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

***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.**

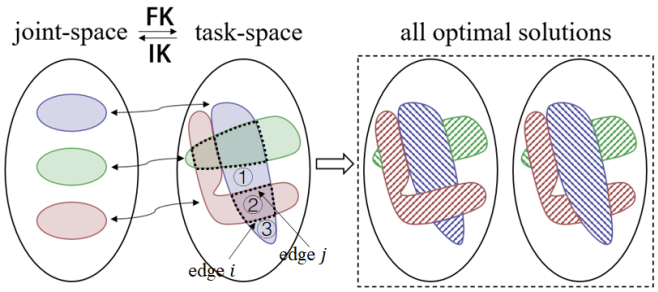

Fig. 1. A toy illustration of the problem arising when planning optimal non-revisiting coverage paths with a manipulator: valid robot configurations form disjoint sets in joint-space, while their task-space FK mapping images (end-effector poses) may overlap. In the example, which for simplicity is illustrated 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 respesctively) 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 VI). 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.

I. Introduction and Related Work

The problem is motivated by the non-revisiting coverage path planning (NCPP) along the surface of an object with a non-redundant manipulator. Given the object's surface, the manipulator, the surrounding obstacles in the workcell and their related poses, for each reachable point on the surface there exist a finite number of inverse kinematic solutions in the manipulator configuration space. Disregarding singular configurations [13], 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 [6], deburring [11], welding [5], 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 transition between joint-space sets, or lift-offs.

Early reports on the generic *coverage path planning* (CPP) problem focused on geometric path designing, particularly for mobile platforms operating in planar surfaces, such as boustrophedon [1] or spiral paths [3]. 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 [2] [4], applied directly onto the area to be covered. This is an ineffective strategy when transfered 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 [10]. Many criteria in the classic CPP problems have been adapted to the manipulator NCPP task, such as time to completion [7] or energy consumption [8], and optimal mobile manipulator pose for a given coverage in task-space has also been investigated [9]. However, optimising end-effector lift-offs for increased smoothness and continuity

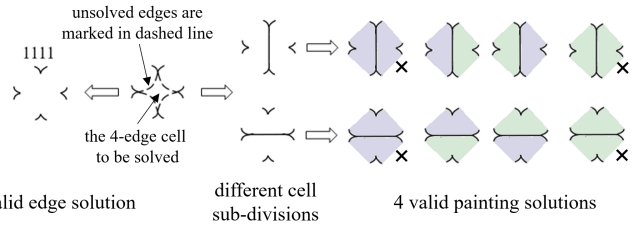

Fig. 2. An illustration of the many-to-one relation between edge and painting solutions. Let a cell with 4 adjacent cells (a 4-edge cell) have 2 possible colours "blue" and "green", then $\alpha = 4, K = 2$. From the point of view of solving edges, the binary number 1111 specifies one solution, but the 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 withe 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 possibly colour for all sub-cells), shown crossed-out, leaving the remaining four valid painting solutions.

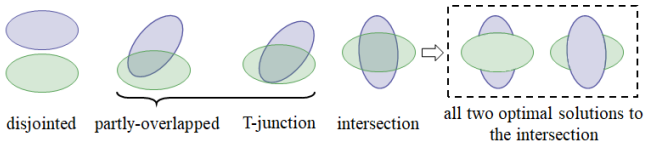

Fig. 3. Illustration of basic relations between the reachable area of two colours. T-junction is a special case of the partly-overlapped distribution of colours, while it can be transformed back to a normal case through continuously modifying the boundary of colours. Only the intersected case needs solving, and it is apparent that the two optimal solutions are not topological equivalent, since they have different number of connected regions in blue and green colour.

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 [12], 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 which 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. 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 [12], 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 appears 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 [12]. 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 (α_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, 2^N.

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 as described in Section IV. Details about the complexity advantage in solving the problem following the proposed strategy is mathematically proven in Section V, whilst experimental results from simulations are collected in Section VI. Final concluding remarks are gathered in Section VII.

II. Optimality via "Topological Intersections"

In this section we introduce a topological invariant variable, *intersection*, whose enumeration is the core element leading to notable inefficiencies in solving the full graph. The simplified representation of all solutions of an intersection-free graph is then made apparent.

A. Defining Intersections

An intersection is a class of overlapping between coverable region with two colours, describing an inevitable interruption of the connectivity of one colour by other colours. The coverable area of two colours may be fully disjointed, partly overlapped or intersected, as shown in Fig.3, 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. And obviously there are two optimal solutions to this

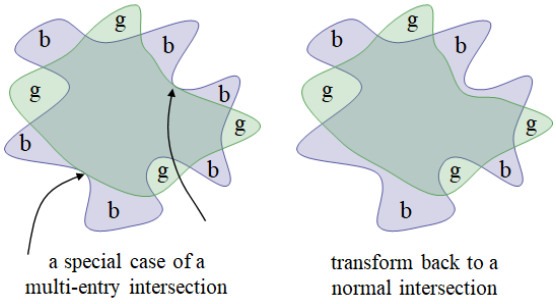

Fig. 4. An intersection with 8 entries. For some special cases that two consecutive adjacent cells can be filled in same colour, through shrinking the boundary of colour, it can be transformed back to the normal case.

intersected case, filling in the central cell full of either blue or green. In more complicated graphs, such an intersection may introduce two set of optimal solutions, in one set all solutions keep one colour connected, while in the other set the other colour remains connected. Seeing the intersection in Fig.3 as 4 entries, the intersection with arbitrary number of entries is intuitive, as shown in Fig.4. After simplifying the intersection 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 an intersection, then another two entries, $(i-1)$ and $(i+1)$, cannot be connected. So the minimal number of continuous 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.

The core role of intersections in graphs can be revealed in solving a simpler case, a separable maximally 2-overlapped graph, illustrated by Fig. 5. The key step is to separate the graph into sub-graphs through 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 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 full combination of the different solutions with intersections, as further described in Section IV.

There are a number of interesting local properties in the concept of graph intersecting. Fig.5 (sub-graph 9) illustrates the case where two parallel (unoverlapping but touching) colour region 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. The generalised cases of intersections that can be encountered are shown in Fig.6. It is apparent how as more colours get involved, intersections become highly coupled, and even identifying intersections is not trivial. There is certainty in the fact that the number of intersections in a partly-solved graph will not reduce under continuous modification of cell cutting paths. Disregarding the position of intersections and how to solve them leads to exhaustive implicit enumeration when all edges and cells are enumerated [12]. 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.

B. Intersection-Free Graphs

An intersection-free graph is a graph where no intersection can be constructed within it. Two types of cells can be distinguished in an intersection-free graph, a *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. It is apparen that all internal cells are shown to have less than 4 cells, or else an intersection could be constructed within. See Fig. 7 for an illustration of the intersection-free graph with 6 boundary cells and 2 internal cells.

A crucial property of intersection-free graphs is that the connectivity of the graph can be uniquely characterised by the colour of the boundary cell. 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 edges being connected. However, this represents an intersection, which contradicts the assumption of intersection-free property of the graph.

The colour of the (ordered) boundary cells can be recorded in a string with length equalling the number of boundary edges of the graph, refered to as the *characteristic string* of the graph. After assigning each boundary cell a colour, all internal cells will be assigned the colour which maximises the connectivity to its adjacent cells. After all cells have their colour, all internal edges are automatically solved. So each characteristic string uniquely corresponds to a unique solution of the intersection-free graph, and the number of characteristic strings we can be used to quantify 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-division after sub-graphs have already been constructed, this constraint may be violated (see Fig.8 for an

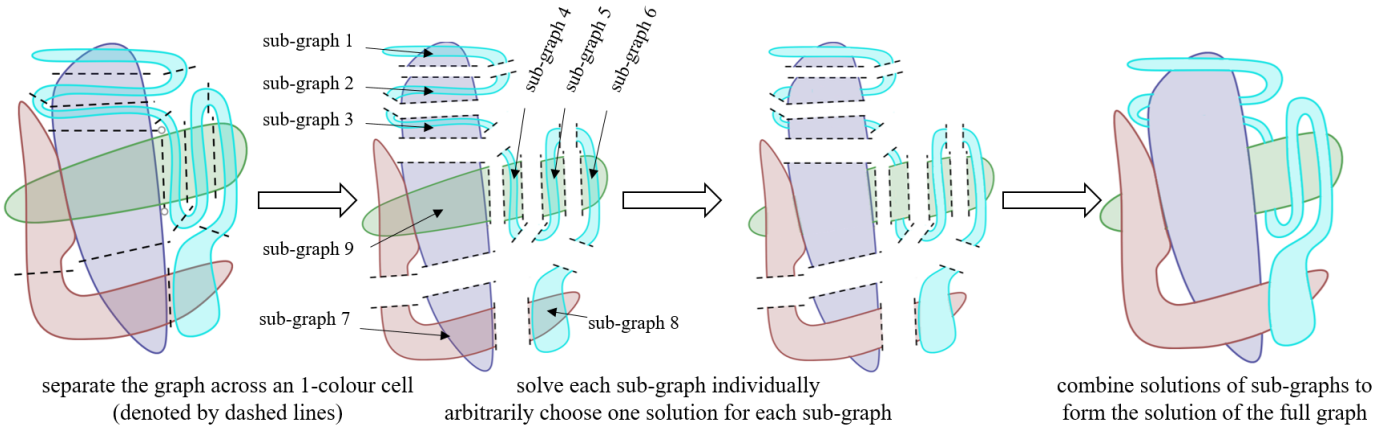

Fig. 5. A maximally 2-overlapped graph to show the role of intersections. The full graph (left) is separated into 9 sub-graphs through 1-colour cells, shown with dashed cutting lines. Sub-graph 1-8 has been shown to have two optimal solutions, while sub-graph 9 gets unique optimal solution after enumerative solving. Note that whatever solution gets choosen 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 is 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.

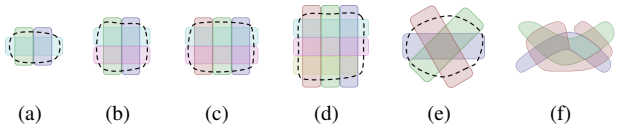

Fig. 6. Demonstrating graphs with various forms of intersections. The dashed circle marks a part of the graph which contains intersections. Intersection formed by: (a) 2&1 parallel colours, (b) 2&2 parallel colours, (c) 2&3 parallel colours and (d) a further special case of 3&3 parallel colours (where whether the orange colour intersects with the blue colour is closely related to the optimal solutions. Now they only form a T-junction), (e) a normal 3-overlapped intersections and (f) a complicated 3-overlapped graph which we cannot even figure out where is the intersection given too many intersections and parallelness of colours in it.

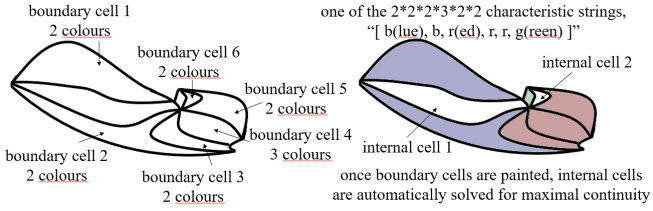

Fig. 7. An intersection-free graph. The internal cell 1 will choose blue $>$ green $>$ other colours. The internal cell 2 will choose green $=$ red $>$ other colours.

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 from this 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 9. 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

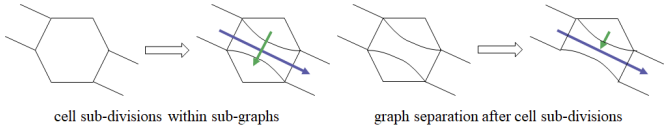

Fig. 8. Illustration of constraint violation due to cell sub-divisions in sub-graphs. If a cell is scheduled to be divided, it should first be divided and the followed by constructing the intersection-free sub-graphs.

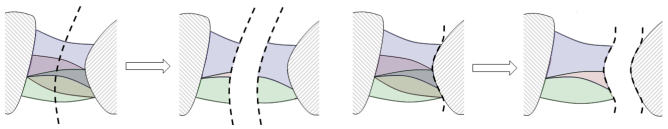

Fig. 9. If a graph is separated through division of multi-colour cells, the constraint of the cell vanishes, then in the sub-graphs there may exist solutions that after being combined is non-optimal, such as the case given in left. If the graph is separated along edges, we will not incur such problem.

Fig 10.

A. *Defining Strips*

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 in the sub-graph it is enforced that (1) each cell has at most 3 adjacent cells, then it can not possibly be an intersection. In noting that two internal 3-edge cells may form a 4-edge cell which breaks this criterion, it is enforced that (2) the internal cells are nonadjacent. A topological graph satisfying the above two constraints is 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. 12 for a visualisation of the separation process with an example. One boundary cells is first selected

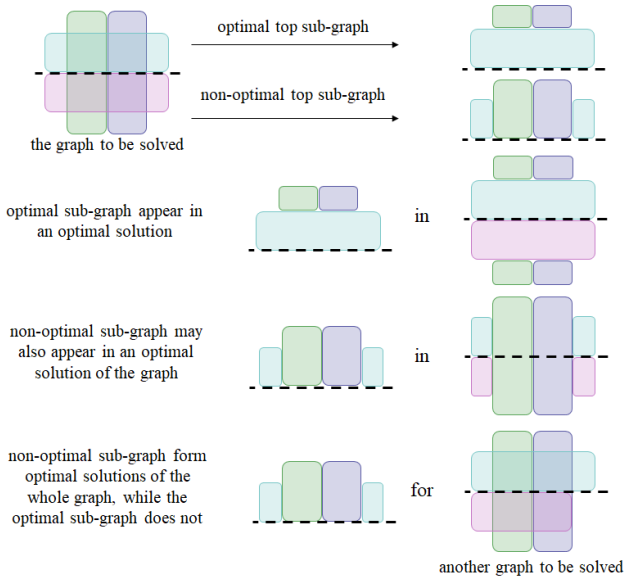

Fig. 10. Separating the graph through the dashed line, we show the optimal solution and a non-optimal solution of the sub-graph. It is noticeable that both of them form optimal solutions of the original graph. An in the bottom case, as a sub-graph of another graph, the non-optimal sub-graph can form optimal solutions while the optimal sub-graph cannot do so.

and regarded as elements of strip 1, shown in green in Fig.12(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.12(a), adjacent cells are iteratively inserted to strip 1 which are shown in blue. If one more cell is inserted, then the boundary cell 5 (illustrated in detail in Fig.7) 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, all adjacent cells to strip 2 are again checked and attemped to be inserted into strip 2. 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.11 where the example graph is separated into 6 sub-graphs.

C. Solving Sub-Graphs

After the graph has been separated, all the solutions for each sub-graph are enumerated via iteraring 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 string, 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.

IV. 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 through comparing the colour of its two adjacent cells. A cost is calculated whose physical meaning is the number of connected regions in the painted part of the (combined) sub-graphs [12]. Once the original graph is reconstructed, the number of continuous regions in the solution is fully known. Its optimality can be trivially judged against all possible combinations of sub-graph solutions to reach to a set with all the optimal solutions.

V. Complexity

As mentioned earlier, cell sub-divisions was 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). The proposed algorithm has been shown to consist of three steps: separating the graph into strips, solving each strip individually and finally constructing all optimal solutions by combining the individual strip solutions. Complexity can thus be calculate individually.

Separating the graph only requires checking on each cell for one time, and we only need one valid graph separation, so the complexity of this part is denoted by $\Phi = O(M)$. Now we calculate the algorithmic complexity of a single strip. Let there be r_i boundary cells and s_i internal cells in the strip i, indexing from $i_1, \cdots, i_{r_i}, i_{r_i+1}, \cdots, i_{r_i+s_i}$, having $\alpha_{i_1}, \cdots, \alpha_{i_{r_i}}, \alpha_{i_{r_i+1}}, \cdots, \alpha_{i_{r_i+s_i}}$ edges, and the number of their coverable colours are written as $K_{i_1}, \cdots, K_{i_{r_i}}, K_{i_{r_i+1}}, \cdots, K_{i_{r_i+s_i}}$. Note that when a list of cells form a chain, other than the starting cell and the ending cell, each cell will have two internal edges, one shared with its precedent cell and the other shared with its successor. So as a 2D topological structure, the strip, except some special cases, cell j in the strip should have at least two internal edges, and at most $(\alpha_{i_j} - 2)$ edges are exposed to the boundary of the strip. So the complexity of solving strip i will be the multiplication of the algorithmic complexity of all boundary cells,

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

Here $\max\{\alpha_{i_j} - 2, 1\}$ intentionally considers the case that multiple edges of a same cell may appear to the strip boundary. Since enumerating solutions of different strips are totally independent, the complexity is just summed up but not muliplied.

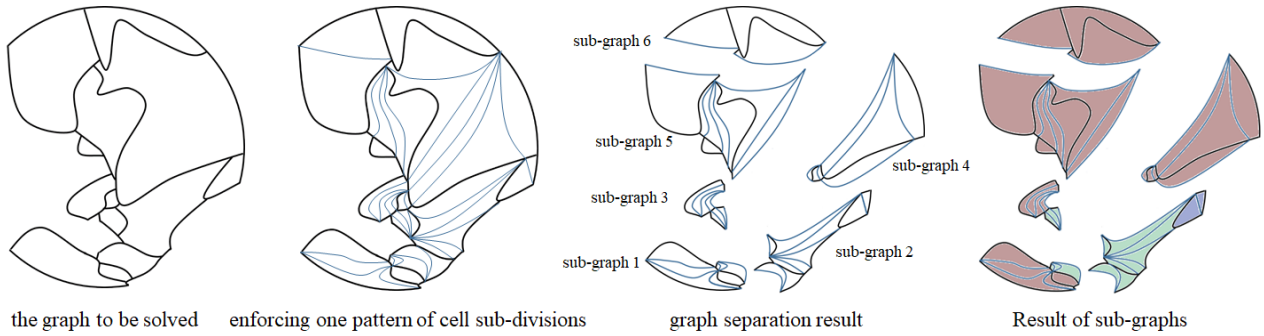

Fig. 11. Illustration of how an unsolved graph is separated into sub-graphs.

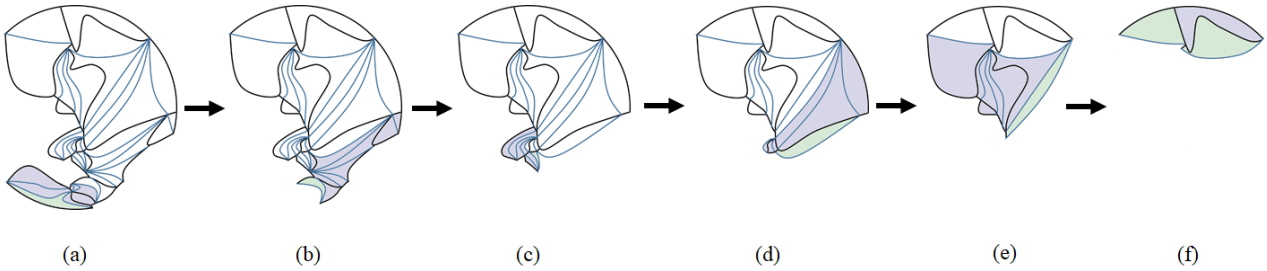

Fig. 12. The concrete steps to separate the graph given in Fig.11 into intersection-free strips.

And note that $M = \sum(r_i + s_i)$,

$$\begin{aligned}\Psi = \sum \Psi_i &= \sum_i \left(\prod_{j=1}^{r_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \right) \\ &\ll \sum_i \left(\prod_{j=1}^{r_i+s_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \right) \\ &\ll \prod_i \left(\prod_{j=1}^{r_i+s_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \right) \\ &= \prod_{j=1}^{M} K_j^{\max\{\alpha_j-2,1\}}\end{aligned} \tag{3}$$

For the final part of the algorithm, the complexity is the multiplication of the algorithmic complexity of all sub-graphs,

$$\begin{aligned}\Xi = \prod \Psi_i &= \prod_i \left(\prod_{j=1}^{r_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \right) \\ &\ll \prod_i \left(\prod_{j=1}^{r_i+s_i} K_{i_j}^{\max\{\alpha_{i_j}-2,1\}} \right) \\ &\ll \prod_{j=1}^{M} K_j^{\max\{\alpha_j-2,1\}}\end{aligned} \tag{4}$$

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

$$\Gamma = \Phi + \Psi + \Xi \ll \prod_{j=1}^{M} K_j^{\max\{\alpha_j-2,1\}} \tag{5}$$

Compared to previous solution [12] whose algorithmic complexity is

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

we provide an exponential improvement in the order of 2^N, where N is the number of topological edges in the graph.

VI. Experimental Results

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

A. Optimal NCPP on a Hat-shape Object

An illustration of the solving process of a hat-shape object is depicted in Fig. 13. 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 corresponds to the description provided earlier in Fig. 11. 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 [12], 35 edges and all colours of all cells are enumerated, resulting in an algorithmic

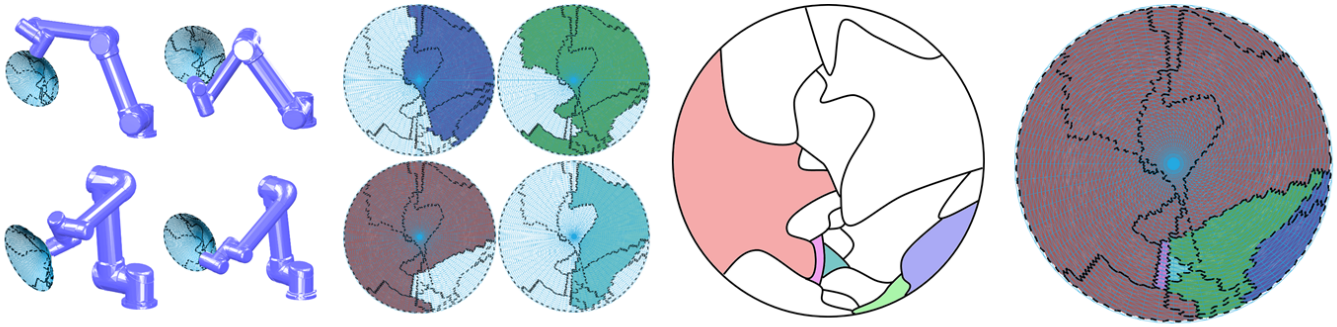

Fig. 13. The coverage task on a hat-shape object. The reachable area of different colours and one of their corresponding configurations are depicted. After painting all 1-colour cells, the unsolved graph is exactly the same one as we have discussed in Section III.

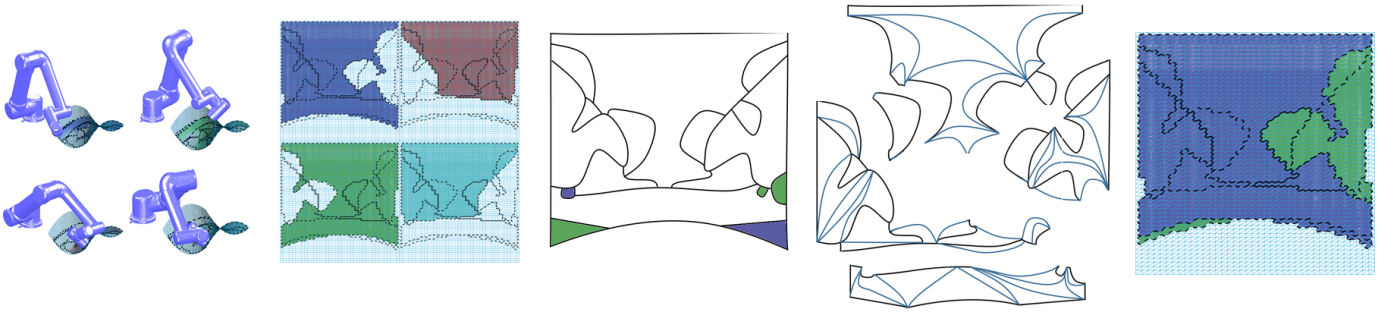

Fig. 14. The coverage task on a saddle surface. The reachable area of different colour and one of their corresponding configurations are depicted. After painting all 1-colour cells, the unsolved grpah is separated into 5 sub-graphs. Finally, all optimal solutions can be collected with 2 end-effector lift-offs.

complexity described by

$$2^{35}* \begin{matrix} 2*2*3*2*2*2*2*2*2* \\ 2*3*3*2*2*3*2*3*2*3* \\ 2*2*2*2*2*2*2*2* \\ 2*3*3*4*4*3*4* \\ 3*3*2*3*3*4*4*3* \\ 3*4*3 \end{matrix} \approx 1.2705*10^{28} \quad (7)$$

Please note the numbers are arraged so that they can be easier to compared with the optimal solution below.

Using the proposed algorithm, the length of the boundary of each sub-graph is $6, 11, 9, 6, 10, 5$, where the case of the sub-graph 1 has been discussed in detail in Sections II and III. The list of colour of the boundary cells of each sub-graph, so the algorithm complexity is the multiplication of complexity of all sub-graphs,

$$\begin{matrix} 2*2*3*2*2*2* \\ 2*3*3*2*2*3*2*3*2* \\ 2*2*2*2*2*2* \\ 2*3*3*4* \\ 3*3*2*3*3*4*4* \\ 3*4*3 \end{matrix} = 4.2797*10^{14} \quad (8)$$

B. Optimal NCPP on a Saddle-shape Object

Another test is given in the coverage task of a saddle-shape object. The surface normal varies hugely as the end-effector moves along the saddle surface. The algorithmic complexity given cell sub-divisions is shown in Fig.14.

For the naive enumeration case, complexity is given by

$$2^{47}* \begin{matrix} 2*2*2*2*2*2*2*2*2*2*2* \\ 3*2*2*2*2*2*2*2* \\ 3*4*4*3*2*2*2*3* \\ 3*2*2*2*2*2*3*3*2* \\ 4*4*4*4*4*4*4*3*4* \\ 3 \end{matrix} \approx 2.924*10^{32} \quad (9)$$

Applying the proposed algorithm, the complexity is reduced to

$$\begin{matrix} 2*2*2*2*2*2*2*2*2*2* \\ 3*2*2*2*2*2*2*2*3* \\ 4*4*3*2*2*2*3* \\ 3*2*2*2*2*3*3*2* \\ 4*4*4*4*4*4*4*3*4* \\ 3 \end{matrix} \approx 4.3283*10^{16} \quad (10)$$

TABLE I
EXPERIMENTAL RESULTS

Object	Number of Iterations	
	Full enumeration [12]	Proposed Optimal
Hat-shaped (Fig. 13)	$1.2705 * 10^{28}$	$4.2797 * 10^{14}$
Saddle-shaped (Fig. 14)	$2.924 * 10^{32}$	$1.0821 * 10^{16}$

The overall results are gather in Table I for easier comparison.

VII. CONCLUSIONS

An advacement 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 permits 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] Howie Choset and Philippe Pignon. Coverage path planning: The boustrophedon cellular decomposition. pages 203–209, 1998.

[2] Howie Choset, Ercan U Acar, A A Rizzi, and Jonathan Luntz. Exact cellular decompositions in terms of critical points of morse functions. 3:2270–2277, 2000.

[3] Mahdi Hassan and Dikai Liu. A deformable spiral based algorithm to smooth coverage path planning for marine growth removal. pages 1913–1918, 2018.

[4] W. H. Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 1, pages 27–32 vol.1, May 2001. doi: 10.1109/ROBOT.2001.932525.

[5] Donghun Lee, TaeWon Seo, and Jongwon Kim. Optimal design and workspace analysis of a mobile welding robot with a 3p3r serial manipulator. *Robotics and Autonomous systems*, 59(10):813–826, 2011.

[6] M.Z. Li, Z.P. Lu, C.F. Sha, and L.Q. Huang. Trajectory generation of spray painting robot using point cloud slicing. *Applied Mechanics and Materials*, 44–47:1290–1294, 2011.

[7] Lei Lu, Jiong Zhang, Jerry Ying Hsi Fuh, Jiang Han, and Hao Wang. Time-optimal tool motion planning with tool-tip kinematic constraints for robotic machining of sculptured surfaces. *Robotics and Computer-Integrated Manufacturing*, 65:101969, 2020.

[8] Yongguo Mei, Yung-Hsiang Lu, Y Charlie Hu, and CS George Lee. Energy-efficient motion planning for mobile robots. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 5, pages 4344–4349. IEEE, 2004.

[9] Fabian Paus, Peter Kaiser, Nikolaus Vahrenkamp, and Tamim 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.

[10] Michael Rososhansky and Fengfeng Xi. Coverage based tool-path planning for automated polishing using contact mechanics theory. *Journal of Manufacturing Systems*, 30 (3):144–153, 2011.

[11] X. Xie and L. Sun. Force control based robotic grinding system and application. In *2016 12th World Congress on Intelligent Control and Automation (WCICA)*, pages 2552–2555, June 2016. doi: 10.1109/WCICA.2016.7578828.

[12] Tong Yang, Jaime Valls Miro, Qianen Lai, Yue Wang, and Rong Xiong. Cellular decomposition for non-repetitive coverage task with minimum discontinuities. *IEEE/ASME Transactions on Mechatronics*, 2020. doi: 10.1109/TMECH.2020.2992685. URL http://doi.org/10.1109/TMECH.2020.2992685.

[13] Tsuneo Yoshikawa. Translational and rotational manipulability of robotic manipulators. *American Control Conference*, 27:228–233, 1990.