Subquadratic Algorithms for the Weighted Maximin Facility Location Problem

(Extended Abstract)

Frank Follert Elmar Schömer Jürgen Sellen[*]

Abstract

Let $\mathcal{S}$ be a set of n points in the plane, and let each point p of $\mathcal{S}$ have a positive weight $w(p)$. We consider the problem of positioning a point $\mathbf{x}$ inside a compact region $\mathcal{R} \subseteq \mathbf{R}^2$ such that $\min\{\ w(p)^{-1} \cdot d(\mathbf{x}, p)\ ;\ p \in \mathcal{S}\ \}$ is maximized. Based on the parametric search paradigm, we give the first subquadratic algorithms for this problem, with running time $O(n \log^4 n)$.
Furthermore, we shall introduce the concept of 'exact approximation' as the bit model counterpart to parametric search. Exploiting ideas from exact computation, we show that the considered problem can be solved in time $O(L\mu(L) n \log n)$, where L denotes the maximal bit-size of input numbers, and $\mu(L)$ the complexity of multiplying two L-bit integers.

1 Introduction

The *(weighted) maximin facility location problem* is a classical problem of operations research, and has also found attention in the computational geometry community. The task is to position a point in the plane such that its distance to a set of (weighted) sites is maximized. As an example, the facility may be a source of pollution, and the input sites cities, weighted by their population.

This problem is unconstrained if the domain of possible locations is the entire plane. Hence one generally considers bounding regions which themselves may have non-constant complexity (in the literature, the bounding region is often defined to be the convex hull of the input points).

Let $||.||$ denote Euclidean norm and d Euclidean distance (for generalizations see 2.3). We formulate the maximin location problem as follows:

Problem Statement *Let $\mathcal{S}$ be a set of n points in the plane, and let each point p of $\mathcal{S}$ have a positive weight $w(p)$. Let $\mathcal{R}$ be either*

(a) the intersection of $\leq n$ discs, or
(b) a simple polygon with $\leq n$ edges.

Find a point $\mathbf{x} \in \mathcal{R}$ which maximizes

$$\min\{\ w(p)^{-1} \cdot d(\mathbf{x}, p)\ ;\ p \in \mathcal{S}\ \}.$$

Even for simple bounding regions, this problem is intrinsically different from its minimax counterpart. Obviously, there can be $O(n)$ distinct optimal locations. Also, the problem has an $\Omega(n \log n)$ lower bound in the algebraic decision tree model (by reduction to MAX-GAP on a line, see [13]). Our algorithms are a substantial step towards this lower bound.

1.1 Previous Work

The weighted maximin location problem has been extensively studied in the operations research literature, for points inside different bounding regions [7, 12] (see also the survey [15]).

The unweighted problem *(largest empty circle in the plane)* can be solved via construction of the nearest-point Voronoi diagram. Toussaint [17] describes an optimal $O(n \log n)$ solution for the case that the bounding region is the convex hull of the input points, and an $O(n^2 \log n)$ solution for case (b).

For weighted sites, the Voronoi diagram is known to have quadratic complexity in the worst case, and it can be constructed in optimal $O(n^2)$ time [4]. The optimal location is either a vertex of this diagram, or it lies on the boundary of the region $\mathcal{R}$. For simple bounding regions with constant complexity, an optimal location can thus be found in $O(n^2)$ time [8].

In this paper, we present the first subquadratic algorithms for this problem. In particular, we also improve on previous solutions for the unweighted problem, case (b).

[*]Universität des Saarlandes, Fachbereich 14, Informatik, Lehrstuhl Prof. G. Hotz, Postfach 151150, 66041 Saarbrücken, Germany. Email: {follert,schoemer,sellen}@cs.uni-sb.de, Fax: +49 681 302 4421. Frank Follert is supported by a Graduiertenkolleg fellowship from DFG, Germany

1.2 Using Parametric Search

Our algorithms are based on the parametric search paradigm as introduced in [11]. This is the obvious technique to apply, and its growing importance in geometric optimization is best reflected in some of its recent applications [3].

The basic step to apply parametric search is to reduce the problem to the *decision problem* (does there exist a feasible location ?) when the optimization parameter (the weighted distance to the sites) is fixed.

This decision problem corresponds to computing the common intersection of regions of allowed locations, and deciding if this intersection is empty. In our case, the 'forbidden regions' are discs around the sites, and we shall decide whether their union covers the bounding region $\mathcal{R}$. The union of a set of discs has only linear complexity [10], and can be computed in $O(n \log n)$ time via inversion. This duality transformation maps the planar circles to planes in 3-space, and reduces the decision problem in case (a) to constructing a convex polytope, and intersecting this polytope with a sphere. The whole process can easily be parallelized, and parametric search yields an $O(n \log^4 n)$ algorithm.

Case (b) requires some additional work. Here, it may happen that the set of allowed locations (i.e., intersected with the bounding region) has quadratic complexity. We shall avoid the computation of this set by a trick: to answer the decision problem, we need - in addition to the computation of the union of discs - only check if two contours with linear complexity intersect. This is solved by a sweep algorithm, and parametric search finally yields the same time bound as above:

Theorem 1 *The weighted maximin location problem (case (a) and (b)) can be solved by parametric search in time $O(n \log^4 n)$.*

Section 2 is devoted to these parametric search solutions. In subsection 2.3 we discuss related problems to which the presented technique can be extended.

1.3 Bit Complexity

Most geometric algorithms are developed within one of two distinct computational frameworks. In the *algebraic framework*, the complexity of an algorithm is measured by the number of algebraic operations on real-valued variables, assuming exact computations. The input size thus corresponds to the number n of input values. In the *bit framework*, the complexity is measured by the number of bitwise boolean operations on binary strings. The input generally consists of integers, and the parameter n is supplemented by an additional parameter L that describes the maximal bit-size of any input value.

Parametric search is an ingenious technique to design optimization algorithms in the algebraic model. However, there are several, well-known disadvantages: algorithms based on parametric search are rather complicated, with high constants hidden in the O–notation. Also, the technique is based on a computing model that is not available in practice. E.g., one of the standard assumptions is that only polynomials of constant degree are involved in comparisons, and that the roots of these polynomials can be found exactly in constant time.

In contrast, the basic decision schemes are readily implemented, and immediately provide approximation algorithms that guarantee a relative error of ε by adding just a factor of $\log \varepsilon$ to the running time.

Problems that parametric search is applied for are generally of 'bounded algebraic depth' [18]. Especially, the algebraic complexity of the solution (and intermediate calculations) does not depend on n. However, with this prerequisite and in a true bit model, exact computation and root bounds provide all the tools that are necessary to make ε-approximation algorithms 'exact'.

We shall demonstrate the practicability of 'exact approximation' for the considered problem. Using inversion and a tricky analysis, we shall deduce constants that lie in a range which make the approach suitable for implementation. The theoretical result is also of interest: the dependency on n meets the known lower bound (though this bound is obtained in an uncomparable model):

Theorem 2 *In the bit complexity model, the weighted maximin location problem (case (a)) can be solved in time $O(L\mu(L) n \log n)$.*

The corresponding bit complexity analysis is subject of section 3.

2 Parametric Search

Parametric search is an optimization technique which can be applied in situations where we seek a maximum parameter ρ^* satisfying certain conditions that are met by all $\rho \leq \rho^*$ but not by any $\rho > \rho^*$. The strategy of parametric search is to give efficient sequential and parallel algorithms for the corresponding decision problem: decide whether a given parameter ρ is equal to, smaller, or larger than the maximum value ρ^* (for brevity in this abstract, we shall deal with the equal-or-smaller case only). Let T_s denote the running time of the sequential decision algorithm, and T_p (resp.,

Figure 1: Illustration of inversion in 2D. The forbidden regions are I1 and I2, the bounding region is I3. Allowed locations on Ez correspond to intersections of P with Sz.

P) the time (resp., number of processors) of the parallel version, then ρ^* can be computed in sequential time $O(PT_p + T_sT_p \log P)$. For a detailed description of parametric search we refer to [3].

2.1 The Basic Algorithm

In the following, let $D(q,r) = \{ \mathbf{x} \in \mathbf{R}^2 \; ; \; d(\mathbf{x},q) \leq r \}$ denote the disc with radius r around q. In this subsection, we deal with the case that the bounding region $\mathcal{R}$ is given by

$$\mathcal{R} = \bigcap_{i=1}^{n} D(q_i, r_i),$$

for given pairs $(q_i, r_i) \in \mathbf{R}^2 \times \mathbf{R}_{>0}$. To rule out the degenerate case that optima form an arc, we assume that $q_i \notin \mathcal{S} \; \forall \; i = 1, \ldots, n$.

In order to apply the parametric search paradigm, we start with the decision problem to determine, for given $\rho > 0$, whether there exists a location $\mathbf{x} \in \mathcal{R}$ whose Euclidean distance to any $p \in \mathcal{S}$ is larger or equal to $\rho \cdot w(p)$.

Each $p \in \mathcal{S}$ defines a *forbidden region* $F(p) = \{ \mathbf{x} \in \mathbf{R}^2 \; ; \; d(\mathbf{x},p) < \rho \cdot w(p) \}$. An *admissible location* exists if and only if

$$\mathcal{A} = \mathcal{R} \cap \bigcap_{p \in \mathcal{S}} \overline{F(p)}$$

is not empty (here, the bar over $F(p)$ denotes complement). To compute this set of admissible locations, we use inversion. As we shall need the derivation in the bit complexity analysis, we go into some details.

First, we embed the planar problem into 3-space (spanned by the coordinates x, y, z) by identifying the 'input plane' with the plane $z = 1$. Let $\mathbf{0} = (0,0,0)$ denote the origin. The *inversion transformation* maps each point $\mathbf{x} \in \mathbf{R}^3 \setminus \{\mathbf{0}\}$ to $\mathbf{x}' = \mathbf{x}/||\mathbf{x}||^2$. Especially, planes are mapped to spheres that pass by the origin and vice versa. The plane $z = 1$, further denoted E_z, is mapped to the sphere S_z with radius $1/2$ around $C_z = (0,0,1/2)$.

Let us consider a circle with center $(m_x, m_y, 1)$ and radius r on E_z. There exists exactly one sphere S which touches the origin and whose intersection with the plane E_z is equal to that circle. The sphere S has center $C = (m_x, m_y, \frac{1}{2}(1 + r^2 - m_x^2 - m_y^2))$ and radius $||C||$. Inversion maps S to the plane E defined by

$$2m_x x + 2m_y y + (1 + r^2 - m_x^2 - m_y^2) z = 1. \qquad (1)$$

Analogous to spheres, inversion maps balls that pass by the origin (and also their complements) to halfspaces. The whole process maps any region $\overline{F(p)}$, $p \in \mathcal{S}$, to a halfspace $H(p)$, and any disc $D(q_i, r_i)$, $i = 1, \ldots, n$, to a halfspace H_i. Let

$$\mathcal{P} = \bigcap_{i=1}^{n} H_i \cap \bigcap_{p \in \mathcal{S}} H(p).$$

$\mathcal{P}$ is a convex polytope, and can be computed in time $O(n \log n)$ via a standard reduction to convex hull.

The intersection of $\mathcal{P}$ with the sphere S_z corresponds to the set of admissible locations $\mathcal{A}$. One easily sees that this intersection can only have linear complexity, and hence we could afford to compute it explicitly. Alternatively, it is possible to simply check for the existence of an admissible location. However, care has to be taken here: though it is usually the case that $\mathcal{P}$ lies inside S_z when $\rho \geq \rho^*$, it can also happen in specific cases that $\mathcal{P}$ lies outside (such a case e.g. occurs if there is only one input point in $\mathcal{S}$).

So far we have seen how to solve the decision problem in sequential $O(n \log n)$ time. A parallel version can be obtained by using the algorithm in [2] which computes the convex hull of n points in 3-space in $O(\log^2 n)$ time with $O(n)$ processors. (Note here that the standard reduction from halfspace intersection to convex hull requires the knowledge of an inner point of the intersection. This point can e.g. be determined by linear programming in $O((\log \log n)^3)$ parallel time on n processors [1]). The intersection test between $\mathcal{P}$ and S_z is easily done in constant time with $O(n)$ processors.

By applying the parametric search paradigm, we finally obtain running time $O(n \log^4 n)$.

This time bound crucially depends on the parallel time complexity of computing intersections of halfspaces. There are many results on this basic problem, including a *randomized* algorithm with $O(\log n)$ parallel time complexity [14]. By leaving the firm ground of deterministic algorithms, the time bound in theorem 1 can thus be improved to $O(n \log^3 n)$ for case (a).

2.2 Polygonal Bounding Regions

Let $G = (v_1, \ldots, v_n)$ be a simple, closed polygon with n vertices. Here, simple means that edges do not intersect. In this subsection, we consider the case that $\mathcal{R}$ is the region enclosed by G.

Compared to case (a), there are two difficulties. First, in case (a) the set of admissible locations $\mathcal{A}$ could be expressed as intersection of a polytope $\mathcal{P}$ and the sphere S_z. This is not possible any more. Second, and more crucial, the set of admissible locations can now have quadratic complexity. As an example, consider n sites with equal weights that are arranged in 2 rows on a grid. For adequate choices of ρ, these sites induce $O(n)$ 'rhombs' of admissible locations in the plane. If we overlay these sets by an adequate comb-shaped polygon, then $\mathcal{A}$ will consist of $O(n^2)$ connected components.

The first problem only introduces technical complications. In the sequential decision algorithm, we first compute

$$\mathcal{P}_{\mathcal{S}} = \bigcap_{p \in \mathcal{S}} H(p),$$

intersect $\mathcal{P}_{\mathcal{S}}$ with S_z, and then *retransform* the intersection into the original plane by inversion. Note that the topology of this region does not change by the transformation, which simply maps intersections of facets of $\mathcal{P}_{\mathcal{S}}$ with S_z to circular arcs in E_z. Let

$$\mathcal{A}_{\mathcal{S}} = \bigcap_{p \in \mathcal{S}} \overline{F(p)}$$

denote the resulting region, represented by a list of circular arcs for each connected component. The remaining part is to detect if the regions $\mathcal{A}_{\mathcal{S}}$ and $\mathcal{R}$ intersect.

Our approach to the second problem is based on the observation that $\mathcal{A}_{\mathcal{S}} \cap \mathcal{R} \neq \emptyset$ if and only if one of the following conditions holds:

(i) The contours of both regions intersect.

(ii) $\mathcal{R}$ contains a connected component of $\mathcal{A}_{\mathcal{S}}$.

(iii) A connected component of $\mathcal{A}_{\mathcal{S}}$ contains $\mathcal{R}$.

We proceed as follows: first, we check for case (i) by a simple sweep-line algorithm which is terminated once an intersection has been detected (note that each contour has only linear complexity). If this fails, cases (ii) and (iii) can be checked separately by planar point location (testing during the sweep is also possible).

Each step of the sequential algorithm takes $O(n \log n)$ time. The basic parts, convex hull, plane sweep and point location can be done in parallel $O(\log^2 n)$ time with n processors [9]. Summing up, parametric search yields time complexity $O(n \log^4 n)$.

2.3 Related Problems

The solutions in the previous subsections are based on the fact that the union of forbidden regions has only linear complexity and that it can be computed by a 'parallelizable' subquadratic algorithm. Direct extensions to the considered problem – yielding the same time bound $O(n \log^4 n)$ – include:

(i) Monotone weight functions:
For point site p, we may define the weighted distance to the location $\mathbf{x}$ as $f_p(d(\mathbf{x}, p))$, for any strictly increasing function $f_p : \mathbf{R} \to \mathbf{R}$.

(ii) Different metrics:
If we replace the Euclidean distance by the L_1– or L_∞–metric, then forbidden regions get squares instead of discs. Again, the union has only linear complexity, and can be computed by the algorithm in [13] in sequential/parallel $O(n \log n)/O(\log^2 n)$ time.

(iii) Different domains:
Instead of optimizing in Euclidean 2-space, our domain may be the *unit sphere*:
given a set $\mathcal{S}$ of n weighted points on the unit sphere $S^2 = \{ \mathbf{x} \in \mathbf{R}^3 \; ; \; ||\mathbf{x}|| = 1 \}$, find a location $\mathbf{x} \in S^2$ which maximizes $\min\{ w(p)^{-1} \cdot d(\mathbf{x}, p) \; ; \; p \in \mathcal{S} \}$.

The solution to the latter problem can be obtained analogous to subsection 2.1, and is even more intuitive. The forbidden region $F(p)$, for $p \in \mathcal{S}$, is a spherical cap on S^2, and can be 'cut out' by a plane. The admissible locations thus naturally correspond to the intersection of a sphere (here, S^2) with a convex polytope.

3 Exact Approximation

In this section, we assume that all input numbers (i.e., coordinates of input points and weights) are given as L-bit integers. For simplicity in this abstract, we shall only consider case (a). We seek an algorithm that finds the optimum by using integer arithmetic.

As a first hurdle on the way to such an 'exact algorithm', note that the optimal locations as well as the value ρ^* can be algebraic numbers. We elude this problem by redefining the solution in a more combinatorial way: as output of an exact algorithm, we shall expect the sites that define the global maxima of the optimization function. To be more precise, let $\mathbf{x}^*$ be an optimal location. Then $\mathcal{S}^* = \{ p \in \mathcal{S} \; ; \; w(p)^{-1} \cdot d(\mathbf{x}^*, p) = \rho^* \}$ is assumed to be output.

In order to make an ε-approximation exact, we need to know the gap between any two values of ρ at which the topological structure of the boundary of the set of admissible locations $\mathcal{A}$ changes (we call such values 'critical values' of ρ). In the following, let δ be this gap.

Using the decision algorithm from subsection 2.1, we can approximate ρ^* by binary search. Let $\rho^* \in I = [\rho_1, \rho_2]$, with I an interval of length $\leq \delta$ and with rational limits. Then the output sites can be detected from the arrangement of boundary arcs of the set of admissible locations, computed for $\rho = \rho_1$.

Assume we know δ. Then the decision algorithm in subsection 2.1, with rational ρ as input, can directly be adapted to the need of exact computation: the polytope $\mathcal{P} = \mathcal{P}(\rho)$ can be described purely by rational numbers, and the intersection test with S_z can be done by computing the necessary distances with up to $O(|\log \delta|)$ bits of precision. All elementary operations can be performed in time $O(\mu(L))$ [6]. With $\delta = 2^{-O(L)}$, this scheme thus has running time $O(L\mu(L) n \log n)$.

The practicability of the exact approach hinges on the separation gap δ. To obtain a good bound is more subtle than the algorithm above may suggest: the radius ρ^* (as well as the other critical values for ρ) is a square-root which again contains a square-root. Our first approach to compute δ was based on the fact that optimal (resp., critical) locations in the plane lie at the intersections of Appolonius circles of pairs of input points. This can be used to characterize the critical values of ρ, and yielded a gap of roughly $800L$. The next subsection shows how a gap of roughly $50L$ can be deduced by using inversion.

3.1 The Separation Gap

In this subsection, we use the O–notation to suppress additive constants in the bit length of integer numbers.

Our goal is to separate the critical values of ρ. Each critical value is a value of ρ at which a vertex, an edge or a face of $\mathcal{P}(\rho)$ touches the sphere S_z. It is easy to see that the third case is impossible as forbidden discs can not emerge or vanish for $\rho > 0$.

Let us consider the first case. A vertex of $\mathcal{P}(\rho)$ is the common intersection of 3 planes $E_1(\rho)$, $E_2(\rho)$, $E_3(\rho)$, each defined by an equation of the form (1), with $(m_x, m_y) = q_i$ and $r = r_i$ for some i, or $(m_x, m_y) = p$ and $r = \rho \cdot w(p)$ for some $p \in \mathcal{S}$. Vice versa, each choice of 3 parametric planes $E_1(\rho)$, $E_2(\rho)$, $E_3(\rho)$ defines a parametric vertex $v(\rho)$, and this vertex determines potential critical values of ρ. These critical values are zeroes of the polynomial

$$f(\rho) = \text{numerator}\,(d(v(\rho), C_z)^2 - \frac{1}{4}).$$

This polynomial is of the form

$$f(\rho) = \alpha + \beta\rho^2 + \gamma\rho^4,$$

where α,β and γ are $O(6L)$-bit integers.

Analogous to this case, also the second case (edge of $\mathcal{P}$ touches S_z) leads to quadratic polynomials in ρ^2, this time with $O(4L)$-bit integer coefficients.

Now assume that ρ_1 and ρ_2 are two distinct critical values, defined as zeroes of two polynomials $f_1(\rho)$ and $f_2(\rho)$. One way to separate these roots is by applying Rump's bound to $(f_1 \cdot f_2)(\rho)$. A better way is as follows: We first derive a lower bound for $|\varepsilon|$ in

$$\varepsilon = \rho_1^2 - \rho_2^2.$$

Here, ρ_i^2 can be calculated from f_i as

$$\rho_i^2 = \frac{\alpha_i' + \sqrt{\beta_i'}}{\gamma_i'},$$

where α_i', β_i' and γ_i' are $O(6L)$, $O(12L)$ and $O(6L)$-bit integers, respectively. By repeated squaring, the above expression can be transformed to a polynomial equation

$$g(\varepsilon) = a_0 + a_1\varepsilon + \ldots + a_4\varepsilon^4 = 0$$

with integer coefficients $a_0, \ldots, a_4$ of bit-size $O(48L)$ each. Cauchy's bound applied to g gives

$$|\varepsilon| \geq \frac{1}{1 + \max\{|a_0|, \ldots, |a_4|\}} = \Omega(2^{-48L}).$$

Finally, by estimating $\rho_i \leq 2^{2L}$, we get

$$\delta = |\rho_1 - \rho_2| = \frac{|\rho_1^2 - \rho_2^2|}{|\rho_1 + \rho_2|} = \Omega(2^{-50L}).$$

Lemma 1 *In the weighted maximin location problem, two distinct maximum values can be separated with an absolute precision of $O(50L)$ bits.*

Though this is not a low constant, it is not totally impractical, and it reflects the worst case. If the minimum is unique, then it may be detected with lower precision – implementations can often be made sensitive to this (see e.g. [5]).

4 Conclusion

In this paper, we presented the first subquadratic algorithms for the weighted maximin location problem. While parametric search is the method of choice in an algebraic computing model, the bit framework seems more suitable to develop practical algorithms for the considered class of optimization problems - and is theoretically challenging as well.

In an exact implementation, the parametric search solution would require calculations with $O(L)$-bit integers, increasing the running time in theorem 1 by a factor $\mu(L)$. Comparing both approaches in the bit framework, we get a tradeoff in running times between a factor of L and a factor of $\log^3 n$.

We conjecture that other problems that are traditionally solved by parametric search do likewise yield to the exact approach. It would be interesting to derive some general results on this.

As a final remark, we note that the complexity of the maximin location problem is still somewhat mysterious: by using floor functions combined with addressing, the 1-dimensional counterpart (MAXGAP on a line) can be solved in linear time – despite the lower bound. It is an open question if this result generalizes to 2D. Alternatively, we state as open problem if the maximin location problem can be solved in the bit model with a time complexity that depends only linearly on n.

5 Acknowledgements

Thanks are due to Günter Hotz and Chee Yap for many helpful discussions and – not least – for pointing out to us some of the topics of this paper.

References

[1] M. Ajtai, N. Megiddo, "A Deterministic Poly(log log N)-Time N-Processor Algorithm for Linear Programming in Fixed Dimension", *24th Annual ACM STOC*, 1992, pp. 327-338.

[2] N. Amato, F. Preparata, "The Parallel 3D Convex Hull Problem Revisited", *Computational Geometry and Applications*, 2, 1992, pp. 163-173.

[3] P. Agarwal, M. Sharir, S. Toledo, "Applications of parametric searching in geometric optimization", *Journal of Algorithms*, 17, 1994, pp. 292-318.

[4] F. Aurenhammer, H. Edelsbrunner, "An optimal algorithm for constructing the weighted Voronoi diagram in the plane", *Pattern Recognition*, 17(2), 1984, pp. 251-257.

[5] C. Burnikel, K. Mehlhorn, S. Schirra, "How to compute the Voronoi diagram of line segments: theoretical and experimental results", *Proc. ESA 94, LNCS Vol. 855*, 1994, pp. 227-239.

[6] R.P. Brent, "Fast multiple-precision evaluation of elementary functions", *Journal of the ACM*, 23(2), 1976, pp. 242-251.

[7] Z. Drezner, G. Wesolowsky, "A maxmin location problem with maximum distance constraints", *IEE Trans.*, 12, 1980, pp. 249-252.

[8] F. Follert, *Lageoptimierung nach dem Maximin-Kriterium*, Diploma Thesis, Univ. d. Saarlandes, Saarbrücken, 1994.

[9] J. JaJa, *An Introduction to Parallel Algorithms*, Addison-Wesley, 1992.

[10] K. Kedem, R. Livne, J. Pach, M. Sharir, "On the union of jordan regions and collision-free translational motion amidst polygonal obstacles", *Discrete and Computational Geometry*, 1(4), 1986, pp. 59-71.

[11] N. Megiddo, "Applying parallel computation algorithms in the design of serial algorithms", *Journal of the ACM*, 30, 1983, pp. 852-865.

[12] E. Melachrinoudis, *The maximin single facility location problem using an Euclidian metric*, Ph.D. Thesis, Dept. of Industrial Engineering and Operations Research, University of Massachusetts, MA, 1980.

[13] F. Preparata, M. Shamos, *Computational geometry: an introduction*, Springer, 1988.

[14] J. Reif, S. Sen, "Optimal Parallel Randomized Algorithms for Three-Dimensional Convex Hulls and Related Problems", *SIAM Journal on Computing*, 23(3), 1994, pp. 466-448.

[15] J. Smith, P. Winter, "Computational geometry and topological network design", in D. Du, F. Hwang (eds.) *Computing in Euclidian Geometry*, World Scientific Publ. Co., 1992.

[16] R. Tamassia, J. Vitter, "Parallel transitive closure and point location in planar subdivisions", *SIAM Journal on Computing*, 20(4), 1991, pp. 708-725.

[17] G. Toussaint, "Computing Largest Empty Circles with Location Constraints", *Int. Journal of Computer and Information Sciences*, 12(5), 1983, pp. 347-357.

[18] C.-K. Yap, "Towards exact geometric computation", *Proc. 5th Canadian Conf. on Comp. Geom.* , 1993, pp. 405-419.