Solving the Holed Space Budgeted Maximum Coverage Problem with a Discrete Selection Problem

Phillip Smith[a] and Mohammad Zamani[b]
Defence Science and Technology Group, Melbourne, Victoria, Australia

Keywords: Motion and Path Planning, Collision Avoidance, Detouring, Area Coverage, Genetic Algorithm.

Abstract: In this paper, a new heuristic for the budgeted maximum coverage problem is introduced for environments that include obstacles (holed space). This heuristic leads to a solvable but NP-hard problem which requires a series of discrete decisions to be made. These decisions are non-trivial as the quality of each decision option may be impacted by the selected options of other decisions in the series and thus optimal solution formation is NP-hard. The effectiveness of the proposed heuristic is demonstrated by empirically comparing it to another known heuristic for the area coverage problem; finding it to be more effective at covering the space, at the cost of requiring greater computation time.

1 INTRODUCTION

Path planning for area-coverage is a well-known problem with numerous applications from CNC milling to robotic environment exploration. Generally speaking, a solution to this problem is a sequence of coordinates $p_t \in \mathbb{R}^n$ (where $n \in \{2,3\}$) that an agent must travel to such that coordinates c in a bounded area $\mathcal{B} \subset \mathbb{R}^n$ are 'covered' by the agent. That is,

$$\sup_t \|p_t - c\| \leq r,\ c \in \mathcal{B} \tag{1}$$

where r is the coverage radius. As an example, an agent equipped with a LIDAR with range r travels a path $p_t|_{t=0}^{t=T}$ (over time 0 to T). Criterion (1) then defines that all space is covered by the path and thus any objects in $\mathcal{B}$ would be detected by the LIDAR. Additionally, this area-coverage path has a travel distance, δ, found via:

$$\delta = \sum_{t=0}^{T-1} \|p_{t+1} - p_t\| \tag{2}$$

assuming the agent travels on a straight line between each pair of consequent waypoints.

In this simple form (with no limit on δ), such a path can be solved in polynomial time (P). However, the complexity is often increased with either distance minimisation (known as Shortest Tour Coverage (STC) (Arkin et al., 2000)) or with a constrained distance (known as Budgeted Maximum Coverage (BMC) (Khuller et al., 1999)). The former aims to find the path-solution with smallest δ meeting (1), which is NP-hard (Arkin et al., 2000). The latter limits δ to a travel budget Δ, $\delta \leq \Delta$, which is chosen in accordance with real-world constraints such as robot battery capacity or task time-limits. The overall optimisation problem becomes:

$$\begin{aligned} &\max_{C \subseteq \mathcal{B}} \text{Volume}(C) \\ &s.t.\ \delta \leq \Delta, \\ &C = \{c\ :\ \sup_t \|p_t - c\| \leq r\}, \end{aligned} \tag{3}$$

which is also NP-hard (Nemhauser et al., 1978).

Due to the fact both sub-problems are NP-hard, many heuristic path generators have been proposed to solve (most instances of) these problems. These include boustrophedon paths (sweeping) (Choset and Pignon, 1998), spiral paths (Cabreira et al., 2018) and fractal planning such as Hilbert Curves (HCs) (Hilbert, 1935). Due to the similarities in STC and BMC, these heuristics may be applied to either problem. However, some heuristics are more suited to one over the other. These heuristics produce a solution in polynomial time at the cost of not guaranteeing global optimally in irregular spaces. Further, these heuristics require modification to overcome challenges such as spaces with holes (e.g. obstacles).

In this work, a modification process for these heuristic-based paths is derived for BMC in holed spaces. Further, this process is demonstrated on HC

[a] https://orcid.org/0000-0002-5330-1830
[b] https://orcid.org/0000-0003-2979-7132

Smith, P. and Zamani, M.
Solving the Holed Space Budgeted Maximum Coverage Problem with a Discrete Selection Problem.
DOI: 10.5220/0012889300003822
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 21st International Conference on Informatics in Control, Automation and Robotics (ICINCO 2024) - Volume 1*, pages 15-24
ISBN: 978-989-758-717-7; ISSN: 2184-2809
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

paths, being found to be more coverage-effective and tolerant of hole shapes and sizes than the holed space path modifier found in (Nair et al., 2017). However, the implemented method requires a set of combinatorial decisions to be made, which is itself an NP-hard problem. This leads to the main problem of this paper; a discrete, non-monotonic decision selector. It is believed this new problem will be of interest to the area-coverage community as it produces a locally optimal solution to the parent problem of BMC with considerably less complexity. Additionally, the discrete decisions of this problem pose a new benchmarking application for numerous meta- and hyper-heuristics algorithms such as Genetic Algorithm (GA) (Holland, 1992), Artificial Ant/Bee Algorithms (Dorigo and Stützle, 2019) (Karaboga, 2005) and Particle Swarm Optimisation (Kennedy and Eberhart, 1995).

In this paper, the background of this topic is further discussed and a formal definition of the proposed path-modifying heuristic is presented in Section 2 and Section 3. In Section 4 an empirical demonstration of this method is made by comparing it to both another HC modifying technique from literature (Nair et al., 2017) and to a tree-search (Boyd and Mattingley, 2007). Finally, in Section 5 this work is concluded.

2 BACKGROUND

2.1 Hilbert Curves

The Hilbert space-filling curve heuristic is a path planner approach that aims to uniformly examine an environment (Hilbert, 1935). The algorithm uses a square pattern, recursively repeated to the h^{th} degrees ($h \in \mathbb{Z}_+$); as the degree increases both the coverage resolution and path length increase, as shown in Figure 1. This path length, δ_{HC}, is defined as:

$$\delta_{HC} = w(2^h - 2^{-h}) \tag{4}$$

where w is the size of the square space being covered and it can be noted that $\delta_{HC}(h)$ is a monotonic function. Additionally, it is noted that applying the coverage circle r to this path produces the coverage volume, which is also monotonic until (1) is satisfied.

To define an optimal h value for square area coverage, (Viana and De Amorim, 2013) observes that a Hilbert curve of degree h divides the space, $\mathcal{B}$, into 2^{h+1} cells, each with size $l = w \cdot 2^{-h}$. The resulting path has the agent travel to each cell centre, and thus the agent's circular coverage of range r fully covers the square cells if

$$l \leq r\sqrt{2} \tag{5}$$

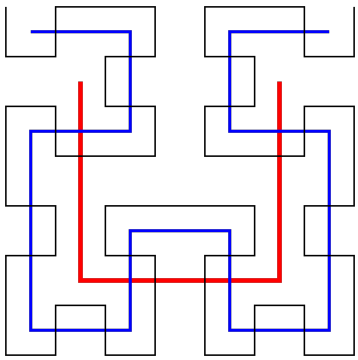


Figure 1: HC scaled to space with degrees 1 (red), 2 (blue) and 3 (black). Each degree covers more area and has greater length. Figure use granted under unrestricted 'public domain' licensing (Richards, 2022).

In contrast, the agent's coverage range does not spread into neighbouring cells (which may have already been explored) if

$$l \geq 2r \tag{6}$$

Substituting w and h into these limits, it is guaranteed a Hilbert curve completely covers an area if

$$h \geq \lceil log_2(\frac{w}{r}) - 0.5 \rceil \tag{7}$$

Similarly, it is observed that the path is guaranteed to have no overlapping coverage (and thus no redundant travel) if

$$h \leq \lfloor log_2(\frac{w}{r}) - 1 \rfloor \tag{8}$$

From these two limits with conflicting rounding functions, it is noted that the proposed h limits are always 1 apart. Further, both (7) *and* (8) cannot be satisfied by a h selection simultaneously.

As an example, consider a space with $w = 16$ covered by a path with $r = 2$ and assume the path can be executed with omnidirectional travel (ignore the cost of rotations). Eq. (7) produces the restriction $h \geq 3$ while Eq. (8) sets $h \leq 2$. In Figure 2 it can be seen that the $h = 3$ path covers all space within $\mathcal{B}$, however, there is significant area covered multiple times and the total path length is much longer than $h = 2$. In contrast, the $h = 2$ has no sub-effective travel from re-covering area but fails to cover the corners of some HC cells.

Using these qualities, it can be observed that the HC lends itself well to the STC and BMC problems in square space. However, as (7) and (8) cannot be satisfied simultaneously, h must be tuned for the specific problem variant.

In terms of STC, a locally optimal solution minimising h such that (7) is satisfied will also minimise length δ_{HC} as per the monotonic natures of (4). The

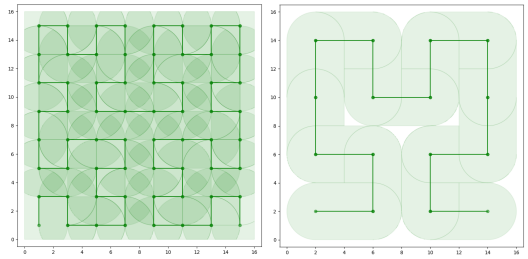


Figure 2: Area coverage with h=3 (left) and 2 (right).

resulting path-solution may not be globally optimal to the STC problem but can be effectively produced in linear time via the optimisation of a single integer.

For BMC, a locally optimal solution is generated by combining (3) and (4) into

$$\max(h) \mid (2^h - 2^{-h}) \leq \frac{\Delta}{w} \tag{9}$$

Again, this approach requires linear time and, if the h produced by (9) can also satisfy (7), the HC is globally optimal due to the travel budget of the problem instance being too relaxed for the give w. Similarly, if we expand the h of (9) to $h \in \mathcal{R}$ and find (8) still holds (the real-number h is not between the neighbouring integer limits of (8) and (7)), we can say the HC is globally optimal, given the w and Δ combination.

To conclude, the HC heuristic allows the path-generating problem of STC and BMC to be compressed to one degree of freedom, h. Not only is this much cheaper in terms of computation but also makes the constraints and objective functions monotonic and hence makes it easier to solve using efficient optimisation algorithms.

2.2 Holed Space Area Coverage Heuristics

Several works have modified path-generating heuristics for area-coverage in holed spaces, e.g. spaces with obstacles. However, these works either compromise path length efficiency to circumvent the holes or make significant assumptions on space discretisation and hole alignment which limit the applicability of these heuristics.

In (Choset and Pignon, 1998) and (Achat et al., 2023) hole-tolerant variants of the boustrophedon heuristic are proposed which segmented the environment as the path intersected holes and sequentially swept each subsection, guaranteeing all spaces are covered. However, in the former, travelling between each subsection used a greedy planner, and in the latter, the subsection order follows the global sweeping pattern. As such, agents could move through prior-covered subsections. As a result, the total path length would be non-optimal in STC and all sub-sections may not be effectively swept in BMC.

Similarly, in (Gabriely and Rimon, 2001) a spiral path was produced by a depth-first spanning trees of unoccupied, unexplored cells. This work escaped dead-ends (caused by obstacles) by reverse-travelling the path until unexplored cell neighbours became available. This, again, reduced the effectiveness of the overall path, making the heuristic sub-optimal for both STC and BMC.

Finally, (Nair et al., 2017) explored modifying HCs by removing, adding, or rearranging path vertices around obstacles with the modifying action chosen via the vertices' index in the recursive HC path. This approach maximised the exploration of unoccupied cells with minimal increase to path length. However, it also assumed the space could be easily discretised, with the holes being of equal size and shape to the occupancy cells. In this work, it was assumed h was provided to the algorithm, though explored the use of each quartile of the space utilising a different h. Further, in a follow-up paper, (Joshi et al., 2019), the heuristic was extended to overcome rectangular obstacles, occupying two adjacent occupancy cells. However, the rectangles still required to be aligned to the occupancy grid and thus HC vertices.

From this review, no prior work, to the best of our knowledge, has been found that modifies a path generation heuristic for resilience to holed spaces without incurring similar shortcomings.

3 HILBERT CURVE FOR HOLED SPACE

3.1 Problem Formulation

In this section, the coverage path planning problem in holed space is revisit and the proposed path adjustment process is described. For simplicity, it is assumed the agent is ground based and thus the space is two-dimensional ($n = 2$). Further, this study is limited to circular and rectangular obstacles. That being said, higher dimensional spaces and irregular obstacles circumvention is possible with this approach. Additionally, it should be highlighted that the obstacles of this demonstration have centroids at any given $c \in \mathbb{R}^2$. That is, the obstacles are not bound to a discrete occupancy grid; this allows a closer representation of real-world obstacle environments.

Given a set of obstacles, the hole-space of the environment is defined as $\mathcal{H}$. This $\mathcal{H}$ represents all obstacles the agent must circumvent plus the collision-

avoiding buffer spaces around them. The remaining free space is $\mathcal{B}\backslash\mathcal{H}$.

3.1.1 Path Initialisation

The proposed process starts with a path not necessarily robust to holes projected onto the space (ignoring the holes). In this study, this takes the form of a h degree HC, maximising h as in (9) to optimise for BMC. It can be noted that this approach is not limited to HC paths; any path planning heuristic (see Section 2) could be utilised at this initial point. As defined in Section 1, the initial path of length T^0, $\mathbf{p}^0$, is expressed as a series of waypoints,

$$\mathbf{p^0} = [p_i^0 \in \mathbb{R}^2, i \in \mathbb{Z}_{T^0}] \tag{10}$$

and the 0 superscript is a preemptive addition for later in this section. In this (and all further discussed) path(s), each pair of consequent waypoints (p_i^j, p_{i+1}^j) (where j denotes an arbitrary path $\mathbf{p^j}$) is connected via a straight line, expressed as:

$$l_i^j = \{p \in \mathbb{R}^2 \,:\, p = p_i^j + t(p_{i+1}^j - p_i^j), t \in \mathbb{R}_1\}. \tag{11}$$

3.1.2 Path Breaks

To prevent the agent colliding with an obstacle while travelling the path, at each point $\mathbf{p}^0$ intersects $\mathcal{H}$ the corresponding straight-line path is split into subpaths, each being expressed as a subarray of waypoints. The overall path is then the array of these subarrays, initialised as $\mathbf{P} = [\mathbf{p}^0]$.

Collision detection is sequentially performed on each subpath, $\mathbf{p}^j$, in $\mathbf{P}$. A collision occurs on $\mathbf{p}^j$ if γ exists such that:

$$\gamma = \underset{i \in \mathbb{Z}_{Tj}}{\operatorname{argmin}}(l_i^j \cap \mathcal{H} \neq \emptyset) \tag{12}$$

If γ exists, the first collision-point along l_γ^j can be found by combining (11) and (12):

$$t_\alpha = \underset{0 \leq t \leq 1}{\operatorname{argmin}}(p_\gamma^j + t \cdot (p_{\gamma+1}^j - p_\gamma^j)) \in \mathcal{H} \tag{13}$$

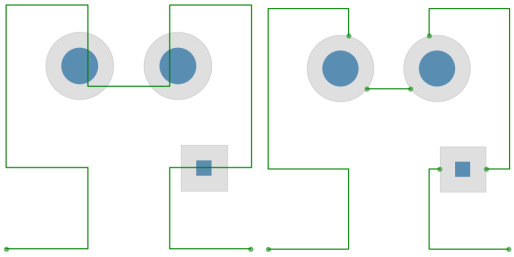
If $t_\alpha > 0$, it can be used to find the 'last safe point', p_α^j, on l_γ^j: [1]

$$p_\alpha^j = \max_{t < t_\alpha}(p_\gamma^j + t \cdot (p_{\gamma+1}^j - p_\gamma^j)) : p_\alpha^j \notin \mathcal{H} \tag{14}$$

The first collision-free point after t_α is labelled p_β and found via:

$$p_\beta^j = \min_{t_\alpha < t \leq 1}(p_\gamma^j + t \cdot (p_{\gamma+1}^j - p_\gamma^j)) : p_\beta^j \notin \mathcal{H} \tag{15}$$

[1] In relation to implementation, t may be incremented in discrete steps, $\hat{t}$. In which case $p_\alpha = p_\gamma^j + (\hat{t} \cdot \lfloor t_\alpha \rfloor) \cdot (p_{\gamma+1}^j - p_\gamma^j))$



original path post path-breaking

Figure 3: HC, split into four subarrays such that hole-collisions are removed. Path presented as green lines, obstacles as blue circles. Grey area around obstacles is the hole space.

The two corner cases of this process are $t_\alpha = 0$ and p_β being undefined. These cases are two results of the unmodified path having a waypoint inside $\mathcal{H}$ and are addressed by (16) and (17), respectively. The former case, $t_\alpha = 0$, occurs when l_γ^j begins inside $\mathcal{H}$ and thus only arises when $\gamma = 0$.[2] The latter case, p_β^j being undefined, occurs if all points of l_γ^j after p_α^j are inside $\mathcal{H}$ and thus is coupled with the former case ($t_\alpha = 0$) occurring in the following straight-line.

Using p_α and p_β, the currently examined subarray $\mathbf{p}^j = [p_0^j, p_1^j, \ldots, p_\alpha^j, p_\gamma^j, \ldots, p_{Tj-1}^j]$ is reduced to the waypoints prior to the collision:

$$\mathbf{p}^{j\prime} = \begin{cases} [p_0^j, \ldots, p_\alpha^j] & \text{, if } t_\alpha > 0 \\ [\,] & \text{, otherwise} \end{cases} \tag{16}$$

and a new subarray, $\mathbf{p}^{j+1}$, is appended to $\mathbf{P}$:

$$\mathbf{p}^{j+1} = \begin{cases} [p_\beta^j, p_{\gamma+1}^j, \ldots, p_{Tj-1}^j], & \text{if } p_\beta^j \text{ is defined} \\ [p_{\gamma+1}^j, \ldots, p_{Tj-1}^j], & \text{otherwise} \end{cases} \tag{17}$$

From the sequential nature of this possess, the modified subpath, $\mathbf{p}^{j\prime}$, is guaranteed to be collision free. When (12) reports a γ does not exit for any $\mathbf{p}^j$, the entire path is assured to be collision free and will be in the form:

$$\mathbf{P'} = [\mathbf{p}^{0\prime}, \mathbf{p}^{1\prime}, \ldots, \mathbf{p}^{\eta}] \tag{18}$$

where $\eta = |\mathbf{P'}| - 1$. However, the set of straight lines defining $\mathbf{P}$ are no longer a continuous path, *i.e.* $p_{Tj-1}^j \neq p_0^{j+1}$. This path-splitting process is demonstrated in Figure 3.

3.1.3 Re-Joining the Path

To make the HC a continuous path once more, heuristic-based detours are added between each sub-

[2] if $a > 0$ (12) assures there are no collisions for l_0^j to $l_{\gamma-1}^j$, and $l_{\gamma-1}^j(t=1) == l_\gamma^j(t=0)$, therefore it is assured that $\gamma > 0 \implies t_\alpha \neq 0$.

array of $\mathbf{P}'$. At each reconnection, $\mu \in \mathbb{Z}_+$ detour options are available, but only one may be selected per reconnection. Here a reconnection is defined as a decision, d^j, with detour options $\{\mathbf{m}_0^j, \mathbf{m}_1^j, \ldots, \mathbf{m}_{\mu-1}^j\}$, where a detour is a heuristic-based array of waypoints, $\mathbf{m}_u^j = [m_{u,0}^j, m_{u,1}^j, \ldots, m_{u,T_u^j}^j]$ such that $\mathbf{p}_{T^j}^{j\prime} == m_{u,0}^j$ and $\mathbf{p}_0^{j+1\prime} == m_{u,T_u^j}^j$ (u denoting an arbitrary detour in the option-set, with length $T_u^j + 1$). The resulting path, $\mathbf{P}''$, can be represented with detour option super-positions:

$$\begin{aligned}\mathbf{P}'' &= [\mathbf{p}^{0\prime}, d^0, \mathbf{p}^{1\prime}, d^1, \ldots, \mathbf{p}^{\eta-1\prime}, d^{\eta-1}, \mathbf{p}^{\eta}] \\ &= [\mathbf{p}^{0\prime}, \begin{bmatrix} \mathbf{m}_0^0 \\ \mathbf{m}_1^0 \\ \ldots \\ \mathbf{m}_{\mu-1}^0 \end{bmatrix}^\top, \ldots, \begin{bmatrix} \mathbf{m}_0^{\eta-1} \\ \mathbf{m}_1^{\eta-1} \\ \ldots \\ \mathbf{m}_{\mu-1}^{\eta-1} \end{bmatrix}^\top, \mathbf{p}^{\eta}] \end{aligned} \quad (19)$$

As η detour options must be selected, the detour solution space is size μ^η. As such, the size complexity of this selection process grows exponentially as more detour options are implemented and grows monotonically (polynomial to μ) as the environments become more densely filled with obstacles.

In relation to the detour options, any path-generating heuristic that connects two points while avoiding hole intersections may be utilised. That said, due to the exponential complexity growth, this demonstration will assume only a 'left' and 'right' circumvention along the borders of $\mathcal{B}$ and $\mathcal{H}$[3]. This reduces d_j to:

$$d_j = \{\mathbf{m}_{j,\text{left}}, \mathbf{m}_{j,\text{right}}\} \quad (20)$$

Applying this $\mu = 2$ option set to the example problem of Figure 3, the graph shown in Figure 4 is produced.

A solution to the decision-making problem, S, can now be defined as the array of detour options chosen to fully reconnect the HC. *i.e.* for the problem in Figure 4, a solution is "[left, right, left]". This solution collapses the super-positions of (19) into a standard, continuous path $\mathbf{P}_S''$. However, discovering an effective and valid solution to this problem remains complex, as discussed next.

3.2 Problem Constraints

3.2.1 Selection Limit

As stated above, a detour selection solution must have exactly one detour option selected for each decision point. Formally, this is defined as the following constraint:

$$\forall j \in \{0, 1, \ldots, \eta-1\}, \sum_{u=0}^{\mu} x(S, \mathbf{m}_u^j) = 1 \quad (21)$$

$$x(S, \mathbf{m}) \triangleq \begin{cases} 1, & \text{if } \mathbf{m} \in S, \\ 0, & \text{otherwise.} \end{cases} \quad (22)$$

[3]Additional detours potentially include 'wide-left' and 'wide-right'; these see the agent circumvents wider, such that the borders of the obstacle is w from the agent, maximising area coverage during detour

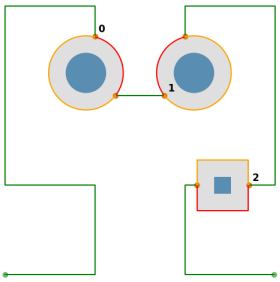


Figure 4: All detour options added to HC, each obstacle must select the left (red) or right (yellow) circumvention path.

3.2.2 Travel Distance

As this work is exploring the BMC problem, any solution path $\mathbf{P}_S''$ must adhere to the travel budget, Δ,

$$\begin{aligned} &\|\mathbf{P}''_S\| \leq \Delta, \\ &\|\mathbf{P}''_S\| \triangleq \sum_{j=0}^{\eta} \sum_{i=0}^{T^j} \|p_{i+1}^j - p_i^j\| \\ &\qquad + \sum_{j=0}^{\eta-1} \sum_{u=0}^{\mu} \left(x(S, \mathbf{m}_{j,u}) \cdot \sum_{i=0}^{T_u^j} \|m_{u,i+1}^j - m_{u,i}^j\| \right), \end{aligned} \quad (23)$$

where the first sum is the remaining HC, $\mathbf{P}'$, and the second sum is the detour paths with $x(S, \mathbf{m}_u^j)$ removing non-selected detours. As $\mathbf{P}'$ is fixed during the detour selection process, we can simplify (23) to,

$$\begin{aligned} &\Delta' \triangleq \Delta - \|\mathbf{P}'\| \\ &\sum_{j=0}^{\eta-1} \sum_{u=0}^{\mu} \left(x(S, \mathbf{m}_{j,u}) \cdot \sum_{i=0}^{T_u^j} \|m_{u,i+1}^j - m_{u,i}^j\| \right) \leq \Delta' \end{aligned} \quad (24)$$

3.3 Space Coverage

As the BMC problem aims to maximise the coverage of the path, the coverage of $\mathbf{P}_S''$ is defined as the union of covered areas (inside $\mathcal{B}$) produced by travelling along the lines of the path, l_i^j (defined in (11)), with the vision range r. As the geometrical calculations are beyond this discussion, the exposition is

simplified to: line l_i^j has a given area $A(l_i^j, r) \subset \mathcal{B}$, and subpath $\mathbf{p}^j$ have area defined as,

$$A(\mathbf{p}^j) = \bigcup_{i=0}^{T^j} A(l_i^j, r) \tag{25}$$

Here, $\cup$ denotes the union of the spacial subsets.

Continuing this notation, the path of solution S is said to have area coverage:

$$A(\mathbf{P}''_S) = \mathcal{B} \cap \Big(\bigcup_{j=0}^{\eta} A(\mathbf{p}_j) \cup \bigcup_{j=0}^{\eta-1} \bigcup_{u=0}^{\mu} \big(x(S, \mathbf{m}_u^j) \cdot A(\mathbf{m}_u^j) \big) \Big), \tag{26}$$

which, again, can be simplified by removing the $\mathbf{P}'$ constants:

$$\begin{aligned} A(\tilde{\mathbf{P}}'_S) &\triangleq A(\mathbf{P}''_S) - A(\mathbf{P}') \\ &= \mathcal{B} \cap \bigcup_{j=0}^{\eta-1} \bigcup_{u=0}^{\mu} \Big(x(S, \mathbf{m}_u^j) \cdot \begin{pmatrix} A(\mathbf{m}_u^j) - \\ A(\mathbf{P}') \cap A(\mathbf{m}_u^j) \end{pmatrix} \Big) \end{aligned} \tag{27}$$

This simplified space coverage value is only the additional coverage of the detours. That is, the area not already covered by the remains of the HC. The coverage of the example problem in Figure 3 is shown in Figure 5. This figure highlights the complexity of this problem space: detour coverage areas may overlap, resulting in area unions less than the sum of individual detour areas. *id est*, if both $\mathbf{m}_{0,left}$ and $\mathbf{m}_{1,left}$ are selected,
$A(\mathbf{m}_{0,left}) \cup A(\mathbf{m}_{0,left})) \ll A(\mathbf{m}_{0,left}) + A(\mathbf{m}_{1,left})$.

3.4 Problem Summary

To summarise the problem requirements, given a collection of decisions $[d_0, d_1, \ldots, d_{\eta-1}]$, each containing μ heuristic-based detour paths, $\{\mathbf{m}_0^j, \mathbf{m}_1^j, \ldots, \mathbf{m}_{\mu-1}^j\}$, a solution to this problem is an array of detour selections of length η. This solution aims to maximise the area coverage while keeping the total path length below a budget:

$$\begin{aligned} &\max_{\mathbf{m}_{j,u}} \mathcal{B} \cap \bigcup_{j=0}^{\eta-1} \bigcup_{u=0}^{\mu} \Big(x(S, \mathbf{m}_u^j) \cdot \begin{pmatrix} A(\mathbf{m}_u^j) - \\ A(\mathbf{P}') \cap A(\mathbf{m}_u^j) \end{pmatrix} \Big) \\ &\text{s.t} \quad \sum_{j=0}^{\eta-1} \sum_{u=0}^{\mu} \Big(x(S, \mathbf{m}_{j,u}) \cdot \sum_{i=0}^{T_u^j} \| m_{u,i+1}^j - m_{u,i}^j \| \Big) \leq \Delta' \\ &\forall j \in \{0, 1, \ldots, \eta - 1\}, \sum_{u=0}^{\mu} x(S, \mathbf{m}_u^j) = 1. \end{aligned} \tag{28}$$

It can be noted that because the detour options have no inherent ordering (the cost-reward function is not monotone), a gradient-based or greedy heuristic can not be derived (Garrido-Merchán and Hernández-Lobato, 2020). Additionally, due to the above discussed detour area overlap, the selection process is non-convex and thus difficult to solve in a sequential greedy fashion, or with tree-search algorithms (such as branch and bound (Boyd and Mattingley, 2007)). That is, partial-solution area-coverage must be recalculated (from scratch) as each decision is iteratively added to a solution with these algorithms. This results in a worst case of $\sum_{i=1}^{\eta}(\mu^i)$ area calculations. Further, a globally optimal solution is only guaranteed if all valid solutions are evaluated in a combinatorial fashion. This may be reduced by testing for budget violation in polynomial time, but the worst-case instance requires all μ^η to undergo area optimality testing, and thus the problem is NP-hard.

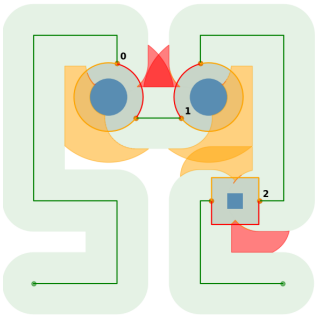


Figure 5: Space coverage of example problem. Coverage coloured as per the line colouring of Figure 4.

As a final note on this detour problem, because a solution is a fixed-length string of values from a finite set, many meta-heuristics can be employed to search such a landscape. Furthermore, the solution landscape is expected to be rugged, with both jagged cliffs (poor and good solutions close together) and smooth concavities. Therefore, this problem lends itself well to meta-heuristic algorithms which effectively explore the space via both local search (finding optima) and large leaps (escaping local optima and crossing jagged cliffs). Such meta-heuristics include: GA (Holland, 1992), Artificial Bee Colony (Karaboga, 2005) , Simulated Annealing (Van Laarhoven and Aarts, 1987), and Particle Swarm Optimisation (Kennedy and Eberhart, 1995). As such, this detour selecting problem is seen as both a new heuristic for the BMC problem in holed space and as a new application for the meta-heuristics field.

4 VALIDATING THE HEURISTIC

4.1 Experiment Design

4.1.1 Corner Cutting Hilbert Curve Comparison

To validate the proposed process, a comparison is made for both efficiency and effectiveness between this method and the hole-tolerant HC modification of Nair *et. al* (Nair et al., 2017). For conciseness, Nair *et. al*'s algorithm is hereon refereed to as Corner Cutting Hilbert Curve (CCHC) and the proposed detour method as Detour Selection Hilbert Curve (DSHC). To solve the detour selection problem, both a Brute-Force search (BF) and GA solver are trialled. The comparison study consists of 20 randomly generated holed spaces for each of the 24 configurations of the following variables:

- HC degree, $h \in \{1,2,3,4\}$
- Budget, $\Delta \in \{\delta_{HC}\ ,\ 1.1\delta_{HC}\ ,\ 1.5\delta_{HC}\}$
- Hole Alignment $\in$ {Aligned, Randomised}

where δ_{HC} is the length of the unmodified HC as defined in (4); 'Aligned' holes are only placed on the vertices of the HC, sized to only restrict the path to/from that vertex, and never adjacent to one another (as seen in (Nair et al., 2017)); and 'Randomised' holes are either rectangular or circular, with size range $(0.02w, 0.05w)$ and randomly placed in the space with centroid coordinates in $\mathbf{R}^2 \in (0.025w, 0.975w)$. Examples of these hole alignments are presented in Figure 6. In addition to these variables, the agent has a fixed vision range $r = 1$.

When generating the environments of this study, $\mathcal{B}$ has width and length $w = r \cdot 2^{h+1}$ which allow an unmodified HC to fully cover the space (see Section 2). This choice of w is seen as the least effective use-case for DSHC and thus presents the greatest comparative challenge. That is, a $\mathcal{B}$ being fully covered by the original HC causes (27) to be relatively small and flat across the detour solution landscape, which minimises the impact of (28). In contrast, a $\mathcal{B}$ only partially covered by the original HC allows the detours to significantly contribute to the coverage and thus (28) has greater optimisation potential. For example, in Figure 5 the left and right detours of the first two decisions produce significantly different coverage. By using this least effective case the performance comparison of this study is strengthened, as any results in favour of DSHC would only be increased in a preferred $\mathcal{B}$.

For the GA implementation, a population of 100 solutions is maintained and each generation produces 10% of the population (10) new solutions. A history

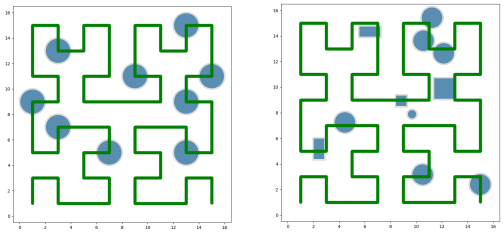


Figure 6: Example of Aligned (left) and Randomised (right) obstacle layouts.

of all explored solutions is additionally maintained, ensuring each generation contains unexplored solutions. Offspring are produced via one-point crossover, with parents selected using roulette wheel. The resulting children are added to the generation in both mutated and unmutated forms (if not found in the search's history); where mutation is an element-wise decision switch with probability $\frac{1}{\eta}$. The GA is terminated after 1000 generations without a solution fitness improvement or if the history contains all 2^η solutions.

To analyse the efficiency results of this comparison, the CPU clock-times relative to the number of detours (η) are plotted. For DSHC, this plotting is done for both the time to generate the selection problem (as discussed in Section 3.1) and the full operation time (both creating the problem and solving it). A best-fit function of type: polynomial (degree 1 to 6), logarithmic or exponential is reported for each algorithm's run time. For fair comparisons, all algorithms are run without multi-threading on a single i5-8365U (1.60GHz) processor.

The effectiveness of the algorithms is quantified via the percentage of $\mathcal{B}$ covered by the resulting path, using (26). The mean and standard deviation for each variable configuration is reported.

As CCHC is not designed for randomised-hole locations, a marginal adjustment has been made to CCHC when operating in random-hole environments. This adjustment recursively decreases the h value of path segments at any point where hole intersections still exist after corner-cutting. It has been implemented to build on Nair *et. al*'s use of different h values in each environment quadrant. Further, it is found that this method performs better than repeatedly removing waypoints until all obstacles are avoided on a fixed h HC, and this adjustment is only made to give a fairer comparison between the algorithms. Further details of this algorithm modification are provided upon request.

In addition to this adjustment to CCHC, both algorithms are modified to strongly adhere to the travel budget; if the modified path of CCHC or the shortest path of all DSHC solutions exceed Δ, the end of

the respective path is trimmed to meet the budget requirement. This reduces the coverage performance of the path but prevents either approach from having an unfair advantage or not being able to produce valid solutions to an environment.

4.1.2 Tree-Search Comparisons

In addition to the above comparison, this study also validates the hypothesis that tree-search algorithms, such as Branch and Bound (B&B), are ill-suited for solving the proposed detour selection problem. This is confirmed by analysing the CPU clock-time and number of area calculations for the DSHC-BF search and a B&B search. This analysis is conducted for budgets: $\Delta = 1.1\delta_{HC}$ and $\Delta = 1.5\delta_{HC}$. For each budget, 1,000 randomly generated problem instances are solved by each algorithm. It can be noted that the coverage results are not compared, as both algorithms produce a globally optimal detour solution.

4.2 Experiment Results

4.2.1 DSHC vs. CCHC Computation Time

Figure 7 presents the execution times and best-fit function for each algorithm. CCHC reported the fastest execution times, faster than either DSHC solver by several order of magnitude. However, these execution times still most closely fit an exponential growth model, despite the theoretical polynomial algorithm. In contrast, the proposed detour problem can be built in linear time and solved with a GA approach in polynomial time. Only the BF solver grows exponentially, as expected.

4.2.2 DSHC vs. CCHC Coverage

Figure 8 presents the mean coverage effectiveness of the paths after heuristic modification.

In relation to the two solvers applied to DSHC, equivalent coverage results are produced. As BF is guaranteed to discover the global optima of the detour-selection problem, it is concluded that the GA meta-heuristic is also consistently discovering the global optima, and is thus well suited to this problem.

In regards to DSHC and CCHC in the various environment configurations, the former outperforms the latter in all cases explored. DSHC is minimally impacted by the HC degree or the placement of obstacles, though sees better performance with a budget significantly larger than δ_{HC}. In contrast, CCHC shows little variation from the budget but is significantly impacted by the holes not aligning with the HC vertices and with smaller h curves.

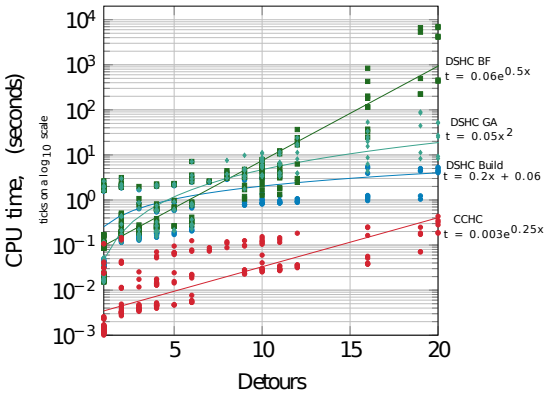


Figure 7: Time-detour relation between obstacle avoiding CCHC (Red), DSHC with a BF solver (green), DSHC with a GA solver (teal), and just the problem creation time of DSHC (blue).

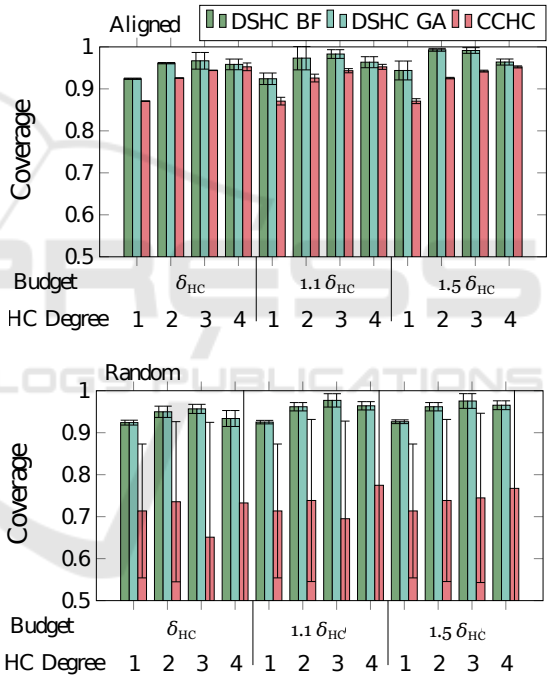
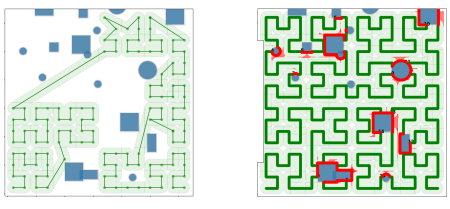


Figure 8: Normalised coverage of $\mathcal{B}$ by the three methods. DSHC shows greater mean coverage in all cases than CCHC.



CCHC. DSHC.

Figure 9: Result of removing vertices to avoid collisions in a dense, randomly scattered, hole environment.

The performance growth of DSHC relative to the budget increase is due to $\delta_{HC} \approx \Delta$ causing $\Delta' \approx 0$ (see (24)). This restricts (28) to the single shortest path solution or a small subset of solutions. Further, if larger obstacles must be circumvented, Δ' may become negative which triggers the path-trimming modification discussed above.

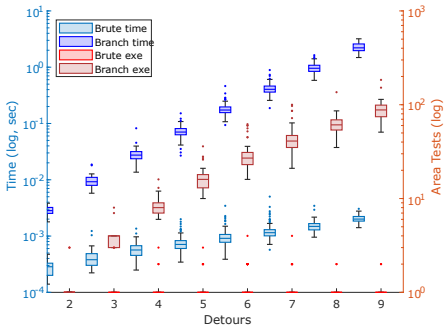
In relation to the poor robustness of CCHC to randomised obstacle placement, this is due to obstacles intersecting the path at edges (rather than vertices) and larger obstacles triggering multiple HC vertices to be modified. As most of the CCHC modifications have a vertex removed, these multiple modifications can accumulate to large sections of the path being discarded. This is in contrast to DSHC, which can detour an obstacle irrespective of size or the intersection location. An extreme case of this over-removal by CCHC is shown in Figure 9. In *(a)* (CCHC), the upper left corner must be treated as a $h = 1$ HC and removed accordingly to prevent collision with all obstacles of the cluster. This results in the entire quadrant being unexplored. In contrast *(b)* (DSHC), shows a series of small detours that allow the quadrant to still be explored as the agent weaves around the many obstacles.

4.3 DSHC vs. B&B Computation Time

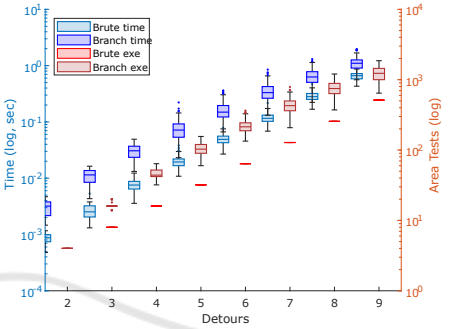
In Figure 10 the computation time (left axis) and number of area computations (right axis) are presented for both DSHC-BF and B&B. In both instances, B&B performs more area calculations and thus requires more computation time; this computation time difference is more significant for low-budget cases.

For the low-budget cases, $\Delta = 1.1\delta_{HC}$, it is observed that very few area calculations are performed by DSHC-BF for all problem instances, irrespective of η. This is due to many solutions not meeting the small Δ', and thus being filtered out prior to area calculation. As such, the execution time is much faster than B&B. In contrast, the number of area calculations performed by B&B grows exponentially to η due to many partial solutions' areas still being calculated as the tree is expanded in order to find the single (or small set of) valid solution(s). Due to these partial solution explorations, the B&B search shows rapid exponential growth relative to η.

In the high budget cases, $\Delta = 1.5\delta_{HC}$, the majority of solutions are within the travel budget and thus the area calculations of DSHC-BF search is exponential relative to detours. As such, the variation between the two search algorithms is reduced with noticeable overlap at high η. That being said, the B&B search has a much wider variation in both area



(a) $\Delta = 1.1\delta_{HC}$.



(b) $\Delta = 1.5\delta_{HC}$.

Figure 10: Time and Area calculations vs number of Detours for both BF DSHC and B&B with both small and large travel budgets.

calculations performed and computation time due to the tree search having a best-case computation count (with complete bounding) of μ and a worst-case (with no bounding) of $\sum_{i=1}^{\eta}(\mu^i)$.

From this experiment, it can be concluded that tree-search algorithms, even B&B, are inappropriate for the proposed problem, as hypothesised. This is because the repeated calculation of each node's 'value' (while expanding the tree with partial detour solutions) is more computationally expensive than the reductions observed by bounding the search.

5 CONCLUSION

In this paper, a new heuristic was defined for modifying a space-filling path which circumvents holed spaces. In doing so, a non-convex optimisation problem is produced which is NP-hard but simpler to solve than the parent Budgeted Maximum Coverage (BMC) path-planning problem. This paper demonstrates the proposed path modification process on the well-established Hilbert Curve (HC) path, finding considerable area can be covered despite the space

having holes. That being said, it is noted that the produced solutions are not guaranteed to be globally optimal for the original BMC problem.

Via empirical evaluation, it has been shown that this approach produces a path that, on average, covers more area than a HC modifying approach found in literature (Nair et al., 2017). However, the proposed approach requires considerably longer computation times.

Additionally, it has been demonstrated that the proposed problem cannot be effectively solved via traditional tree-search techniques due to the non-convex nature of the detour overlapping areas.

In addition to this work acting as a new approach to the BMC problem, it is the authors' belief that the detour selection problem produced by this method may act as a new test-bed for meta- and hyper-heuristic research. In this initial examination, it was shown that GA often found the global-optimal solution in polynomial time, and there is potential for other meta-heuristics to do likewise. Future work in this domain aims to explore hyper-heuristics on more challenging instances of this problem, specifically cases with more than two detour options and hole spaces based on real-world obstacle environments.

REFERENCES

Achat, S., Marzat, J., and Moras, J. (2023). Curved surface inspection by a climbing robot: Path planning approach for aircraft applications. In *ICINCO 2023*.

Arkin, E. M., Fekete, S. P., and Mitchell, J. S. (2000). Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17(1):25–50.

Boyd, S. and Mattingley, J. (2007). Branch and bound methods. *Notes for EE364b, Stanford University*, 2006:07.

Cabreira, T. M., Di Franco, C., Ferreira, P. R., and Buttazzo, G. C. (2018). Energy-aware spiral coverage path planning for uav photogrammetric applications. *IEEE Robotics and automation letters*, 3(4):3662–3668.

Choset, H. and Pignon, P. (1998). Coverage path planning: The boustrophedon cellular decomposition. In Zelinsky, A., editor, *Field and Service Robotics*, pages 203–209, London. Springer London.

Dorigo, M. and Stützle, T. (2019). Ant colony optimization: overview and recent advances. *Handbook of metaheuristics*, pages 311–351.

Gabriely, Y. and Rimon, E. (2001). Spanning-tree based coverage of continuous areas by a mobile robot. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 2, pages 1927–1933 vol.2.

Garrido-Merchán, E. C. and Hernández-Lobato, D. (2020). Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes. *Neurocomputing*, 380:20–35.

Hilbert, D. (1935). Über die stetige abbildung einer linie auf ein flächenstück. In *Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes*, pages 1–2. Springer.

Holland, J. H. (1992). Genetic algorithms. *Scientific american*, 267(1):66–73.

Joshi, A. A., Bhatt, M. C., and Sinha, A. (2019). Modification of hilbert's space-filling curve to avoid obstacles: A robotic path-planning strategy. In *2019 Sixth Indian Control Conference (ICC)*, pages 338–343. IEEE.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical report, Erciyes University, Engineering Faculty Computer Engineering Department,Kayseri/Türkiye.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE.

Khuller, S., Moss, A., and Naor, J. S. (1999). The budgeted maximum coverage problem. *Information processing letters*, 70(1):39–45.

Nair, S. H., Sinha, A., and Vachhani, L. (2017). Hilbert's space-filling curve for regions with holes. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 313–319. IEEE.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming*, 14(1):265–294.

Richards, G. (2022). Hilbert curves, first to third orders. https://commons.wikimedia.org/wiki/File:Hilbert_curve_3.svg. Last checked on Aug 09, 2024.

Van Laarhoven, P. J. and Aarts, E. H. (1987). Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer.

Viana, A. C. and De Amorim, M. D. (2013). Coverage strategy for periodic readings in robotic-assisted monitoring systems. *Ad hoc networks*, 11(7):1907–1918.