IADIS International Journal on Computer Science and Information Systems
Vol. 14, No. 2, pp. 95-114
ISSN: 1646-3692

SOLUTION OF AN INTEGRATED TRAVELING SALESMAN AND COVERAGE PATH PLANNING PROBLEM BY USING A GENETIC ALGORITHM WITH MODIFIED OPERATORS

Wen-Chieh Tung and Jing-Sin Liu
Institute of Information Science, Academia Sinica, Nankang, Taipei, Taiwan

ABSTRACT

Coverage path planning (CPP) is a fundamental task that is conducted in many applications for machining, cleaning, mine sweeping, lawn mowing, and performing missions by using unmanned aerial vehicles such as mapping, surveillance, search and rescue, and air-quality monitoring. An approach for conducting CPP for a known environment with obstacles involves decomposing the environment into cells such that each cell can be covered individually. The visiting order of the cells can then be decided to connect those intracell paths together. Finding the shortest intercell path that visits every cell and returns to the origin cell is similar to the traveling salesman problem (TSP). However, an additional variation from TSP that should be considered is that there are multiple intracell paths for each cell. These paths result from different selections of entry and exit points in each cell and thus affect the intercell path. This integrated TSP and CPP problem is known as TSP-CPP and is similar to the TSP with neighborhoods (TSPN). To solve TSP-CPP, one must simultaneously determine the visiting order of sites with minimal repetition and the transition points of each visiting site. The current approaches for solving TSP-CPP are as follows: (i) adapting dynamic programming (DP) for TSP to TSP-CPP, which is excellent for obtaining the optimal route and (ii) determining the optimal route by conducting a brute force enumerative search on entry and exit point combinations for every cell and then solving each combination of entry and exit points with a TSP solver. For large numbers of cells, approaches (i) and (ii) both suffer from exponential complexity and are impractical for complex environments. In this study, we proposed an appropriate genetic algorithm implementation for TSP-CPP to achieve an optimal balance between time efficiency and path optimality to eliminate the curse of dimensionality in DP. Our approach is demonstrated to find the true optimal solution as DP in all simulation environments that can be solved by both DP and GA, and GA is one hundred times faster than DP approach for maps decomposed with large cell number.

KEYWORDS

Coverage Path Planning, Traveling Salesman Problem (TSP), TSP-CPP, Genetic Algorithm, Dynamic Programming, Modified Operators

1. INTRODUCTION

Coverage path planning (CPP) is a fundamental task of mobility used for a given or unknown cluttered environment of interest that must be covered using one or more mobile vehicles (Choset, 2001; Galceran and Carreras, 2013; Dasgupta, 2015; Das *et al.*, 2016). This task is necessary in various real-world civilian, military, and commercial sensing applications that use autonomous or manned ground vehicles, surface or underwater vehicles, and unmanned aerial vehicles (UAVs). The broad applications of CPP, such as machining, cleaning, mine sweeping or humanitarian demining, lawn mowing, farming, search and rescue or disaster management, mapping tasks, surveillance, inspection (e.g., forest fire or aging infrastructure such as bridges), and monitoring (e.g., air quality or climate monitoring by using UAVs with onboard sensory and navigation systems), are emerging (Oksanen and Visala, 2009; Mansouri *et al*., 2017; Otto *et al.*, 2018). Many CPP problems have been studied for different application scenarios (Khan *et al*., 2017). A commonly encountered problem of CPP pertains to the completion of CPP, that is, fully covering the portion of the two-dimensional environment with known static obstacles and returning to the starting position with or without repeatedly covering the same regions that have already been covered. An efficient approach for conducting CPP is to first decompose the polygonal environment into cells such that the entire region is covered and the resulting structure is simple. Then, each cell can be covered individually by a simple intracell path, and the intracell paths of each cell are connected to each other via intercell path. Decomposition methods such as the boustrophedon method (Choset and Pignon, 1998), multiple sequence alignment decomposition (Huang, 2001), trapezoidal method (Jimenez *et al.*, 2007; Oksanen and Visala, 2009), and Morse decomposition (Galceran and Carreras, 2013) were used.

The shape of the map and obstacles influence the direction of sweeping to completely cover the entire area. For example, a rectangular map can be covered by following a sweeping direction along the longest side that is perpendicular to the width of the rectilinear area. The coverage path is not unique. Thus, in addition to the decomposition methods, some studies have attempted to find the optimal sweeping direction for an intracell path in terms of conserving energy consumption or have proposed shorter paths with few turns in which the vehicle switches its moving direction by selecting from multiple sweeping directions (Huang, 2001). The A* search algorithm with an admissible heuristic design was applied to conduct an enumerative search in a grid environment to find a coverage path with few turns (Dogru and Marques, 2017) or with high coverage and low repetition (Cai *et al.*, 2014). The intercell path length can be optimized by deciding the visiting order of cells that visits every cell exactly once and returns to the origin cell with the lowest possible sum of the Euclidean distances along the path. This is the traveling salesman problem (TSP). Specifying an a priori sweeping pattern is practically useful for an intracell path that covers a cell completely. In practice, a commonly used sweeping pattern is the zig-zag path (parallel lines) or back-and-forth motion in certain orientations (or more generally oriented line sweeping; Huang, 2001) with a predictable vehicle behavior. This integrated TSP and CPP problem is known as the TSP-CPP and was coined by Xie *et al.* (2019). In this problem, both TSP and CPP are NP-hard. The additional variation to consider in TSP-CPP is that multiple options intracell paths exist for each cell that result from different selections of the entry and exit points. These paths influence the length of the intercell path during cell-to-cell transitions resulting from the visiting order of cells by providing a solution to TSP. To consider the visiting order and the

selection of entry and exit points of each cell, TSP-CPP comprises two asymmetric subproblems: determining the visiting order (TSP, an NP-hard problem) and determining the exit and entry point pairs for incoming or outgoing intercell paths in each cell, which also represent the starting and ending points of the intracell path (an optimization problem). TSP-CPP has an exponentially larger search space than TSP; the selection of entry and exit points of each cell is challenging. The two problems are coupled; that is, the solution of one subproblem requires the solution of the other subproblem. These problems can be solved separately, but in full generality they should be solved simultaneously. Recent approaches for solving TSP-CPP are based on the solutions of TSP. Thus, dynamic programming (DP) for TSP can be adapted to TSP-CPP to find the optimal solution (Xie *et al.*, 2019), or a two-step procedure from Khsheibun *et al.* (2018) can be conducted. In this procedure, first a brute force enumerative search is performed on all the possible combinations of the entry and exit points of every cell. Then, each combination of entry and exit points is solved using a TSP solver. Both methods suffer from exponential complexity and thus are prohibitive for complex environments with large numbers of cells; this computational burden remains a major obstacle to broad applications of DP in practice.

The TSP with neighborhoods (TSPN) is a variant formulation of TSP; TSPN is relevant to data collection in a wireless sensor network by using data mule (Wu and Liu, 2014), task sequencing relevant to industrial and service robotics (Alatartsev et al., 2015) such as that motivated by robot remote laser welding (Kovács, 2016), CPP (Yu, 2015), and other problems (such as Otto *et al.*, 2018). In TSPN, the shortest tour that visits the given sites must be found. A visited site is no longer counted as a point but a spatial region, known as a neighborhood. That is, not only the order of visiting (or task sequencing) sites is set but also the entry and exit points of a tour within a site (at the most two points) are determined simultaneously for TSPN. In most formulations, the entry point can be the same as the exit point. In a study by Yu (2015), a simplified TSPN related to CPP for a workspace without obstacles was solved by first specifying a single via point (the center) and sweeping pattern for each cell (a disk) to reduce TSPN into TSP. Then, the resulting TSP is solved for all cells to determine the optimal visiting order. For the optimal visiting order, the via point is modified to further shorten the tour. In general, two subproblems are involved in TSPN: (1) determination of the visiting order of neighborhoods and (2) determination of precise via points of nighborhoods to enter and exit. The latter subproblem in full generality is a continuous optimization problem over theneighborhood, because the via points can be located at any point in the neighborhood and the neighborhood can have an arbitrary shape. Therefore, TSP-CPP is similar to TSPN in terms of solving the problem of interest with different underlying meanings. Because both methods are variants of TSP, they are NP-hard problems. Moreover, their solutions require a solution to TSP with additional complexity of searching via points (one via point for TSPN and two via points for TSP-CPP with a prespecified sweeping pattern that is selected from a finite selected set of sweeping patterns for each cell) for each cell to connect the intercell paths together for enabling cell-to-cell transition.

Metaheuristics are proved to be practically useful for efficiently computing approximate solutions to NP-hard problems. In general, metaheuristics quickly provide an approximate solution but are susceptible to getting trapped in local optima. Genetic algorithms (GAs) constitute a class of metaheuristics used for stochastic search and derivative-free optimization and belong to a larger family of population-based evolutionary algorithms inspired by nature and biology. A practical advantage of GA-based approaches is that they can find multiple suitable solutions when limited time and memory are available even when the data is large.

GA-based approaches have been developed as effective approaches to TSP (Potvin, 1996; Scholz, 2019) and TSPN (Wu and Liu, 2014). GAs have already been considered for a problem similar to TSP-CPP in a study by Jimenez *et al.* (2007). Their GA only applied a simple single-point crossover and swap mutation, and no performance evaluation was presented. In this paper, we propose a genetic algorithm approach for TSP-CPP. In the context of DP, the contributions of this paper are as follows: 1. Even though GA metaheuristics do not guarantee optimal solutions in general, our GA approach with modified operators can find a solution identical to the optimal solution of DP for TSP-CPP in all simulation environments that can be solved by both approaches. For any problem with a large number of cells, the time and space requirements of DP increase exponentially (the well-known curse of dimensionality or search space explosion). Our approach has the potential to successfully find one solution and is more than one hundred times faster than a DP approach for large problems that suffer from the curse of dimensionality. 2.We provide a complete workflow and implemented code to transform our original CPP problem to the TSP-CPP problem; the DP and proposed GA code for TSP-CPP are available at (https://github.com/WJTung/GA-TSPCPP). In addition, we believe GA can also be used for a more difficult TSP-CPP, such as for handling multiple conflicting objectives and change in workspace, such as obstacles motion and extension with constrained vehicle motion.

The paper is organized as follows. In section 2, we describe how to decompose the environment into cells. In section 3, we display different choices for entry and exit points of each cell and how they affect the inter-cell path. In section 4, the proposed GA approach is presented along with a brief introduction to DP. Time and space complexity of GA is given. In section 5, performance of GA and DP in six simulation environments is compared to present the advantages of GA. Section 6 presents the conclusion.

2. SPACE DECOMPOSITION AND COVERAGE PATH OF A SINGLE CELL

The workspace or target area to be covered is a closed and bounded region W in R^2. W is cluttered with a finite number of static disjoint polygonal obstacles $\{P_i\}$ and is a priori known. For simplicity, the vehicle or the robot V is modeled as a point with a constant sensing range that can translate in any direction so that a piecewise linear path or a 90^0 turn is executable. In this paper, we assume the robot's task is to completely cover the environment W with no collisions with the obstacles$\{P_i\}$. One common approach to coverage of an area of interest W is the decomposition of W into a collection of connected, possibly irregular cells served as the independent target regions. There still remains a small area in W that cannot be reached by the robot due to motion constraints. We let $\overline{W}_{free} \approx W_{free} := W \backslash \{P_i\}$ denoting the free space that can be fully covered by the robot motion, i.e. any point in $\overline{W}_{free}$ can be covered. We apply the boustrophedon cellular decomposition to break the robot's achievable free space $\overline{W}_{free}$ down into cells, whose union is the complete covering region $\overline{W}_{free}$. To implement the boustrophedon cellular decomposition, we use a vertical slice to sweep through the environment, and open and close cells according to the connectivity change of the slice. For cell decomposition of workspace, the motion is restricted from a current cell to its neighboring cells that can be directly reached from the current cell. Assume for simplicity that coverage path pattern is back-and-forth motion in accordance with boustrophedon cellular

decomposition (Choset and Pignon, 1998) so that only one path enters each cell and only one path exits from each cell. The distance between the boustrophedon path and the boundary of cell must be smaller than the sensing radius, and the distance between two adjacent vertical lines in the boustrophedon path must be smaller than twice of the sensing radius. Figure 1 demonstrates the result of the boustrophedon cellular decomposition for one example environment and how each cell can be covered with back-and-forth path pattern in accordance with boustrophedon decomposition of polygonal environment.

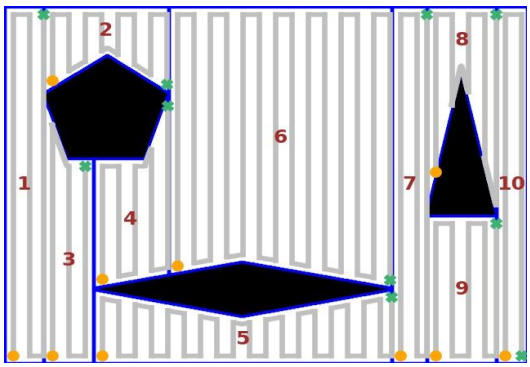


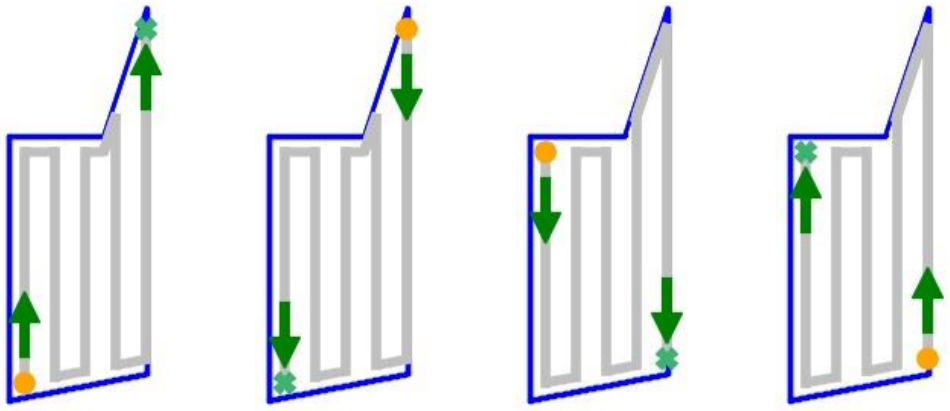
Figure 1. The free space is decomposed into 10 cells (each labelled with an id) by the boustrophedon cellular decomposition. Each cell can be completely covered with a sweeping back-and-forth boustrophedon path from an entry point to an exit point so that the environment can be completely covered

3. DETERMINE THE ENTRY AND EXIT POINTS FOR EACH CELL AND THE VISITING ORDER OF CELLS

After the cluttered environment is decomposed into *N* cells by boustrophedon cellular decomposition, each cell is labeled a cell ID (identification number). The environment to be covered is then represented by a set of *N* cell IDs. By a priori specifying the aforementioned back-and-forth sweeping motion pattern as the intra-path coverage for each cell, the next step is to determine the visiting order to connect intra-cell paths to completely cover every individual cell. We assume that we can only connect the ending point of an intra-cell path to the starting point of the next intra-cell path in some adjacent cell; the next path also ends at another ending point. Therefore, it is assumed that the starting point and ending point of the boustrophedon path in a given cell coincide with the entry point and exit point of the cell-to-cell transition for incoming and outgoing inter-cell paths of the given cell, respectively, that connect together the previous and next cells. There are four alternative pairs of entry and exit points for intra-cell paths of lawn mowing pattern for a cell with opposite traversal direction, which we call cell-path combinations. If the entry and exit points of each cell are fixed, finding the coverage path with the shortest length is actually solving the TSP, i.e. a shortest path search problem in the partitioned cells.. The TSP is a classical NP-hard problem in which one must find the shortest possible route (in the metric of Euclidean distance). This route visits each city and returns to the initial departure city. To find the shortest path between

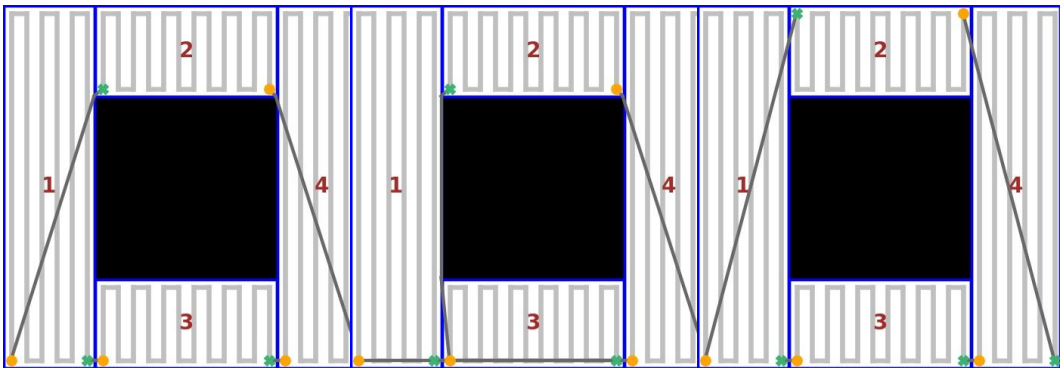
two points in terms of the cell distance when the forbidden regions occupied by the nonoverlapping obstacles are polygonal regions (bounded by straight line segments), we can construct a visibility graph using the vertices of obstacles with the start and goal points used as additional vertices. Then, the edges of the visibility graph can be marked with the corresponding Euclidean lengths and Dijkstra's algorithm can be used to find the shortest path from the start point (Alt and Welzl, 1988).

However, for the CPP problem within each cell, there exists some flexibility in selecting the entry and exit points for the boustrophedon path. The path can be entered or exited from either the top left, top right, bottom left, or bottom right, thus yielding the four candidate coverage paths for each cell with an opposite traversal direction. Figure 2 presents an illustration. In addition to the change in the intracell path length (difference between intracell path lengths of (a) = (b) and (c) = (d), which depends on the cell shape, is usually small for polygonal cell), the determination of the entry and exit points for each cell greatly affects the total inter-cell path length. Figure 3 presents an example that illustrates that the determination of the visiting order of cells and the entry and exit points in each cell should be conducted simultaneously for TSP-CPP. Figure 3(a) is the optimal solution for the simple four path-cell environment. Either a different visiting order that resembles (b) or a different selection of entry and exit points for any cell that resembles (c) can cause an increase in the intercell path length. In the following section, two distinct algorithms, DP and GA, are presented for solving TSP-CPP. These algorithms simultaneously find the entry and exit points for each intracell path within a cell and the visiting order of cells to incur the minimum total sum of lengths over the intracell coverage paths and the intercell transition paths covering the entire workspace.



(a) (b) (c) (d)

Figure 2. Each cell has the same number of (4) cell-path combinations. The coverage intra-cell path for each cell depends on the choice of a pair of entry point and exit point (as its starting and ending points) shown in (a), (b), (c), (d). The length of the intra-cell path corresponding to each choice of cell-path combination is the same. The entry point is marked by the orange circle, and the exit point is marked by the sea green cross. The direction of the lawn mowing path is indicated by the green arrow



(a) (b) (c)

Figure 3. We can see the influence of the visiting order of cells and entry and exit points of each cell on the intercell path length through (a), (b), and (c). (a) Optimal visiting order of 1, 2, 4, and 3 and solution of the entry and exit points for this environment with the shortest intercell path length of 802.78. (b) Entry and exit points of each cell are the same as (a), but the visiting order is 1, 3, 2, and 4, which results in a total intercell path length of 1347.40 (67.8% larger than that in (a)). (c) Visiting order of cells is the same as that in (a), but the intracell path of 1 and 4 starts from the top left instead of the bottom right, which results in a total inter-path length of 1153.72 (43.7% larger than that in (a))

4. SOLUTIONS TO TSP-CPP

4.1 Problem Analysis

In the following text, we denote the cell number as *N*. Moreover, to make the explanation more concise, we denote a specific cell based on the selection of its entry and exit points (intracell path) that specifies a cell-path combination. Each cell has four possible choices of entry and exit points; four combinations are possible. In total, 4*N* cell-path combinations exist for all cells. Suppose that the sum of lengths over intracell paths and over intercell paths are L_{intra} and L_{inter}, respectively. Let the length of a TSP subtour of a TSP-CPP tour T for a given visiting order of the set {cell-path combinations for all cells} with cardinality 4*N* be $L_{TSP}(T)$. The actual length for a given candidate TSP-CPP solution is as follows:

$$L=L_{intra}+ L_{TSP}(T) - S(\overline{W}_{free}) \quad (1)$$

where $S(\overline{W}_{free})$ is the summation of the distance between each pair of entry and exit points for each cell over $\overline{W}_{free}$. Clearly, because $\overline{W}_{free}$ is compact and the *N* decomposed cells are compact, connected, and disjoint, S_{min}, S_{max} exist such that $S_{min} \leq S(\overline{W}_{free}) \leq S_{max}$. Moreover, L_{intra} is the complete coverage path length. This path length for each cell that is fully covered is bounded by a cell area divided by the width of the

back-and-forth motion pattern. This length is the same for any complete coverage path. In particular, for our decomposition and a priori selected intracell path pattern, the selection of the starting and ending points of intracell coverage path can be different. Therefore, we assumed that L_{intra} for $\bar{W}_{free}$ is constant. Moreover, $L_{TSP}(T) \leq \alpha(\bar{W}_{free})\sqrt{N}$, where $\alpha(\bar{W}_{free})$ is a constant (Steele 1990). Therefore, the TSP-CPP route length is upper and lower bounded. If the variation in the intracell path length due to different entry and exit points is small (e.g. tending to be average out for long intracell path) and can be neglected, then the optimal solution of TSP for the least sum of lengths of the intercell paths is an optimal solution used for minimizing the total coverage path length of TSP-CPP. The search for the shortest route for TSP-CPP over *W* for a given decomposition of $\bar{W}_{free}$ is equivalent to the search for the shortest path that results from the shortest TSP tour in combination with the largest *S* resulting from an appropriate selection from {cell-path combinations for all cells} over $\bar{W}_{free}$. In full generality, the solution of TSP-CPP requires the following two steps to be conducted simultaneously: (1) determining the visiting order of cells by solving a TSP of all cells and (2) determining a cell-path combination including the sweeping direction and the starting and ending points of the intracell path for each cell. The two subproblems are coupled, that is, the solution to one subproblem influences the solution to the other. The alternative cell paths and the visiting order of all cells or simplifications conducted by sequentially solving the two subproblems can be enumerated to produce an acceptable solution to TSP-CPP. The use of DP and GA for obtaining the global optimal solution for a given boustrophedon decomposition of *W* and specified intracell sweeping pattern is presented in the following section.

4.2 DP Approach

The DP approach methodology for TSP-CPP and the proof of its optimality are described in the study by Xie *et al*. (2018). The main idea of this approach is similar to that of DP for TSP. We can assign one cell as the starting cell and select one of its four cell-path combinations as a starting cell-path to break the route cycle. We then recursively update shortest path length for a specific visited state (set of cells that have already been visited) and last cell-path. This can be done by checking all possible previous cell-paths with corresponding previous visited states. The time complexity of DP approach is $O(2^N \times (4 \times N)^2)$, and the space complexity is $O(2^N \times 4 \times N)$. These time and space complexities are the same as those for DP for TSP, $O(2^N \times N^2)$and $O(2^N \times N)$, respectively. Although the DP approach can provide the optimal solution for TSP-CPP, the computation quickly becomes prohibitively large as the number *N* becomes moderately large due to the exponential factors in both time and space complexities. Thus, the DP approach is only suggested to be used for up to $N = 17$ (Bellman, 1961; Xie *et al*., 2018). However, numerous real-life applications have more than 20 cells in an environment with many obstacles.

4.3 GA Approach

A GA comprises a population of candidate solutions, a fitness function to rank the solutions, a selection mechanism, reproduction operators to generate the new population, and a termination criterion. A suitable set of solutions can be obtained using the GA with an appropriate setting of the parameters involved, design of the operators, and incorporation of

additional guiding heuristics (such as preference or mating restriction) to bias the search for operating a GA for a given problem. Due to its generality, GAs have been used to solve TSP (e.g., Vahdati *et al*., 2009; Ismkhan and Zamanifar, 2015) and TSPN (Wu and Liu, 2014). Among the many methods for TSP, GA is an efficient approach to obtain a suitable (in general, not optimal) solution within a limited time period or with a limited computation power and memory. Thus, GAs are suitable for path computation onboard UAV. For the operators of the GA, we modify the GA operators for TSP to apply to TSP-CPP.

4.3.1 Encoding and Decoding

For *N* given cells, a valid tour in a TSP is the one that begins and ends at the same cell that initiates the tour and that visits the other ($N-1$) cells exactly once. A chromosome represents a candidate solution of TSP or TSPN that encodes the cells in the order that the vehicle traverses. Each cell is a gene of the chromosome. Various scholars have different methods for representing TSP candidate solutions in a GA. For TSP in a GA, the genes of one individual are usually one ordered permutation $\{\sigma_1, \cdots, \sigma_N\}$of number $\{1, \cdots, N\}$, which can be directly viewed as the visiting order of the entire cells. The search procedure for a solution to TSP is simply to generate and evaluate as many different permutations $\{\sigma_1, \cdots, \sigma_N\}$ as possible so as to retain the optimal permutation as the solution. A candidate solution to TSP-CPP simultaneously comprises both the entry and exit points of each cell and the visiting order of cells, which is similar to TSPN. Let the four alternative pairings of exit and entry points of intracell lawnmower path for σ_i cell be denoted by Δ_{σ_i} with $|\Delta_{\sigma_i}|$=4 (refer to Figure 2). In TSP-CPP, a chromosome T is a TSP-CPP tour naturally represented by a concatination of two substrings

$$T = (v_1 \cdots v_N) \tag{2}$$
$$v_i = (\sigma_i, \theta_{\sigma_i}),\ i=1, \cdots, N,\ \theta_{\sigma_i} \in \Delta_{\sigma_i}$$

where v_i is a gene composed of $\sigma_i \in \{1, \cdots, N\}$ representing the cell ID, i.e. a given visiting order $\{\sigma_1, \cdots, \sigma_N\}$ of each cell associated with one of 4N cell-path combinations $\{\theta_{\sigma_1}, \cdots, \theta_{\sigma_N}\}$ assigned to each cell for a specified visiting order. That is, v_i is an offspring segment of a TSP-CPP route formed by associating each cell with one of four cell-path combinations. The cell ID and the corresponding choice of entry and exit points can be decoded by dividing 4 and modulo 4. Furthermore, for feasibility genes of one individual cannot have repeated cell ID to ensure that each cell is visited once. In order to find the shortest total tour length path, GA searches for the chromosome T defined in (2) minimizing the fitness function L in (1).

4.3.2 Fitness Function

We want to minimize the length of a tour, that is, the sum of the length of an intercell path and the length lengths of an intracell path in the order of visit. The fitness function used in the GA for TSP-CPP is then given by the length of the tour L defined in (1) with which to select and to evaluate a chromosome T. The vehicle may take any cellσ_i from where it is transitioned via a local route TSP_{ij} consisting of an exit point of cell σ_i and an entry point of cell σ_j to cell σ_j, which incurs a path cost $L_{TSPij} := \mathrm{dist}(\sigma_i, \sigma_j)$, where $\mathrm{dist}(\sigma_i, \sigma_j) > 0$ represents the local

transition path length of inter-cell path from cell ID σ_i to cell ID σ_j (i.e. shortest path length between the exit point of cell σ_i and the entry point of cell σ_j obtained by the visibility graph). Then

$$L_{TSP}(T) = \sum_{i,j=1,i\neq j}^{N} L_{TSPij}, \quad (3)$$

The metric dist(σ_i, σ_j) satisfies the triangle inequality. Since dist(σ_i, σ_j) might be different from dist(σ_j, σ_i), in general the distance metric depends on the direction of the motion. The optimal permutation we search is an asymmetric TSP.

4.3.3 Crossover Operator

Crossover is a crucial operation that influences the final outcome in GA. Crossover in the context of permutation is conducted to generate better offsprings by inheriting the optimal segments of parents and adding new segments that do not exist in the parents into the offsprings. Various crossover operators can be used in GA for TSP or TSPN. One simple but powerful crossover operator is the heuristic crossover operator used in the study by Vahdati *et al*. (2009). This heuristic crossover operator performs well even when the number of cities is greater than 100 (Ismkhan and Zamanifar, 2015). Our modified heuristic crossover operator for considering the entry and exit points of each cell can be described with the following steps:

1. Randomly pick one cell as the starting cell. Then, randomly pick the cell-path combination of starting cell from one parent. Add combination to the offspring as the starting cell path.
2. Find four candidates for the next cell, namely cells of the first left neighbor and the first right neighbor of the starting cell in two parents which are still not visited in the offspring. This step can be implemented by retaining four pointers of the current position, as described in the study by Vahdati *et al*. (2009).
3. Consider four alternative choices of the entry and exit points for each candidate cell so that 16 candidates are available for the next cell path. For the heuristic to determine the next cell path of the offspring in addition to the intercell path length between the current cell path of offsprings and candidates of the next cell-path, we consider the change in the intracell path length caused by different entry and exit point selections of a cell. We define the cost of one cell path to be its intracell path length minus the minimum intracell path length among four possible entry and exit point selections of its cell. This cost represents the additional cost for not selecting the entry and exit points of a cell with the shortest intracell path. We add the cell paths with a minimum distance (current cell path of offspring, cell-path) + cost(cell path) of the 16 candidates to the offspring as the next cell path.
4. Repeat steps 2 and 3 until all cells are visited in the offspring.

4.3.4 Mutation Operator

Algorithm 1: Given the visiting order of cells, find the optimal entry and exit points for each cell with the shortest coverage path using DP

Input: Offspring after the crossover and swap mutation
Output: Individual that has the same visiting order as the input with the shortest coverage path

```
Initialize a cell array of size N
for each i = 1 ~ N do
    cell[i] = the cell of individual[i]
Initialize the optimal solution array of size N
for each starting cell path of cell[1] do
    Initialize from an array of a size of 4N
    Initialize the shortest path length array of a size of 4N with infinity
    shortest path length[starting cell path] = intracell path length of the starting cell path
    for each i = 2 ~ N do
        for each current cell path of cell[i] do
            for each previous cell path of cell[i – 1] do
                if the shortest path length[previous cell-path] + distance(previous cell path, current cell
                path) + intracell path length of current cell path is smaller than the shortest path
                length[current cell-path] then
                    update the shortest path length from the current cell path
    for each ending cell path of cell[N] do
        add distance(ending cell path, starting cell path) to the shortest path length[ending cell path]
    find the optimal ending cell path with the shortest path length[ending cell path]
    if the shortest path length[best ending cell path] is smaller than the coverage path length of the optimal
    solution then
        find the whole solution for the current starting cell path by backtracking from the optial ending cell
        path from array
        update the optimal solution
return the optimal solution
```

The mutation operation is used for triggering the diversity of a population and avoiding getting trapped in the local optimum by extending the search space of GA occasionally. The swap mutation operator is a commonly used mutation operator in GA for TSP. In addition to the swap mutation operator, the two-opt local search was proposed for solving TSP and is a widely used and effective mutation operator that improves an individual solution through a local search. In our GA for TSP-CPP, we randomly swap the visiting order of two cell-path combinations under the assigned probability. Suppose the visiting order of cells is fixed; we need not go back to the starting cell. The only factor that influences the following path is our selection of the exit point for current cell. We can permute four possible starting cell-paths to break the route cycle, and use an array to record the shortest path length with regard to each cell-path combination of all cells for the given visiting order. As the minimum distance sum to current cell-path, whatever the choice of current cell-path, is only related to the previous cell-path and its shortest path length, we can update the minimum distance sum array by checking four possible previous cell-path combinations. An additional array is used to record from which previous cell-path we get the minimum distance sum with regard to current cell-path for backtracking. Then we can determine the optimal entry and exit points for each cell with the shortest coverage path through the application of dynamic programming in linear time (see the Algorithm 1) for a fixed visiting order of cells starting from any cell. Although

the result is a local optimal solution as we consider only the entry and exit points of each cell and the visiting order of cells cannot be modified, this optimal entry and exit points mutation will very likely increase our probability of getting a global optimal solution through evolution

4.3.5 Overall Workflow of GA Implementation

The overall workflow of our GA approach is described as follows:

1. Start with a diverse population. Initialize the population of the assigned population size with a random permutation of cells by using Knuth shuffling and randomly select one entry and exit points from four possible selections for each cell.
2. Calculate the coverage path length of each individual in the initial population and determine the current optimal solution.
3. Randomly select two parents from the population by using a roulette wheel selection and generate an offspring with our modified heuristic crossover operator. Repeat this process until there are as many offspring as the original population.
4. For each offspring generated using the crossover operator, randomly swap the visiting order of two cells according to the assigned swap mutation probability. Then, find the optimal entry and exit points for each cell with the same visiting order of cells (call Algorithm 1).
5. Replace the population with the newly generated offspring. Calculate the coverage path length of each individual in the new population and update the current optimal solution if any better solution is found.
6. Repeat step 3 to 5 until the termination criterion is met. Output the optimal solution obtained in the evolution process.

The minor implementation variations of GA could yield different achievable performance levels. Relative to the standard settings, a smaller (larger) population size and generation number can be used for a small (large) cell number, where the length of an individual coding is considerably small (large). In practice, a large population can hold numerous diverse individuals, which prevent important gene combinations from disappearing. The only significant difficulty in using a larger population size or a larger generation number for our current GA implementation on a desktop PC is that except that convergence simply takes too long or the cost of computing is high, because the fitness evaluation comprises the vast majority of computational time.

Note that in step 6, common termination criteria of the GA are run-time limit being reached, number of evolved generations being equal to the assigned generation number, or a predefined fitness value being achieved. For our purpose of simplicity and efficiency of GA implementation, maximum number of generations is adopted as the termination criterion. Once a converged solution is found, it can be checked if the converged solution is a local optimal solution or the global optimal solution (i.e. the same as the solution found by DP). This can be checked by using different initializations for GA to prevent the search from becoming trapped in local minima; one drawback of GA is that GA can generate local minima. An important implementation issue with GA for TSP or TSPN is initialization, which is an initial effort or guess toward the exploration of the solution space. GA initialization methods have been proposed for TSP. A suitable initialization, instead of randomly generating chromosomes, usually improves the convergence and solution quality of a GA. Moreover, a parallel computing approach, such as the island model, capitalizes on diversified initialization. Initialization of TSP is likely to obtain many duplicate individuals. Duplicates might limit further exploration of the search space because duplicates are likely to be paths with short length; thus population diversity will be low. To streamline search, duplicates of individuals can be prohibited. Various methods can generate an initial population but avoid duplicates (Wu and Liu, 2014).

4.3.6 Time and Space Complexity of GA

Although the computational load of our GA depends on the specific implementation details, here we give a complexity analysis. The measure used for time complexity of GA is the number of genetic operations and evaluations performed.
Time complexity:

- Calculate coverage path length: $O(N)$ for each individual
- Modified heuristic crossover: $O(16 \times N)$ for each offspring (16 candidates for each position)
- Optimal entry and exit points mutation: $O(64 \times N)$ for each offspring (Permute 4 possible starting cell-path combinations and consider the four possible previous cell-path combination search time for updating the shortest path length of $4N$ cell-path combinations)
- Overall: $O(N \times \text{population size} \times \text{generation number})$

Space complexity: $O(\text{population size} \times N)$

The path quality and convergence speed of GAs used for solving the instances of TSP-CPP are not sensitive to the shape of the workspace to be covered and the shape of the decomposed cells. We consider GA a potential approach to find the optimal or near-optimal solution for environments with a large number of decomposed cells that prohibit the application of DP. This is demonstrated in the following simulations.

5. SIMULATION RESULTS

To test the ability of a GA of converging to the optimal solution and to examine the GA generated path quality in the metric of path length, DP was used as the baseline of optimality, computation time, and memory usage. We first generated 100 random environments with multiple spatially distributed rectangular regions that cover the region for comparing the coverage path length generated by GA and DP. In the absence of obstacles, there is no requirement of considering the decomposition of an environment, and a visibility graph is not required for finding the shortest path between two points. For comparison simplicity, we assumed that the intracell path length is the same for all four combination selections of a cell. Therefore, the intracell path part is common for GA and DP, regardless of the solution. Only the intercell path length, which is an influential factor, was is used to calculate the relative error, which is also more similar to the original TSP. As seen in Table 1, our GA approach is capable of finding the optimal solution in multiple operations in all generated environments and the near-optimal (within 3%) solution, even in the worst performance. Figure 4 displays two examples of the tests with 21 cells and their corresponding optimal solutions. Figure 5 presents the test case with the maximum relative error of 2.78% (25 cells) in the worst performance scenario.

Table 1. Solution quality of GA for multiple spatially distributed rectangular regions over ten runs. For each cell number, we randomly generated ten corresponding test cases of the same environment size. The number of successes is defined as the number of times that the GA successfully finds the same optimal solution as DP among those ten cases. The relative error reported here is calculated between the intercell path length of DP and GA

Cell Number	Environment Size	Number of Success	Relative Error (intercell path length)	
			Maximum	Average
16	1280×720	10	0.0%	0.0%
17	1080×1080	10	0.0%	0.0%
18	1280×720	10	0.0%	0.0%
19	1080×1080	10	0.0%	0.0%
20	1280×720	9	0.095%	0.01%
21	1080×1080	9	1.063%	0.106%
22	1280×720	8	0.322%	0.057%
23	1080×1080	8	1.217%	0.227%
24	1080×1080	7	1.688%	0.288%
25	1080×1080	4	2.778%	0.750%

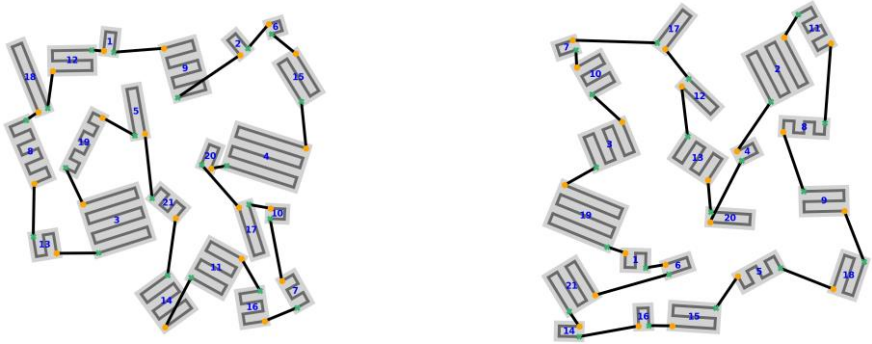
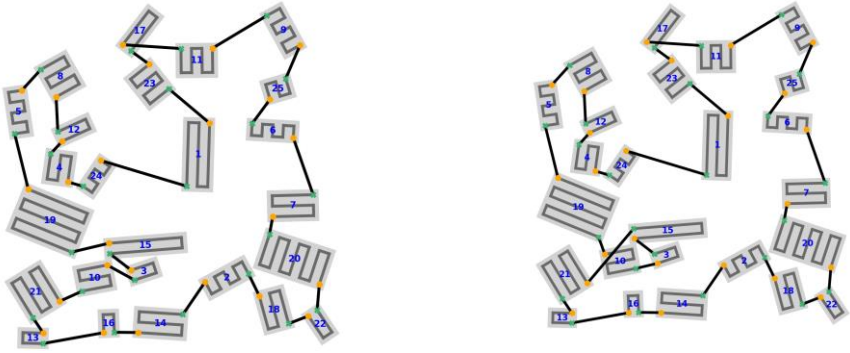


Figure 4. Two example test cases with 21 cells and their corresponding optimal solutions



(a) (b)

Figure 5. Test case (25 cells, as tabulated in Table 1) with the maximum relative error of 2.778% for solutions obtained using the GA-based and DP-based approaches. (a) Optimal solution generated by DP and (b) solution generated using our GA approach. The numbers represent the cell IDs

The solution quality and performance (computation time and memory) of DP and GA approaches to TSP-CPP depend on the task environment of interest. To verify the performance and correctness of the implementation, the simulations were conducted for polygons of various shapes and sizes with different obstacle distributions. Then, we designed six experiment environments of varying complexity comprising polygonal obstacles of various shapes, which are possibly concave, and sizes so that the decomposition yields different numbers of cells (between 10 and 26). We used a sensor radius of 10, population size of 4096, generation number of 16, crossover probability of 0.9, and swap mutation probability of 0.02 for all the experiments. For the performance test, we used the GA five times on each environment. Moreover, if the GA was successful in finding the same solution as the DP, the time and memory required were recorded. The successful results are displayed in Table 2. The table shows the improvement of GA over the DP as the cell number increases up to 26 (the dimensionality limit of DP in the PC) in terms of the computation time and memory. In all the experiments, an agreement in DP and GA is observed for obtaining the same solution without adjusting the parameters, which is guaranteed to be the optimal solution. GA can find the same solution as that of DP for all six experiments within a much less time and with a much lower memory requirement when the number of cells is increased. Figures 6 to 11 display the optimal solutions obtained by both DP and GA in different environments. For the experiment Nº 1 that is illustrated in Figure 6, the entire optimal path is depicted. However, for the experiment Nº 2, 3, 4, 5, and 6 that are illustrated in Figures 7 to 11, respectively, we only depict the intercell path and the entry and exit points for each cell. However, the intracell path is not plotted. Note that Figure 11 is a densely crowded environment that comprises 26 cells after decomposition.

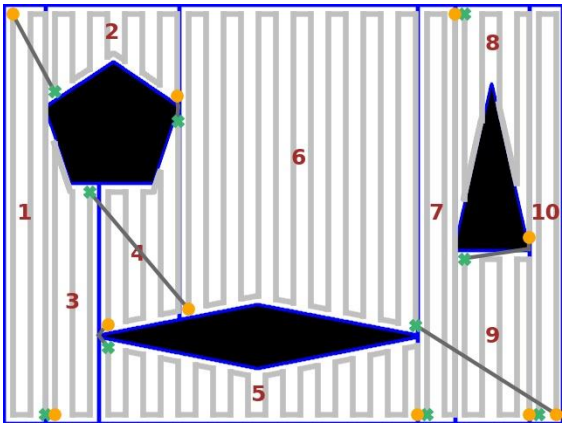


Figure 6. Optimal solution for Experiment 1 with the total coverage path length of 15582. The light gray path is the intracell path (boustrophedon path), and the dark gray path is the intercell path. The numbers represents the cell IDs. The optimal visiting order of cells in this simulation is 1, 3, 6, 10, 9, 8, 7, 5, 4, 2, and 1

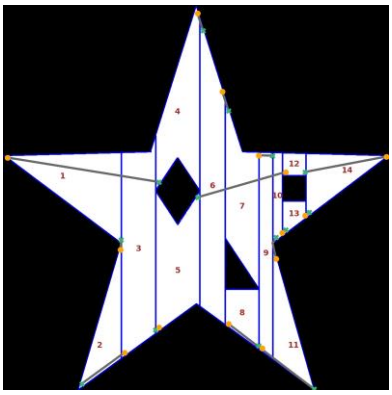


Figure 7. Optimal solution for Experiment 2 with a total coverage path length of 26977.5. The numbers represent the cell IDs.The optimal visiting order is 1, 2, 3, 5, 12, 14, 13, 10, 11, 8, 9, 7, 6, and 4

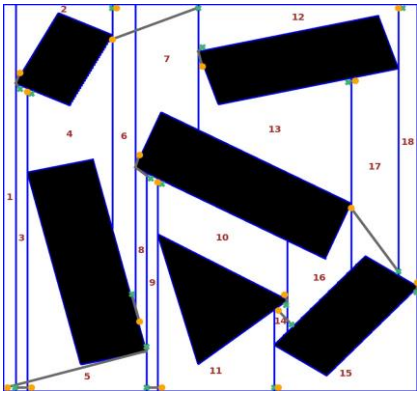


Figure 8. Optimal solution for Experiment 3 with a total coverage path length of 43682.8. The numbers represent the cell IDs. The optimal visiting order is 1, 5, 11, 15, 18, 12, 13, 17, 16, 14, 10, 9, 7, 4, 3, 2, 6, 8, and 1

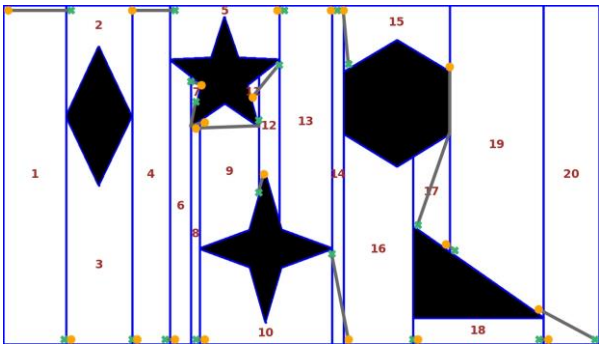


Figure 9. Optimal solution for Experiment 4 with a total coverage path length of 43640.1. There are concave obstacles that increase the number of decomposed cells having a few small cells. The numbers represent the cell IDs. The optimal visiting order is 1, 3, 4, 6, 7, 9, 12, 11, 8, 10, 16, 18, 20, 19, 17, 15, 14, 13, 5, 2, and 1

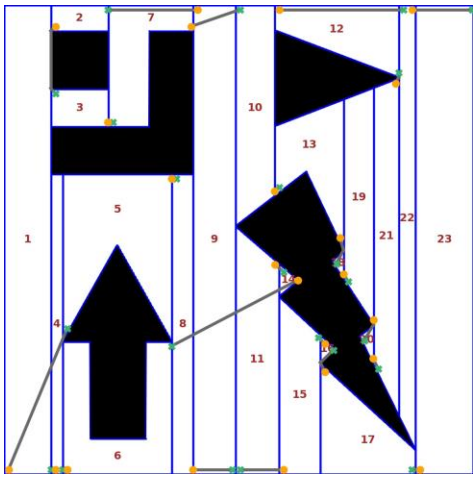


Figure 10. Optimal solution for Experiment 5 with a total coverage path length of 52065.4. The numbers represent the cell IDs. The optimal visiting order is 1, 4, 6, 14, 11, 15, 16, 17, 23, 22, 12, 21, 20, 19, 18, 13, 10, 7, 3, 2, 9, 8, 5, and 1

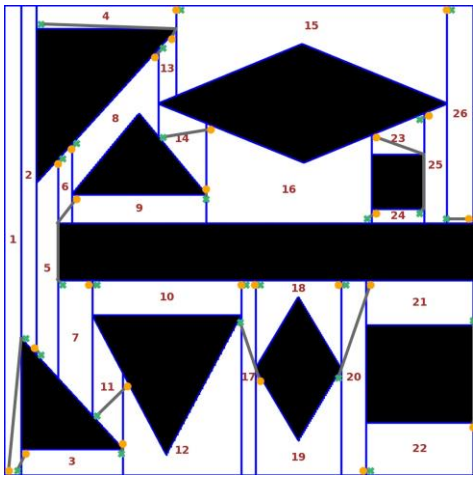


Figure 11. Optimal solution for Experiment 6 with the total coverage path length of 37717.7. The numbers represent the cell IDs. The optimal visiting order is 1, 3, 11, 12, 19, 21, 22, 20, 18, 17, 10, 7, 9, 14, 16, 24, 23, 25, 26, 15, 4, 13, 8, 6, 5, 2, and 1

As presented in Table 2, when the population size and generation number are fixed, the computation time of GA is almost linear to the cell number. GA is determined to be considerably less computationally expensive and much (2–3 orders of magnitude) faster than the DP approach that has an exponential time complexity when the cell number is large. Moreover, the memory of GA is 2–3 orders of magnitude less than that of DP. Due to the exponential time and space complexity (curse of dimensionality or dimensionality limitation due to fast growth of the search space) encountered by DP as the cell number increases, except for maps with a small cell number (that is, small problem dimensionality), the application of

DP is limited to TSP-CPP of low dimensions. For example, application of DP is problematic in real-time applications that require onboard computation by a UAV with limited memory and computational resources. Nevertheless, because the complexity is linear to the cell number, if population size and generation number are fixed, we considered GA, in combination with decomposition, to be an easy-to-implement, suitable metaheuristics approach whose convergence speed is insensitive to the obstacle distribution. GA yields an optimal or near-optimal solution with a high quality of TSP-CPP for environments with a large number of decomposed cells, which prohibit the application of DP.

Table 2. Time and memory comparisons of GA and DP for Figures 6 to 11. GA can find the same solution as the optimal solution of DP in all these six experiments. C++ programs of DP and GA are executed on the same computer (HP Z440 Workstation, CPU: Intel Xeon E5-1630 v3, 64G RAM, OS: Ubuntu 18.04) five times to obtain the average computation time. Parameters for GA: population size = 4096, generation number = 16, crossover probability = 0.9, and swap mutation probability = 0.02

Experiment N°	**Environment Size**	***N***	**Computation Time (s)**			**Memory (Maximum RSS, MB)**		
			DP	**GA**	**DP/GA**	**DP**	**GA**	**DP/ GA**
1 (Figure 6)	640 × 480	10	0.014	1.508	0.009	3.668	4.052	0.905
2 (Figure 7)	1200 × 1200	14	0.289	1.845	0.157	6.836	4.264	1.603
3 (Figure 8)	1080 × 1080	18	7.538	2.210	3.411	78.92	4.284	18.42
4 (Figure 9)	1280 × 720	20	36.53	2.361	15.47	339.1	4.340	78.13
5 (Figure 10)	1080 × 1080	23	383.2	2.629	145.8	3083	4.716	653.8
6 (Figure 11)	1080 × 1080	26	3965	2.900	1367	27790	4.740	5863

6. CONCLUSION

In this study, we pointed out the similarity between TSP-CPP and TSPN in solving the NP-hard problem of interest, which was also seen in Yu (2015). Both approaches are variants of TSP with additional complexity; they involve searching one or two via points over the spatial region for enabling transition between cells. For the TSP-CPP problem, after a boustrophedon decomposition of the environment has been conducted to obtain nonoverlapping cells, a coverage intracell path comprising back-and-forth path pattern with four options of pairs of the starting and ending points for each cell can be determined. To achieve a suitable balance between efficiency, scalability, and path optimality, a sequential two-step optimization procedure based on an appropriate implementation of GA was adopted for full coverage with null or minimum repetition of a known cluttered environment after decomposition into cells. First, a visiting order of cells was determined using GA. Then, optimal entry and exit points according to a given visiting order were determined so that the crossing between the two cells occurs only once. The combination of the GA and DP approaches can help to coordinate and jointly optimize both the selections of the entry and exit points for each cell and the visiting order of the cells simultaneously. Moreover, we demonstrated that GA can successfully find the true optimal solution as found by DP in all six performed experiments in all six performed experiments that can be solved both by DP and GA. Furthermore, GA is hundred times faster than the DP approach when the cell number is large. Therefore, due to the limitations of computation time and computational resources in

applications such as UAV flight, GA offers a simple-to-implement, computationally feasible approach to TSP-CPP in highly complex environments with numbers of decomposed cells beyond the dimensionality limitation of DP. In this study, we did not consider the more realistic vehicle model that the incorporates the size and shape of a vehicle and its kinodynamic constraints, such as curvature or turn-rate constraint, into the motion of vehicle, such as Dubins vehicle (Dubins TSPN) (Sujit *et al*., 2013; Yu, 2015) and that includes the variation in the intracell path length caused by different selections of entry and exit points. For nonpolygonal or arbitrary shape obstacles, convex hull or bounding volume approaches can be implemented at the expense of reduced free space. Some possible further improvements include testing more intracell path patterns by considering the reduction in the number of turns cost (Huang, 2001; Vandermeulen *et al*., 2019), which is related to the travel time and energy consumption, because the complete coverage path in the current implementation comprises a single sweeping direction zig-zag (back-and-forth sweeping) path pattern with frequent turns. A multi-objective GA can also manage multiple conflicting criteria for more difficult coverage path optimization problems. Future research will consider these complicating factors.

REFERENCES

Alatartsev et al., 2015. Robotic Task Sequencing Problem: A survey. *Journal of Intelligent & Robotic Systems*, Vol.*80*, No.2, pp. 279-298.

Alt, H. and Welzl, E., 1988. Visibility Graphs and Obstacle-avoiding Shortest Paths. *In Zeitschriftfür Operations-Research*, Vol. 32, Nos. 3 & 4, pp. 145-164.

Bellman, R.E., 1961. Dynamic Programming Treatment of the Traveling Salesman Problem. *In Journal of the ACM*, Vol. 9, No. 1, pp. 61-63.

Cai, Z. et al., 2014. Research on Complete Coverage Path Planning Algorithms Based on A* Algorithms. *In the Open Cybernetics Systemics Journal*, Vol. 8, pp. 418-426.

Choset, H., 2001. Coverage for Robotics–A Survey of Recent Results. *In Annals of Mathematics and Artificial Intelligence*, Vol. 31, Nos. 1 & 4, pp. 113-126.

Choset, H. and Pignon, P., 1998. *Coverage Path Planning: The Boustrophedon Cellular Decomposition*. *In Book: Field and Service Robotics*. Springer, London, pp. 203-209.

Das, D. et al., 2016. Techniques in Multi-robot Area Coverage: A Comparative Survey. In *Book: Handbook of Research on Design, Control, and Modeling of Swarm Robotics*. IGI Global, pp. 741-765.

Dasgupta, P. (2015). *Coverage Path Planning Using Mobile Robot Team Formations*. In *Book: Emerging Research on Swarm Intelligence and Algorithm Optimization*. IGI Global, pp. 214-247.

Dogru, S. and Marques, L., 2017. *A*-based Solution to the Coverage Path Planning Problem*. *In book: Iberian Robotics Conference*. Springer, Cham, pp. 240-248.

Galceran, E. and Carreras, M., 2013. A Survey on Coverage Path Planning for Robotics. In *Robotics and Autonomous Systems*, Vol. 61, No. 12, pp. 1258-1276.

Huang, W.H., 2001. Optimal Line-sweep-based Decompositions for Coverage Algorithms. *Proceedings 2001 IEEE International Conference on Robotics and Automation*, pp. 27-32.

Ismkhan, H. and Zamanifar, K., 2015. Study of Some Recent Crossovers Effects on Speed and Accuracy of Genetic Algorithm, Using Symmetric Travelling Salesman Problem. *arXiv preprint arXiv:1504.02590.*

Jimenez, P.A. et al., 2007. Optimal Area Covering Using Genetic Algorithms. *2007 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1-5.

Khan, A. et al., 2017. On Complete Coverage Path Planning Algorithms for Non-holonomic Mobile Robots: Survey and Challenges. *In Journal of Information Science and Engineering*, Vol. 33, No. 1, pp. 101-121.

Khsheibun, E. et al., 2018. Efficient Coverage of Unstructured Environments. *International Conference on Intelligent Autonomous Systems*, Springer, Cham, pp. 111-126.

Kovács, A., 2016. Integrated Task Sequencing and Path Planning for Robotic Remote Laser Welding. *In International Journal of Production Research*, Vol. 54, No. 4, pp. 1210-1224.

Mansouri, S.S. et al., 2017. On the Covering of a Polygonal Region with Fixed Size Rectangles with an Application Towards Aerial Inspection. *2017 25th Mediterranean Conference on Control and Automation (MED)*, pp. 1219-1224.

Oksanen, T. and Visala, A., 2009. Coverage Path Planning Algorithms for Agricultural Field Machines. In *Journal of Field Robotics*, Vol. 26, No. 8, pp. 651-668.

Otto, A. et al., 2018. Optimization Approaches for Civil Applications of Unmanned Aerial Vehicles (UAVs) or Aerial Drones: A Survey. *In Networks*, Vol. 72, No. 4, pp. 411-458.

Potvin, J.Y., 1996. Genetic Algorithms for the Traveling Salesman Problem. *In Annals of Operations Research*, Vol. 63, No. 3, pp. 337-370.

Scholz, J., 2019. Genetic Algorithms and the Traveling Salesman Problem a Historical Review. *arXiv preprint arXiv:1901.05737*.

Steele, J.M., 1990. Probabilistic and Worst Case Analyses of Classical Problems of Combinatorial Optimization in Euclidean Space. *In Mathematics of Operations Research*, Vol. 15, No. 4, pp. 749-770.

Sujit, P.B. et al., 2013. Route Planning for Angle Constrained Terrain Mapping Using an Unmanned Aerial Vehicle. *In Journal of Intelligent & Robotic Systems*, Vol. 69, Nos. 1 & 4, pp. 273-283.

Vahdati, G. et al., 2009. A New Approach to Solve Traveling Salesman Problem Using Genetic Algorithm Based on Heuristic Crossover and Mutation Operator. *2009 International Conference of Soft Computing and Pattern Recognition*, pp. 112-116.

Vandermeulen, I. et al., 2019. Turn-minimizing Multirobot Coverage. *2019 IEEE International Conference on Robotics and Automation*.

Wu, S.Y. and Liu, J.S., 2014. Evolutionary Path Planning of a Data Mule in Wireless Sensor Network by Using Shortcuts. *2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2708-2715.

Xie, J. et al., 2018. An Integrated Traveling Salesman and Coverage Path Planning Problem for Unmanned Aircraft Systems. *In IEEE Control Systems Letters*, Vol. 3, No. 1, pp. 67-72.

Yu, X., 2015. *Optimization Approaches for a Dubins Vehicle in Coverage Planning Problem and Traveling Salesman Problems* (Doctoral dissertation). Department of Electrical Engineering, Auburn University, Auburn, AL.