Croatian Operational Research Review
CRORR **6**(2015), 79–89

Efficient parallel implementations of approximation algorithms for guarding 1.5D terrains

Goran Martinović[1], **Domagoj Matijević**[2] and **Domagoj Ševerdija**[2,*]

[1] *Faculty of Electrical Engineering, Josip Juraj Strossmayer University of Osijek*
Kneza Trpimira 2B, 31000 Osijek, Croatia
E-mail: ⟨goran.martinovic@etfos.hr⟩

[2] *Department of Mathematics, Josip Juraj Strossmayer University of Osijek*
Trg Ljudevita Gaja 6, 31000 Osijek, Croatia
E-mail: ⟨{domagoj, dseverdi}@mathos.hr⟩

Abstract. In the 1.5D terrain guarding problem, an x-monotone polygonal line is defined by k vertices and a G set of terrain points, i.e. guards, and a N set of terrain points which guards are to observe (guard). This involves a weighted version of the guarding problem where guards G have weights. The goal is to determine a minimum weight subset of G to cover all the points in N, including a version where points from N have demands. Furthermore, another goal is to determine the smallest subset of G, such that every point in N is observed by the required number of guards. Both problems are NP-hard and have a factor 5 approximation [3, 4]. This paper will show that if the $(1+\epsilon)$-approximate solver for 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. A comparison will be carried out the parallel implementation based on GPU and CPU threads with the Gurobi solver, leading to the conclusion that the respective algorithm outperforms the 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: October 3, 2014; accepted: February 18, 2015; available online: March 30, 2015

DOI:10.17535/crorr.2015.0007

1. Introduction

A *terrain* T is an x-monotone polygonal chain with a set of vertices, i.e., a piecewise linear curve intersecting any vertical line at no more than one point. The terrain polygon P_T determined by T is a closed region in a plane bounded from below by T. The two points p and q in P_T, are described by saying that p *sees* q and $p \sim q$, 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 the optimal placement of antennas for communication networks [1]. Solving the 1.5D-*terrain guarding problem* means selecting the smallest set of guards X from a terrain T such that for every $p \in T$ there is a guard in X that sees p.

*Corresponding author.

The 1.5D terrain guarding problem is acknowledged to be NP-hard [10]. The problem can be approximated within $1+\epsilon$ for any $\epsilon > 0$ using a local search technique [7], but how to extend this approach to the weighted the version and version with demands remains unclear. A first constant factor approximation algorithms based on LP rounding is presented in [3] and [4]. Thus far, there is no knowledge of any attempts at implementations, because all the approaches prior to [3] and [4] are relatively complicated to implement.

This paper presents the implementation of terrain guarding algorithms from the results [3] and [4] and show how approximately solving corresponding LP of the terrain guarding problem in an approximate manner can induce an approximation error in these problems, but also increase efficiency in terms of error.

1.1. Preliminaries

Let T be a $1.5D$ terrain and let $V(T)$ denote the vertices of the T. The complexity of 1.5D terrain is the number of terrain vertices $|V(T)|$. Hence, $p < q$ if p lies to the left of q and in a symmetrical sense, it is true that $p > q$ if p lies to the right of q. Furthermore, $\mathcal{V}(q) := \{p : p \in T, q \sim p\}$ denotes a *visibility region* of a point q. The left and right of the visibility region related to 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.

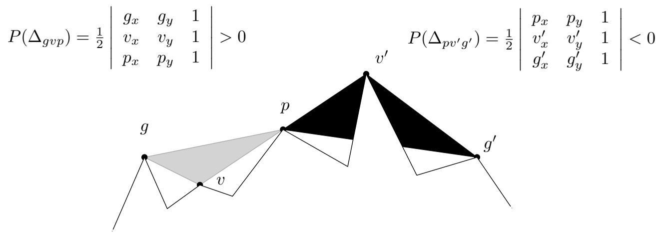

Figure 1: *Visibility on 1.5D Terrains*

Throughout this paper, the discrete version of the problem is considered, i.e. finite set of possible guards $G \subset T$ and a finite set of points $N \subset T$ are given, with the goal of selecting a minimum set of guards $X \subseteq G$ to guard N. It can be shown that the $1.5D$ terrain guarding problem can be reduced to discrete one by assigning extra $O(n^2)$ points to the terrain (see [1]). In this work, implementing 2 variations of the discrete 1.5D terrain guarding problem is presented as follows:

- The *weighted 1.5D terrain guarding problem* refers to 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 $w\colon G \to \mathbb{R}_+$. The goal is to find a minimum weight set of guards $X \subseteq G$ to guard all the points in N.

- The *1.5D terrain guarding problem with demands* refers to 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 $d\colon N \to \mathbb{Z}_+$. 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.

Let A be a non-negative $m \times n$ real matrix, and b, c and u the vectors consisting of non-negative real values. The following special classes of linear programs are considered in this paper:

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

$$\max\{c^T x \colon Ax \leq b, x \geq 0\} = \min\{b^T y \colon A^T y \geq c, y \geq 0\}, \tag{1}$$

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

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

The strong duality property implies equality in (1) and (2) (for more information see [2] and references therein). The interest is to find efficiently an approximate solution to the 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 $c^T x \leq (1+\epsilon)(b^T y - u^T z)$.*

1.2. Programming environment

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

2. Implementation of the 1.5D terrain guarding algorithms

This section presents an implementation model of guarding 1.5D terrains. The generic algorithm for the 1.5D terrain guarding problem can be expressed as an algorithm shown in *Figure 2*.

The Calculate-Visibility procedure receives as input a 1.5D terrain T, a set of guards G and a set of vertices N, and subsequently 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 triangular area formed by these 3 points is non-negative, as is shown in the *Figure 3*. The overall time of this step is $O(mnk)$ where $k = |V(T)|$ (computing the determinant takes $O(1)$ time). Due to the assumption that there are polynomially many guards

procedure Terrain-Guarding(T, G, N, w, d)
$A \leftarrow$ Calculate-Visibilty(T, G, N)
$(A, w, d, u, m, n) \leftarrow$ LP-Form(A, w, d)
$(x^*, y^*) \leftarrow$ LP-Solve($A, w, d, u, m, n, \epsilon$)
$X_0 \leftarrow$ Find-Guard-Points($G \cap N, x^*, \alpha$)
$(G', N', w, d') \leftarrow$ Reduce-Instance(X_0, d)
$(d'_L, d'_R) \leftarrow$ Decompose(G', N', d', x^*)
$X_L \leftarrow$ Left-Guards(T, G', N', w, d'_L)
$X_R \leftarrow$ Right-Guards(T, G', N', w, d'_R)
$X \leftarrow X_0 \cup X_L \cup X_R$
return X

Figure 2: *Generic algorithm for guarding 1.5D terrains*

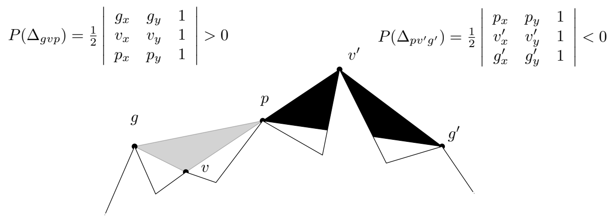

Figure 3: *Determinant method for computing visibility on the terrain.*

and points, the visibility relation is presented as a 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 LP-Form procedure defines an integer linear program relaxation (i.e, a linear program) for the corresponding terrain guarding problem:

$$\min_{x \in \mathbb{R}^n} \{w^T x \colon Ax \le d, x \le u, x \ge 0\} \tag{3}$$

where A is the computed visibility matrix, w are weights and d demands. Vector u is the upper-bound number of copies of any guard that can be seen.

The fundamental part of the algorithm is the LP-Solve procedure, which solves (3) with respect to the error parameter $\epsilon \ge 0$. If $\epsilon = 0$ then the procedure determines 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] where the running-time explicitly depends on the size of problem and the $1/\epsilon$ parameter.

In the Find-Guard-Points procedure, the algorithm chooses the guards acting also as points and which have large fractional values with respect to x^* and the parameter $\alpha > 0$, thus reducing the problem instance and achieving the condition required by the 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. Implementation of weighted 1.5D terrain guarding through algorithms

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 $0 \le x \le u$ results in $x \ge 0$ because every point needs no more than one guard. The problem (3) is then a dual of packing-covering problem (1).

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

---

procedure PC-APX(A, b, c, ϵ)
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 $b^T y^* \le (1+\epsilon)c^T x^*$
$\epsilon' \leftarrow 1 - 1/\sqrt{1+\epsilon}, \quad \delta \leftarrow (1+\epsilon')((1+\epsilon')m)^{-1/\epsilon'}$
$x_0(j) \leftarrow 0, \quad j = 1, \dots, n$
$y_0(i) \leftarrow \delta/b(i), \quad i = 1, \dots, m$
$L_y(j) \leftarrow \sum_i A(i,j)y(i)/c(j)$
$P(0) \leftarrow 0, \quad D(0) \leftarrow m \cdot \delta$ ▷ primal-dual values
while $D(k) < 1$ **do**
  $q \leftarrow \text{argmin}_j \; L_{y_{k-1}}(j)$
  $p \leftarrow \text{argmin}_i \; b(i)/A(i,q)$
  $x_k(q) \leftarrow x_{k-1}(q) + b(p)/A(p,q)$
  $y_k(i) \leftarrow y_{k-1}(i)\left(1 + \epsilon' \frac{b(p)/A(p,q)}{b(i)/A(i,q)}\right), \quad i = 1, \dots, m$
  $P(k) \leftarrow P(k-1) + c(q)b(p)/A(p,q)$
  $D(k) \leftarrow D(k-1) + c(q)b(p)/A(p,q) \cdot \rho(y_{k-1})$
$\rho \leftarrow \min_j L_{y_k}(j)$
$x(j) \leftarrow x_t(j)/\log_{1+\epsilon'}((1+\epsilon')/\delta), \quad j = 1, 2, \dots, n$
$y(i) \leftarrow y_t(i)/\rho, \quad i = 1, 2, \dots, m$
return (x, y)

Figure 4: *The approximation scheme for packing-covering problems from [6]*

Suppose, further on, the case when the algorithm in *Figure 4* returns a $(1+\epsilon)$-approximation of (3) (treated as a covering problem, and appropriate dual packing problem), namely, (x', y').

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

The DECOMPOSE procedure defines the partition of points N' as two sets N_L and N_R as

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

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 *Figure 5* 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 a 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.

```
procedure WEIGHTED-LEFT-GUARDING(T, G, N, w)
    PROCESSING FROM THE LEFT:
    X ← ∅, Y ← ∅
    w′(g) ← w(g),   ∀g ∈ G
    for p ∈ N processed from left to right do
        if V_L(p) ∩ X = ∅ then
            g_p ← arg min{w′(g): g ∈ V_L(p)}
            w′(g) ← w′(g) − w′(g_p), ∀g ∈ V_L(p) \ {g_p}
            X ← X ∪ {g_p},   Y ← Y ∪ {p}
    PRUNING STEP:
    for p ∈ Y processed from right to left do
        if (X \ {g_p}) ∩ V_L(p) ≠ ∅ then
            X ← X \ {g_p}
    return X
```

Figure 5: *Finding an optimal set of left guards*

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

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

The argument is put forward that using the $(1+\epsilon)$-approximation can produce an approximate solution that arbitrarily is not far from the optimal solution.

Theorem 1. *The weighted 1.5D terrain guarding problem with m points and n guards can be approximated by the $5(1+\epsilon)$ 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 the $(1+\epsilon)$-approximation of the corresponding LP solution.*

Proof. The analysis strictly follows that of [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, overall and using (5):

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

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

□

2.2. Implementation of the 1.5D terrain guarding problem with demands through algorithms

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 the multi-cover problem with boxed constraints (2).

If the error parameter $\epsilon = 0$, then the procedure LP-SOLVE uses the GUROBI LP solver to solve optimally LP, otherwise, an approximate solver for (2) from [5] is used.

Fleischer [5] proposed the approximation algorithm based on primal-dual updates described in *Figure 6*.

---

procedure MULTI-COVER-APX(A, b, c, ϵ)
  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).
  $\delta \leftarrow (1+\epsilon)((1+\epsilon)c^T u)^{-1/\epsilon}$
  $x(j) \leftarrow u(j)\delta, \quad j = 1, 2, \ldots, n,\ x^* \leftarrow x, \quad (y, z) = (\mathbf{0}, \mathbf{0})$
  $L_x(i) := \sum_j A(i,j)x(j)/b(i)$
  $\alpha^* \leftarrow \min_i L_x(i), \quad p \leftarrow \arg\min_i L_x(i)$
  while $c^T x < 1$ **do**
    $\alpha \leftarrow (1+\epsilon)L_x(p)$
    while $L_x(p) < \alpha$ and $c^T x < 1$ **do**
      $Q(p) = \{1 \le j \le n : x(j) < u(j)\alpha\}$
      $\eta \leftarrow \min_{j\in Q(p)} \frac{c(j)}{A(p,j)} \min\{1, \frac{u(j)\alpha - x(j)}{\epsilon x(j)}\}$
      $y(p) \leftarrow y(p) + \eta$
      $x(j) \leftarrow x(j)(1 + \epsilon\frac{\eta A(p,j)}{c(j)}), \quad j \in Q(p)$
      $z(j) \leftarrow z(j) + \eta A(p,j), \quad j \notin Q(p)$
      $p \leftarrow \arg\min_i L_x(i)$
    if $c^T x/\alpha < c^T x^*/\alpha^*$ **then**
      $x^* \leftarrow x, \alpha^* \leftarrow \alpha$
  $x^* \leftarrow x^*/\alpha^*, (y^*, z^*) \leftarrow \frac{\epsilon}{\ln(\frac{1+\epsilon}{\delta})}(y, z)$
  return $(x^*, (y^*, z^*))$

Figure 6: *The approximation scheme for multi-cover problems with boxed constraints from [5]*

Theorem 2 ([5]). *Algorithm given in* Figure 6 *in* $O(\epsilon^{-2}n^2 m \log(c^T u))$ *time returns an* $(1+\epsilon)$*-approximation of* (2).

Let x' be a feasible solution from a $(1+\epsilon)$-approximation of (3) returned by *Algorithm 6*.

An update is applied to the problem instance with $X_0 = \{x_g : g \in G \cap N, x'_g \ge \alpha\}$ 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 DECOMPOSE procedure defines the portions of demand that should be met from left and right with respect to the fractional value x':

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

A combinatorial algorithm is given in *Figure 7* from [4] that can be used to 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 guard, namely, RIGHT-MULTI-GUARDING algorithm produces a set of optimal right guards. Both algorithms run in $O(mn)$ time.

```
procedure LEFT-MULTI-GUARDING(T, G, N, d_L)
    X ← ∅
    for p ∈ N processed from left to right do
        while |X ∩ V_L(p)| ≤ d_{p,L} do
            X ← X ∪ L(p)
    return X
```

Figure 7: *Finding an optimal set of left guards where points have demands.*

Let $(x', (y', z'))$ be a primal-dual $(1+\epsilon)$-approximation returned by the LP-SOLVE procedure implemented in *Figure 6*. By definition of the $(1+\epsilon)$-approximation and the strong LP duality property, the following condition is achieved:

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

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* $O(mnk+\epsilon^{-2}n^2 m \log n)$ *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

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

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

□

3. Experiments

This section tests the terrain guarding approximation algorithms against the GUROBI integer Linear Programming solver for 1.5D terrain guarding instances. The implementation presented here uses the CUDA programming environment for the algorithm given in *Figure 4* and POSIX threads to solve the left and right guarding problems in parallel.

3.1. Platform

All the measurements were carried out on the quad-core Intel 2.8 MHz i5 processor with 8 Gb RAM coupled with a GPU processing unit type TESLA C2070 and 6 GB of DDR5 RAM having 448 massively threaded processing cores at a clock rate of 1.15 GHz that runs 30000 threads in parallel, and a high memory bandwidth of 144 GB/s.

3.2. Tests and comparisons

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

Table 1 and *Table 2* show the obtained experimental results for the testbeds. The number of terrain vertices is denoted as $|V(T)|$ and d represents a density of the corresponding visibility matrix. The optimal solution value returned by the GUROBI ILP solver is given as GRB-OPT with a running time of GRB-TIME. The approximation value returned by the 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 the algorithm and the GUROBI solver is given in $\frac{\text{GRB-TIME}}{\text{TG-TIME}}$. Time measurements 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 above results, it is evident that the algorithm outperforms the Gurobi ILP solver for larger and denser inputs. The reason for this is that the current algorithm implementation has been developed for dense matrices and the overhead for initialization of GPU computation (mapping threads, copying data to device memory etc.) as 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 algorithm iterations given in *Figure 4* is much smaller for a greater ϵ. The approximation ratio achieved in these examples is not as strict as in the analysis.

4. Conclusion

This paper has presented implementation of state-of-the-art approximation algorithms for 2 variants of the 1.5D terrain guarding problem in a multi-threaded parallel environment using CUDA and POSIX threads. In applying the $(1+\epsilon)$-approximation from LP solvers, an error in overall approximation was induced but an explicit dependency of the approximation algorithms in terms of $1/\epsilon^2$ was obtained. The implementation was tested with the Gurobi integer linear programming solver and has led to the conclusion that the algorithm behaves well on large and dense inputs depending on the choices of ϵ.

Future work on the topic should focus on rewriting CUDA programs for sparse matrices, which would also improve the performance of the algorithms applied to terrain inputs with sparse visibility matrices. The implementation can be used in heuristics for solving guarding problems on 3D terrains.

References

[1] Ben-moshe, B., Katz, M. J., and Mitchell, J. S. B. (2007). A constant factor approximation algorithm for optimal 1.5D terrain guarding. SIAM Journal on Computing, 36(6), 1631–1647. doi:10.1137/s0097539704446384.

[2] Boyd, S., and Vandenberghe, L. (2004). Convex Optimization. Cambridge University Press.

[3] Elbassioni, K., Krohn, E., Matijević, D., Mestre, J., and Ševerdija, D. (2011). Improved approximations for guarding 1.5-dimensional terrains. Algorithmica (New York), 60(2), 451–463. doi:10.1007/s00453-009-9358-4.

[4] Elbassioni, K., Matijević, D., and Ševerdija, D. (2012). Guarding 1.5D terrains with demands. International Journal of Computer Mathematics, 89(16), 2143–2151. doi:10.1080/00207160.2012.707800.

[5] Fleischer, L. (2004). A fast approximation scheme for fractional covering problems with variable upper bounds. Symposium on Discrete Algorithms, 1001–1010.

[6] Garg, N., and Könemann, J. (2007). Faster and simpler algorithms for multicommodity flow and other fractional packing problems. SIAM Journal on Computing, 37(2), 630–652. doi:10.1137/s0097539704446232.

[7] Gibson, M., Kanade, G., Krohn, E., and Varadarajan, K. (2009). An approximation scheme for terrain guarding. Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5687 LNCS, 140–148. doi:10.1007/978-3-642-03685-9_11.

[8] Gurobi Optimization. (2014). Gurobi optimizer reference manual version 5.6. http://www.gurobi.com/documentation/5.6/reference-manual/ [Accessed on 20 October 2014].

[9] Khachian, L. G. (1979). A polynomial time algorithm for linear programming. Doklady Akademia Nauk SSR, 224, 1093–1096.

[10] King, J., and Krohn, E. (2011). Terrain guarding is NP-hard. SIAM Journal on Computing, 40(5), 1316–1339. doi:10.1137/100791506.

[11] Martinovic, G., Matijević, D. and Ševerdija, D. (2014). CPU-GPU implementation of a fast approximation scheme for covering and packing problems. Submitted to Journal of Parallel and Distirbuted Computing.

[12] Ševerdija, D., (2013). Improved approximation algorithms for guarding 1.5D terrains. PhD thesis, Faculty of Electrical Engineering, Josip Juraj Strossmayer University of Osijek.

[13] Wilt, N. (2013). The CUDA Handbook: A Comprehensive Guide to GPU Programming. Addison-Wesley.