An Improved Maximal Continuity Graph Solver for Non-repetitive Manipulator Coverage Path Planning

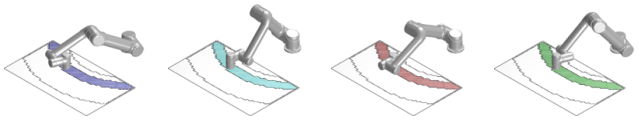
Tong Yang, Jaime Valls Miro, Yue Wang* and Rong Xiong

***Abstract*—A provable computational improvement to the problem of maximal continuity during non-repetitive object coverage with non-redundant manipulators is proposed in this work, where the physical meaning of optimality translates to the minimal number of end-effector lift-offs. Existing solutions enumerate each point on the surface with multiple "colours" according to the joint-configuration adopted, and model the problem as a painting problem of a graph with M topological cells (a fully-connected section of end-effector points that can be painted with the same set of possible colours) and N topological edges (indicating the different choices of colour available). These works have proven that all optimal solutions can be collected in a finite number of steps. However, the solution grows exponentially in the size of M, N, becoming potentially intractable even for relatively simple graphs. The proposed solution aims to avoid the need to enumerate all the edges by exploiting a topological invariance observed at colour intersections: the solution of an intersection-free graph can be uniquely represented by the colour of its boundary cells (in order), while enumerating internal edges and cells are shown to make no difference to the optimality of the solution, and can be safely omitted. A novel strategy is thus proposed to separate the graph into intersection-free sub-graphs. After enumerating and combining the solutions to each sub-graph to form the set of all optimal solutions, the complexity is proven to be reduced by a factor of 2^N. Challenging scenarios are presented to validate the computational advantage of the proposed strategy. An open source implementation has also been provided for the benefit of the robotics community.**

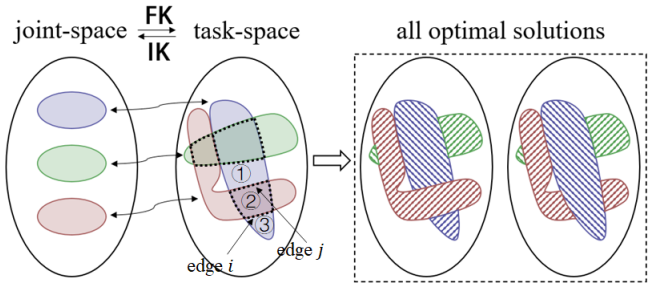
***Index Terms*—Optimal Cellular Decomposition, Manipulator Coverage Task**

I. Introduction

The work in this manuscript is motivated by the *non-revisiting coverage path planning* (NCPP) along the surface of an object [1] [2] with a non-redundant manipulator, as illustrated by Fig. 1(a). NCPP can be paramount in practice to avoid the undesirable effects of over-coverage, such as excessive removal of surface material form repetitive inspection [3] [4], or the formation of fatigue cracks incurred by invasive handling operations such as water jetting [5]. However, coming up with such paths for a manipulator end-effector to track is non-trivial: a continuous task-space coverage path may not admit a continuous, non-singular, collision-free inverse kinematic solution. Likewise, the opposite is also true: a continuous joint-space path may have overlapped task-space images under the nonlinear manipulator kinematics.



(a) Coverage illustration of a planar rectangular region with a non-redundant manipulator, depicting the four possible configurations that the manipulator may adopt to do so: s(houlder)-l(eft)&w(rist)-f(lipped), s-r(ight)&w-u(nflipped), s-r&w-f, and s-l&w-u. A transition between any pair imposes the adoption of a singular configuration thus preventing continuous traversability and a manipulator lift-off from the surface. By representing disconnectable manipulator configurations as different colours, an abstract topological graph with task-space colour distributions can be constructed.

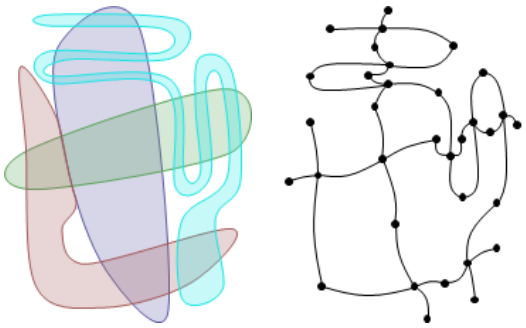


(b) Valid robot configurations form disjoint sets in joint-space, while their task-space FK mapping images (end-effector poses) may overlap. In the example, which for simplicity is illustrated with cells that are maximally 2-overlapped, the resulting task-space graph conveys $N = 11$ undetermined edges constructed as shown by the dashed lines, thus separating the graph into $M = 11$ cells. To facilitate the understanding whilst limiting clutter in the graph, 3 cells and 2 edges (i and j in dashed blue-green and blue-red respectively) are singled out. Solving the NCPP problem for the set of edges in the example can easily reveal all the optimal solutions in this case (two, depicted), demarcating the minimal bound for lift-offs as four. It is easy to appreciate how as more colours "intersect", the algorithmic complexity to find the optimal solutions will soon escalate. It is also intuitive that should the green colour remain a complete set, both the red and blue colour cells would be split into further disconnected parts, leading to extra lift-offs. This hints at the fact how to optimally resolve the "intersecting" of colours can be exploited to reduce the complexity of the solution.

Fig. 1. A toy illustration of the problem arising when planning optimal non-revisiting coverage paths with a manipulator: (a) non-revisiting coverage of an object is dictated by the available configurations the manipulator can adopt. This leads to (b) disjoint sets in task-space, which conform the input to the NCPP problem. The optimal slicing of these sets leads to minimal discontinuities in tracing the surface with the end-effector. (Please note the above separate examples are independently chosen to best illustrate the problem and do not correspond to each other).

An illustration of these challenges is depicted by the mapping example provided in Fig. 1(b), left. Given the object's surface, the manipulator, the surrounding obstacles in the workcell and their related poses, for each reachable point on the surface there exist a finite number of inverse kinematic solutions in the manipulator configuration space. Disregarding singular configurations [6], the collection of all valid configu-

[1] Tong Yang, Yue Wang and Rong Xiong are with the State Key Laboratory of Industrial Control and Technology, Zhejiang University, P.R. China.
[2] Jaime Valls Miro is with the Robotics Institute at the University of Technology Sydney (UTS:RI), Sydney, Australia.
* Corresponding Author.
E-mail address: wangyue@iipc.zju.edu.cn



(a) a cell-graph to be solved (b) the node-graph

Fig. 2. Quantitive relationship between the cell-graph notation adopted in the manipulator coverage task and the traditional node-graph concepts in classic graph theory. In this example there are 33 vertices, 39 edges, and 8 faces (7 cycles + 1 outer region reaching the infinity) in the node-graph, which correspond to 33 cells and 39 edges in the cell-graph.

rations form disjoint sets in joint-space. When the manipulator transits from one set to another, it adopts a new pose altogether whereby the end-effector (EE) cannot remain on the object, being forcibly detached from the surface. This is an undesirable effect for tasks where smoothness and continuity are critical, such as painting [7], deburring [8], welding [9], scanning [10] [11], etc. In these instances, the desired solution translates to designing manipulator trajectories such that the end-effector (EE) visits each point on the surface exactly one time, whilst ensuring a minimum number of transitions between joint-space sets, or lift-offs.

A. Contributions

Careful examination of the NCPP problem [12] has given birth to a topological graph representation where choosing the optimal manipulator IK solutions for the NCPP task has been transformed into choosing the optimal colours for graph painting. Further modelling of the overlap of colours leads in this work to the introduction of a higher-level abstraction, the topological intersection which is the origin of the multiplicity of the optimal solutions. See Fig. 1(b) for an intuitive analysis of the phenomenon. It is hereby proven that separating the graph into intersection-free sub-graphs leads to all intersections being implicitly enumerated when the sub-graphs are recombined later to calculate the final solutions. The contributions of this paper can be summarised as follows:

1) We propose a novel solver for the NCCP problem, hereby referred to as the Improved Maximal Continuity Graph Solver (IMCGS). The algorithmic complexity is proven to be reduced by a factor of 2^N with respect to the state-of-the-art approach in [12], where N is the number of internal edges in the initial topological graph.
2) Complexity is substantially enhanced with no trade-offs in the optimality of the solution: the improved graph solver remains globally optimal and complete, thus collecting all optimal solutions as in [12]).
3) Whilst the improved graph solver is laid out assuming that all cells in the graph are simply-connected, it has already been proven [13] that for graphs with multiply-connected cells, iterative enumeration can reduce the problem to that of a simply-connected cell graph. Hence the proposed algorithmic improvement is generic to any type of topological graphs.
4) An open source implementation has been supplied, where a supplementary video is also included [1].

B. Significance of Contribution

The significance of the algorithmic improvement can also be revealed by invoking the cell-graph notation promoted in this work and the traditional concept of node-graph in classic graph theory. This equivalence is pictorially shown in Fig. 2, whereby a one-to-one correspondence between the cells and edges in a cell-graph and the vertices and edges in a node-graph can be established. The Euler's formula [14] for planar graphs yields:

$$V - E + F = 2 \tag{1}$$

where V, E, and F are the number of vertices, edges, and faces in the node-graph, respectively. This relationship highlights the large number of edges in a cell-graph, $F - 2$ greater than the number of cells:

$$N - M = E - V = F - 2 \tag{2}$$

It can be observed how in complicated graphs where F will be necessarily large, reducing algorithmic complexity by 2^N represents a notable breakthrough. This relationship to classic graph theory also points towards a potential contribution in other related research areas besides the manipulator NCPP problem, which is beyond the scope of this work.

The remainder of this paper is organised as follows. Section II reviews existing literature. Section III formally states the non-repetitive manipulator coverage problem, with a revisiting of state-of-the-art graph solver in Section IV. Section V introduces the concept of *intersection-free* property. Section VI describes concrete steps to separate a graph into intersection-free sub-graphs. Solutions to these can then be combined to construct the optimal solutions for the full graph. Details about the complexity advantage in solving the problem following the proposed strategy are mathematically proven in Section VII, whilst experimental results from simulations are collected in Section VIII. Final concluding remarks are gathered in Section IX.

II. Related Works

Early reports on the generic *coverage path planning* (CPP) problem focused on geometric path designing [15] [16], particularly for mobile platforms operating in planar surfaces or near-flat terrains [17] [18], such as boustrophedon [19] or spiral paths [5]. Additional strategies were later proposed that transformed the coverable region into smaller partitions, or *cells*, where continuous coverage paths could be guaranteed. A body of novel partitioning *cellular decomposition* strategies emerged [20] [21] [22] [23] [24], applied directly to the area to be covered. This is an ineffective strategy when transferred from task to joint-space for manipulator planning since the kinematic mapping between the two spaces is non-bijective:

[1] https://github.com/ZJUTongYang/imcgs

as illustrated in Fig. 1, the forward kinematic relationship from the configuration space to the surface is surjective (many-to-one) and locally flat (one-to-one from each connected component of the valid configuration space to the surface). The problem is further compounded when the EE can only visit each point on task-space once, leading to numerous pose reconfigurations during the motion of the EE [25]. Many criteria in the classic CPP problems have been adapted to the manipulator NCPP task, such as time to completion [26] or energy consumption [27], and optimal mobile manipulator pose for a given coverage in task-space has also been investigated [28] [29] [30]. However, minimising EE lift-offs to decrease the numbers of reconfigurations adopted during a desired task motion has rarely been exploited. Specifically for coverage with multi-axis mechanisms, almost all existing works [31] [32] [33] have neglected the problem and assumed that the target surface is relatively small so as to be entirely embedded into the reachable workcell of a manipulator, which does not generalise well, e.g. for large workpieces. Recent propositions in the literature to tackle the problem decomposed the task-space area into a topological graph ensuring continuous joint-space coverage within each cell, and looked for solutions where maximal continuity between cells existed [12], as intuitively depicted in Fig. 1. For these cases, all maximally-continuous cellular decompositions were proven to be collectable in a finite number of steps - a relevant observation for this work that will be recalled later in Section IV.

III. Problem Statement

The NCPP task is a generalisation of the classic coverage path planning (CPP) problem, in that for each point in the region to be covered there may be not one but a finite number of valid inverse kinematic solutions to perform the coverage.

Definition 1. *(Valid Configuration) A valid configuration is the non-singular [6] collision-free manipulator configuration such that the end-effector rests on the object surface with its orientation parallel to the surface normal.*

Given the kinematics of a non-redundant manipulator K, the pose and surface mesh of the object to be traversed M, the obstacles within the robot workspace in the surrounding environment $\{O_i\}$, each point $m \in M$ on the surface can be reached by the EE through a finite number of Inverse Kinematic (IK) solutions $\mathscr{C}(m) = \{c_{m1}, \cdots, c_{mn}\}$. *Solving* the graph formally refers to the process of preserving only one of the IK solutions $c_m \in \mathscr{C}(m)$ for each point. Traversing between discontinuous configurations calls for an EE lift-off for manipulator pose reconfiguration. An optimal solution to the NCPP problem in one that admits the minimum number of EE lift-offs during the coverage motion.

IV. Existing Algorithm

A. Definitions

Given the NCPP problem to be solved, the tuple $(K, \{O_i\}, M)$, the following topological elements are well-defined:

Definition 2. *(Colour) Each valid manipulator configuration is represented by a colour, with continuous configurations being assigned the same colour.*

When singular configurations are disregarded, the forward kinematics mapping from manipulator configuration to end-effector pose space is locally one-to-one. Thus, assuming that there is a one-to-one correspondence between an end-effector pose and a point on the surface, the colour of a point on the surface represents the corresponding IK solutions with said colour. Early work [12] grouped the points with the same set of possible colours into sub-regions to simplify the problem, namely, cells.

Definition 3. *(Cell) A cell C_1 (where 1 represents its index) is a maximally-connected region on the surface, whereby all points within have the same set of possible colours, i.e.,*

$$\begin{aligned} &\forall m, m' \in C_1, m \text{ and } m' \text{ are task-space continuous} \\ \Leftrightarrow &\begin{cases} \mathscr{C}(m) \triangleq \{c_{m1}, \cdots, c_{mn}\} \\ \mathscr{C}(m') \triangleq \{c_{m'1}, \cdots, c_{m'n'}\} \\ n = n' \\ \forall c_{mi} \in \mathscr{C}(m),\ \exists c_{m'j} \in \mathscr{C}(m') \\ \qquad s.t.\ c_{mi} \text{ and } c_{m'j} \text{ are joint-space continuous} \end{cases} \end{aligned} \tag{3}$$

The possible colours of a cell is the possible colours of points in the cell.

The geometric coverage path within a single cell (e.g., boustrophedon [19], spiral path [5]) can be arbitrarily designed with a continuity guarantee, such that when starting from any configuration of choice within the cell, the manipulator will be able to track the chosen coverage path without the need to resort to any end-effector lift-off. An edge is delineated by the common boundary between two cells, formally defined as

Definition 4. *(Edge) An edge E_1 (where 1 represents its index) is the maximal continuous task-space curve amidst two cells, i.e., within an arbitrary small neighbourhood of the point on edge we may find points belonging to different cells,*

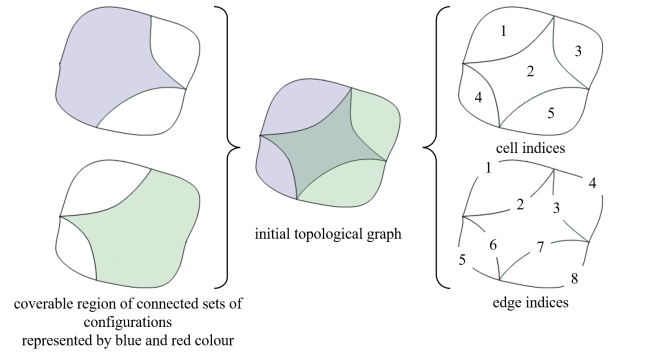
$$\begin{aligned} E_1 &= E_1(C_i, C_j) \\ &= \{p \in M | \forall (p \in) U_p, \exists m_1, m_2 \in U_p, m_1 \in C_i, m_2 \in C_j\} \end{aligned} \tag{4}$$

Finally, the initial topological graph can be constructed by collecting all cells and edges.

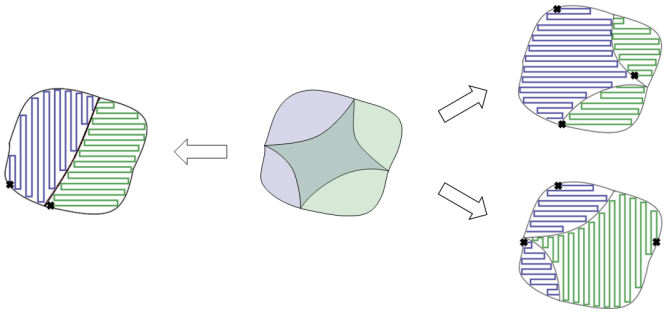
Definition 5. *(Graph) A graph is the combination of all cells and their connectivity, i.e., the edges.*

Earlier work [12] has proven that the number of cells in the initial topological graph is finite.

A visual illustration of a cell and its subdivision is supplied in the example shown in Fig. 3. The cell representation in this case can be characterised as



(a)



(b)

Fig. 3. (a) A simple example to illustrate various cell concepts, where the central cell is the only cell with non-unique possible colours. (b) Just filling in the central cell with either blue or green colour would be non-optimal (right-hand side examples). The resulting solutions will have 3 connected regions, i.e., 2 extra EE lift-offs. In contrast, dividing the cell into two parts (left-hand side) enables it to connect two pairs of its adjacent cells, leading to an optimal cellular decomposition with 2 regions (thus only 1 extra EE lift-off). Examples of coverage paths have been illustrated inside each of the final cells, with bold × denoting the places where the EE would leave the object surface.

$$
\begin{array}{ll}
\text{Cell: } \{\text{colours}, \text{edges}\} & \text{Edge: } \{\text{cells}, \text{constraint}\} \\
C_1 : \{[b], [1,2]\} & E_1 : \{[-1,1], \text{''}\} \quad E_5 : \{[-1,4], \text{''}\} \\
C_2 : \{[b,g], [3,2,6,7]\} & E_2 : \{[1,2], \text{''}\} \quad E_6 : \{[2,4], \text{''}\} \\
C_3 : \{[g], [4,3]\} & E_3 : \{[2,3], \text{''}\} \quad E_7 : \{[2,5], \text{''}\} \\
C_4 : \{[b], [5,6]\} & E_4 : \{[-1,3], \text{''}\} \quad E_8 : \{[-1,5], \text{''}\} \\
C_5 : \{[g], [7,8]\} &
\end{array} \tag{5}
$$

where the edges connecting to -1 are the boundaries of the graph. In the initial topological graph, there were no constraints imposed on the edges, so the set is empty (shown in single quotes).

B. Cell Sub-division and Enumeration

Unlike the usual setting in classic graph theory where nodes are regarded as inseparable entities, cells in this work can be arbitrarily divided into sub-cells (and then painted with different possible colours) to be able to arrive at a better solution. Fig. 3(b) shows one such example. We refer to the *cell sub-division* process as the designing of cutting paths within a cell.

Definition 6. *(Cutting Path) The cutting path of a cell is a path starting and ending at the boundary of a cell, enforcing its sides having different colours. Taking sub-cells as distinct cells, the cutting path is equivalent to a manually created edge.*

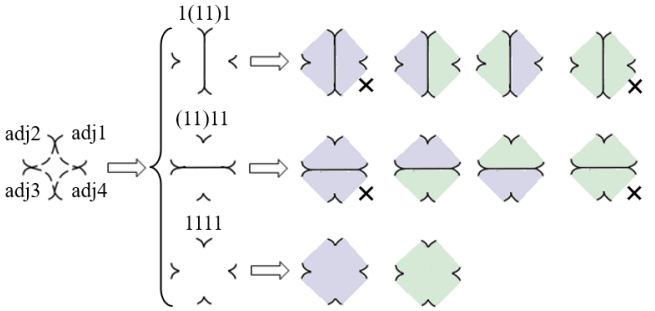


Fig. 4. An illustration of the many-to-one relation between edge and painting solutions. Let a cell with 4 adjacent cells (a 4-edge cell) have 2 possible colours "blue" and "green", then $\alpha = 4, K = 2$. From the point of view of solving edges, the binary number 1111 specifies one solution, but the corresponding valid painting solutions are multiple, 6 to be precise: there can be no cell subdivisions, or $E = 2$ ways to divide the multi-edge cell into 3-edge sub-cells, and $\alpha - 2 = 2$ sub-cells are generated which can be filled with the $K(=2)$ possible colours. After enumerating $E \cdot K^{\max\{\alpha-2,1\}}$ cases, four of them are removed (adjacent sub-cells are separated by cutting paths thus cannot be painted by the same colour), shown crossed-out, leaving the remaining six valid painting solutions.

Moreover, the following lemma ensures the finiteness of all possible topological-distinct cellular decompositions.

Lemma 7. *The following principles for designing cutting paths in a cell have been proven in [12]:*

1) *We only need to consider cutting paths which connect edge endpoints.*
2) *We only need to consider cutting paths which go across edges.*
3) *We only need to consider intersecting cutting paths.*

A finite-step graph solver has been proposed to collect all optimal solutions. In each iteration, for an unpainted cell (say it has α edges and K possible colours), we firstly enumerate all its possible cell sub-divisions (say there are $E = E(\alpha)$ ways) which construct different branches. It has been shown how a binary array of length α can be used to index all possible different cell sub-divisions [12], whereby 0 at a given position enforces the cell having a different colour to its adjacent cell, whilst 1 compels the same choice of colour. However, as supported by the illustrative example shown in Fig. 4, for one of the edge solutions "1111" and 2 possible colours available for painting, the binary array does not uniquely represent a valid painting solution: the cell may be divided into parts with differing colours, while each part themselves connect to their two adjacent cells. Hence, in generic, there may be up to $E(\alpha) \cdot 2^{\alpha} \cdot K^{\max\{\alpha-2,1\}}$ different branches after solving an α-edge cell with K possible colours. In the next iteration, an unsolved branch is selected, and an unsolved cell within it, and repeat the same process.

An example for this operation in the optimal graph solution

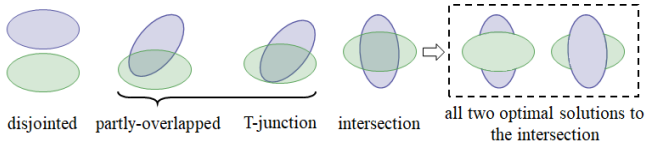


Fig. 5. Illustration of the basic relationships between the reachable area for a two-colour case. In all but the intersection scenario, each colour can keep itself fully connected. T-junction can be seen as a special instance of the generic partly-overlapped case, as it can be transformed back to the generic case through continuously modifying the boundary of the colours. Only the intersection case needs solving, and it is apparent that the two optimal solutions are not topological equivalent: they have a different number of connected regions for each colour, depending on whether the blue or the green colour is chosen in the intersecting area.

shown in Fig. 3(b) can be listed as follows:

$$
\begin{aligned}
&\text{(only changed elements are listed)}\\
&C_2:\{[b,g],[3,2,6,7]\} \quad E_2:\{[1,2],\text{`'}\} \quad E_6:\{[2,4],\text{`'}\}\\
&\Downarrow \text{(cell subdivision)} \Rightarrow \text{(other branches)}\cdots\\
&C_2:\{[b,g],[3,9,7]\} \quad E_2:\{[1,6],\text{`'}\} \quad E_6:\{[4,6],\text{`'}\}\\
&C_6:\{[b,g],[2,6,9]\} \quad E_9:\{[2,6],\text{`0'}\}\\
&\Downarrow \text{(tree pruning, the adjacent cells must be connected)}\\
&C_2:\{[b,g],[3,9,7]\} \quad E_2:\{[1,6],\text{`1'}\} \quad E_6:\{[4,6],\text{`1'}\}\\
&C_6:\{[b,g],[2,6,9]\} \quad E_9:\{[2,6],\text{`0'}\}\\
&\Downarrow \text{(tree pruning, the colour of sub-cell is constrained)}\\
&C_2:\{[b,g],[3,9,7]\} \quad E_2:\{[1,6],\text{`1'}\} \quad E_6:\{[4,6],\text{`1'}\}\\
&C_6:\{[b],[2,6,9]\} \quad E_9:\{[2,6],\text{`0'}\}\\
&\Downarrow \text{(tree pruning, sub-cells cannot have the same colour)}\\
&C_2:\{[g],[3,9,7]\} \quad E_2:\{[1,6],\text{`1'}\} \quad E_6:\{[4,6],\text{`1'}\}\\
&C_6:\{[b],[2,6,9]\} \quad E_9:\{[2,6],\text{`0'}\}\\
&\text{(all cells have a unique color,}\\
&\text{there are 2 connected regions)}
\end{aligned}
\tag{6}
$$

C. Algorithmic Complexity

Let there be M topological cells and N internal topological edges in the modelled graph to be solved. The number of edges for the i-th cell is α_i, and denote the number of possible colours to fill cell i as K_i. Considering a worst case scenario, let there be E_i different ways to divide cell i into $(\alpha_i - 2)$ sub-cells, then $E_i \cdot K_i^{\max\{\alpha_i-2,1\}} \cdot 2^{\alpha_i}$ steps are required for a single cell. Iteratively enumerating each cell and noting that each edge will only be enumerated once in the process, the upper bound of algorithmic complexity can be calculated as

$$
\prod_{\substack{i=1\\ \alpha_i \geq 4}}^{M} E_i \cdot \prod_{i=1}^{M} K_i^{\max\{\alpha_i-2,1\}} \cdot 2^N \tag{7}
$$

V. Proposed Method: Optimality via "Topological Intersections"

To construct the globally optimal painting schemes of the topological graph, one will naturally consider how the solution of a part in the graph contributes to that of the entire graph. Typically, for the simplest case, the coverable area of two colours may be fully disjointed, partly overlapped, or intersected, as shown in Fig. 5. There are two obvious optimal solutions to the right-most intersected case: filling in the central cell fully with either blue or green. It can be observed how colours can be kept connected in the disjointed and partly overlapped cases, but critically in the intersection case one colour being connected will truncate the other colour in two parts. Intuitively, having all other parts of the graph been already painted, the remaining part of the graph should be painted such that it maximally "connects" its adjacent cells. In this regard, the concept of intersection-free graphs plays a crucial role.

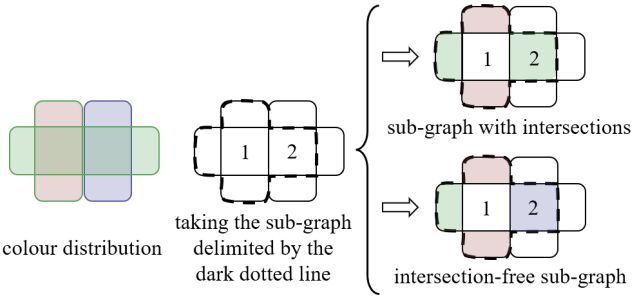


Fig. 6. Illustration of intersection-free sub-graph. By definition, the intersection-free property is subject to the colouring of boundary cells. In this example, taking the sub-graph delimited by the dark dotted line, if cell 2 is assigned as blue, then the sub-graph is intersection-free, as cell 1 being filling in with red is the only optimal solution. Or else, with cell 2 being in the green colour, either green or red in cell 1 will constitute an optimal solution of the sub-graph, where by the sub-graph is not intersection-free.

A. Intersection-free Sub-graphs

Definition 8. *(Sub-Graph) A sub-graph is a connected subset of cells in the graph, together with all edges that connect these cells. The adjacent cells of the sub-graph are the cells adjacent to any cell within the sub-graph. The boundary of a sub-graph is the set of edges between cells inside/outside the sub-graph, listed in order.*

To simplify the discussion, all sub-graphs in the paper are assumed simply-connected, hence the adjacent cells of a sub-graph are well-ordered, and the boundary of a sub-graph is a close chain of edges.

Definition 9. *(Boundary Cell, Internal Cell) Boundary cells are the cells which have edges forming the boundary of the sub-graph. The remaining cells are internal cells which have no edge exposed to the sub-graph boundary.*

We formally define the *intersection-free* property of a sub-graph by the peculiarity that, subject to a specified colour assignment of its boundary cells, the optimal solution for its internal cells is unique.

Definition 10. *(Intersection-free Sub-graph) Let a sub-graph have r boundary cells and s internal cells, indexed as $1,\cdots,r,r+1,\cdots,r+s$ and having $K_1,\cdots,K_{r+s}$ possible colours. A sub-graph can be distinguished as an intersection-free sub-graph only if*

1) *$K_1 = \cdots = K_r = 1$.*
2) *There exists only one topology-distinct optimal solution among all enumerations of internal cells $r+1,\cdots,r+s$ for the sub-graph.*

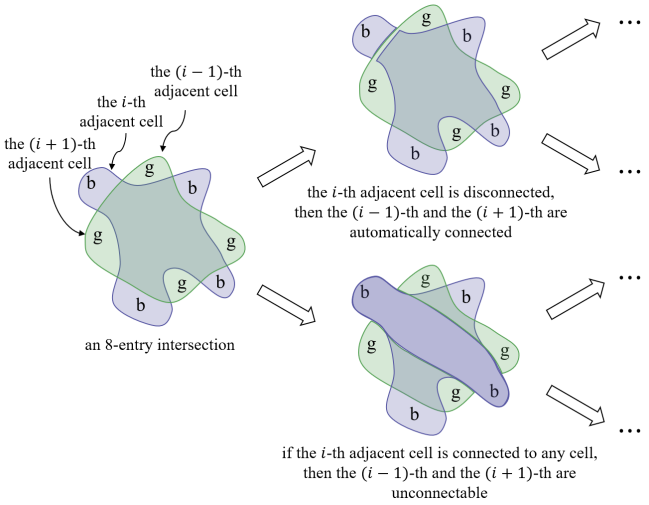


Fig. 7. An example of intersection with 8 adjacent cells is depicted on the left. The figure also illustrates the process of solving an intersection with a simpler example where adjacent cells have been assigned alternate colours "b(lue), g(reen), b, g, ···". It is apparent from the two token scenarios shown that an exhaustive enumeration will be required to obtain all optimal cellular decompositions. More generically, let the intersection have an even number (say n) of adjacent cells, with consecutive adjacent cells having been assigned alternate colours, "b(lue), g(reen), b, g, ···". It is observed that for each adjacent cell (say the i-th, labelled cyclically), if it is connected to any other adjacent cell through the intersection, then another two adjacent cells, $(i-1)$ and $(i+1)$, cannot be connected. So the minimal number of connected regions is $\frac{n}{2}+1$. However, note that whilst the optimal number can be directly deduced, to get all optimal cellular decompositions the intersection still needs to be enumeratively solved. For the i-th adjacent cell, two scenarios must then be separately regarded: (a) keeping itself disconnected so that adjacent cells $(i-1)$ and $(i+1)$ can be connected (top-right), or (b) connecting it to other adjacent cells (bottom-right). Both of them go to optimal cellular decompositions and they are topologically distinct.

The concept of intersection-free sub-graphs is illustrated in Fig. 6. The underlying reason for the multiplicity of optimal solutions is the different ways that a sub-graph may connect its adjacent cells, even with equal number of connections. As different adjacent cells become connected and disconnected through the sub-graph, the number of connected regions of each colour differs. Hence, these optimal solutions are topologically distinct and have to be collected by enumeration.

In fact, all sub-graphs with intersections have to be enumerated in order to identify all optimal solutions. Refer to the illustration in Fig. 7 where all enumerations of the internal cell may constitute an optimal solution, the illustration in Fig. 8 where the optimal of sub-graph has no relation to the global optimal solution of the whole graph, and the illustration in Fig. 9 where all possible combinations of all enumerations of cells form an optimal solution of the whole graph, by which the necessity of enumeration is indicated.

VI. Proposed Method: Graph Pre-separation

It has been discussed how disregarding the position of intersections and how to solve them leads to exhaustive implicit enumeration when all edges and cells are enumerated. On the other hand, exploiting the interesting complex scenarios that analysing the topology between different colours offer will attenuate this problem. An efficient mechanism called IMCGS to separate a graph into intersection-free sub-graphs

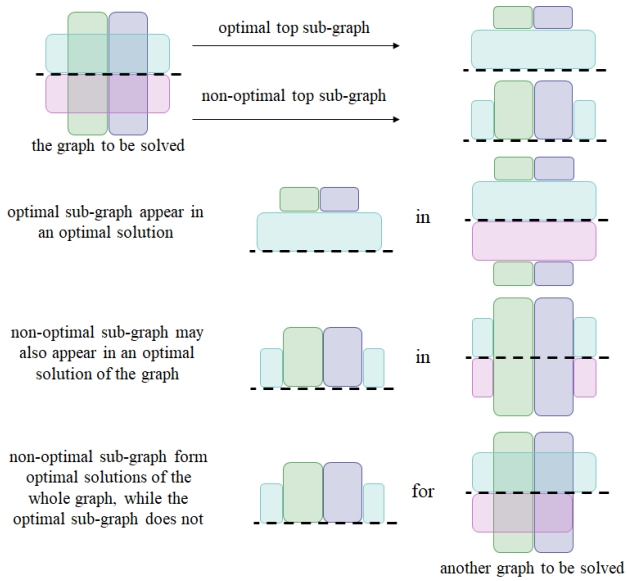


Fig. 8. Separating the graph through the dashed line, we show the optimal solution and a non-optimal solution of the sub-graph. It is noticeable that both of them form optimal solutions of the original graph. An in the bottom case, as a sub-graph of another graph, the non-optimal sub-graph can form optimal solutions while the optimal sub-graph cannot do so.

is described in this section. The reader is referred to the algorithm pseudocode given in **Algorithm** 1.

A. Definition of Strips

Definition 11. *(Strip) A strip is a sub-graph designed such that*

1) *Each internal cell has at most* 3 *connectable adjacent cells.*
2) *Internal cells are non-adjacent or disconnectable.*

Proposition 12. *Given any possible colour of the boundary cells of a strip, the strip is an intersection-free sub-graph.*

Proof. It is observed that for a sub-graph with intersections, its internal cell must be able to be assigned colours such that different pairs of painted adjacent cells can be connected. In other words, it needs to have at least 4 connectable adjacent cells. Then its opposite becomes a sufficient condition for distinguishing intersection-free sub-graphs: If condition 1) in **Definition** 11 is satisfied, then there can not possibly be multiple optimal choices of colours. In noting that two internal 3-edge cells may form a 4-edge cell which breaks this criterion, condition 2) is required. □

Now that the optimal solution for the internal cell of an intersection-free sub-graph is unique, finding it is easy: choosing for each internal cell the colour that maximally connects its adjacent cells. See Fig. 10 as an example. In this case, if internal cell 1 can be painted in blue, then it is optimal to be assigned as blue. Otherwise, if it can be painted in green, then green. Finally, any other colour if there is can also be chosen. This solution is simply represented by the priority of choice: blue > green > other colours. For internal cell 2, the solution is: green = red > other colours.

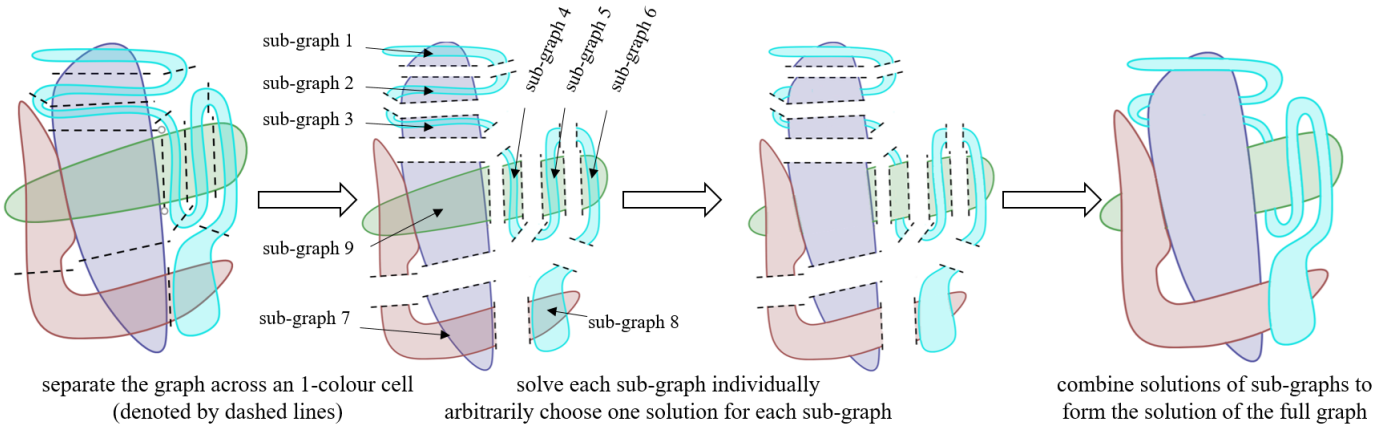


Fig. 9. A maximally 2-overlapped graph to show the role of intersections. The key step is to separate the full graph (left) into 9 sub-graphs through 1-colour cells, shown with dashed graph separating lines. Sub-graph 1-8 has been shown to have two optimal solutions, while sub-graph 9 gets a unique optimal solution after enumerative solving. Whatever solution gets chosen in each sub-graph, when two sub-graphs are combined, the two 1-colour sub-cells separated by the dashed line re-join together, resulting in a reduction of one on the number of continuous regions. In total $4 + (3 \times 8) = 28$ connected regions can be observed after solving the sub-graphs individually, combined with cost -15 (number of dashed lines). The nonrepetitive coverage problem of this graph ends up with $28 - 15 = 13$ connected regions, i.e., 12 lift-offs. Note that no intersections in the full graph are avoided when solving all the sub-graphs. And different choices of the solution for each sub-graph will double the number of optimal solutions of the full graph: there will be $2^8 = 256$ optimal solutions to the graph. In other words, all sub-graphs with intersections have to be enumerated.

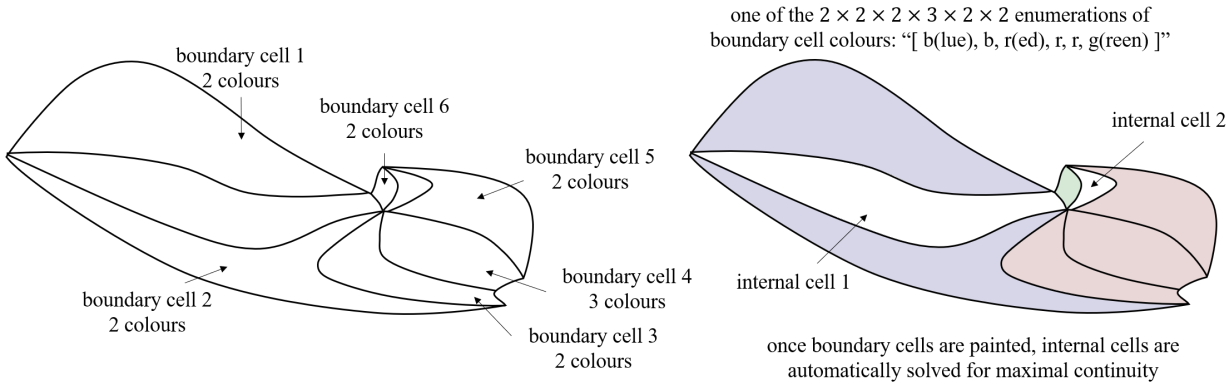


Fig. 10. An intersection-free graph. The internal cell 1 will choose blue > green > other colours. The internal cell 2 will choose green = red > other colours.

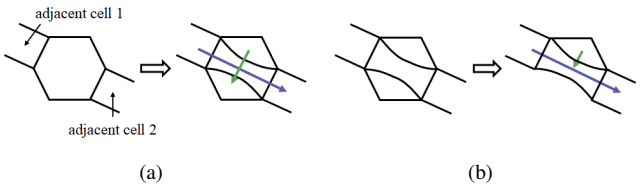


Fig. 11. (a) Illustration of constraint violation due to cell sub-divisions in sub-graphs. One sub-cell will have four adjacent cells. (b) If a cell is scheduled to be divided, it should first be divided and then followed by constructing the intersection-free sub-graphs.

Since non-adjacency of internal cells in sub-graphs will be enforced, but cell sub-divisions after sub-graphs have already been constructed, this constraint may be violated (see Fig. 11 for an illustration of this scenario). So let at this point all cells be assumed properly divided before applying the algorithm. As such, all cells contained in the sub-graphs can only be seen as a whole, and be filled with the same colour.

B. Graph Separation into Strips

At this point in the start of the separation process it can be safely assumed that all multi-edge cells have been divided. One boundary cell is first selected and regarded as an element of strip 1, shown in light green in Fig. 12(a). Each cell adjacent to the strip is checked to validate whether accommodating the cell into the strip will violate the strip constraints (**Definition** 11). Otherwise it is inserted. When no further cells can be inserted, the construction of strip 1 is deemed finished. In this example, as shown in Fig. 12(a), adjacent cells are iteratively inserted to strip 1 which are shown in blue. If one more cell is inserted, then the boundary cell 5 (illustrated in detail in Fig. 10) will become an internal cell which is adjacent to the internal cell 2, so the construction of strip 1 terminates.

To construct strip 2 (Fig. 12(b)), a cell (shown in light green) adjacent to strip 1 is selected, then its adjacent cells are checked one-by-one to see whether they can be inserted respecting the conditions in **Definition** 11. After that, another cell in the left-out graph is chosen into strip 3 (Fig. 12(c)), and other cells are again checked and attempted to be inserted

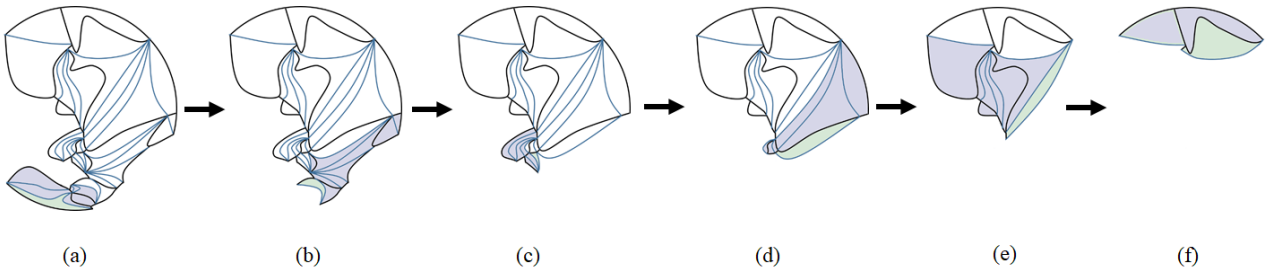


Fig. 12. Step-by-step illustration of the separation of a topological graph into intersection-free strips. The light green cell is the first one selected to form a new strip, and all light purple cells are then inserted into the same strip. The graph ends up with 6 separated strips. See also Fig. 13.

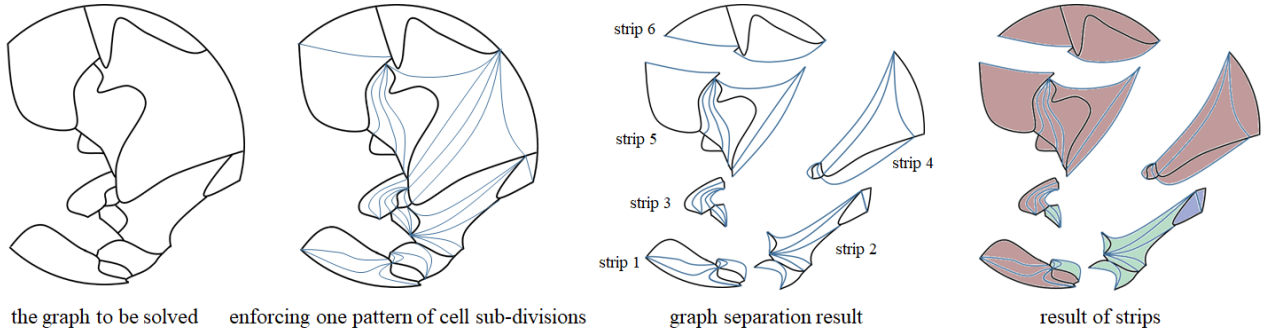


Fig. 13. Illustration of how an unsolved graph is separated into strips.

into strip 3, and so on. By iteratively considering adding the adjacent cells to the previous strip into the current strip and adding more cells satisfying the two constraints, the whole graph can be finally separated into strips, as shown in Fig. 13 where the graph is separated into 6 strips. (The process corresponds to line 11 ~ line 15 in **Algorithm** 1.) Note that there is no extra memory cost for this step - we are to enumerate all boundary cells of all strips and leave all internal cells un-enumerated, hence we need not physically manipulate the graph into strips as shown in figures, but simply mark all internal cells and skip them whilst enumeration. Besides, the enumeration can still be accessed in the depth-first manner. An implementation in C++ has been open sourced at:
https://github.com/ZJUTongYang/imcgs

C. Full Graph Solution

All the solutions for each strip are enumerated via iterating through all possible combinations of colours of its boundary cells, and for each enumerated case, the optimal colour of each internal cell is assigned. Picking up one solution from each sub-graph, they are combined to form a solution of the original graph. The edges identified during the graph separation process are automatically solved by comparing the colour of its two adjacent cells. A cost is calculated whose physical meaning is the number of connected regions in the painted part of the (combined) sub-graphs [12]. Once the original graph is reconstructed, the number of connected regions in the solution is fully known. Its optimality can be trivially judged against all possible combinations of sub-graph solutions to reach a set with all the optimal solutions. (line 18 ~ line 26 in **Algorithm** 1)

VII. Complexity

The advancement that the proposed mechanism brings about in solving the NCPP problem is reflected in an exponential improvement in algorithmic complexity over existing approaches.

As mentioned earlier, cell sub-divisions were undertaken before graph separation to avoid the possible constraint violation caused by cell sub-divisions in strips. So the algorithm complexity must also be calculated based on this (i.e. no $\prod E_i$ is multiplied for that segment of the procedure).

Theorem 13. *Given a topological graph to be solved, it is assumed that it has been first cell sub-divided before running the enumerative solvers. Let there be M topological cells and N internal edges in the graph. The number of edges for the i-th cell is α_i, and denote the number of available colours to fill cell i as K_i. Then, the proposed graph separation-based algorithm encompasses the following number of steps for enumerative solving the graph:*

$$\prod_{j=1}^{M} K_j^{\max\{\alpha_j-2,1\}} \tag{8}$$

Compared to the complexity of the full enumeration solution [12]

$$\prod_{j=1}^{M} K_j^{\max\{\alpha_j-2,1\}} \cdot 2^N \tag{9}$$

the proposed graphing scheme represents an exponential improvement in the order of 2^N.

Proof. The proposed algorithm has been shown to consist on a number of steps, namely:

1) Separating a graph into strips

Algorithm 1 IMCGS

Require: The initial graph $G = (\{C_i\}_{i=1}^{M}, \{E_j\}_{j=1}^{N})$
Ensure: All solved graphs $\{result\}$

```
for all divisions of cell 1 do
    if non-optimality is detected then
        continue
    end if
    ∵.
    for all divisions of cell M do
        if non-optimality is detected then
            continue
        end if
        // Graph Separation into Strips (Section VI)
        while the graph is not intersection-free do
            Choose a cell to be in the new strip
            Attach new cells satisfying constraints
            separate the strip from the graph
        end while
        // Let there be n strips generated
        for all enumerations of boundary cells do
            Assign optimal colour choice to internal cells
            if it is better than {result} then
                {result} = ∅
                push S into {result}
            else
                if it is currently optimal then
                    push S into {result}
                end if
            end if
        end for
    end for
    ∴
end for
```

2) Solving each strip individually.
3) Constructing all optimal solutions of the original graph by combining the individual strip solutions.

The overall complexity can thus be calculated individually for each step.

1) Separating the graph only requires checking on each cell once, and we only need one valid graph separation, so the complexity of this part is denoted by $\Phi = O(M)$.

2) The complexity of a single strip can be derived as follows: Let there be r_i boundary cells and s_i internal cells in the strip i, indexed as $(i_1, \cdots, i_{r_i}, i_{r_i+1}, \cdots, i_{r_i+s_i})$, and having $(\alpha_{i_1}, \cdots, \alpha_{i_{r_i}}, \alpha_{i_{r_i+1}}, \cdots, \alpha_{i_{r_i+s_i}})$ edges. The number of available colours for each cell can be written as $(K_{i_1}, \cdots, K_{i_{r_i}}, K_{i_{r_i+1}}, \cdots, K_{i_{r_i+s_i}})$. Note that the list of cells form a chain, so that other than the start and end cells, each cell will have two internal edges, one shared with its precedent cell and the other shared with its successor. As a 2D topological structure, a cell j in the strip will have at least two internal edges, and at most $(\alpha_{i_j} - 2)$ edges are exposed to the boundary of the strip. Hence, the complexity Ψ_i of solving strip i will be the multiplication of the algorithmic complexity of all boundary cells

$$\Psi_i = \prod_{j=1}^{r_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \tag{10}$$

where $\max\{\alpha_{i_j} - 2, 1\}$ represents the case that multiple edges of the same cell may appear in the strip boundary. Since the enumeration of the solutions for different strips is fully independent, the combined complexity is just summed up but not muliplied. Considering the notation $M = \sum(r_i + s_i)$, the overall complexity of finding the solutions for a sub-graph is given by

$$\begin{aligned}\Psi = \sum \Psi_i &= \sum_i \left(\prod_{j=1}^{r_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \right) \\ &\ll \sum_i \left(\prod_{j=1}^{r_i+s_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \right) \\ &\ll \prod_i \left(\prod_{j=1}^{r_i+s_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \right) \\ &= \prod_{j=1}^{M} K_j^{\max\{\alpha_j-2,1\}}\end{aligned} \tag{11}$$

3) For the final step of the algorithm, the complexity is the multiplication of the algorithmic complexity of all sub-graphs

$$\begin{aligned}\Xi = \prod \Psi_i &= \prod_i \left(\prod_{j=1}^{r_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \right) \\ &\ll \prod_i \left(\prod_{j=1}^{r_i+s_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \right) \\ &\ll \prod_{j=1}^{M} K_j^{\max\{\alpha_j-2,1\}}\end{aligned} \tag{12}$$

The overall complexity of the proposed algorithm is the sum of Φ, Ψ and Ξ,

$$\Gamma = \Phi + \Psi + \Xi \ll \prod_{j=1}^{M} K_j^{\max\{\alpha_j-2,1\}} \tag{13}$$

□

VIII. Experimental Results

The proposed solution is tested experimentally by simulating the NCPP with a non-redundant manipulator (6 DoF Universal Robot UR5) on four arbitrarily shaped objects, a hat-like concave semi-sphere, a saddle surface, a car door like surface, and a wok-like surface, as shown in Fig. 14, Fig. 15, Fig. 16 and Fig. 17. In all experiments, only one side is considered. The reader is referred to the supplemented video for the animation of the manipulator motion.

A comparative analysis is provided to compare the performance of the proposed solver IMCGS against the existing enumerative solver in [12]. All experiments were implemented on a PC computer with I7-8700 CPU, 11 threads in parallel. For fair comparison, both solvers are executed subjected to the same standard tree pruning processes. All statistics have been detailed in Table. I. Specif,

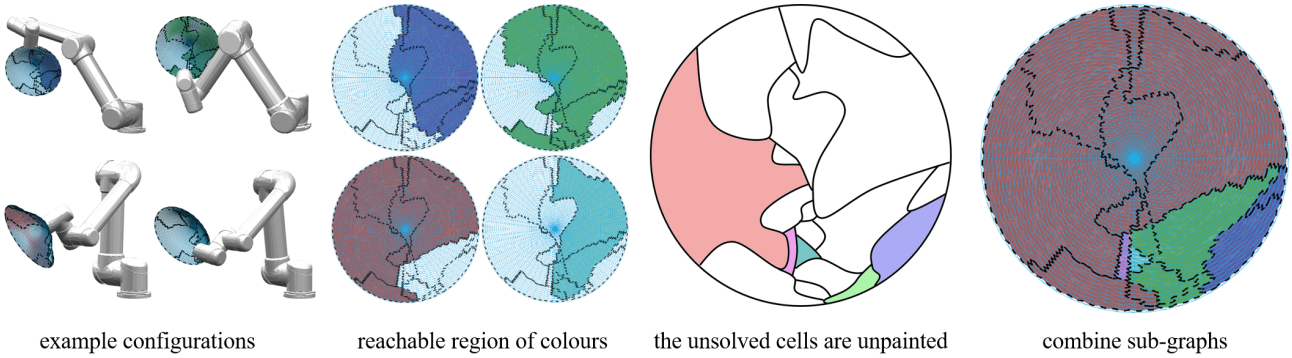


Fig. 14. The coverage task of a hat-shape object. The reachable area of different colours and one of their corresponding configurations are depicted. After painting all 1-colour cells, the unsolved graph is separated and solved as described in Section VI, with the step-by-step process illustrated in Fig. 12.

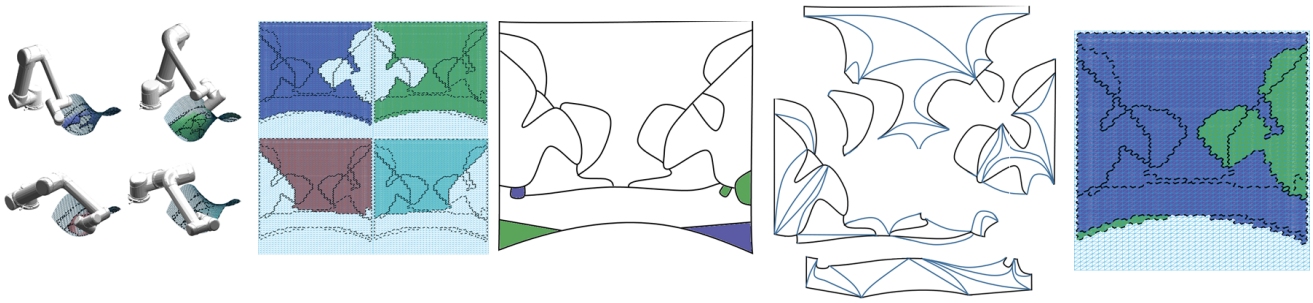


Fig. 15. The coverage task of a saddle surface. The reachable area of different colour and one of their corresponding configurations are depicted. After painting all 1-colour cells, the unsolved graph is separated into 5 sub-graphs. Finally, all optimal solutions can be collected with 2 end-effector lift-offs.

TABLE I
EXPERIMENTAL RESULTS

Object	Num. of Cells[1]	Num. of Edges[2]	Algorithm	Lift-offs[3]	Theoretical Bound[3]	Num. of Iterations[4]	Time[5]
Hat-shaped (Fig. 14)	13(18)	35(51)	[12]	**4**	$1.27 * 10^{28}$	$4.50 * 10^{7}$	26.05s
			Ours	**4**	$\mathbf{4.28 * 10^{14}}$	$\mathbf{2.46 * 10^{4}}$	**94ms**
Saddle-shaped (Fig. 15)	12(17)	37(48)	[12]	**2**	$2.86 * 10^{29}$	$8.71 * 10^{8}$	2.97h
			Ours	**2**	$\mathbf{4.33 * 10^{16}}$	$\mathbf{2.03 * 10^{6}}$	**7.02s**
Car Door-shaped (Fig. 16)	9(10)	16(35)	[12]	**1**	$2.32 * 10^{11}$	$1.54 * 10^{5}$	48.94s
			Ours	**1**	$\mathbf{3.46 * 10^{3}}$	**405**	**125ms**
Wok-shaped (Fig. 17)	9(15)	28(42)	[12]	**4**	$6.68 * 10^{13}$	$1.24 * 10^{4}$	1.13s
			Ours	**4**	$\mathbf{6.91 * 10^{3}}$	**27**	**<1ms**

[1] Num. of Cells: Number of non-unique colour cells (number of all cells).
[2] Num. of Edges: Number of internal edges (number of edges).
[3] All results are optimal in the number of end-effector lift-offs, since both solvers are optimal.
[3] The theoretical bounds are calculated based on a specific designing of cell sub-divisions. The sub-divisions for the hat-shaped object and the saddle-shaped object have been shown in Fig. 14 and Fig. 15, whilst the detailed illustrations for the car door-shaped object and the wok-shaped object are omitted.
[4] One iteration corresponds to sub-dividing a cell or colouring the whole graph.
[5] The computational time is measured on a PC computer with I7-8700 CPU, 11 threads in parallel.

1) When the number of painted colours for a partly-solved graph has been larger than the number of regions in the currently optimal solution, we remove this graph.
2) Cells are reindexed in the edge-number-increasing order, so that cells with large number of edges (hence may generate enormous branches after its sub-division) are less likely to be enumerated.
3) When a sub-cell is enforced to be connected to multiple adjacent cells, its possible colours can only be the intersecting set of the possible colours of all these adjacent cells. Specifically, the intersecting set of an arbitrary set with a single-element set is the single-element set itself, or the empty set (leading to the pruning of this enumerated case). Hence an adjacent cell with a unique colour will simplify the enumeration significantly.
4) For two sub-cells gapped by a cutting path, when one colour is chosen by one sub-cell, it is eliminated from the set of possible colours of the other sub-cell.

A. Optimal NCPP on a Hat-shape Object

An illustration of the solving process for the hat-shape object is depicted in Fig. 14. Four continuous sets of manipulator configurations are represented in blue, red, green and cyan colour separately. 1-colour cells have been directly drawn to the graph. The cell sub-divisions correspond to the description

provided earlier in Fig. 13. For the algorithmic complexity there are 43 cells. The possible colours of each cell are known. As for edges, there are 44 unsolved edges (drawn in black) and 30 manually created edges - enforced to be kept (drawn in blue). Plus 9 edges which are connected to unreachable area (a fifth colour), which need not solving. Using the naive enumeration [12] proposed in the literature, 35 edges and all colours of all cells are enumerated, resulting in an algorithmic complexity described by

$$\begin{aligned} &\overbrace{2\times2\times3\times2\times2\times2\times2\times2\times}^{\text{all cells in sub-graph 1, shown in Fig. 10}}\\ &2\times3\times3\times2\times2\times3\times2\times3\times2\times3\times\\ 2^{35}\times&2\times2\times2\times2\times2\times2\times2\times2\times \qquad \approx 1.27\times10^{28} \quad (14)\\ &2\times3\times3\times4\times4\times3\times4\times\\ &3\times3\times2\times3\times3\times4\times4\times3\times\\ &3\times4\times3 \end{aligned}$$

Using the proposed algorithm, the length of the boundary for each sub-graph is $6, 11, 9, 6, 10, 5$ respectively. The reader is referred to the sub-graph 1 example employed to discuss the process in detail in Sections V and VI. Detail results for the other cases are ommited for lack of space, but follow the same process. The number of colours for the boundary cells in each sub-graph is included in the multiplicative factor, so the algorithm complexity derives in the multiplication of the complexity of all sub-graphs, given by

$$\begin{aligned} &\overbrace{2\times2\times3\times2\times2\times2\times2\times}^{\text{internal cells shown in Fig. 10}}\\ &2\times3\times3\times2\times2\times3\times2\times3\times2\times\\ &2\times2\times2\times2\times2\times2\times2\times \qquad \approx 4.28\times10^{14} \quad (15)\\ &2\times3\times3\times4\times4\times\\ &3\times3\times2\times3\times3\times4\times4\times\\ &3\times4\times3 \end{aligned}$$

B. Optimal NCPP on a Saddle-shape Object

Another example is provided with the analysis of the outer shell of a saddle-shape object. In this case the surface normal varies greatly as the end-effector moves along the surface, representing a challenging manipulator planning problem in adopting poses that would benefit the continuous coverage this paper is interested in. The algorithmic complexity given cell sub-divisions is shown in Fig. 15. For the naive enumeration case [12], the overall complexity is given by

$$\begin{aligned} &2\times2\times2\times2\times2\times2\times2\times2\times2\times2\times2\times\\ &3\times2\times2\times2\times2\times2\times2\times2\times\\ &3\times4\times4\times3\times2\times2\times2\times3\times3\times\\ 2^{37}*&3\times2\times2\times2\times2\times3\times3\times2\times2\times \qquad (16)\\ &4\times4\times4\times4\times4\times4\times3\times4\times4\times\\ &3\\ \approx&2.86\times10^{29} \end{aligned}$$

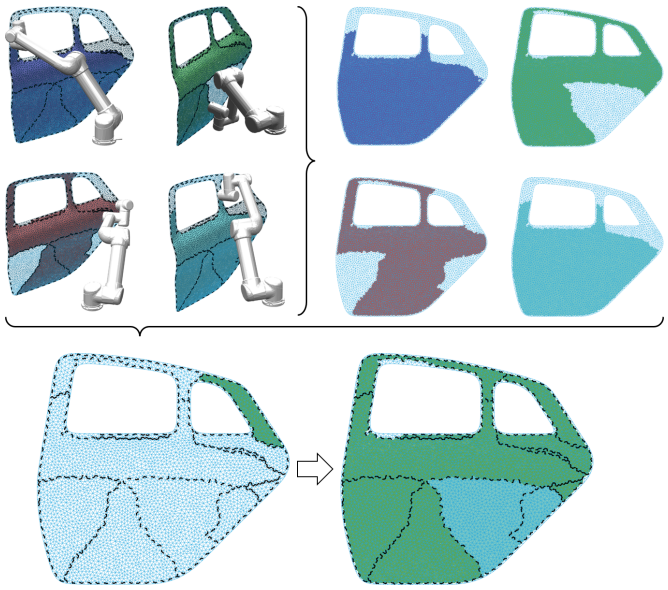


Fig. 16. Car door example. Single-colour cells have been directly drawn to the initial graph.

Applying the proposed algorithm, the complexity is reduced to

$$\begin{aligned} &2\times2\times2\times2\times2\times2\times2\times2\times2\times2\times\\ &3\times2\times2\times2\times2\times2\times2\times2\times\\ &3\times4\times4\times3\times2\times2\times2\times3\times\\ &3\times2\times2\times2\times2\times3\times3\times2\times \qquad \approx 4.33\times10^{16} \quad (17)\\ &4\times4\times4\times4\times4\times4\times3\times4\times\\ &3 \end{aligned}$$

The overall results are gathered in Table I for easier comparison, collecting the substantial computational improvement in the number of edges for each problem (35 and 37 repectively). As the geometry of the object of interest becomes more intricate, the benefit of the proposed scheme to be able to find paths with minimal discontinuities in joint-space with a reduced computational effort becomes also more apparent.

C. Further Evaluations

Further simulation tests are shown in Fig. 16 and Fig. 17. The object surfaces are now unstructured meshes, where the proposed solver is still applicable.

In the former test, a car door mesh is considered. Among the 10 cells in this graph, one cell (referred to as Cell 1 in this experiment) has 16 edges, whilst all other cells have no more than 4 adjacent connectable cells. Recalling that cells with ≤ 3 edges need not be sub-divided, and the number of sub-divisions for a 4-edge cell can be easily verified as 2 (the diagonals of the topological quadrangle). Hence, enumerating Cell 1 consumes the largest proportion of computational time for the solver. As indicated before, Cell 1 has the largest number of edges, so it is the last cell to be enumerated. In contrast to the existing solver, the proposed improved solver does not enumerate edges nor possible colours of cells, hence creates a significant smaller number of enumerated cases that

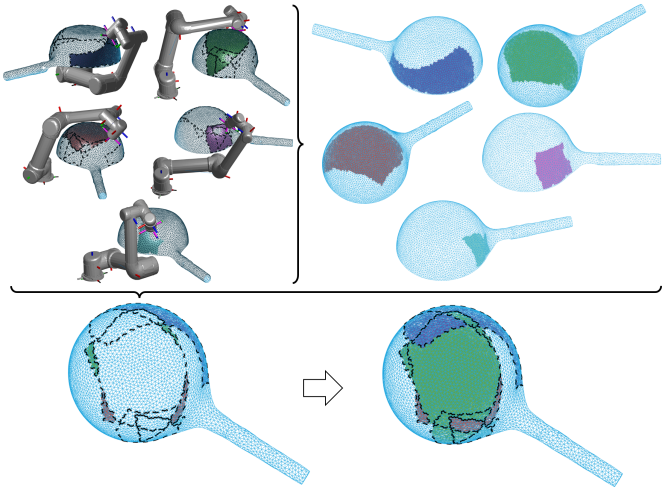


Fig. 17. Wok example. Single-colour cells have been directly drawn to the initial graph.

requires the enumeration of Cell 1. Result shows that the computational time of the improved solver IMCGS (125ms) is much more efficient than the state-of-the-art method (48.94s).

In the latter, a wok-like mesh is tested. For simplicity in this example, the handle of the wok is not considered part of the mesh for coverage, and it is set as an obstacle. To form a more challenging set of valid configurations, we assume that the wok floats in the air without support, so the manipulator can perform the coverage motion with its forearm below the wok. Although there are 15 cells in the graph, 6 of them only have a unique colour (painted in the initial graph in Fig. 17), which give birth to a much more efficient tree pruning processes (see the huge reduction of the number of iterations against the theoretical bound). Finally, the proposed solver is able to solve the graph within 1ms.

IX. Conclusions

An advancement to efficiently solve non-repetitive coverage tasks with a non-redundant manipulator is proposed in this work. The core concept is the introduction of intersection-free sub-graphs, which permit the separation of a graph into sub-graphs at the points where the multiplicity of optimal solutions originate. The implicit enumeration when the sub-graphs are recombined in search of the full solution derives in a mathematically proven exponential improvement in the order of 2^N, N representing the number of topological edges in the graph. This is particularly relevant for intricate task-space graphs, where an exhaustive search for solutions with all the cells and edges is unattainable. The proposed graph solver, called IMCGS, has been supplemented by a detailed video and an open source implementation in C++ for the benefit of the research community. Comprehensive experimental results with step-by-step derivations to illustrate the proposed algorithm are supplied that demonstrate the validity of the novel scheme.

Appendix: Glossary

1) Colour: A continuous set of collision-free, non-singular manipulator configurations with the end-effector staying on the task-space surface.
2) Cell: A part of task-space reachable region, where all points having the same set of possible colours.
3) Edge: The common boundary between two cells.
4) Graph: The combination of cells and edges.
5) Cutting Path: A task-space curve that enforces its two sides cannot be painted the same colour.
6) Cell Sub-division: Dividing the region of a cell into multiple sub-parts by cutting paths.
7) Graph Solution: An assignment for each cell a colour within its set of possible colours.
8) Cost: Grouping the same-colour adjacent cells in one group, the cost is the number of groups, visualised as the number of connected painted regions.
9) Optimal Solution: The graph solution with minimum cost.
10) Sub-graph: a connected subset of cells, together with all edges that connect them.
11) Boundary of Sub-graph: A closed chain of the edges that connect a cell inside the sub-graph and a cell outside the graph.
12) Boundary Cell: The cells in a sub-graph which are adjacent to cells outside this sub-graph.
13) Internal Cell: The non-boundary cells in a sub-graph.
14) Intersection-free Sub-graph: A sub-graph with all its boundary cells having a unique colour, such that the number of optimal solution of the sub-graph is unique.
15) Strip: A kind of sub-graph that is intersection-free given any possible colours of its boundary cells.

References

[1] G. Paul, N. Kwok, and D. Liu, "A novel surface segmentation approach for robotic manipulator-based maintenance operation planning," *Automation in Construction*, vol. 29, pp. 136–147, 2013.

[2] P. M. Bhatt, A. Kulkarni, R. K. Malhan, B. C. Shah, Y. J. Yoon, and S. K. Gupta, "Automated planning for robotic multi-resolution additive manufacturing," *Journal of Computing and Information Science in Engineering*, vol. 22, no. 2, 2022.

[3] Y. Han, L. Zhang, M. Guo, C. Fan, and F. Liang, "Tool paths generation strategy for polishing of freeform surface with physically uniform coverage," *The International Journal of Advanced Manufacturing Technology*, vol. 95, no. 5, pp. 2125–2144, 2018.

[4] Y. Wen, D. J. Jaeger, and P. R. Pagilla, "Uniform coverage tool path generation for robotic surface finishing of curved surfaces," *IEEE Robotics and Automation Letters*, 2022.

[5] M. Hassan and D. Liu, "A deformable spiral based algorithm to smooth coverage path planning for marine growth removal," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1913–1918, IEEE, 2018.

[6] T. Yoshikawa, "Translational and rotational manipulability of robotic manipulators," *American Control Conference*, vol. 27, pp. 228–233, 1990.

[7] M. Li, Z. Lu, C. Sha, and L. Huang, "Trajectory generation of spray painting robot using point cloud slicing,"

Applied Mechanics and Materials, vol. 44–47, pp. 1290–1294, 2011.

[8] X. Xie and L. Sun, "Force control based robotic grinding system and application," in *Proceedings of the 2016 World Congress on Intelligent Control and Automation*, pp. 2552–2555, June 2016.

[9] D. Lee, T. Seo, and J. Kim, "Optimal design and workspace analysis of a mobile welding robot with a 3p3r serial manipulator," *Robotics and Autonomous systems*, vol. 59, no. 10, pp. 813–826, 2011.

[10] P. Giataganas, V. Vitiello, V. Simaiaki, E. Lopez, and G.-Z. Yang, "Cooperative in situ microscopic scanning and simultaneous tissue surface reconstruction using a compliant robotic manipulator," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5378–5383, IEEE, 2013.

[11] S. Huo, D. Navarro-Alarcon, and D. T. Chik, "A robotic defect inspection system for free-form specular surfaces," in *Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11364–11370, IEEE, 2021.

[12] T. Yang, J. Valls Miro, Q. Lai, Y. Wang, and R. Xiong, "Cellular decomposition for non-repetitive coverage task with minimum discontinuities," *IEEE/ASME Transactions on Mechatronics*, 2020.

[13] T. Yang, J. Valls Miro, Y. Wang, and R. Xiong, "Non-revisiting coverage task with minimal discontinuities for non-redundant manipulators," in *Proceedings of the 16th Conference on Robotics-Science and Systems*, MIT PRESS, 2020.

[14] J. A. Bondy, U. S. R. Murty, *et al.*, *Graph theory with applications*, vol. 290. Macmillan London, 1976.

[15] D. Kaljaca, B. Vroegindeweij, and E. van Henten, "Coverage trajectory planning for a bush trimming robot arm," *Journal of Field Robotics*, vol. 37, no. 2, pp. 283–308, 2020.

[16] G. Oriolo and C. Mongillo, "Motion planning for mobile manipulators along given end-effector paths," in *2005 IEEE Conference on Robotics and Automation (ICRA)*, 2005.

[17] E. Eiben, I. Godage, I. Kanj, and G. Xia, "On the problem of covering a 3-d terrain," in *Proceedings of the 2020 AAAI Conference on Artificial Intelligence*, vol. 34, pp. 10361–10368, 2020.

[18] C. Wu, C. Dai, X. Gong, Y.-J. Liu, J. Wang, X. D. Gu, and C. C. Wang, "Energy-efficient coverage path planning for general terrain surfaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2584–2591, 2019.

[19] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and service robotics*, pp. 203–209, Springer, 1998.

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

[21] E. U. Acar, H. Choset, and J. Y. Lee, "Sensor-based coverage with extended range detectors," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 189–198, 2006.

[22] W. H. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, vol. 1, May 2001.

[23] P. Atkar, D. Conner, A. Greenfield, H. Choset, and A. Rizzi, "Hierarchical segmentation of piecewise pseudoextruded surfaces for uniform coverage," *IEEE Transactions on Automation Science & Engineering*, vol. 6, no. 1, pp. 107–120.

[24] M. Hassan and D. Liu, "Simultaneous area partitioning and allocation for complete coverage by multiple autonomous industrial robots," *Autonomous Robots*, vol. 41, no. 8, pp. 1609–1628, 2017.

[25] M. Rososhansky and F. Xi, "Coverage based tool-path planning for automated polishing using contact mechanics theory," *Journal of Manufacturing Systems*, vol. 30, no. 3, pp. 144–153, 2011.

[26] L. Lu, J. Zhang, J. Y. H. Fuh, J. Han, and H. Wang, "Time-optimal tool motion planning with tool-tip kinematic constraints for robotic machining of sculptured surfaces," *Robotics and Computer-Integrated Manufacturing*, vol. 65, p. 101969, 2020.

[27] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, "Deployment of mobile robots with energy and timing constraints," *IEEE Transactions on robotics*, vol. 22, no. 3, pp. 507–522, 2006.

[28] F. Paus, P. Kaiser, N. Vahrenkamp, and T. Asfour, "A combined approach for robot placement and coverage path planning for mobile manipulation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[29] Q. Fan, Z. Gong, B. Tao, Y. Gao, Z. Yin, and H. Ding, "Base position optimization of mobile manipulators for machining large complex components," *Robotics and Computer-Integrated Manufacturing*, vol. 70, p. 102138, 2021.

[30] R. Kalawoun, S. Lengagne, and Y. Mezouar, "Optimal robot base placements for coverage tasks," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pp. 235–240, IEEE, 2018.

[31] Y. Liu, W. Zhao, R. Sun, and X. Yue, "Optimal path planning for automated dimensional inspection of free-form surfaces," *Journal of Manufacturing Systems*, vol. 56, pp. 84–92, 2020.

[32] Z. Zhou, Y. Zhang, and K. Tang, "Sweep scan path planning for efficient freeform surface inspection on five-axis cmm," *Computer-Aided Design*, vol. 77, pp. 1–17, 2016.

[33] P. M. Bhatt, C. Gong, A. M. Kabir, R. K. Malhan, B. C. Shah, and S. K. Gupta, "Incorporating tool contact considerations in tool-path planning for robotic operations," in *Proceedings of the 2020 International Manufacturing Science and Engineering Conference*, vol. 84256, p. V001T05A018, American Society of Mechanical Engineers, 2020.