes number per chromosome depends on $\mathcal{R}$'s sub-cell divisions. Algorithm.2 shows how MOGA followed steps for finding an optimal solution. First, MOGA generates a population of initial coverage paths for the surrounding area; these paths are create using traditional approaches like zigzag and fixed direction algorithms (lines:2-9). The passed parameters for these algorithms are to indicate the coverage directions. Afterwards, MOGA locates the best fitting chromosome (lines:11-16) and performs a traditional crossover for it and creates a new population by cross-overing the current population (lines:10-18). Finally, it finds the best fitting coverage path and returns it (lines:19-25)

1) Fitness Function: The multi-objective fitness function (evaluation function) determines how efficient the sub-path is. It is a summation of a set of sub-path attributes. Each attribute evaluates part of the sub-paths efficiency and has a weight value representing its weight (importance) and priority among other attributes. The proposed approach presents a ten attributes fitness function. Together, these attributes with their weights ensure satisfying the given CPP KPIs and reaching an optimally efficient complete coverage path. The fitness function is calculated as follows:

$$\begin{aligned} F(i) = {} & w_1 * totalDist(i) + w_2 * freeCells(i) \\ & + w_3 * relativeDist_x(i) + w_4 * turnsNum(i) \\ & + w_5 * visitedCellsAroundPath(i) \\ & + w_6 * revisitsNum(i) \\ & + w_7 * freeCellsAroundLastCell(i) \\ & + w_8 * obstacleCellsAroundPath(i) \\ & + w_9 * trappedCellsAroundPath(i) \\ & + w_{10} * isPathSplitsArea(i) \\ & + w_{11} * isPathClosesArea(i) \end{aligned} \tag{1}$$

Where $w_1, w_2, \ldots, w_{11}$ are constant values representing weight of the targeted attribute; while each attribute is defined as follows:

a) $totalDist(i)$*:* minimizes number and angles of turns. a sub-path with less total length is better as it helps in reducing time and energy required by the robot to follow the path.

b) $freeCells(i)$*:* defines the number of free non-obstacle cells in the sub-path, free-cells are non-obstacle unvisited cells (i.e. with zero number of visits). As it increases, the sub-path efficiency rises, as the robot will pass through new unvisited-cells.

c) $relativeDist_x(i)$*:* indicates the total relative-distance from the start-cell's position to each cell in the sub-path on the x-axis, a better sub-path is the one with less total relative-distance, as it makes sure that the robot finishes parts of the area on the left before going further to the right (i.e. it ensures left-gravitation).

d) $turnsNum(i)$*:* defines number of turns in the sub-path, as it increases the energy consumed and required time rises which negatively affects the performance.

e) $visitedCellsAroundPAth(i)$*:* embraces the total number of previously visited-cells around the sub-path, as the number of already visited cells around the sub-path increases, it means that this sub-path completes the total-path for covering the environment perfectly as it aligns to other parts of the total-path. A sub-path with more cells around it is already-visited is a better one.

f) $revisitsNum(i)$*:* shows the total number of visits for cells in the sub-path, some cells in the sub-path may be previously visited, once or more than one time, if the robot has to pass through previously visited-cells to get out of a trapped position it is better to revisit the cells that are less previously visited. A better path is the one with less total number of revisits to all its cells.

g) $freeCellsAroundLastCell(i)$*:* counts the number of unvisited free-cells around the last-cell in the sub-path, this makes sure that the robot will reach the last cell in the sub-path and still has unvisited free-cells to go though, and not trapped at the end of the sub-path in a situation with no exit to complete the coverage. A sub-path with no unvisited free-cells around the last-cell is not preferable to be followed.

h) $obstacleCellsAroundPath(i)$*:* keeps track of the number of obstacle-cells around the sub-path, as the number of obstacle-cells around the sub-path increases it becomes more likely to follow the wall and not get stuck in a closed area. A slightly better sub-path is the one with higher number of obstacle-cells around it.

i) $trappedCellsAroundPath(i)$*:* computes the number of trapped-cells around the sub-path, a cell is trapped if the robot cannot reach it, this mainly depends on number and direction of unvisited free-cells around that cell, a cell can be fully or partially trapped. A sub-path that results in making cells around it trapped is not preferable at all.

j) $isPathSplitsArea(i)$*:* checks if the sub-path splits the area. If the sub-path results in splitting the area into islands with unvisited free-cells, which can ruin the coverage path totally as the robot chooses one side to cover and the other will be fully isolated from the part being covered by that sub-path as previously visited cells. A sub-path that results in splitting

the area under coverage into two islands must not be chosen at all.

k) $isPathClosesArea(i)$*:* checks if the sub-path finish covering some visible area on the map. If it does so it is preferable to finish some area and exit after covering completely.

Algorithm 2 MOGA

Input: $\mathcal{P}, \mathcal{M}, \mathcal{V}$
Output: $coverage - sub - path$

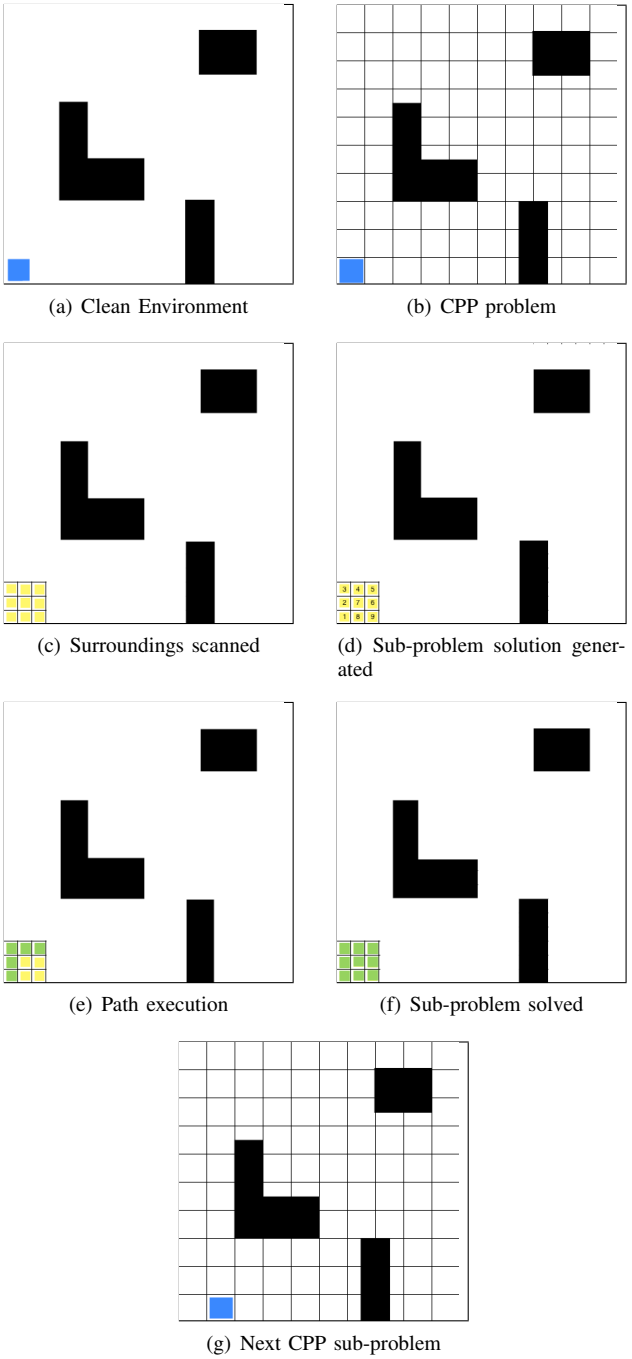
$pop \leftarrow \emptyset$
while $i = 1, \ldots, N$ **do**
  $pop \leftarrow \{Zigzag(V, R)\} \cup \{Zigzag(V, L)\}$
  $pop \leftarrow pop \cup \{Zigzag(H, D)\} \cup \{Zigzag(H, U)\}$
  $pop \leftarrow pop \cup \{Spiral(R, N = 8)\}$
  $pop \leftarrow pop \cup \{Spiral(L, N = 8)\}$
  $pop \leftarrow pop \cup \{Spiral(R, N = 4)\}$
  $pop \leftarrow pop \cup \{Spiral(L, N = 4)\}$
end while
while $i = 1, \ldots, N$ **do**
  $best - chromosome \leftarrow pop(1)$
  for each chromosome **in** pop **do**
    if $F(chromosome) > F(best - chromosome)$ **then**
      $best - chromosome \leftarrow chromosome$
    end if
  end for
  $pop \leftarrow pop \cup CrossOver(best - chromosome)$
end while
$coverage - sub - path \leftarrow POP(1)$
for each chromosome **in** population **do**
  if $F(chromosome) > F(coverage - sub - path)$ **then**
    $coverage - sub - path \leftarrow chromosome$
  end if
end for
return $coverage - sub - path$

B. Runtime Analysis

Since both $ScanSurroundings$ and $ExecutePath$ are implementation-dependent based on the used robot, therefore, their complexities are assumed to be $O(1)$. For analysis simplification we assume the termination condition for $Augmented - CPP$ is bounded by executing n iterations and $MOGA$ is bounded by population size m. Therefore, the overall time complexity for $Augmented - CPP$ is bounded by $O(n \times m^2)$ and space complexity if bounded by $O(|\mathcal{M}| + |\mathcal{V}| + |\mathcal{S}|)$.

TABLE I. MEASUREMENTS RESULT

Map - Approach	Completeness ratio (%)	path length	#turns	#revisits
a - Augmented-CPP	100	349.4	92	0
a - PPRC	100	351.46	62	0
b - Augmented-CPP	100	598	100	0
b - PPRC	100	600.9	103	0
c - Augmented-CPP	100	1123.24	326	29
c - PPRC	9.5	-	∞	∞
d - Augmented-CPP	100	223	71	0
d - PPRC	91.9	-	∞	∞



(a) Clean Environment (b) CPP problem (c) Surroundings scanned (d) Sub-problem solution generated (e) Path execution (f) Sub-problem solved (g) Next CPP sub-problem

Fig. 1. Augmented-CPP life cycle: Blue color denotes the robot's position, the yellow color shows a scanned area, while the green color marks the area as covered area.

V. EXPERIMENTS

The proposed algorithm is verified and tested using MATLAB R2016b with a holonomic robot model in $\mathbb{R}^2$. Multiple environments with multiple obstacles of different shapes are used to prove the efficiency of the proposed algorithm which is displayed in fig.2(a)-2(c). Regarding both the environment configurations and robot's used parameters are listed in table II and III respectively. Regarding the fitness function weights are selected empirically with values $\langle 40, -50, 7, 0.5, 80, -1, -0.5, 20, 100, -1 \rangle$. We are bench-

marking our results based on four parameters which are: completeness ratio, obstacle avoidance, time to complete, and finally, energy consumption. Completeness ratio is defined by how many cells are covered concerning a total number of free cells. Obstacle avoidance is a requirement more than a benchmarking KPI; since it indicates that the algorithm has avoided non-free cells in it is execution. Time to complete is measured in a number of revisited cells, not in seconds. Energy consumption is measured regarding path length and the number of revisited cells. Also, we are benchmarking the PPCR algorithm using the same maps and benchmarks KPIs. CPP problem solution paths are visualized in fig.3 while the measurements result are presented in table I.

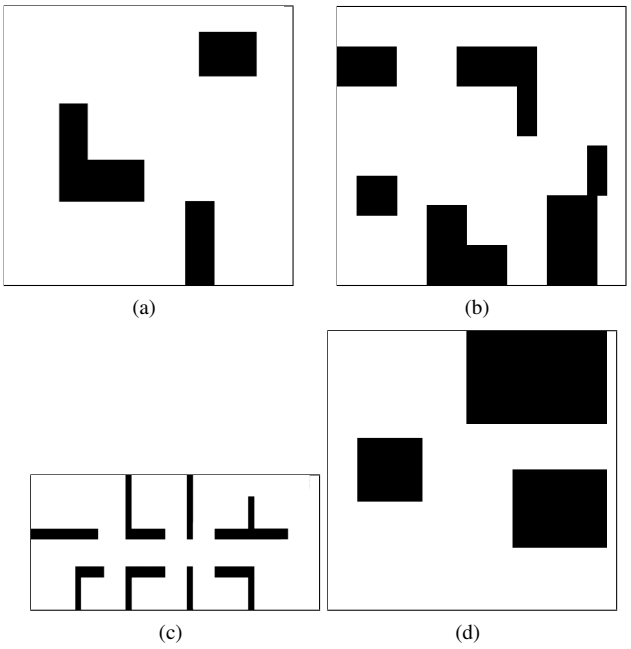


Fig. 2. Benchmarking environments

TABLE II. ENVIRONMENT CONFIGURATION PARAMETERS

Environment	Dimensions ($dim_1 \times dim_2$)
a	(28×28)
b	(20×20)
c	(50×25)
d	(26×31)

TABLE III. ROBOT CONFIGURATION PARAMETERS

Parameter	Value
$\mathcal{R}$	4
$\mathcal{D}$	4
$\mathcal{F}$	1

VI. DISCUSSIONS

Table I provides comparing results of the proposed approach with those of the PPCR algorithm on the same environments they used (fig.3(a)-3(d)) which shows very close performance. While when running on more complicated environments (fig.3(e)-3(h)) with more complex obstacles structure shows great improvements using the proposed approach over the PPCR algorithm.

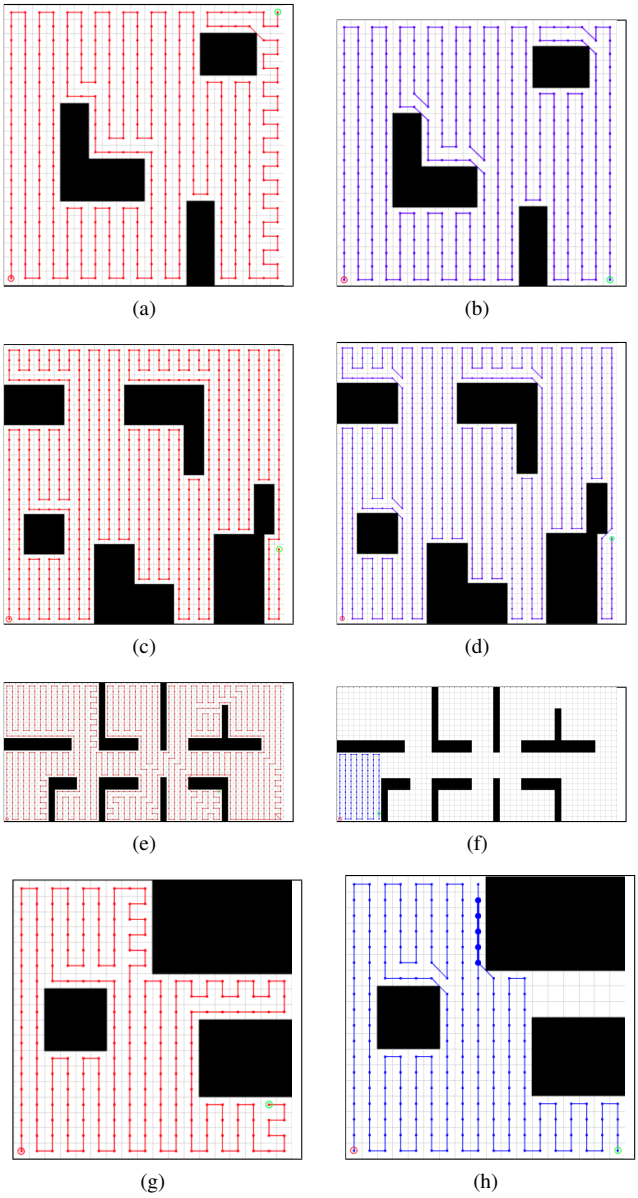


Fig. 3. Experiments Results: the left column embraces our proposed approach while the right column hold the PPCR algorithm results

In simple structured environments (fig.2(a), 2(b)) the results are very close (fig.3(a)-3(d)), this is due to the environment and obstacle structure. The use of $totalDist$, $freeCells$ and $relativDist_x$, as the evaluation function's parameters, provides very well performance in such simple obstacle structure, where the environmental dimensions in terms of cells number at x-axis are of even values and make dimensions of all obstacles and squared free spaces are of even length in terms of cells number, and where the obstacle structure does not divide the environment into separate closed areas with few cells for its entry.

However, this is not always applicable in real environments, especially when the environment is unknown to the robot, and it discovers it on the go, where the robot has no idea where the obstacles will be or how it will look like.

The proposed approach shows a much better performance

compared with PPCR in more realistic and complicated environments (fig.2(c), 2(d)) show the much better performance of the proposed algorithm (fig.3(e)-3(h)). Because of the added parameters to the evaluation function which make the robot make sure that it will not get stuck in a closed space in the environment and still finds a near-optimal coverage path to adequately cover the given area.

Parameters used in the proposed approach provides a proper evaluation for each sub-path's quality with the target to find the near-optimum total coverage path that satisfies the main CPP key performance indicator factors, by increasing the completeness ratio, while avoiding obstacles and decreasing the energy consumed and the time to complete. Each parameter has its weight defining its positive or negative impact on the total cost considering decreasing that cost to find better paths as explained in table IV.

TABLE IV. IMPACT OF EACH PARAMETER'S WEIGHT IN THE FITNESS FUNCTION ON THE TARGETED KPIS

Target KPI	Weights of parameters and their impact
Completeness ratio	setting positive weights for: $relativeDist_x$, $trappedCellsAroundPath$, $isPathSplitsArea$ and negative weights for: $freeCells$, $visitedCellsAroundPath$, $freeCellsAroundLastCell$, $obstacleCellsAroundPath$ with appropriate values ensures that the robot does not get stuck at any space or leave unvisited cells in the environment till finishing.
Obstacle avoidance	by generating initial sub-paths never passing in any obstacle cell thanks to sensors scanned areas visible to the robot.
Energy consumption and time to complete	setting positive weights for: $totalDist$, $relativeDist_x$, $revisitsNum$, $turnsNum$ and negative weights for: $freeCells$ with appropriate values provides shorter paths and decrease the need for the robot to stop and move again by decreasing number and angle of turns. This leads to minimizing the amount of energy consumed and the time required.

VII. CONCLUSION

In this paper, a new approach for solving the CPP problem in unknown environments is proposed. In which, Dynamic Programming (DP) was used to divide the complete problem of finding a complete coverage path into a set of sub-problems of finding coverage sub-paths, which together, the coverage sub-paths, construct the complete coverage path. That the planning decision is not made only one time, but multiple times and it has multiple decision steps. At each decision step, the robot sensed the surrounding environment with limited range on board sensors and used Multi-Objective Genetic Algorithm (MOGA) to find the best coverage sub-path that covers that area. This keeps repeating until gradually the environment gets fully covered and served. MOGA enabled us to find the near optimal coverage paths that satisfied multiple CPP criteria, such as complete coverage, obstacle avoidance and minimum time and energy consumption.

REFERENCES

[1] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1258–1276, Dec. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2013.09.004

[2] F. Yasutomi, M. Yamada, and K. Tsukamoto, "Cleaning robot control," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, Apr 1988, pp. 1839–1841 vol.3.

[3] M. Ollis and A. Stentz, "Vision-based perception for an automated harvester," in *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 3, Sep 1997, pp. 1838–1844 vol.3.

[4] B. Englot and F. Hover, "Sampling-based coverage path planning for inspection of complex structures," 2012. [Online]. Available: https://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/view/4728

[5] A. Khan, I. Noreen, and Z. Habib, "On complete coverage path planning algorithms for non-holonomic mobile robots: Survey and challenges," *J. Inf. Sci. Eng.*, vol. 33, pp. 101–121, 2017.

[6] H. Choset, "Coverage for robotics – a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, pp. 113–126, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1016639210559

[7] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[8] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and Service Robotics*, A. Zelinsky, Ed. London: Springer London, 1998, pp. 203–209.

[9] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull, "Morse decompositions for coverage tasks," *The International Journal of Robotics Research*, vol. 21, no. 4, pp. 331–344, apr 2002.

[10] 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, 2002, pp. 954–960 vol.1.

[11] M. Waanders, "Coverage path planning for mobile cleaning robots," *15th Twente Student Conference on IT, Enschede*, 2011.

[12] E. Gonzalez, M. Alarcon, P. Aristizabal, and C. Parra, "Bsa: a coverage algorithm," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 2, Oct 2003, pp. 1679–1684 vol.2.

[13] 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*, April 2005, pp. 2040–2044.

[14] S. Bochkarev and S. L. Smith, "On minimizing turns in robot coverage path planning," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug 2016, pp. 1237–1242.

[15] I. A. Hameed, D. Bochtis, and C. A. Srensen, "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. [Online]. Available: https://doi.org/10.5772/56248

[16] P. A. Jimenez, B. Shirinzadeh, A. Nicholson, and G. Alici, "Optimal area covering using genetic algorithms," in *2007 IEEE/ASME international conference on advanced intelligent mechatronics*, Sept 2007, pp. 1–5.

[17] P. Zhou, Z.-m. Wang, Z.-n. Li, and Y. Li, "Complete coverage path planning of mobile robot based on dynamic programming algorithm," 09 2012.

[18] T. R. Schfle, S. Mohamed, N. Uchiyama, and O. Sawodny, "Coverage path planning for mobile robots using genetic algorithm with energy optimization," in *2016 International Electronics Symposium (IES)*, Sept 2016, pp. 99–104.

[19] M. A. Yakoubi and M. T. Laskri, "The path planning of cleaner robot for coverage region using genetic algorithms," *J. Innovation in Digital Ecosystems*, vol. 3, pp. 37–43, 2016.