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

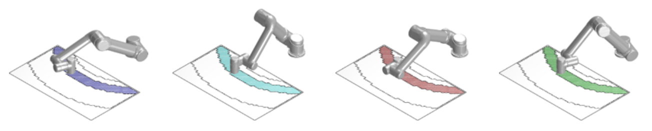
Tong Yang, Jaime Valls Miro, Yue Wang* and Rong Xiong

***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.**

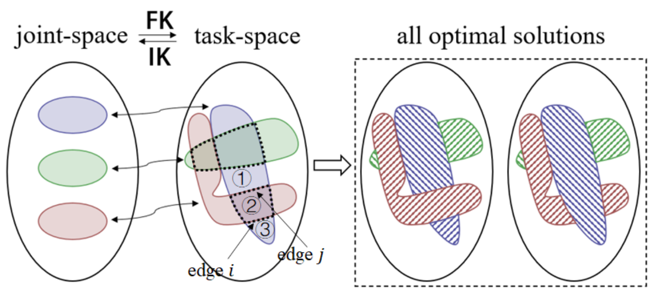
***Index Terms*—Optimal Cellular Decomposition, Manipulator Coverage Task**

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 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.



(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. The 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 correspond to each other).

[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

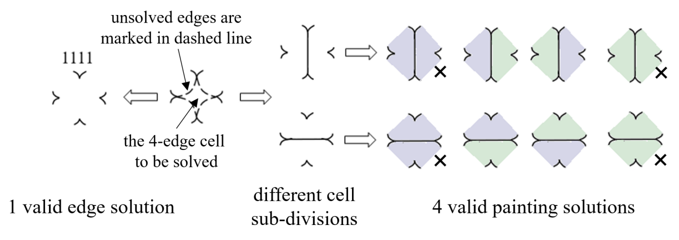

unsolved edges are marked in dashed line
1111
the 4-edge cell to be solved
1 valid edge solution
different cell sub-divisions
4 valid painting solutions

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.

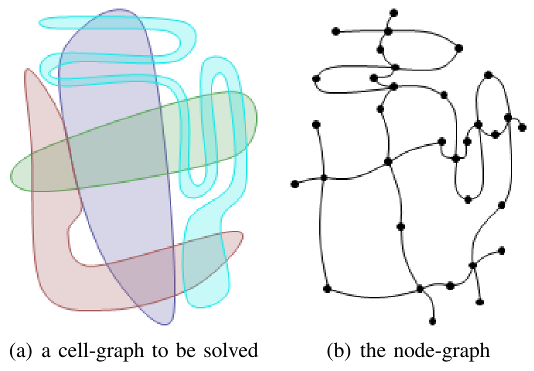


Fig. 3. Quantitive relationship between the cell-graph notation adopted in the manipulator coverage task and the traditional node-graph concepts in classic graph theory. In this example there are 33 vertices, 39 edges, and 8 faces (7 cycles + 1 outer region reaching the infinity) in the node-graph, which correspond to 33 cells and 39 edges in the cell-graph.

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.

This is however an intensely computational process, growing exponentially in the size of the graph [1].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.

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

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

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, in the order of 2^N.

[1]For a better understanding of the ensuing terminology, the reader is referred to the preliminary definitions set out in Appendix.

It should be noted that an assumption of simply-connected cells underlies the structure of the proposed solution. As it has already been proven [24] that for graphs with multiply-connected cells, iterative enumeration can reduce the problem to that of a simply-connected cell graph, the proposed solver improvement is generic to any type of cell connectivity.

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.3, 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 [25] 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 relationship highlights the large number of edges in a cell-graph, $F - 2$ greater than the number of cells:

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

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 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 are 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, named *intersection*. The enumeration of intersections will be shown to be the core element in the graph solver leading to notable inefficiencies when solving a full graph with intersections, versus solving an equivalent intersection-free graph.

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. A typical application of topological invariants in mathematics is for distinguishing two different objects: first the topological invariant is defined, then two objects can be shown to be different topological invariants, hence not equivalent under the modifications that preserve the topological invariant. However, unlike this mainstream usage of topological invariants, in this work the key observation is not leveraging intersections, but how to avoid intersections altogether.

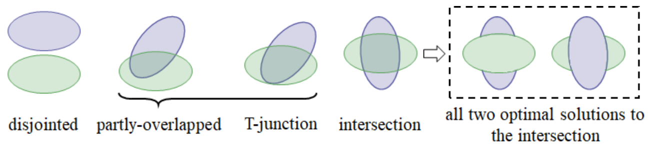


Fig. 4. Illustration of the basic relationships between the reachable area for a two-colour case. In all but the intersection scenario, each colour can keep itself fully connected. T-junction can be seen as a special instance of the generic partly-overlapped case, as it can be transformed back to the generic case through continuously modifying the boundary of the colours. Only the intersected 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 connecting area.

The methodology in this section aims at constructing "counter-examples" that restrict the potential directions any algorithmic complexity improvements may take. As will be concluded, revealing all optimal solutions requires the complete set of intersections to be enumerated. Yet enumerating intersections increases the complexity of any solver by a multiplicative factor. As such, existing algorithms and any likely improvements to solve the graph should note the "explicit" reduction in the enumeration of intersections, and look for a minimalistic set of them to be solved. In that regard, it makes the simplified representation adopted in this work of a wholly intersection-free graph apparent, and points towards an intersection-free graph separation scheme which ensures all intersections are placed at common boundaries of sub-graphs, presented in detail in Section III.

A. Definition of Intersection

Definition 1. *(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

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

In regards to intersections and graph solving it is worth noticing that painting the colour of one cell is equivalent to removing all the other possible colours it could be painted with. Hence, as a graph is being solved the distribution of coverable region by the different colours keeps changing, unconsciously leading to the possible elimination of some of the intersections. Generally, a graph to be solved admits intersections, whilst a solved graph where each cell only preserves one colour is intersection-free. Reducing the number of intersections until none remains is the target of the graph solver.

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

Proposition 2. *(Invariance under homotopic cutting paths) In a (partly-solved) graph, any continuous modification of the*

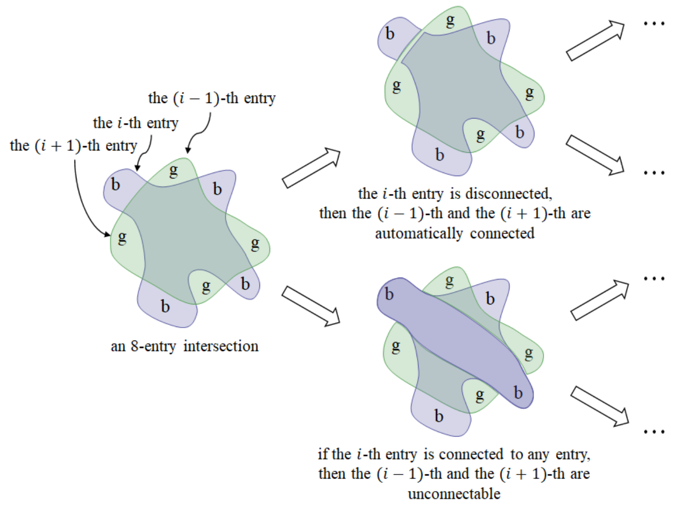

the $(i-1)$-th entry
the i-th entry
the $(i+1)$-th entry
an 8-entry intersection
the i-th entry is disconnected, then the $(i-1)$-th and the $(i+1)$-th are automatically connected
if the i-th entry is connected to any entry, then the $(i-1)$-th and the $(i+1)$-th are unconnectable
b
g
$\cdots$

Fig. 5. An example with an 8-entry intersection is depicted on the left. The figure also illustrates the process of solving a single intersection with a simpler example where adjacent entry cells have been assigned alternate colours "b(lue), g(reen), b, g, $\cdots$". It is apparent from the two token scenarios shown that an exhaustive enumeration will be required to obtain all optimal cellular decompositions. It will be established later in **Corollary** 5) that given $n = 8$ entries, the minimum number of connected regions can be directly calculated as 5.

cell cutting paths will not reduce the number of intersections in the graph.

Proof: It has been shown in **Lemma** 22 that all cutting paths (and preserved edges) need not be intersected, hence the continuous modification of cutting paths will not change the connectivity of any cell. That means there will not be a case whereby two disconnected cells with the same colour become connected after continuous modification of cutting paths, nor a case that a single solved cell is truncated 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. 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.

B. Intersection Properties

The intersection properties defined in this section represent a "drawback" that limits the emergence of more efficient algorithms. Note that these pitfalls do not directly contribute to the algorithm design itself, but indicate a possible upper bound in algorithmic complexity improvements. For convenience in the discussion of subsequent properties, the *entries* of an intersection are first defined.

Definition 3. *(Entries) Given an intersection, its entries are the single-colour surrounding cells, in order.*

Refer to Fig. 5 for a simple illustration of entries in a graph section containing an intersection, with a central 2-colour (green, blue) cell, and 8 entries. More complex scenarios can also be seen depicted in Fig. 8, with increasingly complex combinations of multi-colour cells to illustrate the intrinsic challenge in counting intersections in graphs except for the most trivial case.

Lemma 4. *(Multiplicity of Solutions) The existence of intersections causes the multiplicity of optimal NCPP solutions.*

Proof: For the most apparent example see Fig. 4. There are two obvious optimal solutions to the right-most intersected case: filling in the central cell fully with 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. Assuming a graph has been solved for all but one intersection, with m solutions already constructed, labelled $s_1, \cdots, s_m$. Let there be n different solutions for the left-over intersection, labelled $t_1, \cdots, t_n$, then $m \times n$ solutions for the whole graph will be constructed. Should the solution (s_i, t_j) be (non-)optimal, $(s_i, t_{j'}), \forall j' \in \{1, \cdots, n\}$ would also be (non-)optimal. ■

Corollary 5. *(Unavoidability of Enumeration) All intersections have to be enumerated in order to identify all optimal NCPP solutions.*

We prove this by showing two aspects:

1) Collecting all solutions of a single intersection requires enumeration.

   Proof: It is proven by example that even a structurally simplistic intersection (a generic multi-entry intersection), needs to undergo an enumerative solving mechanism. The intuitive dual-colour intersection shown in Fig. 4, with 4 "entries", is extended to a more generic case with an arbitrary even number of entries, as the case depicted in in Fig. 5, with 8 entries. Let the intersection have n entries, with consecutive adjacent cells having been assigned alternate colours, "b(lue), g(reen), b, g, $\cdots$". It is observed that for each adjacent cell (say the i-th, labelled cyclicly), if it is connected to any other adjacent cell through 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 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 entry, two scenarios must then be separately regarded, as illustrated in Fig. 5: (a) keeping itself disconnected so that entries $(i-1)$ and $(i+1)$ can be connected (Fig. 5, top-right), or (b) connecting it to other entries (Fig. 5, bottom-right). Both of them go to optimal cellular decompositions. It is intuitive that more complex intersections can be devolved into simpler cases through connecting entries, thus transforming higher-entry cases into simpler cases as the example above, ultimately requiring solving through enumerative cellular decomposition regardless. ■

2) Intersections are not removed by graph separation.

   Proof: This can be revealed in solving a simple

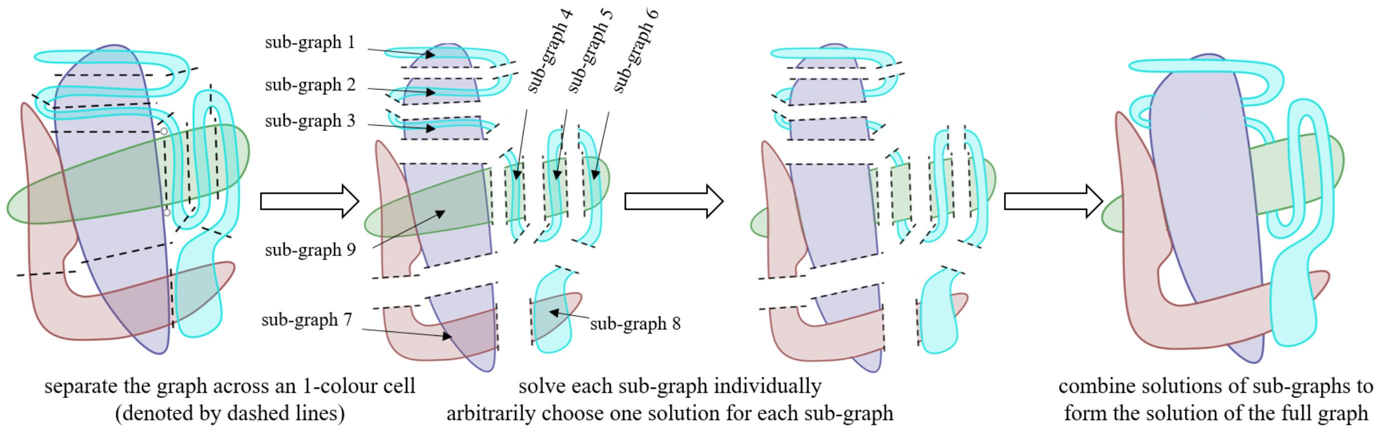

sub-graph 1
sub-graph 2
sub-graph 3
sub-graph 4
sub-graph 5
sub-graph 6
sub-graph 9
sub-graph 7
sub-graph 8
separate the graph across an 1-colour cell (denoted by dashed lines)
solve each sub-graph individually arbitrarily choose one solution for each sub-graph
combine solutions of sub-graphs to form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

graph, a maximally 2-overlapped graph, 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 the 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. ■

The examples employed to illustrate the various scenarios discussed thus far are relatively simplistic, and include graphs where intersections could be easily distinguished. However, for more generic graphs the necessity of solving by full enumeration is compounded by the inability to fully identify the intersections.

Lemma 6. *(Uncountability of Intersections) Given a graph, except for the most trivial cases (with either 0 or 1 intersection), the number of intersections in the graph is not countable.*

Proof: Fig. 6 sub-graph 9 illustrates the case where two parallel (non-overlapping but touching) colour regions exist 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. A more detailed study of this case is provided in Fig. 7. It can be easily observed how the graph has only one intersection, as after the intersection between red colour and green colour is resolved, the graph will have no further intersections. However, a combination of intersection-free graphs can generate an intersection-one graph, namely “0” + “0” = “1” (Fig. 7(b)). Similarly, after the blue colour (with an apparent intersection with the green colour) is removed, the remaining graph still has an intersection, namely “1” - “1” = “1” (Fig. 7(c)). A more generalised set of cases can be encountered as shown in Fig. 8. It is apparent how the introduction of more and more colours leads to highly coupled intersections, where even identifying intersections becomes non-trivial. ■

Corollary 7. *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 any graph solvers (including all existing solvers, and the one to be proposed in this work) will face an enumerative scheme, where there is no guarantee of an explicitly decreasing number of intersections as a graph gets gradually solved.

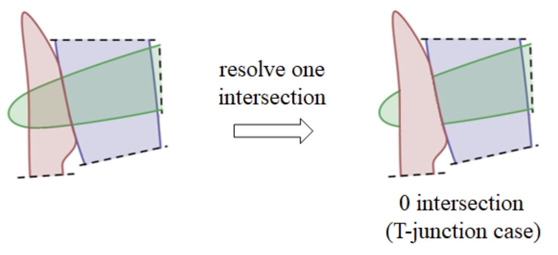
C. Intersection-Free Graphs

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 [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 plays a crucial role.

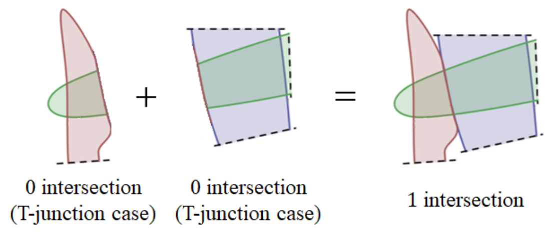
Definition 8. *(Intersection-free Graph) An intersection-free graph is a graph where no intersection can be constructed within it.*

Two types of cells can be distinguished in a graph:

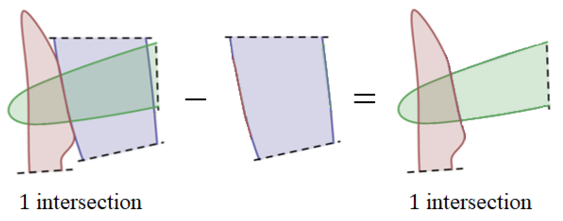
Definition 9. *(Boundary Cell, Internal Cell) Boundary cells are the cells which have edges forming the boundary of the graph. The remaining cells are internal cells which have no edge exposed to the graph boundary.*



(a) A graph with 1 intersection, removed after solving (intersection cell allocated red colour).


0 intersection (T-junction case)
0 intersection (T-junction case)
1 intersection

(b) Two intersection-free graphs form a graph with 1 intersection.



(c) After removing the blue colour together with its intersection with the green colour, the subgraph remaining still has 1 intersection.

Fig. 7. Examples of intersection countability complexity as graphs gets resolved, depicting the interesting scenarios where (b) intersections are created from 0 intersection subgraphs ("0" + "0" = "1"), and (c) the same number of intersections remains even after removal of one of them ("1" - "1" = "1").

See Fig. 9 for illustration of an 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 10. *(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 11. *The solution of the graph, i.e., the assigned colour of all cells in the graph, can be uniquely characterised by the colour of the boundary cells.*

Proof: 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 from being connected. However, this represents an intersection, which contradicts the intersection-free assumption 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 likened to the full quantification of the number of solutions in the graph. ∎

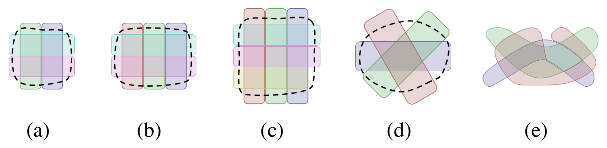


Fig. 8. Graphs with ever increasing complex intersection scenarios. Intersections are contained with the encircled dashed line, whilst intersection entries lay in the outside. Intersection in these examples are formed by: (a) 2&2 parallel colours, (b) 2&3 parallel colours, (c) 3&3 parallel colours, (d) a normal 3-overlapped intersections, and (e) a complex 3-overlapped graph where delineation of the intersecting section in the graph is not apparent given the many intersections and paralleled colours within.

III. Graph Separation

An efficient strategy to separate a graph into intersection-free sub-graphs is described in this section. The reader is referred to the algorithm pseudocode given in **Algorithm** 1. 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 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.

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.

A. Definition of Strips

Definition 12. *(Strip) A strip is a graph designed such that*

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

Proposition 13. *Strips are intersection-free graphs.*

Proof: It is observed that when a section of a graph contains an intersection, there must exist at least 4 entries. Moreover, each entry may not occupy a single edge. These motivate us to consider its opposite: if condition 1) in **Definition** 12 is satisfied, then there can not possibly be an intersection. In noting that two internal 3-edge cells may form a 4-edge cell which breaks this criterion, condition 2) is required. ∎

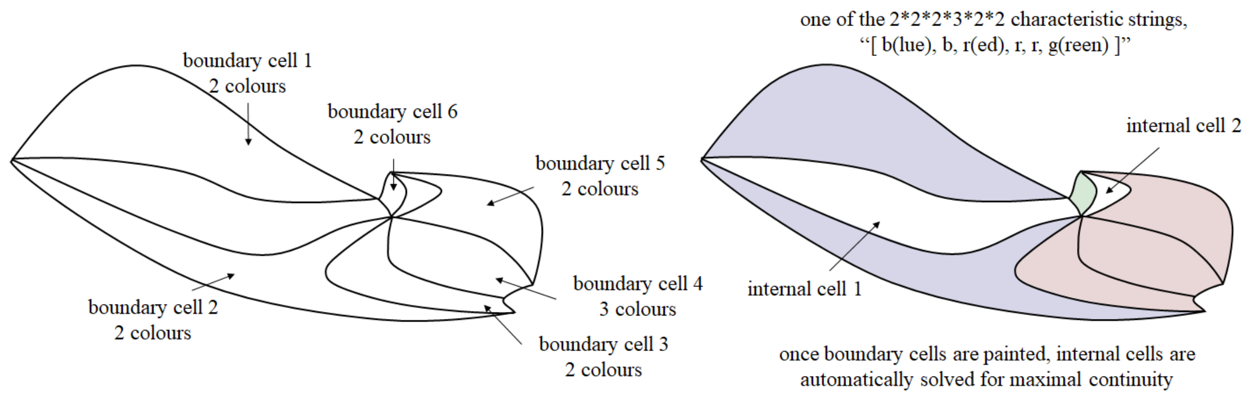

boundary cell 1
2 colours
boundary cell 6
2 colours
boundary cell 5
2 colours
boundary cell 4
3 colours
boundary cell 3
2 colours
boundary cell 2
2 colours
one of the 2*2*2*3*2*2 characteristic strings,
"[b(lue), b, r(ed), r, r, g(reen)]"
internal cell 2
internal cell 1
once boundary cells are painted,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

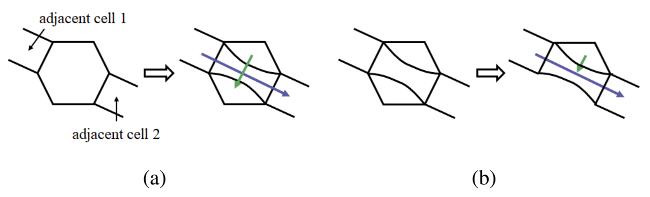


Fig. 10. (a) Illustration of constraint violation due to cell sub-divisions in sub-graphs. One sub-cell will have four adjacent cells. (b) If a cell is scheduled to be divided, it should first be divided and then followed by constructing the intersection-free sub-graphs.

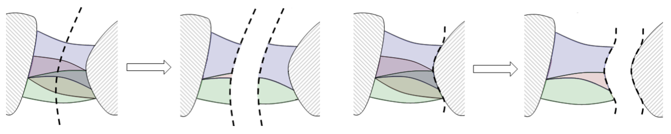


Fig. 11. If a graph is separated through division of multi-colour cells, the constraint of the cell vanishes, then in the sub-graphs there may exist solutions that after being combined is non-optimal, such as the case given in left. If the graph is separated along edges, we will not incur such problem.

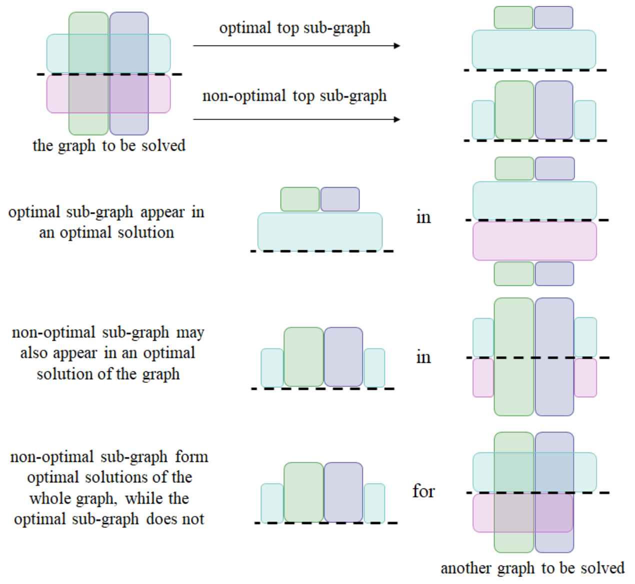

optimal top sub-graph
non-optimal top sub-graph
the graph to be solved
optimal sub-graph appear in an optimal solution
in
non-optimal sub-graph may also appear in an optimal solution of the graph
in
non-optimal sub-graph form optimal solutions of the whole graph, while the optimal sub-graph does not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

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 light green in Fig. 13(a). Each cell adjacent to the strip is checked to validate whether accommodating the cell into the strip will violate the strip constraints (**Definition** 12). 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 (Fig. 13(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** 12. After that, another cell in the left-out graph is chosen into strip 3 (Fig. 13(c)), and other cells are again checked and attempted to be inserted 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. 14 where the graph is separated into 6 strips. (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 strip solutions are indexed by the characteristic strings, so they can be randomly accessed to be able to pick up any solution of one strip to combine with other strips, resulting in a given solution to the original graph.

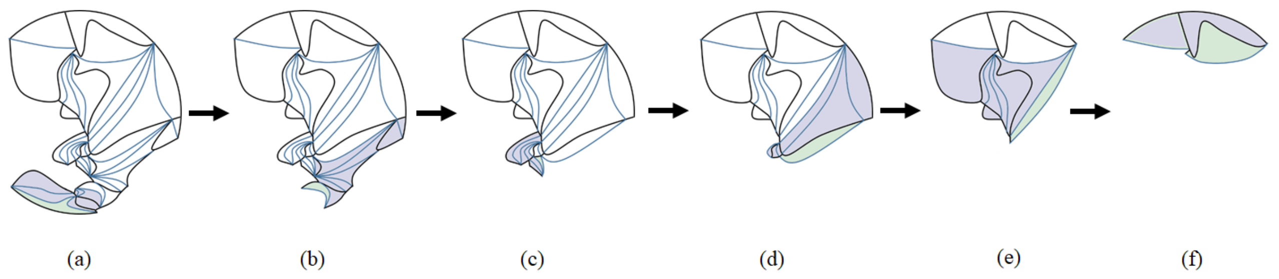


Fig. 13. Step-by-step illustration of the separation of the graph shown in Fig. 14 into intersection-free strips. The light green cell is the first one selected to form a new strip, and all light purple cells are then inserted into the same strip. The graph ends up with 6 separated strips.

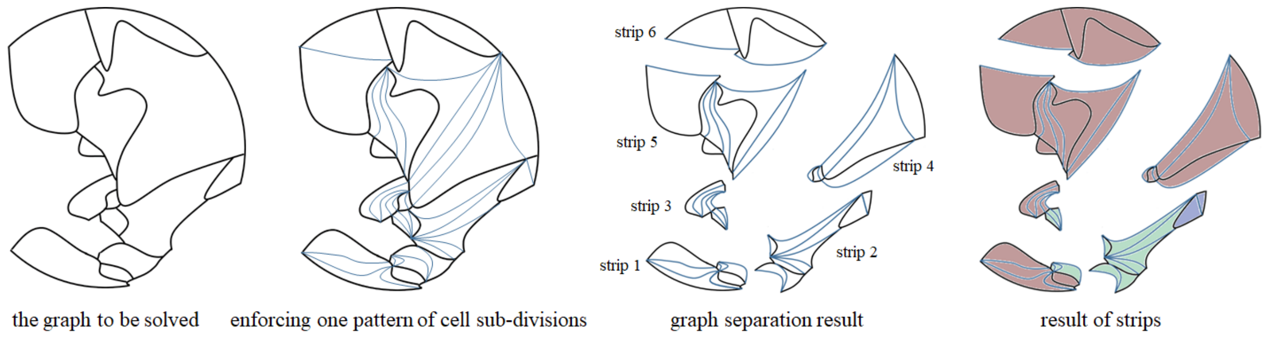


Fig. 14. Illustration of how an unsolved graph is separated into strips.

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 22 ~ line 30 in **Algorithm** 1)

IV. Complexity

The advancement that the proposed mechanism brings about in solving the NCCP 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 14. *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:*

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

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 on a number of steps, namely:

1) Separating a graph into strips
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

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 into Strips (Section III)
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:         // Enumerate solutions by characteristic strings
18:         // (Theorem 11)
19:         for all characteristic strings of strip 1 do
20:             $\ddots$
21:             for all characteristic strings of strip $n$ do
22:                 Combine strips to form a solution $S$
23:                 if it is better than $\{result\}$ then
24:                     $\{result\} = \varnothing$
25:                     push $S$ into $\{result\}$
26:                 else
27:                     if it is currently optimal then
28:                         push $S$ into $\{result\}$
29:                     end if
30:                 end if
31:             end for
32:             $\iddots$
33:         end for
34:     end for
35:     $\iddots$
36: end for
```

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

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

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

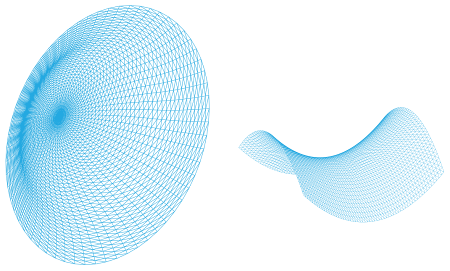

(a) (b)

Fig. 15. The objects used to experimentally illustrate the results, 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

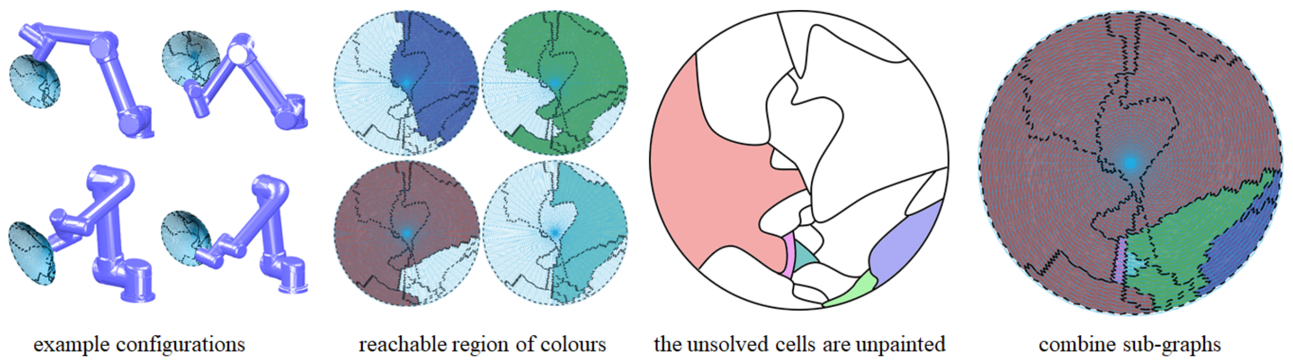


Fig. 16. The coverage task of a hat-shape object. The reachable area of different colours and one of their corresponding configurations are depicted. After painting all 1-colour cells, the unsolved graph is separated and solved as decribed in Section III, with the step-by-step process illustrated in Fig. 13

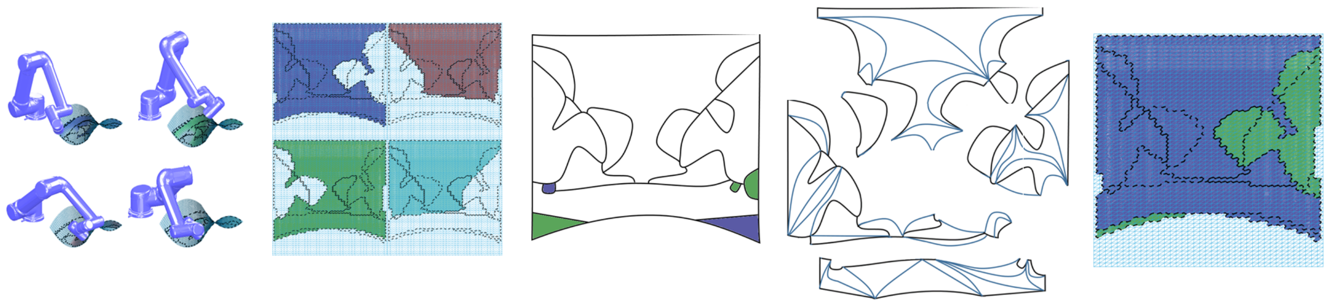


Fig. 17. The coverage task of a saddle surface. The reachable area of different colour and one of their corresponding configurations are depicted. After painting all 1-colour cells, the unsolved grpah is separated into 5 sub-graphs. Finally, all optimal solutions can be collected with 2 end-effector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

$$\begin{array}{ll} & \overbrace{2\times2\times3\times2\times2\times2\times2\times2\times}^{\textbf{all cells in sub-graph 1, shown in Fig. 9}} \\ & 2\times3\times3\times2\times2\times3\times2\times3\times2\times3\times \\ 2^{35}\times & 2\times2\times2\times2\times2\times2\times2\times \qquad \approx 1.2705\times10^{28} \\ & 2\times3\times3\times4\times4\times3\times4\times \\ & 3\times3\times2\times3\times3\times4\times4\times3\times \\ & 3\times4\times3 \end{array} \tag{11}$$

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

$$\begin{array}{l} \overbrace{2\times2\times3\times2\times2\times2\times}^{\textbf{internal cells shown in Fig. 9}} \\ 2\times3\times3\times2\times2\times3\times2\times3\times2\times \\ 2\times2\times2\times2\times2\times2\times \qquad \approx 4.2797\times10^{14} \\ 2\times3\times3\times4\times4\times \\ 3\times3\times2\times3\times3\times4\times4\times \\ 3\times4\times3 \end{array} \tag{12}$$

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

$$\begin{array}{ll} & 2\times2\times2\times2\times2\times2\times2\times2\times2\times2\times \\ & 3\times2\times2\times2\times2\times2\times2\times \\ & \quad 3\times4\times4\times3\times2\times2\times2\times3\times3\times \\ 2^{47}* & 3\times2\times2\times2\times2\times3\times3\times2\times2\times \\ & 4\times4\times4\times4\times4\times4\times3\times4\times4\times \\ & 3 \\ \approx 2.924\times10^{32} & \end{array} \tag{13}$$

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

$$\begin{array}{l} 2\times 2\times 2\times 2\times 2\times 2\times 2\times 2\times 2\times \\ 3\times 2\times 2\times 2\times 2\times 2\times 2\times \\ \quad 3\times 4\times 4\times 3\times 2\times 2\times 2\times 2\times 3\times \\ 3\times 2\times 2\times 2\times 2\times 3\times 3\times 2\times \\ 4\times 4\times 4\times 4\times 4\times 4\times 3\times 4\times \\ 3 \end{array} \approx 4.3283\times 10^{16} \qquad (14)$$

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 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. Comprehensive experimental results with step-by-step derivations to illustrate the proposed algorithm are supplied that demonstrate the validity of the novel scheme.

Appendix: Preliminary Definitions

Given the kinematics of a non-redundant manipulator, the pose and surface mesh of the object to be traversed, the obstacles within the robot workspace in the surrounding environment, and their relative poses, each point on the surface can be reached by the end-effector through a finite number of Inverse Kinematic (IK) solutions. The following topological elements are well-defined:

Definition 15. *(Valid Configuration) A valid configuration is the non-singular [3] collision-free manipulator configuration such that the end-effector rests on the object surface with its orientation parallel to the surface normal.*

Remark 16. *When singular configurations are disregarded, the Forward Kinematics mapping from manipulator configuration to end-effector pose space is locally one-to-one.*

Definition 17. *(Colour) Each valid manipulator configuration is represented by a colour, with continuous configurations being assigned the same colour. 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.*

Definition 18. *(Cell) A cell is a maximally-connected region on the surface, whereby all points within 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 a cell.*

Remark 19. *The geometric coverage path within a single cell (e.g. boustrophedon [10], spiral path [11]) 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.*

Definition 20. *(Edge) An edge is delineated by the common boundary between two cells.*

Definition 21. *(Graph) A graph is the combination of all cells and their connectivities, i.e., the edges.*

Given the NCPP problem to be solved, an initial graph can be constructed. *Solving* the graph formally refers to the process of preserving only one of the possible colours for each point by optimally merging cells to form larger cells, or split them with cutting paths in an iterative fashion to create a task-space cellular decomposition with the minimum number of resulting cells, such that

1) Each resulting cell is connected.
2) Resulting cells are non-overlapped.
3) The union of all resulting cells fill the whole task-space.

Earlier work has proven that [23] the number of cells in the initial topological graph is finite. Moreover, the following lemma ensures the finiteness of all possible topological-distinct cellular decompositions, amongst which all optimal cellular decompositions lie.

Lemma 22. *Different parts of a cell may be painted with different colours provided that the design of the cell-cutting paths satisfy that:*

1) *It is sufficient to consider cutting paths that 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.*

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 mechanics 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. 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.

[24] 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.

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