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

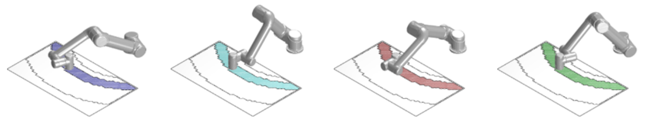
Tong Yang, Jaime Valls Miro, Yue Wang* and Rong Xiong

***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 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.**

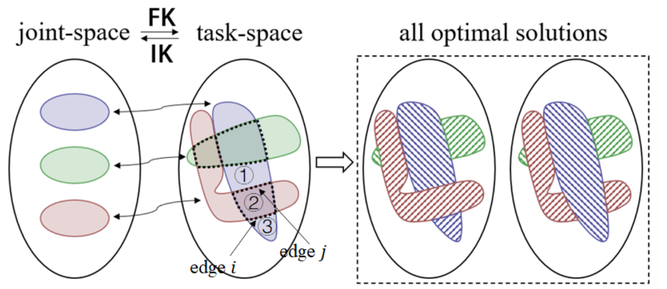
***Index Terms*—Optimal Cellular Decomposition, Manipulator Coverage Task**

I. Introduction and Related Work

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). 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, an example of which is depicted by the example mapping provided in Fig. 1(b), left. Disregarding singular configurations [3], the collection of all valid configurations 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 cannot remain on the object, being forcibly detached from the surface. This is an undesirable effect for tasks where smoothness and continuity



(a) Coverage illustration of a planar rectangular region with a non-redundant manipulator, depicting s(houlder)-l(eft)&w(rist)-f(lipped), s-r(ight)&w-u(nflipped), s-r&w-f, and s-l&w-u configurations. 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 as shown in (b) below.



(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 (more intricate examples are provided in Section V). 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 "intersecting" 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 correpond to each other).

are critical such as painting [4], deburring [5], welding [6], scanning [7] etc. In these instances, the desired solution translates to designing manipulator trajectories such that the end-effector visits each point on the surface exactly one time, whilst ensuring a minimum number of transitions between joint-space sets, or lift-offs.

[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

Early reports on the generic *coverage path planning* (CPP) problem focused on geometric path designing [8] [9], particularly for mobile platforms operating in planar surfaces, such as boustrophedon [10] or spiral paths [11]. 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 [12] [13] [14] [15] [16], applied directly onto 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: 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 end-effector can only visit each point on task-space once, leading to numerous pose reconfigurations during the motion of the end-effector [17]. Many criteria in the classic CPP problems have been adapted to the manipulator NCPP task, such as time to completion [18] or energy consumption [19], and optimal mobile manipulator pose for a given coverage in task-space has also been investigated [20] [21] [22]. However, optimising end-effector lift-offs for increased smoothness and continuity has been rarely exploited. 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 [23], as intuitively depicted in Fig. 1. All maximal-continuous cellular decompositions were proven to be collectable in a finite number of steps, defined by edges that represent the smallest, inseparable elements and cells that could only have a finite number of different sub-divisions.

A. Preliminary and Complexity

Given the kinematics of the non-redundant manipulator, the shape of the surface to be non-revisitably covered, the obstacles in the surrounding environment, and their related poses, each point on the surface can be reached by the end-effector through finite number of Inverse Kinematic (IK) solutions. Following topological elements are well-defined.

Definition 1. *(Valid Configuration) Valid configuration is the collision-free and non-singular [3] manipulator configuration such that the end-effector touches the surface to be manipulated with its orientation parallel to the surface normal.*

Remark 2. *When singular configurations are disregarded, the FK relation from the manipulator configuration space to the space of end-effector poses is locally one-to-one.*

Definition 3. *(Colour) Each valid manipulator configuration is represented by a colour. Continuous configurations are assigned the same colour. Assume that there is an one-to-one correspondence between the EE pose and the point on the surface, the colour of a point on the surface represents the corresponding IK solutions with same colour.*

Definition 4. *(Cell) A cell is a maximal connected region on the surface, within which all points have the same number of valid IK configurations, and the configurations are pairwise continuous. In other words, all points have the same set of possible colours, which is also referred to as the possible colours of the cell.*

Remark 5. *By the definition of cell, within a single cell the geometric coverage path can be arbitrarily designed. And starting from any choice of configuration, the manipulator can continuously track the coverage path without end-effector lift-off.*

Definition 6. *(Edge) The edge is the common boundary of two cells.*

Definition 7. *(Graph) A graph is the combination of all cells and their connectivities, i.e., the edges.*

Given the NCPP problem to be solved, the initial graph can be constructed. By saying *solving the graph* we mean preserving only one of the possible colours for each point. During the solving process, if on the both side of the edge the points have the same possible colour, then it is equivalent to removing the edge. Or else it is equivalent to keeping the edge. Theoretical discussions [23] have be carried out to ensure that

1) The number of cells in the initial topological graph is finite.
2) The intersection of edges is isolated point, thus can be safely disregarded.

The following lemma has ensured the finiteness of all possible topological-distincted cellular decompositions, among which all optimal cellular decompositions lie in.

Lemma 8. *([23]) Different part of a cell may be filled in different colours, while the design of the cell cutting paths satisfies that*

1) *It is sufficient to consider cutting paths which start and end at the edge endpoints.*
2) *It is unnecessary to consider cutting paths that go across edges.*
3) *It is unnecessary to consider intersecting cutting paths.*

This is however an intensely computational process, growing exponentially in the size of the graph. 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. It has been shown how a binary array of length α_i can be used to index all possible different divisions of cell i [23], whereby 0 at a given position enforces the cell having a different color to its adjacent cell, whilst 1 compels the same choice of colour. However, as supported by the illustrative example shown in Fig. 2 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 two parts with differing colours, while each part themselves connect to their two adjacent cells. Following the cell sub-divisions, 4 valid painting solutions of this cell appear corresponding to the edge solution 1111.

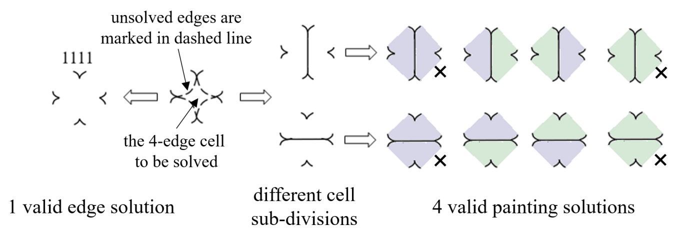
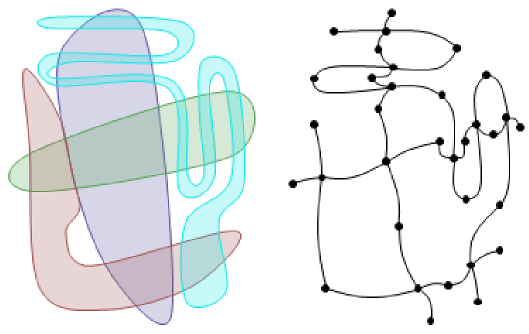


Fig. 2. An illustration of the many-to-one relation between edge and painting solutions. Let a cell with 4 adjacent cells (a 4-edge cell) have 2 possible colours "blue" and "green", then $\alpha = 4, K = 2$. From the point of view of solving edges, the binary number 1111 specifies one solution, but the corresponding valid painting solutions are multiple, 4 to be precise: there are $E = 2$ 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 (no subdivision when there is only one possible colour for all sub-cells), shown crossed-out, leaving the remaining four valid painting solutions.



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

Fig. 3. The quantitive relation between the cell-graph adopted in manipulator coverage task and the node-graph discussed in classic graph theory. 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.

There is a need to enumerate all the edges for each cell, leading to all different colours to be subsequently filled in to create all posissible solutions [23]. This is a costly exercise. For the more generic case of a multi-edge ($\alpha_i \geq 4$) cell, other than the simple case of considering it as a whole, it may be sub-divided into up to $(\alpha_i - 2)$ sub-cells, and each sub-cell can be individually assigned with K_i colours. 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 algorithmic complexity can be calculated as

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

B. Contribution of This Paper

Careful examination of the topological graph leads in this work to the introduction of a high-level abstraction, the *topologcal intersection* which is the origin of the multiplicity of the optimal solutions. It is hereby proven that intersections are indeed unavoidable and have to be enumeratively solved. However, 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 end result is the removal of the substantial complexity in enumerating all edges, 2^N.

The significance of the algorithmic improvement proposed in this work can also be revealed by mentioning the quantitive connection between the cell-graph utilised in manipulator NCPP task and the node-graph in classic graph theory, as shown in Fig.3. There shall be an one-to-one correspondence between the cells and edges of a cell-graph and the vertices and edges in a node-graph, respectively, where then the Euler's formula [24] for planar graphs yields:

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

where V, E and F are the number of vertices, edges and faces in the node-graph, respectively. This indicates the enormous number of edges in the cell-graph, $F - 2$ greater than the number of cells,

$$N - M = E - V = F - 2 \tag{3}$$

Here it has been observed that in complicated graphs F must be large, which acknowledges the contribution of reducing the algorithmic complexity by $1/2^N$. Also, the quantitive relation to the classic graph theory may indicate a potential application in other 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 introduces the concept of *topological intersection* and the *intersection-free graph* property. Section III 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 is mathematically proven in Section IV, whilst experimental results from simulations are collected in Section V. Final concluding remarks are gathered in Section VI.

II. Optimality via "Topological Intersections"

In this section, we introduce a topological invariant variable, *intersection*. A topological invariant variable is a character of the given structure which is invariant under the admissible set of modifications to the structure, such as the connectedness of a region under homeomorphic mappings. Typical application of topological invariants in mathematics is for distinguishing two different objects: First define the topological invariant, then show that two objects have different topological invariants, so they are not equivalent under the modifications that preserve the topological invariant. However, different from the mainstream usage of topological invariants, we show that the enumerative solving of intersections is the core element in the graph solver leading to notable inefficiencies in solving the full graph. The key observation is then not leveraging intersections but how to avoid intersections.

All contents in this sections are to contruct "counterexamples" that reveal the inexistence of a possibly more efficient algorithm which avoids enumerating intersections. As

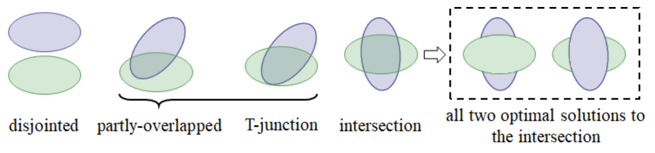


Fig. 4. Illustration of basic relations between the reachable area of two colours. T-junction is a special case of the partly-overlapped distribution of colours, while it can be transformed back to a normal case through continuously modifying the boundary of colours. Only the intersected case needs solving, and it is apparent that the two optimal solutions are not topological equivalent, since they have the different number of connected regions in blue and green colour.

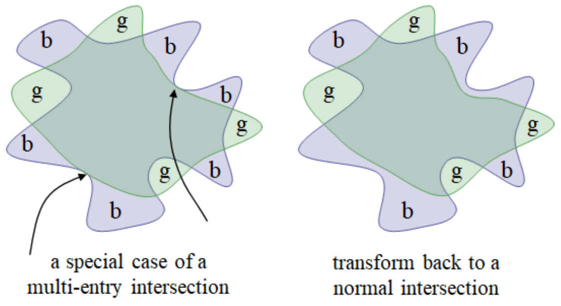

b
g
b
g
b
g
b
g
b
b
a special case of a multi-entry intersection
b
g
b
g
g
g
b
b
transform back to a normal intersection

Fig. 5. An intersection with 8 entries. For some special cases that two consecutive adjacent cells can be filled in same colour, through shrinking the boundary of colour, it can be transformed back to the normal case.

will be concluded, the enumeration of all intersections, with their algorithmic complexity being multiplied, is unavoidable. Finally, all concentrations for possible improvement of the algorithmic complexity over existing algorithms have to be paid on avoiding "explicitly" enumerating intersections, which makes the simplified representation of all solutions of an intersection-free graph apparent, and indicates an efficient algorithm based on intersection-free graph separation in the next section.

A. *Defining Intersections*

Definition 9. *(Intersections) An intersection is a class of distribution, more concretely, overlapping, of the coverable region of multiple colours, describing an inevitable interruption of the connectedness of one colour by other colours.*

Denoting $\#\{\text{NCPP}(G)\}$ as the minimal number of end-effector lift-offs given the graph G, and $\#\{\text{CPP}(G)\}$ as that of the repetitive coverage path planning problem, the existence of intersections can be formally revealed by

$$\text{intersection exists} \Leftrightarrow \#\{\text{NCPP}(G)\} - \#\{\text{CPP}(G)\} > 0 \quad (4)$$

The invariance of intersections is revealed in the connectedness of cells:

Proposition 10. *(Invariance under homotopic cutting paths) In a (partly-solved) graph, any continuous modification of the cell cutting paths will not reduce the number of intersections in the graph.*

Proof: It has been shown in **Lemma** 8 that all cutting paths (and preserved edges) need not to be intersected, hence the continuous modification of cutting paths will not change the connectivity of any cell. I.e., there will not be a case that two disconnected cells with the same colour filled become connected after continuous modification of cutting paths, nor a case that a single solved cell is trancated into two disconnected parts. Hence the intersections will be invariant. ■

Typically, for the simplest case, the coverable area of two colours may be fully disjointed, partly overlapped or intersected, as shown in Fig. 4, whereby colours can be both kept connected in the disjointed and partly overlapped cases, but critically in the intersection case one colour being connected will truncate the other colour into two parts.

B. *Properties of Intersections*

Each result in this subsection presents a crucial property of intersection. These properties are "traps" that prevent an efficient algorithm being proposed. Note that they do not directly contribute to the algorithm design, but as a side effect indicate the possible upper bound of the improvement on the algorithmic complexity.

Lemma 11. *(Multiplicity of Solutions) The existence of intersections causes multiplicity of optimal NCPP solutions.*

Proof: For the most apparent example see Fig. 4. Obviously, there are two optimal solutions to the intersected case, filling in the central cell full of either blue or green. In more complicated graphs, such an intersection may introduce two sets of optimal solutions, in one set all solutions keep the blue colour connected, while in the other set the green colour remains connected.

Note that the example is simple but not a special case. Assume that except one intersection, the left-out graph has been solved and m solutions have been constructed, labelled as $s_1, \cdots, s_m$. Let there be n different solutions of an intersection labelled by $t_1, \cdots, t_n$, then $m \times n$ solutions for the whole graph are constructed. Should the solution (s_i, t_j) be (non-)optimal, $(s_i, t_{j'}), \forall j' \in \{1, \cdots, n\}$ would be also (non-)optimal. ■

Corollary 12. *(Unavoidability of Enumeration) All intersections have to be enumerated in order to collect all optimal NCPP solutions.*

We prove this by showing two aspects:

1) Collecting all solutions of a single intersection requires enumeration.
2) Each intersection will not be removed by graph separation. Thus it can only be reduced by (enumerative) solving.

Proof: 1) We prove by example that even a kind of simple intersections, the multi-entry intersections, which are so structural to be easily solved, have to undergo an enumerative solving. Seeing the intersection in Fig. 4 as 4 entries, the intersection with arbitrary number of entries is intuitive, as shown in Fig. 5. After simplifying the case by merging consecutive adjacent cells which have the same set of possible colours, the intersection will have n (an even number) entries. It is observed that for each adjacent cell (say the i-th, labelled cyclicly), if it is connected to any other adjacent cell through

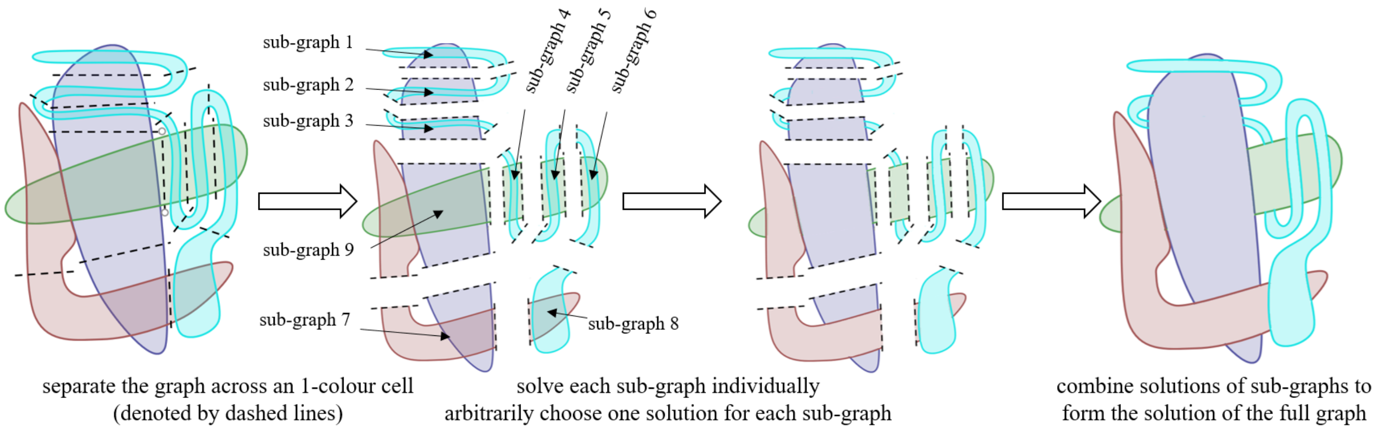

sub-graph 1
sub-graph 2
sub-graph 3
sub-graph 4
sub-graph 5
sub-graph 6
sub-graph 9
sub-graph 7
sub-graph 8
separate the graph across an 1-colour cell
(denoted by dashed lines)
solve each sub-graph individually
arbitrarily choose one solution for each sub-graph
combine solutions of sub-graphs to
form the solution of the full graph

Fig. 6. A maximally 2-overlapped graph to show the role of intersections. The full graph (left) is separated into 9 sub-graphs through 1-colour cells, shown with dashed cutting 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. Note that 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.

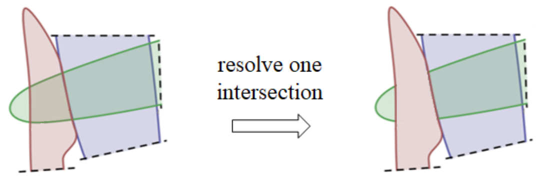
the intersection, then another two entries, $(i-1)$ and $(i+1)$, cannot be connected. So the minimal number of connected regions for an n-entry intersection is $\frac{n}{2}+1$. However, note that even if the optimal number is directly deduced, we still have to enumeratively solving the intersection to get all optimal cellular decompositions. For the i-th entry, we will separately consider (a) connecting it to other entries, or (b) keeping itself disconnected so that entries $(i-1)$ and $(i+1)$ can be connected. Both of them go to optimal cellular decompositions.

2) The role of intersections in graphs can be revealed in solving a kind of simple graphs, separable maximally 2-overlapped graphs, illustrated by Fig. 6. The key step is to separate the graph into sub-graphs by dividing 1-colour cells connecting intersections. Since the divided cell only has a unique coverable colour, we divide one cell into two unconnectable cells (they belong to different sub-graphs), so the number of continuous regions on the graph should be the sum of continuous regions in sub-graphs minus the number of dashed lines. Moreover, we can arbitrarily combine the optimal solutions of sub-graphs to form the optimal solution of the original graph. However, it should be noted that while the computational cost of combining solutions of sub-graphs has been reduced, no intersection was removed. Intersections must be solved within the sub-graphs, and all optimal solutions collected through a full combination of the different solutions with intersections. ■

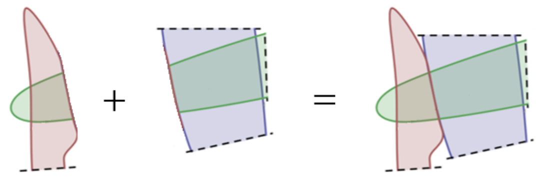
All above results have been proved by raising example graphs where the intersections can be easily distinguished. However, this is only for simple explanation, as in generic graphs the necessity of enumerative solving is compounded by the incountability of intersections.

Lemma 13. *(Incountability of Intersections) Given a graph, except the trivial cases, 0 adn 1, the number of intersections in the graph is not countable.*

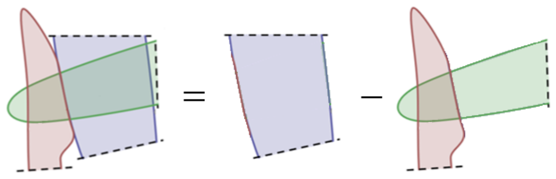
Proof: Fig. 6 sub-graph 9 illustrates the case where two parallel (nonoverlapping but touching) colour regions exist



(a) The graph is observed to have 1 intersection.



(b) Two intersection-free graphs form a graph with one intersection.



(c) After removing one intersection, the left-out part of an intersection-one graph still has one intersection.

Fig. 7. The $0+0=1$ and $1-1=1$ example of the number of intersections between graph and sub-graphs.

which are simultaneously crossed by a third colour. In this case, the graph cannot be trivially separated since the multi-colour cells are adjacent, with no 1-colour cell in between.

Analysing the sub-graph 9 in detail gives us surprising results, as shown in Fig. 7. It can be easily observed that the graph has only one intersection, as after the intersection between red colour and green colour is resolved, the graph will have no intersection. However, a combination of intersection-free graph can generate an intersection-one graph, namely $0+0=1$ (Fig. 7(b)). From another perspective, after the blue

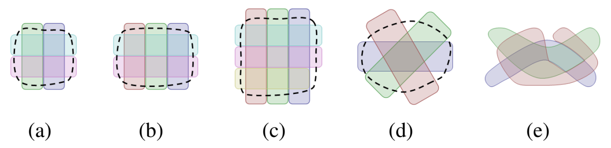


Fig. 8. Demonstrating graphs with various forms of intersections. The dashed circle marks a part of the graph which contains intersections. Intersection formed by: (a) 2&2 parallel colours, (b) 2&3 parallel colours and (c) a further special case of 3&3 parallel colours (where whether the orange colour intersects with the blue colour is closely related to the optimal solutions. Now they only form a T-junction), (d) a normal 3-overlapped intersections and (e) a complicated 3-overlapped graph which we cannot even figure out where is the intersection given too many intersections and parallels of colours in it.

colour is removed which apparently has an intersection with the green colour, the left-out graph still has an intersection, namely $1 - 1 = 1$ (Fig. 7(c)).

In generalised cases encountered as shown in Fig. 8, it is apparent how as more colours get involved, intersections become highly coupled, and even identifying intersections is not trivial. ∎

Corollary 14. *There is no algorithm that can solve a graph with explicit intersection-decreasing, such as transforming an intersection-n graph to an intersection-$(n-1)$ partly-solved graph.*

It can be thus concluded that all solvers (including all existing solvers and ours to be proposed in this work) to this problem have to be in an enumerative scheme, and there is no guarantee of the decreasing of the number of intersections as a graph being gradually solved. Only what we know is that a solved graph has no intersection. Disregarding the position of intersections and how to solve them leads to exhaustive implicit enumeration when all edges and cells are enumerated [23]. On the other hand, exploiting the interesting complex scenarios that analysing the topology between different colours offer will attenuate this problem. In this regard, the concept of intersection-free graphs described next will play a crucial role.

C. Intersection-Free Graphs

Definition 15. *(Intersection-free Graph) An intersection-free graph is a graph where no intersection can be constructed within it.*

(Boundary Cell, Internal Cell) Two types of cells can be distinguished in an intersection-free graph, the boundary cell which has edges forming the boundary of the graph, and the remaining internal cell which has no edge exposed to the graph boundary.

See Fig. 9 for an illustration of the intersection-free graph with 6 boundary cells and 2 internal cells. It is apparent that all internal cells are shown to have less than 4 cells, or else an intersection could be constructed within.

Definition 16. *(Characteristic String) Given a graph, let there be n (ordered) boundary edges, then the characteristic string of the graph is a word of length n, where the i-th letter in the word represents the colour of the boundary cell containing the i-th boundary edge.*

For the example in Fig. 9, the partly painted graph has characteristic string "[b, b, r, r, r, g]". A crucial property of intersection-free graphs is presented as the following theorem.

Theorem 17. *The solution of the graph, more concretely, the assigned colour of all cells in the graph, can be uniquely characterised by the colour of the boundary cells.*

Proof: To prove this, let the colour of all boundary cells be determined. On picking up two boundary cells with the same colour one must be able to judge their connectivity through this graph. Should connectivity be undetermined, one may find another pair of boundary cells whose connection may prevent these two cells being connected. However, this represents an intersection, which contradicts the assumption of intersection-free property of the graph.

After assigning a colour to each boundary cell, all internal cells will be assigned the colour which maximises the connectivity to their adjacent cells. After all cells have had their colour allocated, all the internal edges can be automatically solved. It can thus be said that each characteristic string uniquely corresponds to a solution of the intersection-free graph, and the number of characteristic strings can then be liken to the full quantification of the number of solutions in the graph. ∎

III. Graph Separation

An efficient strategy to separate a graph into intersection-free sub-graphs is described in this section. Since non-adjacency of internal cells in sub-graphs will be enforced, and cell sub-divisions after sub-graphs have already been constructed, this constraint may be violated (see Fig. 10 for an illustration of this scenario). So let at this point all $n(\geq 4)$-edge cells have been properly divided before applying the algorithm, thus all cells in the sub-graphs can only be seen as a whole and be filled with the same colour.

An apparent result for graph separation is that the graph should not be separated across edges, or else the constraints of the original cell vanishes, and non-optimal solutions may appear. See an illustration of this scenario in Fig 11. Another result is that the optimality of part of the graph has no relation to the global optimal solution of the whole graph. This is proved by the example illustrated in Fig 12.

An algorithm block showing the processes of intersection-free graph separation is given in **Algorithm** 1.

A. Defining Strips

Proposition 18. *In a graph enforced that*

1) *Each cell has at most 3 adjacent cells, and*
2) *Internal cells are non-adjacent,*

then the graph is an intersection-free graph.

Proof: It is observed that when a part of the graph contains an intersection, there must exist at least 4 entries. Moreover, each entry may not be a single edge. These motivate us to consider its opposite: if 1) is satisfied, then there can not possibly be an intersection. In noting that two internal 3-edge

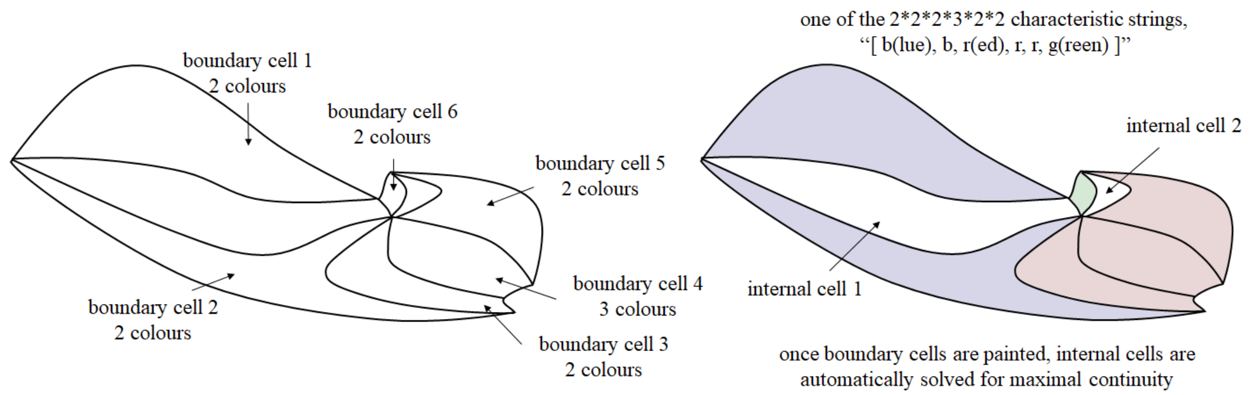

boundary cell 1
2 colours
boundary cell 6
2 colours
boundary cell 5
2 colours
boundary cell 4
3 colours
boundary cell 3
2 colours
boundary cell 2
2 colours
one of the 2*2*2*3*2*2 characteristic strings,
"[b(lue), b, r(ed), r, r, g(reen)]"
internal cell 2
internal cell 1
once boundary cells are painted, internal cells are
automatically solved for maximal continuity

Fig. 9. An intersection-free graph. The internal cell 1 will choose blue > green > other colours. The internal cell 2 will choose green = red > other colours.

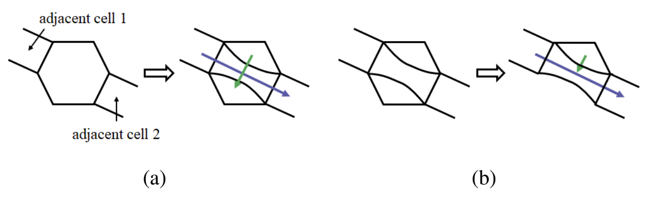


Fig. 10. (a) Illustration of constraint violation due to cell sub-divisions in sub-graphs. One sub-cell will have four adjacent cells. (b) If a cell is scheduled to be divided, it should first be divided and the followed by constructing the intersection-free sub-graphs.

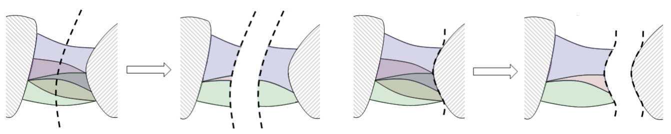


Fig. 11. If a graph is separated through division of multi-colour cells, the constraint of the cell vanishes, then in the sub-graphs there may exist solutions that after being combined is non-optimal, such as the case given in left. If the graph is separated along edges, we will not incur such problem.

cells may form a 4-edge cell which breaks this criterion, 2) is required. ■

With all internal cells being isolated, the graph satisfying the above two constraints is also referred to as a *strip*.

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. Please refer to Fig. 13 for a visualisation of the separation process with an example. One boundary cell is first selected and regarded as an element of strip 1, shown in green in Fig. 13(a). Then, for each adjacent cell of the strip, we check whether accommodating this cell into the strip will violate the constraints discussed earlier. 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. 13(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. 9) will become an internal cell which is adjacent to the internal cell 2, so the construction of strip 1 terminates. To construct strip 2, we check one-by-one whether the cells adjacent to strip 1 can be inserted. After that,

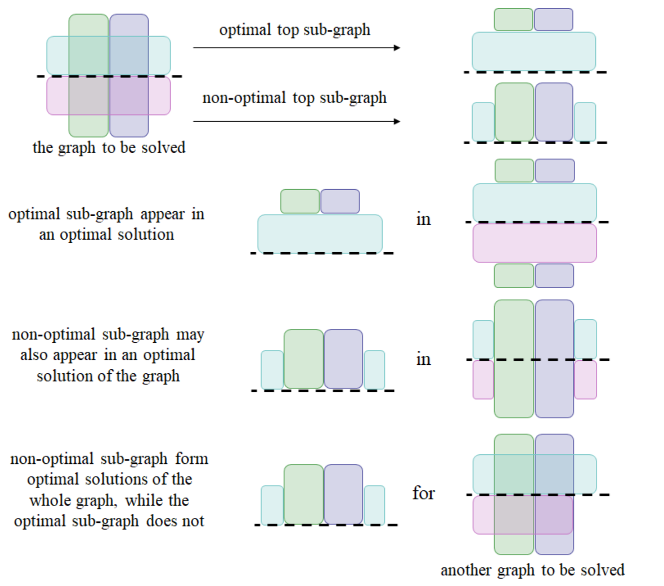

optimal top sub-graph
non-optimal top sub-graph
the graph to be solved
optimal sub-graph appear in an optimal solution
in
non-optimal sub-graph may also appear in an optimal solution of the graph
in
non-optimal sub-graph form optimal solutions of the whole graph, while the optimal sub-graph does not
for
another graph to be solved

Fig. 12. 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.

all adjacent cells to strip 2 are again checked and attempted to be inserted into strip 3. 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 is finally separated into strips, as shown in Fig. 14 where the example graph is separated into 6 sub-graphs. (The process corresponds to line 11 ~ line 15 in **Algorithm** 1.)

C. Solving Strips

After the graph has been separated, all the solutions for each strip are enumerated via iterating through all the possible characteristic strings. Note that there is no extra memory cost for this step - all sub-graph solutions are indexed by the characteristic strings, so they can be randomly accessed to be able to pick up any solution of one sub-graph to combine with another sub-graphs, resulting in a given solution to the original graph.

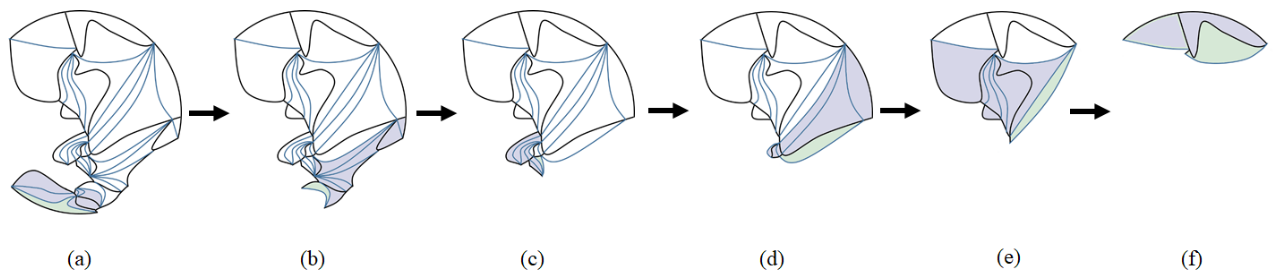


Fig. 13. The concrete steps to separate the graph given in Fig. 14 into intersection-free strips.

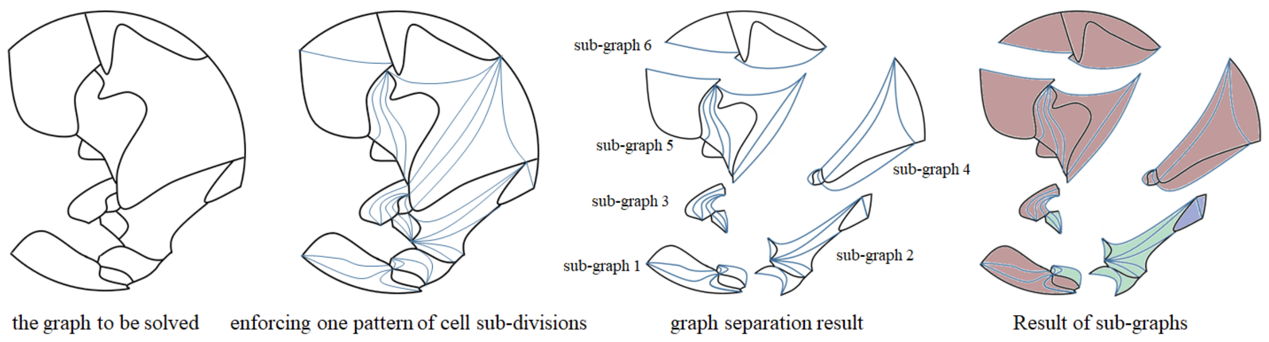


Fig. 14. Illustration of how an unsolved graph is separated into sub-graphs.

D. Full Graph Solution

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 [23]. Once the original graph is re-constructed, 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 20 ~ line 28 in **Algorithm** 1)

IV. Complexity

The target of this section is to prove the exponential improvement on algorithmic complexity over existing approaches, as claimed in abstract and Section I. As mentioned earlier, cell sub-divisions were undertaken before graph separation to avoid the possible constraint violation caused by cell sub-divisions in sub-graphs. 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 19. *Given a topological graph to be solved, assume that it has been properly 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 has following number of steps for enumerative solving*

$$\prod_{j=1}^{M} K_j^{\max\{\alpha_j-2,1\}} \tag{5}$$

Compared to the complexity of the full enumeration solution [23]

$$\prod_{j=1}^{M} K_j^{\max\{\alpha_j-2,1\}} \cdot 2^N \tag{6}$$

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

Proof: The proposed algorithm has been shown to consist of a number of steps:

1) separating a graph into strips,
2) solving each strip individually, and
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

Algorithm 1 Improved Solver

```
Require: The initial graph $G = (\{C_i\}_{i=1}^{M}, \{E_j\}_{j=1}^{N})$
Ensure: All solved graphs $\{result\}$
 1: for all $E_1$ ways of divisions of cell 1 do
 2:     if non-optimality is detected then
 3:         continue
 4:     end if
 5:     $\ddots$
 6:     for all $E_M$ ways of divisions of cell $M$ do
 7:         if non-optimality is detected then
 8:             continue
 9:         end if
10:         // Graph Separation
11:         while the graph is not intersection-free do
12:             Choose a cell to be in the new strip
13:             Attach new cells satisfying constraints
14:             separate the strip from the graph
15:         end while
16:         // Let there be $n$ strips generated
17:         for all characteristic strings of strip 1 do
18:             $\ddots$
19:             for all characteristic strings of strip $n$ do
20:                 Combine strips to form a solution $S$
21:                 if it is better than $\{result\}$ then
22:                     $\{result\} = \varnothing$
23:                     push $S$ into $\{result\}$
24:                 else
25:                     if it is currently optimal then
26:                         push $S$ into $\{result\}$
27:                     end if
28:                 end if
29:             end for
30:             $\iddots$
31:         end for
32:     end for
33:     $\iddots$
34: end for
```

boundary cells

$$\Psi_i = \prod_{j=1}^{r_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \tag{7}$$

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

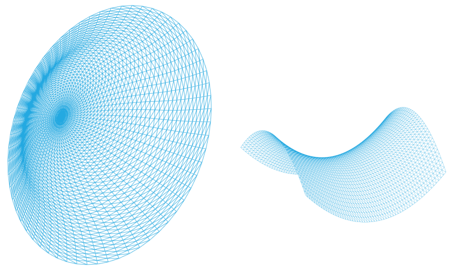

(a) (b)

Fig. 15. The objects used for illustration in Section V.

$$\begin{aligned}\Psi = \sum \Psi_i &= \sum_i \left(\prod_{j=1}^{r_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \right) \\ &\ll \sum_i \left(\prod_{j=1}^{r_i+s_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \right) \\ &\ll \prod_i \left(\prod_{j=1}^{r_i+s_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \right) \\ &= \prod_{j=1}^{M} K_j^{\max\{\alpha_j-2,1\}}\end{aligned} \tag{8}$$

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

$$\begin{aligned}\Xi = \prod \Psi_i &= \prod_i \left(\prod_{j=1}^{r_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \right) \\ &\ll \prod_i \left(\prod_{j=1}^{r_i+s_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \right) \\ &\ll \prod_{j=1}^{M} K_j^{\max\{\alpha_j-2,1\}}\end{aligned} \tag{9}$$

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

$$\Gamma = \Phi + \Psi + \Xi \ll \prod_{j=1}^{M} K_j^{\max\{\alpha_j-2,1\}} \tag{10}$$

■

V. Experimental Results

The proposed solution is tested experimentally by simulating the NCPP with a non-redundant manipulator (6 DoF Universal Robot UR5) on two arbitrarily shaped objects, one a hat-like concave semi-sphere (only one side is considered), and a saddle surface (only one side being considered also), as shown in Fig. 15.

A. Optimal NCPP on a Hat-shape Object

An illustration of the solving process for the hat-shape object is depicted in Fig. 16. 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

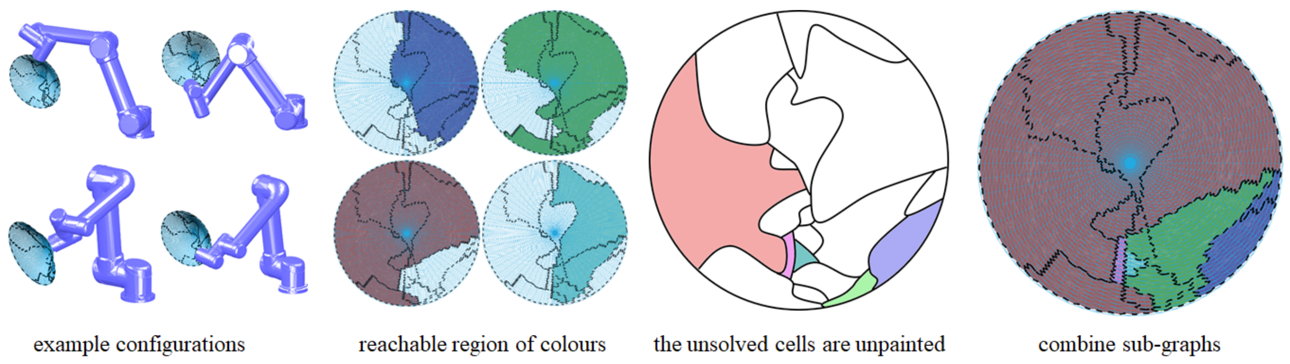


Fig. 16. The coverage task on a hat-shape object. The reachable area of different colours and one of their corresponding configurations are depicted. After painting all 1-colour cells, the unsolved graph is exactly the same one as we have discussed in Section III.

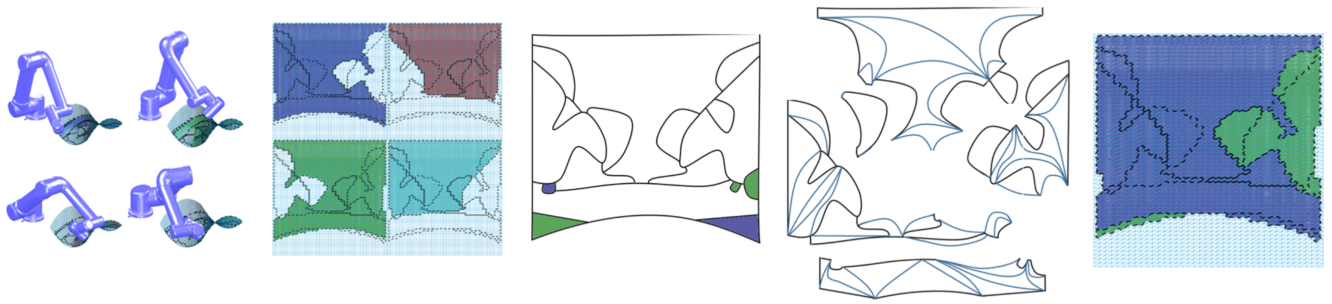


Fig. 17. The coverage task on a saddle surface. The reachable area of different colour and one of their corresponding configurations are depicted. After painting all 1-colour cells, the unsolved grpah is separated into 5 sub-graphs. Finally, all optimal solutions can be collected with 2 end-effector lift-offs.

description provided earlier in Fig. 14. 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 proposed in the literature, 35 edges and all colours of all cells are enumerated, resulting in an algorithmic complexity described by

$$
\begin{aligned}
&\overbrace{2\times 2\times 3\times 2\times 2\times 2\times 2\times 2\times}^{\textbf{all cells in sub-graph 1, shown in Fig. 9}}\\
&2\times 3\times 3\times 2\times 2\times 3\times 2\times 3\times 2\times 3\times\\
2^{35}\times\,&2\times 2\times 2\times 2\times 2\times 2\times 2\times \qquad\qquad \approx 1.2705\times 10^{28}\\
&2\times 3\times 3\times 4\times 4\times 3\times 4\times\\
&3\times 3\times 2\times 3\times 3\times 4\times 4\times 3\times\\
&3\times 4\times 3
\end{aligned}
\tag{11}
$$

Please note there is a minor abuse of notation in the arrangement of the numbers above simply so that they can be easier to compared with the optimal solution below.

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 II and III. Detail results for the others 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

$$
\begin{aligned}
&\overbrace{2\times 2\times 3\times 2\times 2\times 2\times}^{\textbf{internal cells shown in Fig.9}}\\
&2\times 3\times 3\times 2\times 2\times 3\times 2\times 3\times 2\times\\
&2\times 2\times 2\times 2\times 2\times 2\times \qquad\qquad \approx 4.2797\times 10^{14}\\
&2\times 3\times 3\times 4\times 4\times\\
&3\times 3\times 2\times 3\times 3\times 4\times 4\times\\
&3\times 4\times 3
\end{aligned}
\tag{12}
$$

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 challeging 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. 17. For the naive enumeration case, the overall complexity is given by

$$
\begin{aligned}
&2\times 2\times 2\times 2\times 2\times 2\times 2\times 2\times 2\times 2\times\\
&3\times 2\times 2\times 2\times 2\times 2\times 2\times\\
2^{47}*\,&\quad 3\times 4\times 4\times 3\times 2\times 2\times 2\times 3\times 3\times\\
&3\times 2\times 2\times 2\times 2\times 3\times 3\times 2\times 2\times\\
&4\times 4\times 4\times 4\times 4\times 3\times 4\times 4\times\\
&3\\
&\approx 2.924\times 10^{32}
\end{aligned}
\tag{13}
$$

TABLE I
EXPERIMENTAL RESULTS

Object	Number of Iterations	
	Full enumeration [23]	Proposed Optimal
Hat-shaped (Fig. 16)	$1.2705 * 10^{28}$	$4.2797 * 10^{14}$
Saddle-shaped (Fig. 17)	$2.924 * 10^{32}$	$4.3283 * 10^{16}$

Applying the proposed algorithm, the complexity is reduced to

$$\begin{array}{l} 2\times2\times2\times2\times2\times2\times2\times2\times2\times \\ 3\times2\times2\times2\times2\times2\times2\times \\ \quad 3\times4\times4\times3\times2\times2\times2\times3\times \\ 3\times2\times2\times2\times2\times3\times3\times2\times \\ 4\times4\times4\times4\times4\times4\times3\times4\times \\ 3 \end{array} \approx 4.3283\times10^{16} \tag{14}$$

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 47 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.

VI. 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 topological intersections and intersection-free graphs, which permit the separation of a graph into sub-graphs at the points where the multiplicity of optimal solutions originate. The implicit enumeration of these intersections when the graphs are recombined in search of the full solution derive 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. Comprehensive experimental results with step-by-step derivations to illustrate the proposed algorithm are supplied that demonstrate the validity of the novel scheme.

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. M. Kabir, R. K. Malhan, A. V. Shembekar, B. C. Shah, and S. K. Gupta, "Concurrent design of tool-paths and impedance controllers for performing area coverage operations in manufacturing applications under uncertainty," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pp. 1151–1156, IEEE, 2019.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] H. Choset, E. Acar, A. A. Rizzi, and J. Luntz, "Exact cellular decompositions in terms of critical points of morse functions," in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, pp. 2270–2277, IEEE, 2000.

[14] 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.

[15] 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.

[16] 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.

[17] M. Rososhansky and F. Xi, "Coverage based tool-path planning for automated polishing using contact mechan-

ics theory," *Journal of Manufacturing Systems*, vol. 30, no. 3, pp. 144–153, 2011.

[18] 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.

[19] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, "Energy-efficient motion planning for mobile robots," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, vol. 5, pp. 4344–4349, IEEE, 2004.

[20] 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.

[21] 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.

[22] 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.

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

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