# Optimal Perimeter Guarding with Heterogeneous Robot Teams: Complexity Analysis and Effective Algorithms

Si Wei Feng Jingjin Yu [*]

March 22, 2023

**Abstract**

We perform structural and algorithmic studies of significantly generalized versions of the optimal perimeter guarding (OPG) problem [?]. As compared with the original OPG where robots are uniform, in this paper, many mobile robots with heterogeneous sensing capabilities are to be deployed to optimally guard a set of one-dimensional segments. Two complimentary formulations are investigated where one limits the number of available robots (OPG$_{LR}$) and the other seeks to minimize the total deployment cost (OPG$_{MC}$). In contrast to the original OPG which admits low-polynomial time solutions, both OPG$_{LR}$ and OPG$_{MC}$ are computationally intractable with OPG$_{LR}$ being strongly NP-hard. Nevertheless, we develop fairly scalable pseudo-polynomial time algorithms for practical, fixed-parameter subcase of OPG$_{LR}$; we also develop pseudo-polynomial time algorithm for general OPG$_{MC}$ and polynomial time algorithm for the fixed-parameter OPG$_{MC}$ case. The applicability and effectiveness of selected algorithms are demonstrated through extensive numerical experiments.

## 1 Introduction

Consider the scenario where many mobile guards (or sensors) are to be deployed to patrol the perimeter of some 2D regions (Fig. **??**) against intrusion, where each guard may effectively cover a continuous segment of a region's boundary. When part of a boundary need not be secured, e.g., there may already be some existing barriers (the blue segments in Fig. **??**), optimally distributing the robots so that each robot's coverage is minimized becomes an interesting and non-trivial computational task [?]. It is established [?] that, when the guards have the same capabilities, the problem, called the *optimal perimeter guarding* (OPG), resides in the complexity class P (polynomial time class), even when the robots must be distributed across many different boundaries.

In this work, we investigate a significantly more general version of OPG where the mobile guards may be heterogeneous. More specifically, two formulations with different guarding/sensing models are addressed in our study. In the first, the number of available robots is fixed where robots of different types have a fixed ratio of capability (e.g., one type of robot may be able to run faster or may have better sensor). The guarding task must be evenly divided among the robots so that each robot, regardless of type, will not need to bear a too large coverage/capability ratio. This formulation is denoted as *optimal perimeter guarding with limited resources* or OPG$_{LR}$. In the second, the number of robots is unlimited; instead, for each type, the sensing range is fixed with a fixed associated cost. The goal here is to find a deployment plan so as to fully cover the perimeter while minimizing the total cost. We call this the *optimal perimeter guarding with minimum cost* problem, or OPG$_{MC}$.

Unlike the plain vanilla version of the OPG problem, we establish that both OPG$_{LR}$ and OPG$_{MC}$ are NP-hard when the number of robot types is part of the problem input. They are, however, at different hardness levels. OPG$_{LR}$ is shown to be NP-hard in the strong sense, thus reducing the likelihood of finding a fully polynomial time approximation scheme (FPTAS). Nevertheless, for the more practical case where the number of robot types is a constant, we show that OPG$_{LR}$ can be solved using a pseudo-polynomial

[*]S.-W. Feng and J. Yu are with the Department of Computer Science, Rutgers, the State University of New Jersey, Piscataway, NJ, USA. E-Mails: {`siwei.feng, jingjin.yu`} @ rutgers.edu.

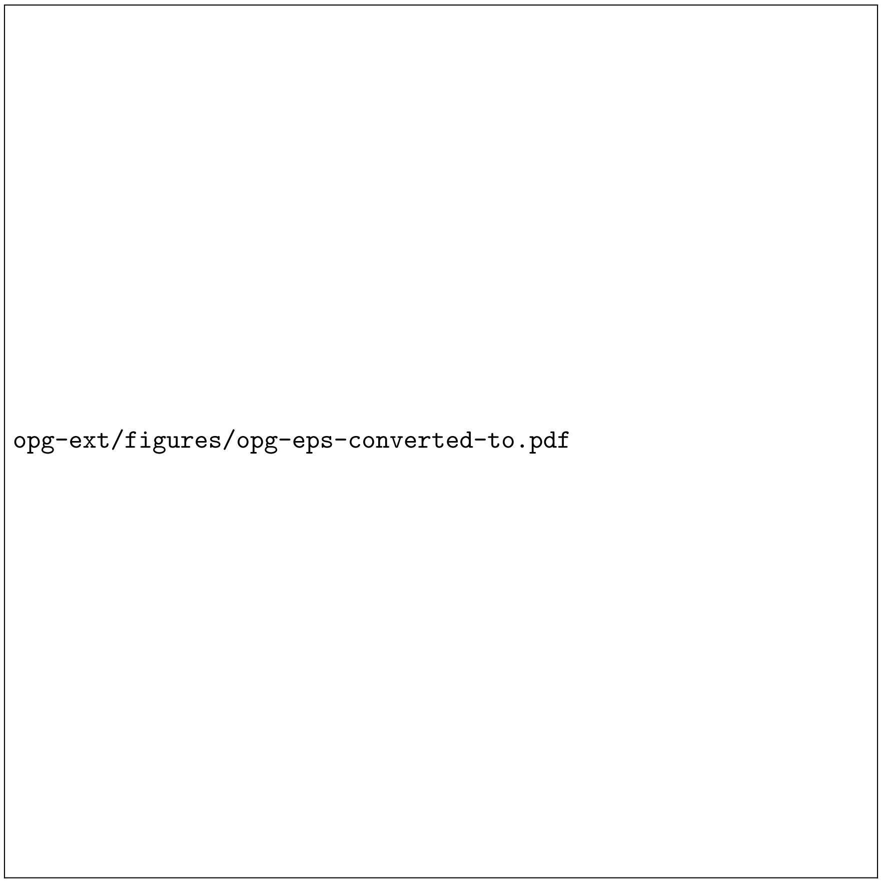

Figure 1: A scenario where boundaries of three (gray) regions must be secured. Zooming in on part of the boundary of one of the regions (the part inside the small circle), portions of the boundary (the red segments) must be guarded while the rest (the blue dotted segments) does not need guarding. For example, the zoomed-in part of the boundary may be monitored by two mobile robots, each patrolling along one of the green segments.

time algorithm with reasonable scalability. On the other hand, we show that OPG$_{MC}$ is weakly NP-hard through the establishment of a pseudo-polynomial time algorithm for OPG$_{MC}$ with arbitrary number of robot types. We further show that, when the number of robot types is fixed, OPG$_{MC}$ can be solved in polynomial time through a fixed-parameter tractable (FPT) approach. This paragraph also summarizes the main contributions of this work.

A main motivation behind our study of the OPG formulations is to address a key missing element in executing autonomous, scalable, and optimal robot deployment tasks. Whereas much research has been devoted to multi-robot motion planning [?, ?] with great success, e.g., [?, ?, ?, ?, ?, ?], existing results in the robotics literature appear to generally assume that a target robot distribution is already provided; the problem of how to effectively generate optimal deployment patterns is largely left unaddressed. It should be noted that control-based solutions to the multi-agent deployment problem do exist, e.g., [?, ?, ?, ?, ?, ?, ?], but the final solutions are obtained through many local iterations and generally do not come with global optimality guarantees. For example, in [?], Voronoi-based iterative methods compute locally optimal target formations for various useful tasks. In contrast, this work, as well as [?], targets the scalable computation of globally optimal solutions.

As a coverage problem, OPG may be characterized as a 1D version of the well-studied Art Gallery problems [?, ?], which commonly assume a sensing model based on line-of-sight visibility [?]; the goal is to ensure that every point in the interior of a given region is visible to at least one of the deployed guards.

Depending on the exact formulation, guards may be placed on boundaries, corners, or the interior of the region. Not surprisingly, Art Gallery problems are typically NP-hard [?]. Other than Art Gallery, 2D coverage problems with other sensing models, e.g., disc-based, have also been considered [?, ?, ?, ?, ?, ?], where some formulations prevent the overlapping of individual sensing ranges [?, ?] while others seek to ensure a full coverage which often requires intersection of sensor ranges. In viewing of these studies, this study helps painting a broader landscape of sensor coverage research.

In terms of structural resemblance, $\text{OPG}_{LR}$ and $\text{OPG}_{MC}$ share many similarities with *bin packing* [?] and other related problems. In a bin packing problem, objects are to be selected to fit within bins of given sizes. Viewing the segments (the red ones in Fig. **??**) as bins, OPG seeks to place guards so that the segments are fully contained in the union of the guards' joint coverage span. In this regard, OPG is a dual problem to bin packing since the former must overfill the bins and the later cannot fully fill the bins. In the extreme, however, both bin packing and OPG converge to a **Subset Sum** [?] like problem where one seeks to partition objects into halves of equal total sizes, i.e., the objects should fit exactly within the bins. With an additional cost term, $\text{OPG}_{MC}$ has further similarities with the **Knapsack** problem [?], which is weakly NP-hard [?].

The rest of the paper is organized as follows. In Section **??**, mathematical formulations of the two OPG variants are fully specified. In Section **??**, both $\text{OPG}_{LR}$ and $\text{OPG}_{MC}$ are shown to be NP-hard. Despite the hardness hurdles, in Section **??**, multiple algorithms are derived for $\text{OPG}_{LR}$ and $\text{OPG}_{MC}$, including effective implementable solutions for both. In Section **??**, we perform numerical evaluation of selected algorithms and demonstrate how they may be applied to address multi-robot deployment problems. We discuss and conclude our study in Section **??**. Please see `https://youtu.be/6gYL0_B3YTk` for an illustration of the problems and selected instances/solutions.

## 2 Preliminaries

Let $\mathcal{W} \subset \mathbb{R}^2$ be a compact (closed and bounded) two-dimensional workspace. There are $m$ pairwise disjoint *regions* $\mathcal{R} = \{R_1, \ldots, R_m\}$ where each region $R_i \subset \mathcal{W}$ is homeomorphic to the closed unit disc, i.e., there exists a continuous bijection $f_i : R_i \to \{(x, y) \mid x^2 + y^2 \leq 1\}$ for all $1 \leq i \leq m$. For a given region $R_i$, let $\partial R_i$ be its (closed) boundary (therefore, $f_i$ maps $\partial R_i$ to the unit circle $\mathbb{S}^1$). With a slight abuse of notation, define $\partial \mathcal{R} = \{\partial R_1, \ldots, \partial R_m\}$. Let $P_i \subset \partial R_i$ be the part of $\partial R_i$ that is accessible, specifially, not blocked by obstacles in $\mathcal{W}$. This means that each $P_i$ is either a single closed curve or formed by a finite number of (possibly curved) line segments. Define $\mathcal{P} = \{P_1, \ldots, P_m\} \subset \mathcal{W}$ as the *perimeter* of $\mathcal{R}$ which must be *guarded*. More formally, each $P_i$ is homeomorphic to a compact subset of the unit circle (i.e., it is assumed that the maximal connected components of $P_i$ are closed line segments). For a given $P_i$, each one of its maximal connected component is called a *perimeter segment* or simply a *segment*, whereas each maximal connected component of $\partial R_i \backslash P_i$ is called a *perimeter gap* or simply a *gap*. An example setting is illustrated in Fig. **??** with two regions.

After deployment, some number of robots are to *cover* the perimeter $\mathcal{P}$ such that a robot $j$ is assigned a continuous closed subset $C_j$ of some $\partial R_i, 1 \leq i \leq m$. All of $\mathcal{P}$ must be *covered* by $\mathcal{C}$, i.e., $\bigcup_{P_i \in \mathcal{P}} P_i \subset \bigcup_{C_j \in \mathcal{C}} C_j$, which implies that elements of $\mathcal{C}$ need not intersect on their interiors. Hence, it is assumed that any two elements of $\mathcal{C}$ may share at most their endpoints. Such a $\mathcal{C}$ is called a *cover* of $\mathcal{P}$. Given a cover $\mathcal{C}$, for a $C_j \in \mathcal{C}$, let $len(C_j)$ denote its length (more formally, measure).

To model heterogeneity of the robots, two models are explored in this study. In either model, there are $t$ types of robots. In the first model, the number of robots of each type is fixed to be $n_1, \ldots, n_t$ with $n = n_1 + \cdots + n_t$. For a robot $1 \leq j \leq n$, let $\tau_j$ denote its type. Each $1 \leq \tau \leq t$ type of robots has some level of *capability* or *ability* $a_\tau \in \mathbb{Z}^+$. We wish to balance the load among all robots based on their capabilities, i.e., the goal is to find cover $\mathcal{C}$ for all robots such that the quantity

$$\max_{C_j \in \mathcal{C}} \frac{len(C_j)}{a_{\tau_j}},$$

which represents the largest coverage-capacity ratio, is minimized. We note that when all capacities are the same, e.g., $a_\tau = 1$ for all robots, this becomes the standard OPG problem studied in [?]. We call this

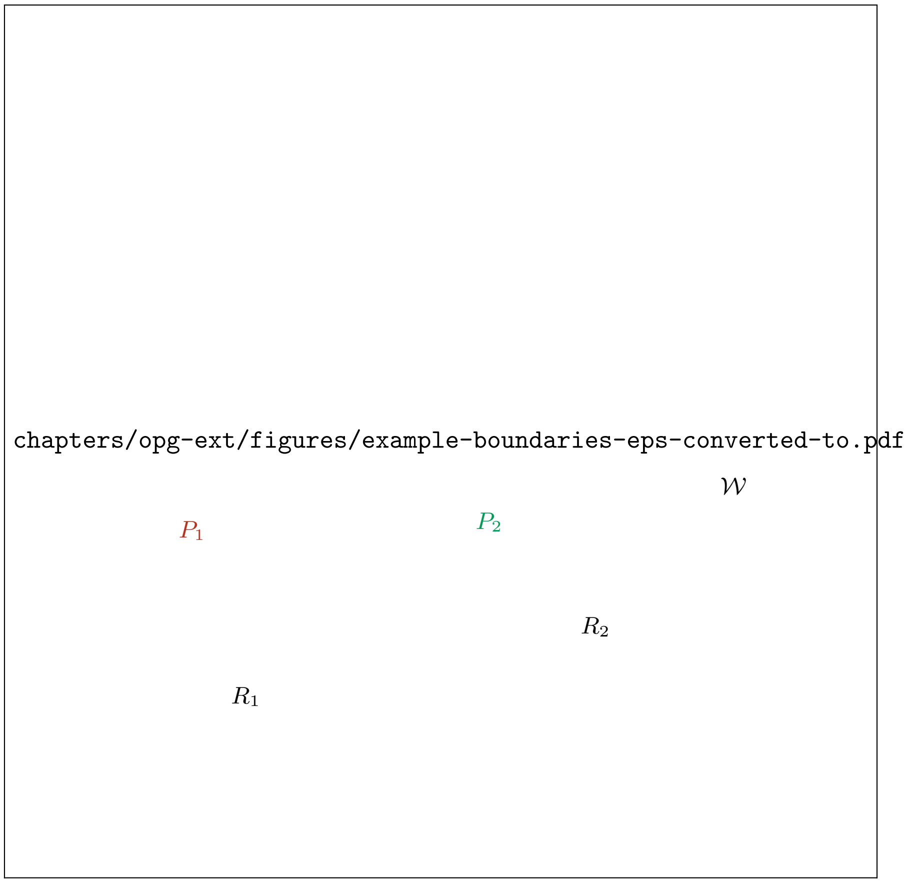
chapters/opg-ext/figures/example-boundaries-eps-converted-to.pdf
$\mathcal{W}$
$P_1$
$P_2$
$R_2$
$R_1$

Figure 2: An example of a workspace $\mathcal{W}$ with two regions $\{R_1, R_2\}$. Due to three *gaps* on $\partial R_1$, marked as dotted lines within long rectangles, $P_1 \subset \partial R_1$ has three *segments* (or maximal connected components); $P_2 = \partial R_2$ has a single segment with no gap.

version of the perimeter guarding problem *optimal perimeter guarding with limited resources* or OPG$_{LR}$. The formal definition is as follows.

**Problem 2.1** (Optimal Perimeter Guarding with Limited Resources (OPG$_{LR}$))**.** *Let there be $t$ types of robots. For each type $1 \leq \tau \leq t$, there are $n_\tau$ such robots, each having the same capability parameter $a_\tau$. Let $n = n_1 + \cdots + n_t$. Given the perimeter set $\mathcal{P} = \{P_1, \ldots, P_m\}$ of a set of 2D regions $\mathcal{R} = \{R_1, \ldots, R_m\}$, find a set of $n$ continuous line segments $\mathcal{C}^* = \{C_1^*, \ldots, C_n^*\}$ such that $\mathcal{C}^*$ covers $\mathcal{P}$, i.e.,*

$$\bigcup_{P_i \in \mathcal{P}} P_i \subset \bigcup_{C_j^* \in \mathcal{C}^*} C_j^*, \tag{1}$$

*such that a $C_j^*$ is covered by robot $j$ of type $\tau_j$, and such that, among all covers $\mathcal{C}$ satisfying* (**??**),

$$\mathcal{C}^* = \operatorname*{argmin}_{\mathcal{C}} \max_{C_j \in \mathcal{C}} \frac{len(C_j)}{a_{\tau_j}}. \tag{2}$$

Whereas the first model caps the number of robots, the second model fixes the maximum coverage of each type of robot. That is, for each robot type $1 \leq \tau \leq t$, $n_\tau$, the number of robots of type $\tau$, is unlimited as long as it is non-negative, but each such robot can only cover a maximum length of $\ell_\tau$. At the same time, using each such robot incurs a cost of $c_\tau$. The goal here is to guard the perimeters with the minimum total cost. We denote this problem *optimal perimeter guarding with minimum cost* or OPG$_{MC}$.

**Problem 2.2** (Optimal Perimeter Guarding with Minimum Cost ($\text{OPG}_{MC}$))**.** *Let there be $t$ types of robots of unlimited quantities. For each robot of type $1 \leq \tau \leq t$, it can guard a length of $\ell_\tau \in \mathbb{Z}^+$ with a cost of $c_\tau \in \mathbb{Z}^+$. Given the perimeter set $\mathcal{P} = \{P_1, \ldots, P_m\}$ of a set of 2D regions $\mathcal{R} = \{R_1, \ldots, R_m\}$, find a set of $n = n_1 + \cdots + n_t$ continuous line segments $\mathcal{C}^* = \{C_1^*, \ldots, C_n^*\}$ where $n_\tau$ such segments are guarded by type $\tau$ robots, such that $\mathcal{C}^*$ covers $\mathcal{P}$, i.e.,*

$$\bigcup_{P_i \in \mathcal{P}} P_i \subset \bigcup_{C_j^* \in \mathcal{C}^*} C_j^*, \tag{3}$$

*such that a $C_j^*$ is covered by robot $j$ of type $\tau_j$, i.e., $C_j^* \leq \ell_{\tau_j}$, and such that, among all covers $\mathcal{C}$ satisfying* (**??**),

$$\mathcal{C}^* = \underset{\mathcal{C}}{\operatorname{argmin}} \sum_{1 \leq \tau \leq t} n_\tau c_\tau. \tag{4}$$

## 3 Computational Complexity for Variable Number of Robot Types

We explore in this section the computational complexity of $\text{OPG}_{LR}$ and $\text{OPG}_{MC}$. Both problems are shown to be NP-hard with $\text{OPG}_{LR}$ being strongly NP-hard. We later confirm that $\text{OPG}_{MC}$ is weakly NP-hard (in Section **??**).

### 3.1 Strong NP-hardness of $\text{OPG}_{LR}$

When the number of types $t$ is a variable, i.e., $t$ is not a constant and may be arbitrarily large, $\text{OPG}_{LR}$ is shown to be NP-hard via the reduction from 3-**Partition** [**?**]:

PROBLEM: 3-**Partition**
INSTANCE: A finite set A of $3m$ elements, a bound $B \in \mathbb{Z}^+$, and a "size" $s(a) \in \mathbb{Z}^+$ for each $a \in A$, such that each $s(a)$ satisfies $B/4 < s(a) < B/2$ and $\sum_{a \in A} s(a) = mB$.
QUESTION: Is there a partition of $S$ into $m$ disjoint subsets $S_1, \ldots, S_m$ such that for $1 \leq i \leq m$, $\sum_{a \in S_i} s(a) = B$?

3-**Partition** is shown to be NP-complete in the strong sense [**?**], i.e., it is NP-complete even when all numeric inputs are bounded by a polynomial of the input size.

For the reduction, it is more convenient to work with a decision version of the $\text{OPG}_{LR}$ problem, denoted as D-$\text{OPG}_{LR}$. In the D-$\text{OPG}_{LR}$ problem, $a_\tau$ is the actual length robot type $\tau$ covers. That is, the coverage length of a robot is fixed. The D-$\text{OPG}_{LR}$ problem is specified as follows.

PROBLEM: D-$\text{OPG}_{LR}$
INSTANCE: $t$ types of robots where there are $n_\tau$ robots for each type $1 \leq \tau \leq t$; $n = n_1 + \cdots + n_t$. A robot of type $\tau$ has a coverage capacity $a_\tau$. A set of perimeters $\mathcal{P} = \{P_1, \ldots, P_m\}$ of a set of 2D regions $\mathcal{R} = \{R_1, \ldots, R_m\}$.
QUESTION: Is there a deployment of $n$ disjoint subsets $C_1, \ldots, C_n$ of $\{\partial R_1, \ldots, \partial R_m\}$ such that $P_1 \cup \ldots \cup P_m \subset C_1 \cup \ldots \cup C_n$, where $C_j$ is a continuous segment for all $1 \leq j \leq n$, and for each $1 \leq j \leq n$, there is a unique robot whose type $\tau$, $1 \leq \tau \leq t$ satisfies $a_\tau \geq len(C_j)$?

**Theorem 3.1.** $\text{OPG}_{LR}$ *is strongly NP-hard.*

*Proof.* A polynomial reduction from 3-**Partition** to D-$\text{OPG}_{LR}$ is constructed by a restriction of D-$\text{OPG}_{LR}$. Given a 3-**Partition** instance with former notations, we apply several restrictions on D-$\text{OPG}_{LR}$: *(i)* there are $3m$ types of robot and there is a single robot for each type, i.e., $n_\tau = 1$ for $1 \leq \tau \leq t$, so $n = t = 3m$ *(ii)* the $3m$ capacities $a_1, \ldots, a_{3m}$ are set to be equal to $s(a)$ for each of the $3m$ elements $a \in A$, and *(iii)* there are $3m$ perimeters and each perimeter $P_i$ is continuous and $len(P_i) = B$ for all $1 \leq i \leq m$.

With the setup, the reduction proof is straightforward. Clearly, the 3-**Partition** instance admits a partition of $A$ into $S_1, \ldots, S_m$ such that $\sum_{a \in S_i} s(a) = B$ for all $1 \leq i \leq m$ if and only if a valid depolyment exists in the corresponding D-$\text{OPG}_{LR}$ instance. It is clear that the reduction from 3-**Partition** to D-$\text{OPG}_{LR}$

is polynomial (in fact, linear). Based on the reduction and because **3-Partition** is strongly NP-hard, so is D-OPG$_{LR}$ and OPG$_{LR}$. □

*Remark.* One may also reduce weakly NP-hard problems, e.g., **Partition** [**?**], to OPG$_{LR}$ for variable number of robot types $t$. Being strongly NP-hard, OPG$_{LR}$ is unlikely to admit pseudo-polynomial time solutions for variable $t$. This contrasts with a later result which provides a pseudo-polynomial time algorithm for OPG$_{LR}$ for constant $t$, as one might expect in practice where robots have limited number of types. We also note that Theorem **??** continues to hold for a single perimeter with multiple segments, each having a length $B$ in previous notation, separated by "long" gaps. Obviously, D-OPG$_{LR}$ is in NP, thus rendering it NP-complete.

## 3.2 NP-hardness of OPG$_{MC}$

The minimum cost OPG variant, OPG$_{MC}$, is also NP-hard, which may be established through reduction from the **Subset Sum** problem [**?**]:

PROBLEM: **Subset Sum**
INSTANCE: A set $B$ with $|B| = n$ and a weight function $w : B \to \mathbb{Z}^+$, and an integer $W$.
QUESTION: Is there a subset $B' \subseteq B$ such that $\sum_{b \in B'} w(b) = W$?

**Theorem 3.2.** OPG$_{MC}$ *is NP-hard.*

*Proof.* Given a **Subset Sum** instance, we construct an OPG$_{MC}$ instance with a single perimeter containing a single segment with length $L$ to be specified shortly. Let there be $t = 2n$ types of robots. For $1 \le i \le n$, let robot type $2i-1$ have $\ell_{2i-1} = c_{2i-1} = w(b_i) + (2^{n+1} + 2^i)W'$ and let robot type $2i$ have $\ell_{2i} = c_{2i} = (2^{n+1} + 2^i)W'$. Here, $W'$ can be any integer number no less than $\sum_{b \in B} w(b)$. Set $L = W + (n2^{n+1} + 2^n + \ldots + 2^1)W'$. We ask the "yes" or "no' decision question of whether there are robots that can be allocated to have a total cost no more than $L$ (equivalently, equal to $L$, as the cost density $c_\tau / l_\tau$ is always 1).

Suppose the **Subset Sum** instance has a yes answer that uses a subset $B' \subseteq B$. Then, the OPG$_{MC}$ instance has a solution with cost $L$ that can be constructed as follows. For each $1 \le i \le n$, a single robot of type $2i-1$ is taken if $b_i \in B'$. Otherwise, a single robot of type $2i$ is taken. This allocation of robots yields a total length and cost of $L$.

For the other direction, we first show that if the OPG$_{MC}$ instance is to be satisfied, it can only use a single robot from type $2i-1$ or $2i$ for all $1 \le i \le n$. First, if more than $n$ robots are used, then the total cost exceeds $(n+1)2^{n+1}W' > L$ as $W \le W'$. Similarly, if less than $n$ robots are used, the total length is at most $(n-1)2^{n+1}W' + (2^{n+1} - 1)W' + W' < L$. Also, to match the $(2^n + \ldots + 2)W'$ part of the cost, exactly one robot from type $2i-1$ or $2i$ for all $1 \le i \le n$ must be taken. Now, if the OPG$_{MC}$ decision instance has a yes answer, if a robot of type $2i-1$ is used, let $b_i \in B$ be part of $B'$, which constructs a $B'$ that gives a yes answer to the **Subset Sum** instance. □

*Remark.* It is also clear that the decision version of the OPG$_{MC}$ problem is NP-complete. The **Subset Sum** is a weakly NP-hard problem that admits a pseudo-polynomial time algorithm [**?**]. As it turns out, OPG$_{MC}$, which shares similarities with **Subset Sum** and **Knapsack** (in particular, **Unbounded Knapsack** [**?**]), though NP-hard, does admit a pseudo-polynomial time algorithm as well.

# 4 Exact Algorithms for OPG$_{LR}$ and OPG$_{MC}$

In this section, we describe three exact algorithms for solving the two variations of the OPG problem. First, we present a pseudo-polynomial time algorithm for OPG$_{LR}$ when the number of robot types, $t$, is a fixed constant. Given that OPG$_{LR}$ is strongly NP-hard, this is in a sense a best possible solution. For OPG$_{MC}$, in addition to providing a pseudo-polynomial algorithm for arbitrary $t$, which confirms that OPG$_{MC}$ is weakly NP-hard, we also provide a polynomial time approximation scheme (PTAS). We then further show the possibility of solving OPG$_{MC}$ in polynomial time when $t$ is a fixed constant. We mention that our development in this section focuses on the single perimeter case, i.e., $m = 1$, as the generalization to arbitrary $m$ is straightforward using techniques described in [?]. With this in mind, we also provide the running times for the general setting with arbitrary $m$ but refer the readers to [?] on how these running times can be derived.

For presenting the analysis and results, for the a perimeter $P$ that we work with, assume that it has $q$ perimeter segments $S_1, \ldots, S_q$ that need to be guarded; these segments are separated by $q$ gaps $G_1, \ldots, G_q$. For $1 \leq i, i' \leq q$, define $S_{i \sim i'} = S_i \cup G_i \cup S_{i+1} \cup \ldots \cup G_{i'-1} \cup S_{i'}$ where $i'$ may be smaller than $i$ (i.e., $S_{i \sim i'}$ may wrap around $G_q$), For the general case with $m$ perimeters, assume that a perimeter $P_i$ has $q_i$ segments.

## 4.1 Pseudo-Polynomial Time Algorithm for $\mathrm{OPG}_{LR}$ with Fixed Number of Robot Types

We set to develop an algorithm for $\mathrm{OPG}_{LR}$ for arbitrary $t$, the number of robot types; the algorithm runs in pseudo-polynomial time when $t$ is a constant. At a higher level, our proposed algorithm works as follows. First, our main effort here goes into deriving a feasibility test for D-$\mathrm{OPG}_{LR}$ as defined in Section **??**. With such a feasibility test, we can then find the optimal $\frac{len(C_j)}{a_{\tau_j}}$ in (**??**) via binary search. Let us denote the optimal value of $\frac{len(C_j)}{a_{\tau_j}}$ as $\ell^*$.

*4.1.1 Feasibility Test for* D-$\mathrm{OPG}_{LR}$The feasibility test for D-$\mathrm{OPG}_{LR}$ essentially tries different candidate $\ell$ to find $\ell^*$. Our implementation uses ideas similar to the pseudo-polynomial time algorithm for the **Knapsack** problem which is based on dynamic programming (DP). In the test, we work with a fixed starting point on $P$, which is set to be the counterclockwise end point of a segment $S_i$, $1 \leq i \leq q$. Essentially, we maintain a $t$ dimensional array $M$ where dimension $\tau$ has a size of $n_\tau + 1$. An element of the array, $M[n'_1] \ldots [n'_t]$, holds the maximal distance starting from $S_i$ that can be covered by $n'_1$ type 1 robots, $n'_2$ type 2 robots, and so on. The DP procedure OPG-LR-FEASIBLE $(i, \ell)$, outlined in Algorithm **??**, incrementally builds this array $M$. For convenience, in the pseudo code, $M[\vec{x}]$ denotes an element of $M$ with $\vec{x}$ being a $t$ dimensional integer vector.

**Algorithm 1:** OPG-LR-FEASIBLE $(i, \ell)$

```
   Data: $n_1, \ldots, n_t$, $a_1, \ldots, a_t$, $S_1, \ldots, S_q$, $G_1, \ldots, G_q$
   Result: true or false, indicating whether $S_1, \ldots, S_q$ can be covered
1  Initialize $M$ as a $t$ dimensional array with dimension $\tau$ having a size of $n_\tau + 1$;
2  $\ell_\tau \leftarrow a_\tau \ell$ for all $1 \leq \tau \leq t$;
3  for $\vec{x} \in [0, n_1] \times \cdots \times [0, n_t]$ do
4      $M[\vec{x}] \leftarrow 0$;
5      for $j = 1$ to $t$ do
6          if $\vec{x}_j = 0$ then continue;
7          $\vec{x'} \leftarrow \vec{x}$; $\vec{x'}_j \leftarrow \vec{x'}_j - 1$;
8          $M[\vec{x}] \leftarrow max(M[\vec{x}],$ INC $(M[\vec{x'}], \ell_j))$;
9      end
10 end
11 return $M[n_1] \ldots [n_t] \geq len(S_{i \sim i-1})$;
```

In Algorithm **??**, the procedure INC $(L, \ell)$ checks how much of the perimeter $P$ can be covered when an additional coverage length $\ell$ is added, assuming that a distance of $L$ (starting from some $S_i$) is already covered. An illustration of how INC $(L, \ell)$ works is given in Fig. **??**.

By simple counting, the complexity of the algorithm is $O(q \cdot t \cdot \Pi_{\tau=1}^{t}(n_\tau + 1))$. However, the amortized complexity of INC $(\cdot)$ for each $\tau$ is $O(q + n_\tau)$; the algorithm thus runs in $O(t \cdot \Pi_{\tau=1}^{t}(n_\tau + 1) + q \cdot \sum_{\tau=1}^{t} \Pi_{\tau' \neq \tau}(n_{\tau'} + 1))$, which is pseudo-polynomial for fixed $t$. After trying every possible starting position $i$ with OPG-LR-FEASIBLE $(i, \ell)$, for a fixed candidate $\ell$, D-$\mathrm{OPG}_{LR}$ is solved in $O(q \cdot t \cdot \Pi_{\tau=1}^{t}(n_\tau + 1) + q^2 \cdot \sum_{\tau=1}^{t} \Pi_{\tau' \neq \tau}(n_{\tau'} + 1))$.

*4.1.2 Solving* $\mathrm{OPG}_{LR}$ *using Feasibility Test for* D-$\mathrm{OPG}_{LR}$Using OPG-LR-FEASIBLE $(i, \ell)$ as a subroutine to check feasibility for a given $\ell$, bisection can be applied over candidate $\ell$ to obtain $\ell^*$. For completing the algorithm, one needs to establish when the bisection will stop (notice that, even though we assume that

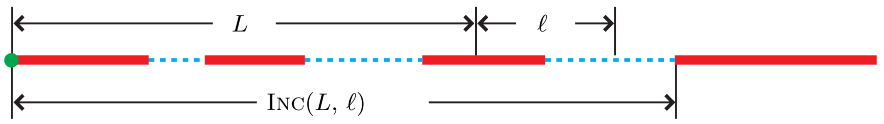

Figure 3: Suppose starting from the fixed left point, a length of $L$ on the boundary is successfully guarded by a group of robots. Then, a robot with coverage capacity $\ell$ is appended to the end of the group of robots to increase the total guarded distance. In the figure, the added additional capacity $\ell$ can fully cover the third red segment plus part of the third (dashed) gap. Because there is no need to cover the rest of the third gap, $\textsc{Inc}(L,\ \ell)$ extends to the end of the gap.

$a_\tau \in \mathbb{Z}^+$, for each $1 \leq \tau \leq t$, $\ell^*$ need not be an integer).

To derive the stop criterion, we note that given the optimal $\ell^*$, there must exist some $S_{i\sim i'}$ that is "exactly" spanned by the allocated robots. That is, assume that $S_{i\sim i'}$ is covered by $n_1'$ of type 1 robots and $n_2'$ of type 2 robots, and so on, then

$$\ell^* = \frac{len(S_{i\sim i'})}{\sum_{1\leq\tau\leq t} a_\tau \cdot n_\tau'}. \tag{5}$$

(**??**) must hold for some $S_{i\sim i'}$ because if not, the solution is not tight and can be further improved. Therefore, the bisection process for locating $\ell^*$ does not need to go on further after reaching a certain granularity [**?**]. With this established, using similar techniques from [**?**] (we omit the technical detail as it is quite complex but without additional new ideas beyond beside what is already covered in [**?**]), we could prove that the full algorithm needs no more than $O(q\log(\sum_\tau n_\tau + q)$ calls to OPG-LR-FEASIBLE $(i, \ell)$. This directly implies that $\mathrm{OPG}_{LR}$ also admits a pseudo-polynomial algorithm for fixed $t$.

*4.1.3 Multiple Perimeters*Also using techniques developed in [**?**], the single perimeter result can be readily generalized to multiple perimeters. We omit the mechanical details of the derivation and point out that the computational complexity in this case becomes $\tilde{O}((m-1)\cdot((\Pi_{\tau=1}^t n_\tau)/\max_\tau n_\tau)^2 + \sum_{k=1}^m (t\cdot q_k \cdot \Pi_{\tau=1}^t (n_\tau + 1) + q_k^2 \sum_{\tau=1}^t \Pi_{\tau'\neq\tau}(n_{\tau'}+1)))$.

## 4.2 Polynomial Time Algorithm for $\mathrm{OPG}_{MC}$ with Fixed Number of Robot Types

The solution to $\mathrm{OPG}_{MC}$ will be discussed here. A method based on DP will be provided first, which leads to a polynomial time algorithm for a fixed number of robot types and a pseudo-polynomial time algorithm when the number of robot types is not fixed. For the latter case, a polynomial time approximation scheme (PTAS) will also be briefly described.

*4.2.1 Dynamic Programming Procedure for* $\mathrm{OPG}_{MC}$When no gaps exist, the optimization problem becomes a covering problem as follows. Let $c_\tau$, $\ell_\tau$, $n_\tau$ correspond to the cost, coverage length, and quantity of robot type $\tau$, respectively, and let total length to cover be $L$. We are to solve the optimization problem

$$\min \sum_\tau c_\tau \cdot n_\tau \quad s.t. \quad \sum_\tau \ell_\tau \cdot n_\tau \geq L, n_\tau \geq 0. \tag{6}$$

Let the solution to the above integer programming problem be $\textsc{Sol}(L)$. Notice that, for $S_{i\sim i'} := \{S_i,\ G_i, \ldots, G_{i'-1}, S_{i'}\}$, the minimum cost cover is by either: *(i)* covering the total boundary without skipping any gaps, or *(ii)* skipping or partially covering some gap, for example $G_k, i \leq k \leq j-1$. In the first case, the minimum cost is exactly $\textsc{Sol}(\lceil len(S_{i\sim(i+k)})\rceil)$. In the second case, the optimal structure for the two subsets of perimeter segments $S_{i\sim k}$ and $S_{(k+1)\sim j}$ still holds. This means that the continuous perimeter segments $S_{i\sim j}$ can be divided into two parts, each of which can be treated separately. This leads to a DP approach for $\mathrm{OPG}_{MC}$. With $M[i][j]$ denoting the minimum cost to cover $S_{i\sim j}$, the DP recursion is given by

$$M[i][j] = \min(\textsc{Sol}(\lceil len(S_{i\sim j})\rceil), \min_k(M[i][k] + M[k+1][j]))$$

The DP procedure is outlined in Algorithm **??**. In the pseudo code, it is assumed that indices of $M$ are modulo $q$, e.g., $M[2][q+1] \equiv M[2][1]$. *tmp* is a temporary variable.

```
Algorithm 2: OPG-MC-DP
   Data: $\ell_1, \ldots, \ell_t$, $c_1, \ldots, c_t$, $S_1, \ldots, S_q$, $G_1, \ldots, G_q$
   Result: $c^*$, the minimum covering cost
1  $M \leftarrow$ a $q \times q$ matrix; $c^* \leftarrow \infty$;
2  for $k \leftarrow 0$ to $q-1$ do
3      for $i \leftarrow 1$ to $q$ do
4          $tmp \leftarrow \textsc{Sol}(\lceil len(S_{i\sim(i+k)}) \rceil)$;
5          for $j \leftarrow i$ to $i+k-1$ do
6              $tmp \leftarrow \min(tmp, M[i][j] + M[j+1][i+k])$;
7          end
8          $M[i][i+k] \leftarrow c$;
9          if $k = q-1$ then $c^* \leftarrow \min(c^*, M[i][i+k])$;
10     end
11 end
12 return $c^*$;
```

*4.2.2 A Polynomial Time Algorithm for* OPG$_{MC}$ *for a Fixed Number of Robot Types*We mention briefly that, by a result of Lenstra [?], the optimization problem (**??**) is in P (i.e., polynomial time) when $t$ is a constant. The running time of the algorithm [?] is however exponential in $t$.

*4.2.3 A Pseudo-polynomial Time Algorithm for Arbitrary t*As demonstrated in the hardness proof, similarities exist between OPG and the **Knapsack** problem. The connection actually allows the derivation of a pseudo-polynomial time algorithm for arbitrary $t$. To achieve this, we use a routine to pre-compute Sol($L$), called PreSolve(), which is itself a DP procedure similar to that for the **Knapsack** problem. The pseudo code of PreSolve() is given in Algorithm **??**. PreSolve() runs in time $O(t \cdot \lceil len(\partial R) \rceil))$. Overall, Algorithm **??** then runs in time $O(q^3 + t \cdot \lceil len(\partial R \rceil))$.

```
Algorithm 3: PreSolve
   Data: $\ell_1, \ldots, \ell_t$, $c_1, \ldots, c_t$
   Result: A lookup table for retrieving Sol($L$)
1  $I_{max} = \lceil len(\partial R) \rceil$; %$I_{max}$ is an integer.
2  $M' \leftarrow$ an array of length $I_{max} + 1$; $M'[0] \leftarrow 0$;
3  for $L \leftarrow 1$ to $I_{max}$ do
4      $M'[L] \leftarrow \infty$;
5      for $\tau \leftarrow 1$ to $t$ do
6          $tmp \leftarrow (L < \ell_\tau \ ?\ 0 \ :\ M'[L - \ell_\tau]) + c_\tau$;
7          $M'[L] \leftarrow min(M'[L], tmp)$;
8      end
9  end
10 return $M'$
```

With the establishment of a pseudo-polynomial time algorithm for OPG$_{MC}$, we have the following corollary.

**Corollary 4.1.** OPG$_{MC}$ *is weakly NP-hard.*

*4.2.4 FPTAS for Arbitrary t*When the number of robot types is not fixed, Lenstra's algorithm [?] or its variants no longer run in polynomial time. We briefly mention that, by slight modifications of a FPTAS for **Unbounded Knapsack** problem from [?], a FPTAS for OPG$_{MC}$ can be obtained that runs in time $O(q^3 + q^2 \cdot \frac{t}{\epsilon^3})$, where $(1+\epsilon)$ is the approximation ratio for both OPG$_{MC}$ and (**??**).

*4.2.5 Multiple Perimeters*For OPG$_{MC}$, when there are multiple perimeters, e.g., $P_1, \ldots, P_m$, a optimal solution can be obtained by optimally solving OPG$_{MC}$ for each perimeter $P_i$ individually and then put together the solutions.

## 5 Performance Evaluation and Applications

In this section, we provide examples illustrating the typical optimal solution structures of OPG$_{LR}$ and OPG$_{MC}$ computed by our DP algorithms. Using an application scenario, solutions to OPG$_{LR}$ and OPG$_{MC}$ are also compared. Then, computational results from extensive numerical evaluations are presented, confirming the effectiveness of these algorithms. The implementation is done using the Python and all computations are performed on an Intel(R) Core(TM) i7-7700 CPU@3.6GHz with 16GB RAM.

### 5.1 Basic Optimal Solution Structure

Fig. **??** shows the typical outcome of solving an OPG$_{LR}$ instance with two perimeters ($m = 2$) for two types of robots with $n_1 = 3, a_1 = 5$, and $n_2 = 5, a_2 = 8$. In the figure, the red segments are parts of the two perimeters that must be guarded. The three orange (resp., five green) segments across the two perimeters indicate the desired coverage regions of the three (resp., five) type 1 (resp., type 2) robots. These coverage regions correspond to the optimal solution returned by the DP algorithm. As may be observed, the optimal solution is somewhat complex with robots of both types on each of the two perimeters; a gap on the second boundary also gets covered. The coverage lengths for a robot type are generally different; this is due to adjustments that shrink some robots' coverage. For example, the first perimeter has a very short orange cover because the corresponding perimeter segment is short and gaps around it need not be covered (The adjustment procedure is also shown in the video).

`chapters/opg-ext/figures/mopglr_shrink-new-eps-converted-to.pd`

Figure 4: An OPG$_{LR}$ problem and an associated optimal solution. The problem has two perimeters and $t = 2$ with $n_1$=3, $n_2$=5, $a_1$=5, $a_2$=8. The boundaries are shown as circles for ease of illustration.

Shifting our attention to OPG$_{MC}$, Fig. **??** illustrates the structure of an optimal solution to a problem with three types of robots with capacities and costs being $\ell_1 = 11, c_1 = 2, \ell_t = 30, c_2 = 4$, and $\ell_3 = 55, c_3 = 7$, respectively. In this case, the majority of the deployed robots are of type 2 with $\ell_2 = 30, c_2 = 4$. Only one type 1 and one type 3 robots are used. The four perimeter segments are covered by three robot groups. The only type 3 robot guards (the purple segment) across two different perimeter segments. Coverage length adjustment is also performed to avoid the unnecessary coverage of some gaps.

`chapters/opg-ext/figures/opgmc-new-t-eps-converted-to.pdf`

Figure 5: An OPG$_{MC}$ problem and an associated optimal solution. The problem has four (red) perimeter segments and three types of robots with $\ell_1 = 11, c_1 = 2$ (orange), $\ell_t = 30, c_2 = 4$ (green), and $\ell_3 = 55, c_3 = 7$ (purple), respectively.

### 5.2 A Robotic Guarding and Patrolling Application

In this subsection, as a potential application, the DP algorithms for OPG$_{LR}$ and OPG$_{MC}$ are employed to solve the problem of securing the perimeter of the Edinburgh castle, an example used in [**?**]. As shown in Fig. **??** (minus the orange and green segments showing the solutions), the central region of the Edinburgh castle has tall buildings on its boundary (the blocks in brick red); these parts of the boundary are the gaps that do not need guarding. In the figure, the top sub-figure shows the optimal solution for an OPG$_{LR}$ instance and an OPG$_{MC}$ instance with a total of 11 robots. The bottom sub-figure is a slightly updated OPG$_{MC}$ instance with slightly higher $c_2$.

Figure 6: [left] OPG$_{LR}$ solution with $n_1 = 4, n_2 = 7, c_1 : c_2 = 2 : 3$ and OPG$_{MC}$ solution with $\ell_1 = 150, c_1 = 100, \ell_2 = 225, c_2 = 145$, and total boundary 3058. Cost of OPG$_{MC}$ solution is 1415. [right] OPG$_{MC}$ solution with $\ell_1 = 150, c_1 = 100, \ell_2 = 225, c_2 = 155$. Cost of solution (13 type 1, 1 type 2) is 1455. In both solutions, covers by type 1 (resp., type 2) robots are shown in orange (resp., green).

It can be observed that the results, while having non-trivial structures, make intuitive sense. For the top sub-figure, solutions to both OPG$_{LR}$ and OPG$_{MC}$ (because robot with larger capacity is slightly lower in relative cost) use mainly higher capacity robots to cover longer perimeter segments and use the lower capacity robots mostly fillers. The solution covers a small gap at the bottom. For the bottom sub-figure, while only small changes are made to the cost, because the longer segment is more expensive to use now, the first type of robot is used mainly.

## 5.3 Computational Performance

With Section **??** fully establishing the correctness and asymptotic complexity of the pseudo-polynomial time algorithms, here, the running time of these algorithms are experimentally evaluated. In doing so, the main goal is demonstrating that, despite the hardness of OPG$_{LR}$ and OPG$_{MC}$, the proposed algorithms could solve the target problems under reasonably broad settings in a scalable way. For results presented in this subsection, each data point is an average over 10 randomly generated instances.

The first two numerical evaluations (Table **??** and Table **??**) focus on the running times of the pseudo-polynomial time algorithms for OPG$_{LR}$ over single and multiple perimeters, respectively. In these two tables, $t$ and $q$ are the number of types and the number of segments, respectively. For each type $\tau$, a capacity ($a_\tau$) is randomly sampled as an integer between 1 and 100, inclusive. The number of robots available for each type ($n_\tau$) is sampled uniformly between 5 and 15, inclusive. For the multiple perimeters case, the parameter $m$ represents the number of perimeters for a given instance.

For the single perimeter case (Table **??**), the results show that the pseudo-polynomial time algorithm is effective for up to five types of robots, for dozens of robots. We expect a more efficient (e.g., C++ based) implementation should be able to effectively handle up to five types of robots with the total number of robots being around a hundred, on a typical PC. This is likely sufficient for many practical applications which have limited types and numbers of robots. Since the algorithm has exponential dependency on $t$, it becomes less efficient for larger $t$ as expected.

| $t$ \ $q$ | 5 | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|
| 2 | 0.022 | 0.044 | 0.131 | 0.208 | 0.326 | 0.516 |
| 3 | 0.281 | 0.714 | 1.670 | 2.577 | 4.107 | 4.708 |
| 4 | 5.504 | 16.07 | 41.68 | 71.55 | 109.9 | 138.9 |
| 5 | 29.53 | 75.60 | 243.6 | 443.4 | 528.0 | 725.0 |

Table 1: Running time in seconds used by the DP algorithm for OPG$_{LR}$ over a single perimeter.

Table **??** illustrates the running time of the DP algorithm for OPG$_{LR}$ over multiple perimeters. As can be readily observed, the impact of the number of perimeters $m$ on the running time is relatively small; the number of robot types is still the determining factor for running time. In this case, our proposed solution is effective for $t$ up to 4 and starts to slow down a robot types become larger than 4.

| $m$ \ $q$ | 10 | | 20 | | 30 | |
|---|---|---|---|---|---|---|
| | $t$=3 | $t$=4 | $t$=3 | $t$=4 | $t$=3 | $t$=4 |
| 2 | 3.148 | 133.2 | 7.077 | 198.4 | 10.33 | 260.0 |
| 3 | 4.828 | 194.1 | 10.125 | 290.6 | 15.52 | 376.7 |
| 4 | 6.131 | 256.8 | 12.485 | 381.3 | 19.75 | 514.3 |
| 5 | 7.622 | 321.7 | 15.355 | 476.2 | 24.31 | 605.8 |

Table 2: Running time in seconds used by the DP algorithm for OPG$_{LR}$ over multiple perimeters.

Table **??** provides performance evaluation of OPG-MC-DP. Since there is no difference between single and multiple perimeters for OPG$_{MC}$, only problems with single perimeters are attempted. Here, for each robot type, the cost is an integer randomly sampled between 1 and 20, and the capacity is computed as five times the cost plus a random integer between 1 and 20. In the table, $L = \partial R$, the total length of the entire boundary. Given OPG$_{MC}$'s lower computational complexity, the DP algorithm, OPG-MC-DP, can effectively deal with over a few hundred types of robots with ease.

| $t$ \ $L$ | $10^2$ | | $10^4$ | | $10^6$ | |
|---|---|---|---|---|---|---|
| | $q$=20 | $q$=50 | $q$=20 | $q$=50 | $q$=20 | $q$=50 |
| 3 | 0.006 | 0.064 | 0.041 | 0.098 | 3.040 | 3.144 |
| 10 | 0.005 | 0.066 | 0.094 | 0.155 | 9.423 | 9.409 |
| 30 | 0.009 | 0.070 | 0.261 | 0.320 | 26.10 | 28.59 |
| 100 | 0.014 | 0.077 | 0.910 | 0.969 | 91.28 | 93.20 |
| 300 | 0.030 | 0.091 | 2.652 | 2.938 | 275.6 | 270.7 |

Table 3: Running time in seconds used by OPG-MC-DP algorithm.

## 6 Conclusion and Discussions

In this paper, we investigate two natural models of optimal perimeter guarding using heterogeneous robots, where one model (OPG$_{LR}$) limits the number of available robots and the second (OPG$_{MC}$) seeks to optimize the total cost of coverage.

These formulations have many potential applications. One application scenario we envision is the deployment of multiple agents or robots as "emergency responders" that are constrained to travel on the boundary. An optimal coverage solution will then translate to minimizing the maximum response time anywhere on the perimeter (the part that needs guarding). The scenario applies to OPG, OPG$_{LR}$, and OPG$_{MC}$.

Another application scenario is the monitoring of the perimeter using robots with different sensing capabilities. A simple heterogeneous sensing model here would be robots equipped with cameras with different resolutions, which may also be approximated as discs of different radii. The model makes sense provided that the region to be covered is much larger than the sensing range of individual robots and assuming that the boundary has relatively small curvature as compared to the inverse of the radius of the smallest sensing disc of the robots. For boundary with relatively small curvature, our solutions would apply well to the sensing model by using the diameter of the sensing disc as the 1D sensing range. As the region to be covered is large, covering the boundary will require much fewer sensors than covering the interior.

On the computational complexity side, we prove that both OPG$_{LR}$ and OPG$_{MC}$ are NP-hard, with OPG$_{LR}$ directly shown to be strongly NP-hard. This is in stark contrast to the homogeneous case, which admits highly efficient low polynomial time solutions [?]. The complexity study also establishes structural similarities between these problems and classical NP-hard problems including **3-Partition**, **Knapsack**, and **Subset Sum**.

On the algorithmic side, we provide methods for solving both OPG$_{LR}$ and OPG$_{MC}$ exactly. For OPG$_{LR}$, the algorithm runs in pseudo-polynomial time in practical settings with limited types of robots. In this case, the approach is shown to be computationally effective. For OPG$_{MC}$, a pseudo-polynomial time algorithm is derived for the general problem, which implies that OPG$_{MC}$ is weakly NP-hard. In practice, this allows us to solve large instances of OPG$_{MC}$. We further show that a polynomial time algorithm is possible for OPG$_{MC}$ when the types of robots are fixed.

With the study of OPG [**?**] for homogeneous and heterogeneous cases, some preliminary understanding has been obtained on how to approach complex 1D guarding problems. Nevertheless, the study so far is limited to *one-shot* settings where the perimeters do not change. In future research, we would like to explore the more challenging case where the perimeters evolve over time, which requires the solution to be dynamic as well. Given the results on the one-shot settings, we expect the dynamic setting to be generally intractable if global optimal solutions are desired, potentially calling for iterative and/or approximate solutions.

We recognize that our work does not readily apply to a visibility-based sensing model, which is also of interest. Currently, we are also exploring covering of the interior using range-based sensing. As with the OPG work, we want to push for optimal or near-optimal solutions when possible.