2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Well-Connected Set and Its Application to Multi-Robot Path Planning

Teng Guo Jingjin Yu

***Abstract*— Parking lots and autonomous warehouses for accommodating many vehicles/robots adopt designs in which the underlying graphs are *well-connected* to simplify planning and reduce congestion. In this study, we formulate and delve into the *largest well-connected set* (LWCS) problem and explore its applications in layout design for multi-robot path planning. Roughly speaking, a well-connected set over a connected graph is a set of vertices such that there is a path on the graph connecting any pair of vertices in the set without passing through any additional vertices of the set. Identifying an LWCS has many potential high-utility applications, e.g., for determining parking garage layout and capacity, as prioritized planning can be shown to be complete when start/goal configurations belong to an LWCS. In this work, we establish that computing an LWCS is NP-complete. We further develop optimal and near-optimal LWCS algorithms, with the near-optimal algorithm targeting large maps. A complete prioritized planning method is given for planning paths for multiple robots residing on an LWCS.**

I. Introduction

Designing infrastructures that accommodate many mobile entities (e.g., vehicles, robots, and so on) without causing frequent congestion or deadlock is critical for improving system throughput in real-world applications, e.g., in an autonomous warehouse where many robots roam around. A good design generally entails good *environment connectivity* in some sense. This paper captures the intuition of "good connectivity" with the concept of *well-connect set* (WCS), presents a comprehensive study on computing largest well-connected sets (LWCS), and highlights the application of LWCS in multi-robot path planning (MRPP).

To illustrate what a WCS is, let's consider a parking garage. It is essential to design it so vehicles can park without blocking each other, and retrieving a parked vehicle doesn't require moving other vehicles. Roughly speaking, the parking spots satisfying these requirements form a WCS, and finding an LWCS is instrumental in determining maximum parking capacity while minimizing congestion. Well-formed infrastructures based on WCSs are encountered in a broad array of real-world scenarios, including fulfillment warehouses, parking structures, storage systems [1]–[4], and so on. These infrastructures, designed properly, efficiently facilitate the movement of the enclosed entities, ensuring smooth operations and avoiding blockages.

WCS is especially relevant to MRPP, which involves finding collision-free paths for many mobile robots [5]–[10]. Here, the challenge lies in finding feasible paths connecting each robot's start and target positions. The concept of WCS becomes crucial, as it ensures that each robot can reach its destination without traversing other robots' positions, thereby guaranteeing a deadlock-free solution for easily realized prioritized planners.

(a) (b)

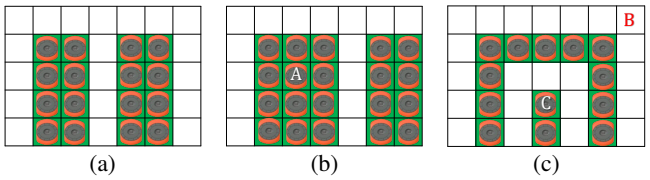
Fig. 1: Examples of well-formed infrastructures. (a) Amazon fulfillment warehouse. (b) A typical parking lot.

In this paper, we provide a rigorous formulation for the LWCS problem and establish its computational intractability. We then propose two algorithms for tackling this challenging combinatorial optimization problem. The first algorithm is an exact optimal approach that guarantees finding a largest well-connected vertex set, while the second algorithm offers a suboptimal but highly efficient solution for large-scale instances. As an application, LWCS readily provides prioritized MRPP with completeness guarantees.

Related work. The concept of well-connected vertex sets is inspired by well-formed infrastructures [10]. In well-formed infrastructures, such as parking lots and fulfillment warehouses [1], the endpoints are designed to allow multiple robots (vehicles) to move between them without completely obstructing each other, where the endpoint can be a parking slot, a pickup station, or a delivery station. Many real-world infrastructures are built in this way to benefit pathfinding and collision avoidance. Planning collision-free paths that move robots from their start positions to target positions, known as multi-robot path planning or MRPP, is generally NP-hard to optimally solve [11], [12]. In real applications, prioritized planning [13], [14] is one of the most popular methods used to find collision-free paths for multiple moving robots where the robots are ordered into a sequence and planned one by one such that each robot avoids collisions with the higher-priority robots. The method performs well in uncluttered settings but is generally incomplete and can fail due to deadlocks in dense environments. Prior studies [9], [10] show that prioritized planning with arbitrary ordering is guaranteed to find deadlock-free paths in well-formed environments. When a problem is not well-formed, it may be possible to find a solution using prioritized planning with a specific priority ordering, as proposed in [9]. However, finding such a priority order can be time-consuming or even impossible. To our knowledge, no previous studies investigated how to efficiently

G. Teng, and J. Yu are with the Department of Computer Science, Rutgers, the State University of New Jersey, Piscataway, NJ, USA. Emails: {teng.guo, jingjin.yu}@rutgers.edu. This work was supported by NSF award IIS-1845888 and an Amazon Research Award.

design a well-formed layout that fully utilizes the workspace.



(a) (b) (c)

Fig. 2: (a) The green cells form a WCS. Any robot parked at one of these cells does not block others' move. (b) An example of a non-WCS. Retrieving one robot, A, requires moving some other robots. (c) An example of a SWCS. B has no access to a robot at C without moving others.

Organization. Section II provides detailed problem formulations. In Section III, we investigate the theoretical properties and establish the proof of NP-hardness. Next, in Section IV, we present our algorithms for finding WCS while optimizing the size of the set. Section V explores the application of WCS in multi-robot navigation. In Section VI, we evaluate the effectiveness of our algorithms on various maps. Finally, we conclude the paper in Section VII and discuss future directions for research.

II. Problem Formulation

A. Well-Connected Set

Let $G(V, E)$ be a connected undirected graph representing the environment with vertex set V and edge set E the edge set. A well-connected set (WCS) is defined as follows.

Definition (Well-Connected Set (WCS))**.** *On a graph $G(V, E)$, a vertex set $M \subset V$ is well-connected if (i). $\forall u, v \in M, u \neq v$, there exists a path connecting u, v without passing through any $w \in M - \{u, v\}$, (ii) the induced subgraph of G by the vertex subset $V - M$ is connected.*

WCS enforces a stronger connectivity requirement. If a vertex set M satisfies (i) but violates (ii), we call M a *semi-well-connected set* (SWCS). Any WCS is a SWCS set but the opposite is not true (see, e.g., Fig. 2(c)).

For a given G, there are many WCSs. We are particularly interested in computing a largest such set. Toward that, We introduce two related concepts: the *maximal well-connected set* and the *largest well-connected set*.

Definition (Maximal Well-Connected Set (MWCS))**.** *A* WCS *M is maximal if for any $v \in V - M$, $\{v\} \bigcup M$ is not a well-connected set.*

Definition (Largest Well-Connected Set (LWCS))**.** *A largest well-connected set M is a WCS with maximum cardinality.*

By definition, a LWCS is also a MWCS; the opposite is not necessarily true. In this paper, we focus on maximizing the "capacity" of well-formed infrastructures, or in other words, maximizing the cardinality of the WCS. Besides capacity, we also introduce the *path efficiency ratio* (PER) for evaluating how good a layout is from the path-length perspective.

Definition (Well-Connected Path (WCP))**.** *Let M be a WCS. A path $p = (p_0, ..., p_k)$ is a well-connected path connecting p_0 and p_k if its subpath $(p_1, ..., p_{k-1})$ does not pass through any vertex in M.*

If M is a WCS, a WCP connects any two vertices $u, v \in M$. We denote $d_w(u, v)$ as the shortest WCP distance between u, v and $d(u, v)$ as the shortest path distance.

Definition (Path Efficiency Ratio)**.** *Let M be a WCS of G and $u \in M$ as the reference point (i.e. I/O port), the* path efficiency ratio *w.r.t vertex u is defined as $\frac{\sum_{v \in M} d(u,v)}{\sum_{v \in M} d_w(u,v)}$.*

III. Theoretic Study

In this section, we investigate the property of WCS and prove that finding LWCS is NP-hard.

A vertex in an undirected connected graph is an *articulation point* (or cut vertex) if removing it (and edges through it) disconnects the graph or increases the number of connected components. We observe if a WCS contains a node v, then v should not be an articulation point so as not to violate property (ii) in the definition of WCS.

Proposition III.1. *If M is a* WCS*, for any $M' \subseteq M, v \in M'$, v is not an articulation point of the subgraph induced by $V - M' + \{v\}$.*

Next, we investigate the property of the node in WCS and its neighbors.

Proposition III.2. *Let M be a* WCS*. For any $v \in M$, if $|M| > |N(v)| + 1$ where $N(v)$ denotes the set of neighbors of v, then at least one of its neighbors is not in M.*

Proof. Assume $N(v) \subset M$. Because $|M| > |N(v)| + 1$, $M - (N(v) \bigcup \{v\}) \neq \emptyset$. Let $w \in M - (N(v) \bigcup \{v\})$, then every path from w to v has to pass through a neighboring node of v, which contradicts property (i) of WCS. □

We now prove that finding a LWCS is NP-hard.

Theorem III.1 (Intractability)**.** *Finding the* LWCS *is NP-hard.*

Proof. We give the proof by using a reduction from *3SAT* [15]. Let (X, C) be an arbitrary instance of 3SAT with $|X| = n$ variables $x_1, ..., x_n$ and $|C| = m$ clauses $c_1, ..., c_m$, in which $c_j = l_j^1 \vee l_j^2 \vee l_j^3$. Without loss of generality, we may assume that the set of all literals, l_j^k's, contain both unnegated and negated forms of each variable x_i.

From the 3SAT instance, a LWCS instance is constructed as follows. For each variable x_i, we create three nodes, one node is for x_i, one node is for its negation $\bar{x}_i$, and y_i which is a gadget node. The three nodes are connected with each other with edges and form a triangle gadget. For each clause variable $c_i = l_j^1 \vee l_j^2 \vee l_j^3$, we create a node and connect it to the three nodes that are associated with $\bar{l}_j^1, \bar{l}_j^2, \bar{l}_j^3$. Finally, we create an auxiliary node z and add an edge between z and each literal node. Fig. 3 gives the complete graph constructed from the 3CNF formula $(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_4)$.

To find a LWCS, we observe that for each triangle gadget formed by $(x_i, \bar{x}_i, y_i)$, the node $x_i, \bar{x}_i$ should not be selected at the same time. Otherwise, if node y_i is not selected, it would be completely isolated, which violates Proposition III.1. If y_i is also selected, then every path to y_i from other nodes

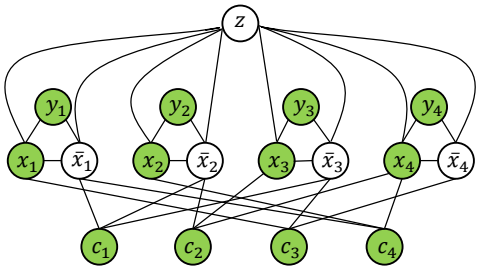


Fig. 3: The graph derived from the 3SAT instance $(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_4)$. The green vertices form a LWCS of size 12. By setting the literals of green vertices to true, the 3SAT instance is satisfied.

must always pass through either node x_i or node $\bar{x}_i$, violating WCS definition. This implies that for each triangle, at most two nodes can be selected and added to the set, and one of the nodes must be y_i. A second observation is if the clause node $c_j = l_j^1 \vee l_j^2 \vee l_j^3$ is selected as a vertex of the WCS, the nodes $\{\bar{l}_j^1, \bar{l}_j^2, \bar{l}_j^3\}$ connected to c_j cannot be selected simultaneously. Otherwise, node c_i would be blocked by the three nodes.

If the 3SAT instance is satisfiable, let $\widetilde{X} = \{\widetilde{x}_1, ..., \widetilde{x}_n\}$ be an assignment of the truth values to the variables. Based on these observations, the MWCS of size $2n+m$ is constructed as $\widetilde{X} \bigcup C \bigcup Y$, where $Y = \{y_1, ..., y_n\}$. On the other hand, if the reduced graph has a WCS of size $2n+m$, for each triangle gadget, we can only choose two, and one of them must be the gadget node. Thus, all the gadget nodes should be selected and this contributes $2n$ nodes. For the remaining m nodes, we hope that all nodes in C are selected. To not violate the well-connectedness, for each clause node $c_j = l_j^1 \vee l_j^2 \vee l_j^3$ to be selected, at least one of the node from $(\bar{l}_j^1, \bar{l}_j^2, \bar{l}_j^3)$ should not be selected. This is equivalent to ensuring that $c_j = l_j^1 \vee l_j^2 \vee l_j^3$ is true. Therefore, if the node $\widetilde{x}_i$ is selected for each i, we set $\widetilde{x}_i = true$, then the resulting assignment $\widetilde{X}$ satisfies the 3SAT instance. □

Next, we investigate the upper bound for the LWCS denoted as M^*.

Proposition III.3. *Denote Δ as the maximum degree of the node of G. We have $|M^*| \leq \max(\frac{\Delta-1}{\Delta}|V|, \Delta+1)$.*

Proof. For each $v \in M^*$, at least one of its neighbors u should be in $V - M^*$. Charge v to u. A node $u \in V - M^*$ can be charged at most $\Delta - 1$ times. Hence, we have $|V - M^*| \geq \frac{|M^*|}{\Delta-1}$. Since every node is either in M^* or $V - M^*$ we have $|M^*| + |V - M^*| = |V| \geq (1 + \frac{1}{\Delta-1})|M^*|$. On the other hand, it is possible that $M^* = N(v) \bigcup \{v\}$ for a node v of degree Δ. Therefore, $|M^*| \leq \max(\frac{\Delta-1}{\Delta}|V|, \Delta+1)$. □

IV. Algorithms for finding well-connected sets

A. Algorithm for Finding a MWCS

We first present an algorithm to find a MWCS in a graph, which begins by initializing two sets: M and P. Set M contains vertices currently included in the WCS, while P contains those available for adding to the set. Initially, M is empty, and P contains all the vertices of G. The algorithm selects a vertex from P and adds it to M while maintaining it as a WCS. At each step, we use Tarjan's algorithm [16] to compute the set of articulation vertices for the subgraph induced by $V - M$ and remove all the articulation vertices from P. Assume $v \in M$ and $u \in N(v)$, u is called an orphan neighbor of v if $u \notin M$ and $N(v) - u \subset M$. We iterate over all the neighboring vertices of M to remove all orphan neighbors from P, according to Proposition III.2. If $|P| = 0$, the algorithm cannot add another vertex to M while keeping it well-connected. To ensure that M is indeed maximal, we check if a $v \in M$ exists, such as $M \subset N(v) \bigcup \{v\}$. If so, we add the remaining neighbor of v to M so that M is maximal. Finding the set of articulation vertices using Tarjan's algorithm takes $O(|V| + |E|)$. As a result, the algorithm for finding a MWCS takes $O(|M|(|V| + |E|)) = O(|V|(|V| + |E|))$.

Algorithm 1: Maximal Well-Connected Set

```
Function MaximalWVS(G):
    M ← {}, P ← V
    while |P|! = 0 do
        AP ← Tarjan(G'(V − M))
        NB ← FindOrphanNeighbors(M)
        P ← P − (AP ⋃ NB)
        u ← ChooseOne(P)
        M.add(u) and P.pop(u)
    M ← AdditionalCheck(M)
    return M
```

Next, we establish lower bounds for the MWCS algorithm.

Proposition IV.1. *Denote W as the set of terminal nodes that contains nodes of degree one. Then $W \subseteq M$.*

Proof. Every terminal node u is neither an orphan neighbor nor an articulation point of the induced subgraph $G'(V - M + \{u\})$. Thus $W \subseteq M$. □

Proposition IV.2. *Denote L as the length of the longest induced path of G. Then $|M| \geq \frac{|V|}{L}$.*

Proof. Let $u \in V - M$. Then u is either an orphan neighbor or an articulation point of $G'(V-M)$. For every $v \in M$, there exists a WCP p_v connecting u and v. Every vertex in $V - M$ should be passed through by at least one of those WCPs. Otherwise, let w be the vertex that is not passed through by any of the WCPs. Clearly, w cannot be an articulation point or an orphan neighbor. Then w can be added to M, which contradicts that M is maximal. Let $p'_v = p_v - v$. Then $\bigcup_{v \in M} p'_v = V - M$. Meanwhile, $|p_v| \leq diam(G(V-M)) \leq L$, where $diam(G(V - M))$ is the diameter of the induced subgraph by $V - M$. We have $|V - M| \leq \sum_{v \in M} |p'_v| \leq |M|(L-1)$. On the other hand $|V - M| + |M| = |V|$. Hence $|M| \geq |V|/L$. □

B. Exact Search-Based Algorithm

We now establish a complete search algorithm to find a LWCS. The algorithm employs a depth-first search (DFS) approach to exhaustively explore all possible vertex sets and select the one with the maximum size and the highest path efficiency ratio (PER). The algorithm starts by defining three empty sets, M, M^*, and $visited$, where M represents the current set of vertices being explored, M^* represents the set

with the maximum size and highest PER found so far, and $visited$ stores all previously explored vertex sets to avoid repeated explorations. Then, the DFS search is initiated by calling the DFS function with the current set M, the set with the maximum size and highest PER found so far M^*, and the set of visited vertex sets visited as parameters. The DFS function starts by checking if the current set M has already been explored before. If it has, the function returns immediately. Otherwise, M is added to the visited set. The function then checks if the current set M is larger than the set with the maximum size and highest PER found so far M^*. If it is, then we update M as the current best M^*. If M has the same size as M^*, then the function compares their PER values, and the set with the higher PER becomes M^*.

Otherwise, the function generates a list of candidate vertices P that can be added to the current set M without violating the WCS condition. The function then loops over the candidate vertices and recursively calls the DFS function with the current set M unioned with the current candidate vertex and the same M^* and visited sets.

The algorithm continues until all possible vertex sets have been explored, or the condition for returning early is met. Similar to the algorithm for the MWCS, we also perform additional checks. For each $v \in M$, we check if $N(v) \bigcup \{v\}$ can be larger than the current solution found to ensure that the final vertex set is the maximum one. The algorithm's time complexity is dependent on the size and density of the graph G, as well as the number of candidate vertices generated at each recursive call. In the worst-case scenario, the algorithm has a time complexity of $O(2^{|V|}(|V| + |E|))$, where V is the number of vertices in the graph G. However, the early termination condition in the DFS function helps avoid exploring unnecessary vertex sets and can significantly reduce the algorithm's execution time.

Algorithm 2: Largest Well-Connected Set

Function DFSSEARCH(G)**:**
  $M, M^*, visited \leftarrow \emptyset, \emptyset, \emptyset$
  DFS($G, M, M^*, visited$)
  $M^* \leftarrow$ AdditionalCheck(M^*)
  return M^*
Function DFS($G, M, M^*, visited$)**:**
  if $M \in visited$ **then return**
  $visited.add(M)$
  if $|M| > |M^*|$ *or* ($|M| = |M^*|$ *and* $PER(M) < PER(M^*)$) **then** $M^* \leftarrow M$
  $AP \leftarrow$ Tarjan($G'(V - M)$)
  $NB \leftarrow$ FindOrphanNeighbor(M)
  $P \leftarrow P - (AP \bigcup NB)$
  if $|P| + |M| < |M^*|$ **then return**
  foreach v *in* P **do** DFS($G, M \bigcup \{v\}, M^*, visited$)

V. Applications in multi-robot navigation

We demonstrate how WCS benefits prioritized multi-robot path planning (MRPP) on graphs. In a legal move, a robot may cross an edge if the edge is not used by another robot during the same move and the target vertex is not occupied by another robot at the end of the move. The task is to plan paths with legal moves for all robots to reach their respective goals. The makespan (the time for all robots to reach their goals) and the total arrival time are two common criteria to evaluate the solution quality. Previous studies [9], [10] have established the completeness of prioritized planning in well-formed infrastructures. Building on the foundation, we provide algorithms with completeness guarantees for non-well-formed environments.

Definition (Well-Formed MRPP)**.** *An* MRPP *instance is well-formed if, for any robot i, a path connects its start and goal without traversing any other robots' start or goal vertex.*

Theorem V.1. *Well-formed* MRPP *is solvable using prioritized planning with any total priority ordering [9], [10].*

When addressing non-well-formed MRPP, we adopt a simple, effective strategy similar to [5] to convert start and goal configurations to intermediate well-connected configurations so that the resulting problems are guaranteed to be solvable by prioritized planning with any total priority ordering. We call the algorithm UNPP- **un**labeled **p**rioritized **p**lanning.

We compute a MWCS/LWCS M offline. The first step in the algorithm is to assign the $2n$ vertices, $\mathcal{S}'', \mathcal{G}''$ in M as the intermediate start vertices and goal vertices. This is done by solving a min-cost matching problem using the Hungarian algorithm [17]. Collision-free paths are easy to plan in the unlabeled setting with optimality guarantees on makespan and total distance [18], [19]. The output of this function is a set of collision-free paths for the robots that route them to a well-formed configuration and the intermediate labeled starting and goal positions $\mathcal{S}', \mathcal{G}'$. The PrioritizedPlanning function is then called on the resulting intermediate starting and goal positions to generate a deadlock-free path for each robot. Finally, the paths generated by the Unlabeled Multi-Robot Path Planning and Prioritized Planning functions are concatenated to produce a final solution.

Algorithm 3: UNPP

Input: Starts $\mathcal{S}$, goals $\mathcal{G}$, LWCS/MWCS M
Function UNPP($\mathcal{S}, \mathcal{G}$)**:**
  $\mathcal{S}'', \mathcal{G}'' \leftarrow$ Assignment($\mathcal{S}, \mathcal{G}, M$)
  $P_s, \mathcal{S}' \leftarrow$ UnlabeledMRPP($\mathcal{S}, \mathcal{S}''$)
  $P_g, \mathcal{G}' \leftarrow$ UnlabeledMRPP($\mathcal{G}, \mathcal{G}''$)
  $P_m \leftarrow$ PrioritizedPlanning($\mathcal{S}', \mathcal{G}'$)
  $solution \leftarrow$ Concat(P_s, P_g, P_m)
  return $solution$

Theorem V.2. *Denote n_c as the size of the* LWCS *of graph G, for any* MRPP *instance with number of robots less than $n_c/2$, regardless of the distribution of starts and goals,* UNPP *is complete with respect to any priority ordering.*

Proof. When the number of robots $n \leq n_c/2$, it is always possible to select such $\mathcal{S}', \mathcal{G}'$ where $\mathcal{S}' \bigcap \mathcal{G}' = \emptyset$. Since $\mathcal{S}' \bigcup \mathcal{G}' \subseteq M$, by the definition of WCS, $\mathcal{S}' \bigcup \mathcal{G}'$ is also a WCS. Therefore, the resulting MRPP problem which requires routing robots from $\mathcal{S}'$ to $\mathcal{G}'$ is well-formed. By Theorem. V.1, prioritized planning is guaranteed to solve the subproblem

using any priority ordering. □

UNPP runs in polynomial time for MRPP instances with $n \leq n_c/2$. We can use the max-flow-based algorithm [18] to solve the unlabeled MRPP, which takes $O(n|E|D(G))$ where $D(G)$ is the diameter of the graph G, if we use [20] (faster max-flow algorithm can also be used here) to solve the max-flow problem. In the worst case, an unlabeled MRPP requires $n+|V|-1$ makespan to solve [19]. For the prioritized planning applied on well-formed instances, the makespan is upper bounded by $nD(G)$. As we use spatiotemporal A* to plan the individual paths while avoiding collisions with higher-priority robots on a time-expanded graph with edges no more than $n|E|D(G)$ and such a solution is guaranteed to exist, the worst time complexity is $O(n^2|E|D(G))$. In summary, UNPP yields worst case time complexity of $O(n^2|E|D(G))$ and its makespan is upper bounded by $2(n+|V|-1)+nD(G)$.

For $\frac{n_c}{2} < n < n_c$, arbitrary priority ordering does not guarantee a solution. For some robots, its intermediate start vertex in $\mathcal{S}'$ has to be the intermediate goal vertex in $\mathcal{G}'$ of another robot when assigned from the WCS M. The resulting subproblem is not well-formed. However, it is possible to solve such an instance by breaking it into several sub-problems and using specific priority ordering to solve it. To do this, we can first establish a dependency graph to determine the priority ordering of robots. The dependency graph consists n nodes representing the robots. If $s'_i = g'_j$, we add a directed edge from node i to node j, meaning that j should have higher priority than i. If the resulting dependency graph is a DAG, topological sort can be performed on it to get the priority ordering. When encountering a cycle, as $n < n_c$, we can break the cycle by moving one of the robots in this cycle to a buffer vertex in $C - \{\mathcal{S}' \bigcup \mathcal{G}'\}$ and perform the topological sort on the remaining robots.

Finally, we briefly illustrate the application of WCS in multi-robot pickup and delivery (MAPD) [8]. In well-formed MAPD, each robot can rest in a non-task endpoint forever without preventing other robots from going to their task endpoints, i.e. pickup stations. The layout of the endpoints forms a WCS of the graph of the environment. While it is desirable to increase the number of robots and the endpoints as many as possible to maximize space utilization, it is also important to keep a well-connected layout so that the robots will not block each other for better pathfinding. Thus, the maximum number of endpoints is equal to n_c, and at most $n_c - 1$ robots can be used in the MAPD.

VI. Experiments

In our evaluation, we first test algorithms that compute WCS for grids and a set of benchmark maps and then perform evaluations on MRPP problems. Since the grids and maps used in our experiments are either 4-connected or 8-connected, the solution we find without additional checks will always be larger than the number of neighbors of a vertex v plus one (i.e., $|N(v)|+1$). Therefore, we can safely omit the additional checks in our solution. All experiments are performed on an Intel® CoreTM i7-6900K CPU at 3.2GHz with 32GB RAM in Ubuntu 18.4 LTS and implemented in C++.

A. Grid Experiments

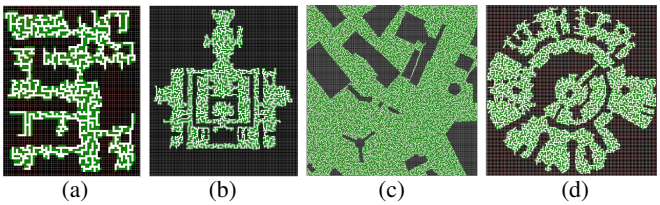
We test the algorithms on $m \times m$ 4/8-connected grids with varying side lengths m. The result is shown in Table I. In "Random" and "Greedy", we run the MWCS algorithm 50 times and return the set with the maximum size. Random randomly chooses a node from the candidates P and adds it to the set. In Greedy, to select the next candidate to add to the current WCS, we sort the candidate nodes in ascending order based on their total shortest distance from any node in the current WCS. We then select the candidate with the smallest total shortest distance as the next node to add to the WCS. To evaluate the running time of Random and Greedy, we take the average of the total execution time over the 50 runs of the algorithm. To evaluate the path efficiency of the maximum(maximal) WCS found by each algorithm, we treat each node in V as the reference point and compute the PER for each node. We then take the average of the PER values to obtain a measure of the overall path efficiency of the algorithms. In DFS, we set a time limit of 600 seconds to search for a solution. If the time limit is reached before a solution is found, we report the best solution found so far.

Though DFS only finds guaranteed optimal solutions on small grids, the final vertex size it returned is usually larger than the other two methods and has better PER. Greedy finds larger WCS on 4-connected grids than Random. Interestingly, on 8-connected grids, the MWCS found by Greedy is smaller. PER of Greedy is generally better than Random.

Through linear regression, we found that on grids, the size of the maximum(maximal) WCS $|M|$ founded by these algorithms is linearly related to the number of vertices $|V|$. Specifically, on 4-connected grids, $|M| \sim 0.63|V|$, and on 8-connected grids, $|M| \sim 0.72|V|$. This means that in a square parking lot (or other well-formed infrastructures), if it is considered a 4-connected grid, at most about 63% of the space can be used for parking.

B. Benchmark Maps

We select several maps from 2D Pathfinding Benchmarks [21]. Here, we use Greedy to compute the suboptimal LWCS as most of the maps are too large to perform DFS search. For maps that are not connected, the largest connected component is used. The result is presented in Table. II. And some examples are shown in Fig. 4. Our algorithm is efficient on large and complex maps with tens of thousands of vertices. The computed vertex set size is roughly 50%-60% of $|V|$ for 4-connected graphs, and 60%-70% for 8-connected graphs.



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

Fig. 4: Examples of the computed MWCS (colored in green) in different 4-connected grid maps. (a) den312d. (b) ht_chantry. (c) Shanghai_0_256. (d) lak503d.Zoom in on the digital version to see more details.

Side Len	Random$_4$			Greedy$_4$			DFS$_4$			Random$_8$			Greedy$_8$			DFS$_8$		
	$\|M\|$	PER	Time	$\|M\|$	PER	Time	$\|M\|$	PER	Time	$\|M\|$	PER	Time	$\|M\|$	PER	Time	$\|M\|$	PER	Time
5	14	0.89	0	11	0.89	0	14	0.89	17.79	20	0.97	0	20	0.93	0	20	0.97	336.2
10	55	0.60	0	52	0.69	0	**60**	0.67	600	72	0.52	0	73	0.96	0	**74**	0.85	600
20	220	0.73	0.07	238	0.74	0.2	**242**	0.67	600	283	0.46	0.1	280	0.82	0.3	**285**	0.85	600
30	487	0.53	0.4	561	0.62	2.0	**561**	0.68	600	**645**	0.47	0.7	622	0.84	2.3	642	0.54	600
40	872	0.47	1.4	992	0.72	9.8	**992**	0.70	600	1137	0.56	2.4	1097	0.79	11.0	**1139**	0.49	600
50	1360	0.33	3.9	1588	0.64	35.9	**1588**	0.66	600	1778	0.51	6.5	1705	0.82	39.8	**1785**	0.50	600

TABLE I: Grid Experiment on 4/8-connected square grids. The numbers in red color are optimal solutions found by DFS.

Map Name	**Map grid size**	**#Vertices $\|V\|$**	**#Edges $\|E_4\|$**	**Time$_4$(s)**	**$\|M_4\|$**	**PER$_4$**	**#Edges $\|E_8\|$**	**Time$_8$(s)**	**$\|M_8\|$**	**PER$_8$**
arena	49 × 49	2,054	3,955	2.33	1,113	0.68	7813	3.76	1455	0.52
brc202d	481 × 530	43,151	81,512	1,685	22,659	0.61	160,277	2,668	29,973	0.63
den001d	80 × 211	8,895	16,980	20.1	4859	0.77	33392	92.55	6233	0.51
den020d	118 × 89	3102	0.12	4.48	1599	0.89	10869	9.04	2104	0.699
den312d	81 × 65	2,445	4,391	3.18	1,247	0.701	8,464	5.11	1,663	0.708
hrt002d	50 × 49	754	1300	0.24	377	0.87	2489	0.38	510	0.68
ht_chantry	141 × 162	7,461	13,963	38.87	3,889	0.45	27222	60.95	5183	0.37
lak103d	49 × 49	859	1509	0.32	438	0.84	2869	0.51	584	0.58
lak503d	194 × 194	17,953	33,781	258.89	9484	0.58	66,734	415.60	12,482	0.48
lt_warehouse	130 × 194	5,534	10,397	18.67	2,895	0.87	20306	31.72	3858	0.63
NewYork_0_256	256 × 256	48299	94068	2000	26025	0.40	186935	3469	34054	0.35
orz201d	45 × 47	745	1342	0.23	389	0.73	2604	0.39	513	0.61
ost003d	194 × 194	13,214	24,999	131.82	7,004	0.88	49,437	206.30	9,221	0.59
random-32-32-20	32 × 32	819	1270	0.26	375	0.66	2,487	0.44	533	0.55
Shanghai_0_256	256 × 256	48,708	95,649	2,119	26,453	0.32	190,581	3,542	34,501	0.29

TABLE II: Computed suboptimal LWCS size in selected 4/8-connected maps.

C. *Evaluations of* MRPP

Lastly, we examine the effectiveness of our proposed MRPP method on selected benchmarks. We compare our proposed method with two other prioritized planners, HCA* [14] and PIBT [22]. For each map, a maximal vertex set is precomputed using the Greedy method. To conduct the experiments, we randomly generate 50 instances for each map and the number of robots n. The results are shown in Fig. 5-6. The experimental results demonstrate that our proposed method significantly improves the success rate compared to HCA* and PIBT. Furthermore, although the solution quality is not optimal, it is still reasonably good.

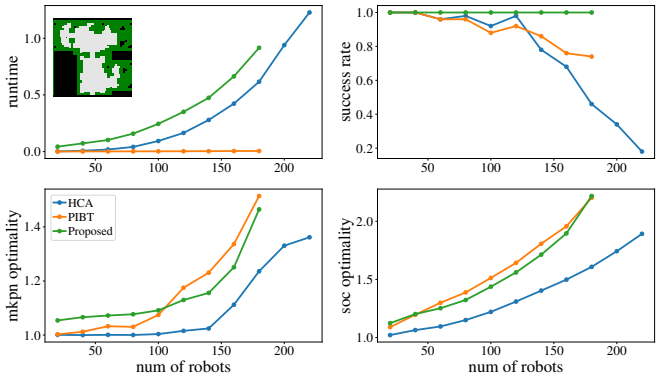


Fig. 5: Experimental results for map orz201d, including computation time, success rate, makespan optimality, and soc optimality, for HCA, PIBT, and the proposed method.

VII. Conclusions

In this paper, we have presented a comprehensive study of the LWCS problem and its applications in multi-robot path planning. We provided a rigorous problem formulation and developed two algorithms, an exact optimal algorithm

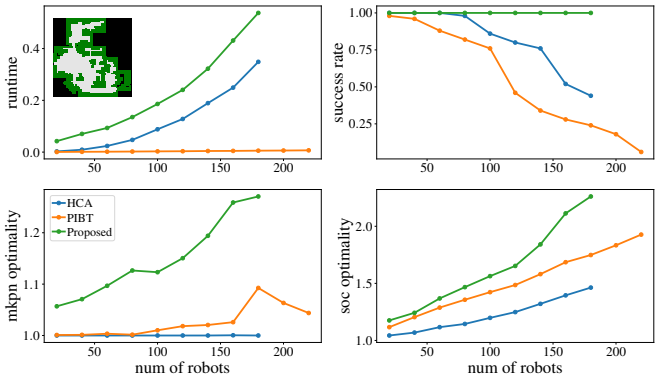


Fig. 6: Experimental results for map hrt002d, including computation time, success rate, makespan optimality, and soc optimality, for HCA, PIBT, and the proposed method.

and a suboptimal algorithm, to solve the problem efficiently. We have shown that the problem has various real-world applications, such as parking and storage systems, multi-robot coordination, and path planning. Our algorithms have been evaluated on various maps to demonstrate their effectiveness in finding solutions. Moreover, we have integrated the LWCS problem with prioritized planning to plan paths for multi-robot systems without encountering deadlocks. Our study enhances comprehension of the relationship between multi-robot path planning complexity, the number of robots, and graph topology, laying a robust groundwork for future research in this domain. In future work, we plan to investigate the performance of our algorithms in more complex environments and explore their scalability in solving larger instances of the problem. Additionally, we aim to explore the potential of our algorithms in real-world applications and examine their robustness against uncertainties and disruptions.

REFERENCES

[1] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI magazine*, vol. 29, no. 1, pp. 9–9, 2008.

[2] T. Guo and J. Yu, "Toward efficient physical and algorithmic design of automated garages," *arXiv preprint arXiv:2302.01305*, 2023.

[3] K. Azadeh, R. De Koster, and D. Roy, "Robotized and automated warehouse systems: Review and recent developments," *Transportation Science*, vol. 53, no. 4, pp. 917–945, 2019.

[4] F. Caron, G. Marchet, and A. Perego, "Optimal layout in low-level picker-to-part systems," *International Journal of Production Research*, vol. 38, no. 1, pp. 101–117, 2000.

[5] T. Guo and J. Yu, "Sub-1.5 Time-Optimal Multi-Robot Path Planning on Grids in Polynomial Time," in *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.

[6] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, 2013.

[7] W. Sheng, Q. Yang, J. Tan, and N. Xi, "Distributed multi-robot coordination in area exploration," *Robotics and autonomous systems*, vol. 54, no. 12, pp. 945–955, 2006.

[8] H. Ma, J. Li, T. K. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding for online pickup and delivery tasks," in *AAMAS*, 2017.

[9] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 7643–7650.

[10] M. Čáp, J. Vokřínek, and A. Kleiner, "Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures," in *Twenty-Fifth International Conference on Automated Planning and Scheduling*, 2015.

[11] P. Surynek, "An optimization variant of multi-robot path planning is intractable," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, no. 1, 2010.

[12] J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[13] M. A. Erdmann and T. Lozano-Pérez, "On multiple moving objects," *Algorithmica*, vol. 2, no. 4, pp. 477–521, 1987.

[14] D. Silver, "Cooperative pathfinding." *Aiide*, vol. 1, pp. 117–122, 2005.

[15] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[16] R. E. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.

[17] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[18] J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," in *Algorithmic foundations of robotics X*. Springer, 2013, pp. 157–173.

[19] J. Yu and M. LaValle, "Distance optimal formation control on graphs with a tight convergence time guarantee," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 4023–4028.

[20] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian journal of Mathematics*, vol. 8, pp. 399–404, 1956.

[21] N. Sturtevant, "Benchmarks for grid-based pathfinding," *Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 144 – 148, 2012. [Online]. Available: http://web.cs.du.edu/~sturtevant/papers/benchmarks.pdf

[22] K. Okumura, M. Machida, X. Défago, and Y. Tamura, "Priority inheritance with backtracking for iterative multi-agent path finding," in *IJCAI*, 2019.