

An Efficient Parallel Implementations of Approximation Algorithms for Guarding 1.5D Terrains

Abstract. In the 1.5D Terrain Guarding Problem we are given an x -monotone polygonal line defined by k vertices and a set G of points from the terrain, i.e. guards, and a set N of points from the terrain which are to be seen (guarded) by guards. We deal with a weighted version of the guarding problem where guards G have weights and the goal is to find a minimum weight subset of G to cover all the points in N , and a version where points from N have demands, and the goal is to find the smallest subset from G such that every point in N is seen by the demanded number of guards. Both problems are NP-hard and have a factor 5 approximation ([3], [4]). We show that if $(1 + \epsilon)$ -approximate solver to the corresponding linear program is a computer, for any $\epsilon > 0$, an extra $1 + \epsilon$ factor will appear in the final approximation factor for both problems. We compare our parallel implementation based on GPU and CPU threads with GUROBI solver and conclude that our algorithm outperforms GUROBI solver on large and dense inputs typically by one order of magnitude.

Key words: 1.5D terrain guarding, linear programming, CUDA, approximation algorithm

Received: xx xx, 201x; accepted: yy yy, 201x; available online: zz zz, 201x

1. Introduction

A *terrain* T is an x -monotone polygonal chain with set of vertices i.e., a piecewise linear curve intersecting any vertical line in at most one point. The terrain polygon P_T determined by T is the closed region in the plane bounded from below by T . For two points p and q in P_T , we say that p *sees* q and write $p \sim q$, if the line segment connecting p and q is contained in P_T , (see *Figure 1*). This kind of guarding problem and its generalizations to 3-dimensions are motivated by optimal placement of antennas for communication networks [1]. The *1.5D-terrain guarding problem* is to select a smallest set of guards X from terrain T such that for every $p \in T$ there is a guard in X that sees p .

Previous work For the 1.5D terrain guarding problem it is known to be NP-hard [10]. The problem can be approximated within $1 + \epsilon$ for any $\epsilon > 0$ using a local search technique [7] but it is not clear how this approach can be extended to the weighted version and version with demands. In [3] and [4] we presented a first constant factor approximation algorithms based on LP rounding. Thus far, we are not aware of any implementations attempts because all the approaches prior to [3] and [4] are relatively complicated to implement.

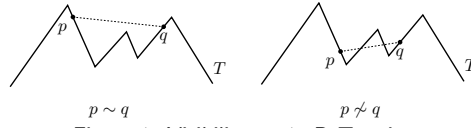


Figure 1: Visibility on 1.5D Terrains

Contribution In this paper we present an implementation of terrain guarding algorithms from our results [3] and [4] and show how approximately solving corresponding LP of terrain guarding problem can induce an error in approximation of these problems but gaining on efficiency with respect to the error.

1.1. Preliminaries

Let T be a 1.5D terrain and let $V(T)$ denote the vertices of T . The complexity of 1.5D terrain is the number of terrain vertices $|V(T)|$. We write $p < q$ if p lies to the left of q and symmetrically, we write $p > q$ if p lies to the right of q . Also, let $\mathcal{V}(q) := \{p : p \in T, q \sim p\}$ denote a *visibility region* of a point q . The left and right visibility region of a point q is defined as $\mathcal{V}_L(q) = \{p : p \in \mathcal{V}(q), p < q\}$ and $\mathcal{V}_R(q) = \{p : p \in \mathcal{V}(q), p > q\}$ respectively.

Throughout this paper, we consider the discrete version of the problem, i.e. we are given a finite set of possible guards $G \subset T$ and a finite set of points $N \subset T$ and our goal is to select a minimum set of guards $X \subseteq G$ to guard N . It can be shown that the 1.5D terrain guarding problem instance can be reduced to discrete one by incurring an extra $O(n^2)$ points to the terrain (see [1]). In this work, the implementations of the 2 variants of the discrete 1.5D terrain guarding problem are presented:

- In the *weighted 1.5D terrain guarding problem* we are given a 1.5D terrain instance T with a set of points $N \subset T$ and a set of guards $G \subset T$ with associated weights $w: G \rightarrow \mathbb{R}_+$. The goal is to find a minimum weight set of guards $X \subseteq G$ to guard all the points in N .
- In the *1.5D terrain guarding problem with demands* we are given a 1.5D terrain instance T with a set of guards $G \subset T$ and a set of points $N \subset T$ with an associated demand function $d: N \rightarrow \mathbb{Z}_+$. The goal is to find a minimum guard set $X \subseteq G$ such that every point $p \in N$ is guarded by at least $d_p := d(p)$ different guards.

Special classes of linear programs. Let A be a non-negative $m \times n$ real matrix, and b, c and u are vectors consisting of non-negative real values.

- A *packing-covering* problem is a primal-dual linear program (LP) pair:

$$\max\{c^T x : Ax \leq b, x \geq 0\} = \min\{b^T y : A^T y \geq c, y \geq 0\}, \quad (1)$$

- A *multi-cover problem with boxed constraints* (as a special case of mixed packing-covering problem) is a primal-dual LP pair:

$$\min_{x \in \mathbb{R}^n} \{c^T x : Ax \geq b, x \leq u, x \geq 0\} = \max_{y \in \mathbb{R}^m, u \in \mathbb{R}^n} \{b^T y - u^T z : A^T y - z \leq c, y \geq 0, z \geq 0\} \quad (2)$$

The strong duality property implies the equality in (1) and (2) (for more information see [2] and references therein). We are interested in finding efficiently an approximate solution of problem (1) and (2) respectively (note that these linear programs can be solved in polynomial time due to [9]).

Definition 1. *Let $\epsilon > 0$ be some arbitrary constant. An $(1 + \epsilon)$ -approximation for (1) is a primal-dual feasible pair (x, y) such that $b^T y \leq (1 + \epsilon)c^T x$.*

Definition 2. *Let $\epsilon > 0$ be some arbitrary constant. An $(1 + \epsilon)$ -approximation for (2) is a primal-dual feasible pair $(x, (y, z))$ such that $c^T x \leq (1 + \epsilon)(b^T y - u^T z)$.*

Programming environment GUROBI is the state-of-the-art solver for mixed integer linear and mixed integer quadratic programming. The tool employs several methods in parallel to find efficiently solution of the corresponding optimization problem (more details in [8]). CUDA (*Compute Unified Device Architecture*) represents a computing engine developed for NVIDIA graphics processing units (GPUs) supporting both graphics and general computing. The CUDA API enables us to implement parallel computation over the large number of threads running on GPU cores. Extensive description of CUDA architecture and CUDA API usage can be found in textbook [13].

2. Implementation of the 1.5D Terrain Guarding algorithms

In this section we present a implementation model of guarding 1.5D terrains. The generic algorithm for the 1.5D terrain guarding problem can be expressed as *Algorithm 1*.

```

1: procedure TERRAIN-GUARDING( $T, G, N, w, d$ )
2:    $A \leftarrow$  CALCULATE-VISIBILITY( $T, G, N$ )
3:    $(A, w, d, u, m, n) \leftarrow$  LP-FORM( $A, w, d$ )
4:    $(x^*, y^*) \leftarrow$  LP-SOLVE( $A, w, d, u, m, n, \epsilon$ )
5:    $X_0 \leftarrow$  FIND-GUARD-POINTS( $G \cap N, x^*, \alpha$ )
6:    $(G', N', w, d') \leftarrow$  REDUCE-INSTANCE( $X_0, d$ )
7:    $(d'_L, d'_R) \leftarrow$  DECOMPOSE( $G', N', d', x^*$ )
8:    $X_L \leftarrow$  LEFT-GUARDS( $T, G', N', w, d'_L$ )
9:    $X_R \leftarrow$  RIGHT-GUARDS( $T, G', N', w, d'_R$ )
10:   $X \leftarrow X_0 \cup X_L \cup X_R$ 
11:  return  $X$ 

```

Algorithm 1: Generic algorithm for guarding 1.5D terrains

The CALCULATE-VISIBILITY procedure receives as an input a 1.5D terrain T and a set of guards G and a set of vertices N and calculates a visibility matrix for the pairs G, N of terrain T , i.e. for every point $p \in N$ and every guard $g \in G$ it calculates the relation $g \sim p$ by checking whether all terrain vertices v between g and p lie strictly below this segment. This step can be done by checking if the triangle area formed by these 3 points is non-negative, as seen in the *Figure 2*. The overall time of this step is $O(mnk)$ where $k = |V(T)|$ (computing determinant takes $O(1)$ time). Due to our assumption that there are polynomially many guards and points,

$$P(\Delta_{gvp}) = \frac{1}{2} \begin{vmatrix} g_x & g_y & 1 \\ v_x & v_y & 1 \\ p_x & p_y & 1 \end{vmatrix} > 0$$

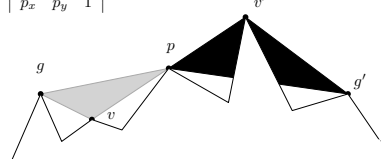
$$P(\Delta_{pv'g'}) = \frac{1}{2} \begin{vmatrix} p_x & p_y & 1 \\ v'_x & v'_y & 1 \\ g'_x & g'_y & 1 \end{vmatrix} < 0$$


Figure 2: Determinant method to compute visibility on the terrain.

we present the visibility relation as an binary matrix $A \in \mathbb{R}^{m \times n}$ for the purpose of fast visibility queries ($A[p][g] = 1 \Leftrightarrow p \sim g$).

The procedure LP-FORM defines an integer linear program relaxation (i.e, a linear program) of the corresponding terrain guarding problem:

$$\min_{x \in \mathbb{R}^n} \{w^T x : Ax \leq d, x \leq u, x \geq 0\} \quad (3)$$

where A is the computed visibility matrix, w are weights and d demands. Vector u is an upper-bound on the number of copies of any guard we are able to use.

The fundamental part of the algorithm is LP-SOLVE procedure which solves (3) with respect to error parameter $\epsilon \geq 0$. If $\epsilon = 0$ then the procedure finds an optimal solution using the state-of-the art LP solvers, else, it returns $(1 + \epsilon)$ -approximation using the approximate solvers presented in [6] and [5] which running-time explicitly depends on the size of problem and the $1/\epsilon$ parameter.

In the FIND-GUARD-POINTS procedure, algorithm chooses the guards that are also points which have large fractional values with respect to x^* and parameter $\alpha > 0$ reducing the problem instance and achieving the condition needed by combinatorial algorithms developed in [3] and [4].

The LEFT-GUARDING and RIGHT-GUARDING procedures are implementations of the combinatorial algorithms for finding an optimal set of left and right guards. It's implementation varies depending on the problem instance (weighted guarding or guarding with demands).

2.1. Algorithm implementation of Weighted 1.5D Terrain Guarding

The problem instance is represented by (3) where $d_p = 1, \forall p \in N$ and $u_g = 1, \forall g \in G$. The constraint $0 \leq x \leq u$ is then $x \geq 0$ because every point needs no more than one guard. The problem (3) is than a dual of packing-covering problem (1).

Solving covering-packing problems The LP-SOLVE for $\epsilon > 0$ gives an $(1 + \epsilon)$ -approximation but with $1/\epsilon^2$ dependency in the running time [6]. The complete CUDA algorithm for the packing-covering approximate solver is given in [11].

Let us consider, further on, the case when *Algorithm 2* returned an $(1 + \epsilon)$ -approximation of the (3) (treated as a covering problem, and appropriate dual as packing problem), namely, (x', y') .

```

1: procedure PC-APX( $A, b, c, \epsilon$ )
   INPUT:  $A \in \mathbb{R}_+^{m \times n}, b \in \mathbb{R}_+^m, c \in \mathbb{R}_+^n, \epsilon > 0$ 
   OUTPUT:  $(x^*, y^*)$  such that  $b^T y^* \leq (1 + \epsilon) c^T x^*$ 
2:    $\epsilon' \leftarrow 1 - 1/\sqrt{1 + \epsilon}, \delta \leftarrow (1 + \epsilon')((1 + \epsilon')m)^{-1/\epsilon'}$ 
3:    $x_0(j) \leftarrow 0, j = 1, \dots, n$ 
4:    $y_0(i) \leftarrow \delta/b(i), i = 1, \dots, m$ 
5:    $L_y(j) \leftarrow \sum_i A(i, j)y(i)/c(j)$ 
6:    $P(0) \leftarrow 0, D(0) \leftarrow m \cdot \delta$ 
7:   while  $D(k) < 1$  do
8:      $q \leftarrow \operatorname{argmin}_j L_{y_{k-1}}(j)$ 
9:      $p \leftarrow \operatorname{argmin}_i b(i)/A(i, q)$ 
10:     $x_k(q) \leftarrow x_{k-1}(q) + b(p)/A(p, q)$ 
11:     $y_k(i) \leftarrow y_{k-1}(i) \left(1 + \epsilon' \frac{b(p)/A(p, q)}{b(i)/A(i, q)}\right), i = 1, \dots, m$ 
12:     $P(k) \leftarrow P(k-1) + c(q)b(p)/A(p, q)$ 
13:     $D(k) \leftarrow D(k-1) + c(q)b(p)/A(p, q) \cdot \rho(y_{k-1})$ 
14:     $\rho \leftarrow \min_j L_{y_k}(j)$ 
15:     $x(j) \leftarrow x_t(j)/\log_{1+\epsilon'}((1 + \epsilon')/\delta), j = 1, 2, \dots, n$ 
16:     $y(i) \leftarrow y_t(i)/\rho, i = 1, 2, \dots, m$ 
17:   return  $(x, y)$ 

```

Algorithm 2: Approximation scheme for packing-covering problems from [6]

Weighted problem instance reduction FIND-GUARD-POINTS finds point-guards $X_0 = \{g: g \in G \cap N, x'_g \geq \alpha\}$ where $\alpha = 1/5$. Updated terrain guarding instance is $N' = N \setminus \{p: g \sim p, g \in X_0\}$ and $G' = G \setminus X_0$ obtaining the condition $G' \cap N' = \emptyset$.

Left and right guarding problems The DECOMPOSE procedure defines the partition of points N' into two sets N_L and N_R as

$$N_L = \left\{ p \in N \mid \sum_{g \in \mathcal{V}_L(p) \cap G'} x'_g \geq \frac{1}{2} \right\}, N_R = \left\{ p \in N \mid \sum_{g \in \mathcal{V}_R(p) \cap G'} x'_g \geq \frac{1}{2} \right\}. \quad (4)$$

```

1: procedure WEIGHTED-LEFT-GUARDING( $T, G, N, w$ )
2:   PROCESSING FROM THE LEFT:
3:    $X \leftarrow \emptyset, Y \leftarrow \emptyset$ 
4:    $w'(g) \leftarrow w(g), \forall g \in G$ 
5:   for  $p \in N$  processed from left to right do
6:     if  $\mathcal{V}_L(p) \cap X = \emptyset$  then
7:        $g_p \leftarrow \operatorname{argmin}\{w'(g): g \in \mathcal{V}_L(p)\}$ 
8:        $w'(g) \leftarrow w'(g) - w'(g_p), \forall g \in \mathcal{V}_L(p) \setminus \{g_p\}$ 
9:        $X \leftarrow X \cup \{g_p\}, Y \leftarrow Y \cup \{p\}$ 
10:  PRUNING STEP:
11:  for  $p \in Y$  processed from right to left do
12:    if  $(X \setminus \{g_p\}) \cap \mathcal{V}_L(p) \neq \emptyset$  then
13:       $X \leftarrow X \setminus \{g_p\}$ 
14:  return  $X$ 

```

Algorithm 3: Finding an optimal set of left guards

In terms of generic algorithm notation, we say $p \in N_L \Leftrightarrow d_{p,L} = 1$ and $p \in N_R \Leftrightarrow d_{p,R} = 1$. The left guarding problem can be solved in polynomial time optimally as shown in [3]. The simple procedure shown in *Algorithm 3* is a greedy algorithm that finds a optimal set of left guards X_L from G' that guard all the points in N_L (see [12, on croatian] for a complete proof). By symmetric formulation, the WEIGHTED-RIGHT-GUARDING procedure finds an optimal set of right guards X_R from G' that guards N_R . Both algorithms run in $O(mn)$ time.

Overall approximation of the Weighted 1.5D Terrain Guarding problem

Let (x^*, y^*) denote an optimal (fractional) solution of (3). The LP-SOLVE procedure returns (x', y') such that

$$\sum_{p \in N} y'_p \leq \sum_{p \in N} y_p^* = \sum_{g \in G} w_g x_g^* \leq \sum_{g \in G} w_g x'_g \leq (1 + \epsilon) \sum_{p \in N} y'_p \leq (1 + \epsilon) \sum_{p \in N} y_p^* \quad (5)$$

We argue that using $(1 + \epsilon)$ -approximation can produce an approximate solution not arbitrarily far from the optimal solution.

Theorem 1. *The Weighted 1.5D Terrain Guarding problem with m points and n guards can be approximated within $5(1 + \epsilon)$ factor in $O(mnk + \epsilon^{-2}n^2m \log n)$ time on a RAM machine, and in $O(mnk + \epsilon^{-2}mn \log n \log m)$ time on the PRAM machine where $\epsilon > 0$ is an error in $(1 + \epsilon)$ -approximation of the corresponding LP solution.*

Proof. The analysis tightly follows the analysis from [3] which is given for $\epsilon = 0$. The cost of rounded point-guards is $w(X_0) \leq \frac{1}{\alpha} \sum_{g \in X_0} w_g x'_g \leq \frac{1}{\alpha}(1 + \epsilon) \sum_{p \in N} y'_p$. Moreover, using the same analysis, it can be shown that $w(X_L) \leq 5/2 \sum_{g \in G'} w_g x'_g$, therefore, in overall, and using (5):

$$\begin{aligned} w(X_0) + w(X_L) + w(X_R) &\leq 5 \sum_{g \in X_0} w_g x'_g + 5 \sum_{g \in G'} w_g x'_g \leq 5 \left(\sum_{g \in X_0} w_g x'_g + \sum_{g \in G'} w_g x'_g \right) \\ &\leq 5(1 + \epsilon) \sum_{p \in N} y'_p \leq 5(1 + \epsilon) \sum_{p \in N} y_p^* \leq 5(1 + \epsilon) \text{OPT} \end{aligned}$$

where OPT is an optimal solution of the weighted problem. Running time of the algorithm takes $O(\epsilon^{-2}n^2m \log n)$ steps on the RAM machine due to [6] and CUDA algorithm takes $O(\epsilon^{-2}mn \log n \log m)$ time due to [11]. \square

3. Algorithm implementation of 1.5D Terrain Guarding problem with Demands

The problem instance is represented by (3) where $w_g = 1, u_g = 1, \forall g \in G$. Due to non-trivial demands, the condition $0 \leq x \leq u$ cannot be simplified and (3) is then a primal of multi-cover problem with boxed constraints (2).

Solving LPs of multi-cover problems with boxed constraints If the error parameter $\epsilon = 0$ the procedure LP-SOLVE uses GUROBI LP solver to solve optimally LP, otherwise, we use an approximate solver for (2) from [5].

Approximating multi-cover problems with boxed constraints Fleischer [5] proposed the approximation algorithm based on primal-dual updates described in *Algorithm 4*.

Theorem 2 ([5]). *Algorithm 4 in $O(\epsilon^{-2}n^2m \log(c^T u))$ time returns an $(1 + \epsilon)$ -approximation of (2).*

Let x' be a feasible solution from a $(1 + \epsilon)$ -approximation of (3) returned by *Algorithm 4*.

```

1: procedure MULTI-COVER-APX( $A, b, c, \epsilon$ )
   INPUT:  $A \in \mathbb{R}_+^{m \times n}, b \in \mathbb{Z}_+^m, c \in \mathbb{R}_+^n, u \in \mathbb{Z}_+^n, \epsilon > 0$ 
   OUTPUT:  $(x^*, (y^*, z^*))$  as  $(1 + \epsilon)$ -approximation of (2).
2:    $\delta \leftarrow (1 + \epsilon)((1 + \epsilon)c^T u)^{-1/\epsilon}$ 
3:    $x(j) \leftarrow u(j)\delta, j = 1, 2, \dots, n, x^* \leftarrow x, (y, z) = (0, 0)$ 
4:    $L_x(i) := \sum_j A(i, j)x(j)/b(i)$ 
5:    $\alpha^* \leftarrow \min_i L_x(i), p \leftarrow \arg \min_i L_x(i)$ 
6:   while  $c^T x < 1$  do
7:      $\alpha \leftarrow (1 + \epsilon)L_x(p)$ 
8:     while  $L_x(p) < \alpha$  and  $c^T x < 1$  do
9:        $Q(p) = \{1 \leq j \leq n: x(j) < u(j)\alpha\}$ 
10:       $\eta \leftarrow \min_{j \in Q(p)} \frac{c(j)}{A(p, j)} \min\{1, \frac{u(j)\alpha - x(j)}{\epsilon x(j)}\}$ 
11:       $y(p) \leftarrow y(p) + \eta$ 
12:       $x(j) \leftarrow x(j)(1 + \epsilon \frac{\eta A(p, j)}{c(j)}), j \in Q(p)$ 
13:       $z(j) \leftarrow z(j) + \eta A(p, j), j \notin Q(p)$ 
14:       $p \leftarrow \arg \min_i L_x(i)$ 
15:      if  $c^T x/\alpha < c^T x^*/\alpha^*$  then
16:         $x^* \leftarrow x, \alpha^* \leftarrow \alpha$ 
17:       $x^* \leftarrow x^*/\alpha^*, (y^*, z^*) \leftarrow \frac{\epsilon}{\ln(\frac{1+\epsilon}{\delta})}(y, z)$ 
18:   return  $(x^*, (y^*, z^*))$ 

```

Algorithm 4: An approximation scheme for multi-cover problems with boxed constraints from [5]

Reduction of the version with demands We update the problem instance with $X_0 = \{x_g: g \in G \cap N, x'_g \geq \alpha\}$ where $\alpha = \frac{2}{5} \frac{d_{\min}}{d_{\min} + 1}$ (d_{\min} is a minimum demand of points from N) such that $G = G \setminus X_0, G' = G' \setminus X_0, d_p = d_p - |\{g: g \sim p, g \in X_0\}|$ achieving the $G' \cap N' = \emptyset$ condition.

The left and right multi-guarding The DECOMPOSE procedure defines the portions of demand that should be met from left and right with respect to the fractional value x' :

$$d_{p,L} = \left\lceil \left(1 + \frac{1}{d_{\min}}\right) \left(\sum_{g \in \mathcal{V}'_L(p)} x'_g + \frac{1}{2}x'_p\right) \right\rceil, d_{p,R} = \left\lceil \left(1 + \frac{1}{d_{\min}}\right) \left(\sum_{g \in \mathcal{V}'_R(p)} x'_g + \frac{1}{2}x'_p\right) \right\rceil \quad (6)$$

There is a combinatorial algorithm given as *Algorithm 5* from [4] that can find a minimum set of left guards X_L such that for every point $p \in N'$ there are $d'_{p,L}$ different guards in X_L that see the point p . By symmetry, the version for the right guarding, namely, RIGHT-MULTI-GUARDING algorithm produces set of optimal right guards. Both algorithms run in $O(mn)$ time.

```

1: procedure LEFT-MULTI-GUARDING( $T, G, N, d_L$ )
2:    $X \leftarrow \emptyset$ 
3:   for  $p \in N$  processed from left to right do
4:     while  $|X \cap \mathcal{V}_L(p)| \leq d_{p,L}$  do
5:        $X \leftarrow X \cup L(p)$ 
6:   return  $X$ 

```

Algorithm 5: Finding optimal set of left guards where points have demands.

Overall approximation of the 1.5D Terrain guarding with demands Let $(x', (y', z'))$ be a primal-dual $(1 + \epsilon)$ -approximation returned by LP-SOLVE procedure

implemented as *Algorithm 4*. By the definition of $(1 + \epsilon)$ -approximation and the strong LP duality property we have the following condition:

$$\sum_{p \in N} d_p y_p^* - \sum_{g \in G} z_g^* = \sum_{g \in G} x_g^* \leq \sum_{g \in G} x'_g \leq (1 + \epsilon) \sum_{p \in N} (d_p y_p^* - \sum_{g \in G} z_g^*)$$

Theorem 3. *For the 1.5D Terrain Guarding problem with demands there is a $5/2(1 + \epsilon)(1 + 1/d_{\min})$ -approximation algorithm in $O(mnk + \epsilon^{-2}n^2m \log n)$ time on the RAM machine.*

Proof. Following the analysis from [4] we can construct a feasible solution for left and right multi-guarding problem (as an LP formulation) with respect to x' . The cost of the returned solution is

$$\begin{aligned} |X_0| + |X_L| + |X_R| &\leq \frac{1}{\alpha} \left(\sum_{g \in X_0} x'_g + \sum_{g \in G'} x'_g \right) \leq \frac{1}{\alpha} (1 + \epsilon) \left(\sum_{p \in N'} d_p y'_p - \sum_{g \in G'} z'_g \right) \\ &\leq \frac{5}{2} \left(1 + \frac{1}{d_{\min}} \right) (1 + \epsilon) \text{OPT} \end{aligned}$$

where OPT denotes the optimal solution of guarding problem with demands. \square

4. Experiments

In this section we test our terrain guarding approximation algorithms against GUROBI Integer Linear Programming solver for the 1.5D terrain guarding instances. Our implementation uses CUDA programming environment for *Algorithm 2* and POSIX threads to solve in parallel left and right guarding problems.

Platform All the measurements are taken on the quad-core Intel 2.8 MHz i5 processor with 8 Gb RAM coupled with GPU processing unit TESLA C2070 with 6 GB of DDR5 RAM having 448 massively threaded processing cores with 1.15 GHz clock rate that can run in parallel 30000 threads and high memory bandwidth of 144 GB/s.

Tests and Comparisons Testbeds are 1.5D terrains and every vertex of the terrain is guard and point, i.e. $G = N = V(T)$ with trivial weights and demands. The input varies on the size of terrains which are randomly generated.

In *Table 1* and *Table 2* are given experimental results for our testbeds. Number of terrain vertices is denoted as $|V(T)|$ and d represents a density of the corresponding visibility matrix. The value of optimal solution returned by GUROBI ILP solver is given as GRB-OPT with running time GRB-TIME. The value of approximation returned by our algorithm is given in TG-APX and the running time is given as TG-TIME. The approximation ratio is expressed as γ . The performance ratio of our algorithm and GUROBI solver is given in $\frac{\text{GRB-TIME}}{\text{TG-TIME}}$. Time measures are expressed in seconds.

$ V(T) $	d	GRB-OPT	TG-APX	γ	GRB-TIME	TG-TIME	$\frac{\text{GRB-TIME}}{\text{TG-TIME}}$
10	0.44	2	3	1,50	0,00	0,00	0,20
	0.52	2	6	3,00	0,00	0,00	0,20
	0.72	2	3	1,50	0,00	0,00	0,27
100	0.28	4	6	1,50	0,06	0,16	0,38
	0.58	2	2	1,00	0,01	0,04	0,24
	0.76	1	2	2,00	0,01	0,04	0,27
1000	0.20	13	22	1,69	1,58	4,82	0,32
	0.52	1	2	2,00	0,29	0,44	0,65
	0.63	1	2	2,00	0,43	0,43	1,00
2000	0.31	2	2	1,00	1,38	1,10	1,25
	0.55	1	2	2,00	1,83	1,03	1,78
	0.70	1	2	2,00	2,94	1,03	2,87
5000	0.18	2	2	1,00	9,35	5,01	1,86
	0.55	1	2	2,00	93,98	4,76	19,75
	0.72	1	2	2,00	107,96	4,75	22,71
8000	0.19	85	148	1,740	98,00	902,662	0,11
	0.54	1	2	2,00	245,56	11,609	21,15
	0.65	1	2	2,00	284,10	11,572	24,55

Table 1: GUROBI solver vs 5.5-approximation ($\epsilon = 0.1$).

$ V(T) $	d	GRB-OPT	TG-APX	γ	GRB-TIME	TG-TIME	$\frac{\text{GRB-TIME}}{\text{TG-TIME}}$
10	0.44	2	3	1,50	0,00	0,00	0,91
	0.52	2	6	3,00	0,00	0,00	1,00
	0.72	2	3	1,50	0,00	0,00	1,00
100	0.28	4	6	1,50	0,06	0,04	1,43
	0.58	2	2	1,00	0,01	0,01	0,79
	0.76	1	2	2,00	0,01	0,01	0,71
1000	0.20	13	21	1,62	1,58	1,45	1,09
	0.52	1	2	2,00	0,29	0,16	1,80
	0.63	1	2	2,00	0,43	0,16	2,68
2000	0.31	2	2	1,00	1,38	0,37	3,75
	0.55	1	2	2,00	1,83	0,37	4,89
	0.70	1	2	2,00	2,94	0,57	5,20
5000	0.18	2	2	1,00	9,35	1,70	5,51
	0.55	1	2	2,00	93,98	1,63	57,55
	0.72	1	2	2,00	107,96	1,62	66,55
8000	0.19	85	148	1,740	98,00	257,94	0,38
	0.54	1	2	2,00	245,56	3,95	62,19
	0.65	1	2	2,00	284,10	3,93	72,28

Table 2: GUROBI solver vs 6-approximation ($\epsilon = 0.2$).

Based on the results, we can see that our algorithm outperforms GUROBI ILP solver for larger and denser inputs. The reason for that is that the current implementation of algorithm is developed for dense matrices and the overhead for initialization of GPU computation (mapping threads, copying data to device memory etc.) is evident on lower inputs. Moreover, for larger inputs, CUDA achieves better performance due to the large number of threads working in parallel. It is also worth noting, that the number of iteration of *Algorithm 2* is much smaller for greater ϵ . The approximation ratio achieved in these random examples isn't tight as in the analysis.

5. Conclusion

We presented an implementation of state-of-the art approximation algorithms for 2 variants of 1.5D terrain guarding problem in multi-threaded parallel environment

using CUDA and POSIX threads. Using $(1 + \epsilon)$ -approximation of LP solvers we induced an error in overall approximation but gained explicit dependency of our approximation algorithms in terms of $1/\epsilon^2$. We tested our implementation with GUROBI integer linear programming solver and conclude that our algorithm well behaves on large and dense inputs depending on the choices of ϵ .

Future work. Rewriting CUDA programs for sparse matrices would also benefit the performance of our algorithm applied to terrain inputs with sparse visibility matrices. We believe that our implementation can be used in heuristics for solving guarding problem on terrain in 3D.

References

- [1] B. Ben-Moshe, M. J. Katz, and J. S. B. Mitchell. A Constant-Factor Approximation Algorithm for Optimal 1.5D Terrain Guarding. *SIAM J. Comput.*, 36:1631–1647, 2007.
- [2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.
- [3] K. Elbassioni, E. Krohn, D. Matijević, J. Mestre, and D. Ševerdija. Improved Approximations for Guarding 1.5-Dimensional Terrains. *Algorithmica*, 60(2):451–463, 2011.
- [4] K. Elbassioni, D. Matijević, and D. Ševerdija. Guarding 1.5D terrains with demands. *International Journal of Computer Mathematics*, 89(16):2143–2151, 2012.
- [5] L. Fleischer. A fast approximation scheme for fractional covering problems with variable upper bounds. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '04, pages 1001–1010, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.
- [6] Naveen Garg and Jochen Könemann. Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems. *SIAM Journal on Computing*, 37(2):630, 2007.
- [7] M. Gibson, G. Kanade, E. Krohn, and K. Varadarajan. An Approximation Scheme for Terrain Guarding. In *Proceedings of the 12th International Workshop and 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX '09 / RANDOM '09, pages 140–148. Springer-Verlag, 2009.
- [8] Inc. Gurobi Optimization. Gurobi optimizer reference manual version 5.6. Houston, Texas: Gurobi Optimization,, 2014.
- [9] L.G. Khachiyan. A polynomial time algorithm for linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979.
- [10] J. King and E. Krohn. Terrain guarding is NP-hard. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1580–1593. SIAM, 2010.
- [11] G. Martinović, D. Matijević, and D. Ševerdija. CPU-GPU Implementation of a Fast Approximation Scheme for Covering and Packing Problems. submitted to *Journal of Parallel and Distributed Computing*, 2014, 2013.
- [12] D. Ševerdija. *Improved approximation algorithms for guarding 1.5D terrains*. PhD thesis, Faculty of Electrical Engineering, University J. J. Strossmayer of Osijek, 2013.
- [13] N. Wilt. *The CUDA Handbook: A Comprehensive Guide to GPU Programming*. Upper Saddle River, NJ : Addison-Wesley, 2013.