УДК 517.11

T. Galkovskyi, B. Gärtner, B. Rublyov

THE DOMINATION HEURISTIC FOR LP-TYPE PROBLEMS

Certain geometric optimization problems, for example finding the smallest enclosing ellipse of a set of points, can be solved in linear time by simple randomized (or complicated deterministic) combinatorial algorithms. In practice, these algorithms are enhanced or replaced with heuristic variants that are faster but do not come with a theoretical runtime guarantee.

In this paper, we introduce a new speed-up heuristic that can easily be integrated into the known linear-time algorithms, without decreasing their worst-case performance. The heuristic can actually be defined for any problem in the well-known abstract class of LP-type problems; its effectiveness in practice depends on whether and how fast the heuristic can be implemented for the specific problem at hand.

We provide test results showing that for two concrete problems, the new heuristic may lead to significant speedups compared to state-of-the-art implementations that are available in the Computational Geometry Algorithms Library CGAL.

1 Introduction

The first (expected) linear-time algorithm for the problem of finding the smallest enclosing ellipsoid of a set of points in d-dimensional space is due to Welzl [10] and was motivated by an earlier randomized algorithm for linear programming by Seidel [9].

Here is the essential idea of the algorithm (described for dimension 2): to find the smallest enclosing ellipse $E(P)$ of a set P of n points, choose a point $p \in P$ at random, and recursively compute $E := E(P\setminus\{p\})$. If p happens to lie inside E, we are done; otherwise, we can conclude that p must be on the boundary of $E(P)$, and we therefore recursively compute the smallest enclosing ellipse $E(P,\{p\})$ of P with p on the boundary. A generic recursive call computes $E(P, R)$ for a set R of boundary points, where the problem is easy for $|R| = 5$, since an ellipse is uniquely determined by 5 points.

Despite its simplicity and its backtracking flavor, this algorithm achieves expected runtime $O(n)$ for any fixed dimension d. The key observation is that the probability for p not being contained in E is $O(1/n)$, meaning that the computation of $E(P\setminus\{p\})$ happens only with small probability. The same algorithm also works for the similar problem of finding the smallest enclosing *ball* of a set of points.

Concerning practical performance, it was already observed by Welzl in his original paper [10] that the above algorithm is rather slow. Welzl suggested a *move-to-front* variant of his algorithm that performs much better in practice but does not come with a theoretical runtime bound anymore, short of a trivial bound.

Further speedups in practice can be achieved by heuristics that take the geometry into account (in contrast, move-to-front is purely combinatorial). For example, the faster algorithm for computing smallest enclosing balls uses Welzl's move-to-front variant for small problem instances, and a *farthest-point* heuristic for large instances. Given a candidate ball which is not yet enclosing, the algorithm searches for the «worst outlier» and uses it to determine a better candidate ball [3].

The disadvantage of such geometric heuristics is that they do not generalize to more abstract settings. One particular such setting is that of *LP-type problems* due to Matousek, Sharir and Welzl [6]. Essentially, an LP-type problem is an optimization problem over an abstract set of constraints H (points in the case of smallest enclosing ellipsoids and balls). Given the optimal solution subject to a subset G of the constraints, the LP-type framework only requires a test whether this solution violates a given constraint $h \in H\setminus G$ (test whether a point is outside the smallest enclosing ellipse/ball of a subset); there is no abstract notion of «how much» a constraint is violated.

The class of LP-type problems includes a large number of practically relevant geometric optimization problems [6]. Given just two problem-specific primitive operations, the randomized algorithm of Matousek, Sharir and Welzl can be used to solve every LP-type problem in expected linear time (linear in the number of constraints H, given that the *combinatorial dimension* of the problem is fixed; see [6] for details).

The generic LP-type algorithm is actually an improvement of the above algorithms of Seidel [9] and Welzl [10]; it still works for smallest enclosing ellipsoids and balls but also (and this is its main strength) for other problems with somewhat less structure.

Just like Welzl's algorithm, the generic LP-type algorithm can typically be improved in practice, at the cost of losing the theoretical guarantees. For smallest enclosing ellipsoids and balls, this has independently been done by Lyashko, Petunin and Rublev [7, 8, 5] and Friedman [2]; on the level of LP-type problems, both approaches coincide. Here is the overview of their approach (described for ellipses in dimension 2): maintain a set B of at most five points along with their smallest enclosing ellipse E. Then iterate through the remaining points, and whenever a point p outside E = E(B) is found, update E to $E(B\cup\{p\})$ and reset B to the set of boundary points of this new ellipse. Computation of the smallest enclosing ellipsoid $E(B\cup\{p\})$ can be done for example with Welzl's algorithm as a subroutine. As long as at least one update took place during the iteration, continue with another iteration through the points. Upon termination, this method has computed the smallest enclosing ellipse of the whole point set.

In this paper, we show that this method can further be improved for many inputs. Namely, before a point p is tested for containment in E = E(B), we test whether it is contained in the convex hull of the set B. If so, we can be sure that p will not contribute to the final ellipse and can therefore be removed altogether. The benefit is that all subsequent iterations won't see p anymore. We call this the *convex hull heuristic*, and it is obviously valid for all convex bounding volumes.

Whether this is effective largely depends on the distribution of input points, and on the cost of the test «$p \in E(B)$». At this point, we would like to make it clear that we are mainly interested in *exact* computations that deliver the mathematically correct result and not an (more or less meaningful) approximation of it. This is exactly the realm of CGAL, the Computational Geometry Algorithms Library (www.cgal.org). In case of smallest enclosing ellipses, the exact test «$p \notin E(B)$» is fairly involved, since E(B) may have irrational coordinates and is therefore not explicitly computed.

Even if containment tests are relatively cheap (like for smallest enclosing balls of points or other balls), it may pay off to switch on the convex hull heuristic.

From a theoretical point of view, the most interesting feature of the convex hull heuristic is that it can be generalized to arbitrary LP-type problems, and that it can be incorporated into existing algorithms without affecting the theoretical guarantees. We will get to this in the Section 3.3 of this paper. In a nutshell, the generalization (which we call the *domination heuristic)* is this: a constraint $h \in H$ is called *dominated* by $B \subseteq H$ if the following holds: whenever h is violated by the optimal solution subject to some subset G of constraints, then this solution also violates an element of B. Under this definition, a point inside the convex hull of other points is dominated by these points with respect to smallest enclosing ellipses, say.

The domination heuristic (which is strictly combinatorial) simply throws away constraints that have been identified as being dominated during the computations. It depends on the concrete LP-type problem whether an efficient domination test is available at all, and whether it will actually remove constraints. But in any case, the expected runtime will asymptotically not increase.

The remainder of the paper is organized as follows. In Section 2, we review the algorithms of Welzl [10] and of Rublev [7, 8] for smallest enclosing ellipsoids, and we enhance them with the convex hull heuristic; we provide test results that show the performance gain (or sometimes loss) under various input distributions.

In Section 3, we present the generalization to LP-type problems. We review the generic (recursive) LP-type algorithms of Matousek, Sharir and Welzl [6], and the simple iterative one resulting from generalizing the methods of Rublev [7, 8] and Friedman [2]. We show that even when the algorithm of Matousek, Sharir and Welzl is equipped with the domination heuristic, the expected complexity of O(n) still holds.

As an example of practical usefulness and ease of use of the general domination heuristic in Section 4 we describe its application to smallest enclosing sphere of spheres problem.

2 Smallest Enclosing Ellipses

We start from Welzl's algorithm 1 for computing smallest enclosing ellipses, written down formally [10]. As a subroutine, it needs to solve the constant-size problem of computing the smallest ellipse $E_0(S)$ with a set S of at most 5 points on its boundary. The algorithm returns a *basis* S, $R \subseteq S \subseteq P\cup R$ with the property that $E_0(S) = E(P, R)$. To compute E(P), we call the algorithm with $R = \varnothing$.

```
Data: Disjoint sets $P$ and $R$ of points, $|R| \leq 5$
Result: Basis of the smallest ellipse that contains $P$ and has $R$ on its boundary
SmEll($P,R$) begin
    $Q \leftarrow \emptyset$;
    $S \leftarrow R$;
    $E \leftarrow E_0(S)$;
    foreach $p \in P$ in random order do
        if $p \notin E$ then
            $S \leftarrow$ SmEll($Q, R \cup \{p\}$);
            $E \leftarrow E_0(S)$;
        end
        $Q \leftarrow Q \cup \{p\}$;
    end
    return $S$;
end
```

Algorithm 1 Welzl's algorithm

One problem that leads to poor performance of this algorithm in practice is that the recursive call throws away the ellipse computed so far and starts from scratch, only exploiting the additional information that p has to lie on the boundary. Rublev's algorithm [7, 8] tries to reuse the ellipse so far: whenever a point is found to be outside, the working basis S and the corresponding ellipse $E^0(S)$ are updated to also cover this point.

The pseudocode is given in Algorithm 2. The proof of correctness can be found in [7, 8].

```
Data: Point set $P$
Result: Smallest enclosing ellipse of set $P$

begin
    $S \leftarrow \emptyset$;
    $E \leftarrow \varnothing$;
    $done \leftarrow false$;
    while not done do
        $done \leftarrow true$;
        foreach $p \in P$ do
            if $p \notin E$ then
                $S \leftarrow$ SmEll$(S, \{p\})$;
                $E \leftarrow E_0(S)$;
                $done \leftarrow false$;
            end
        end
    end
    return $E$
end
```

Algorithm 2 Rublev's Algorithm

2.1 Computational Complexity

For both algorithms, the runtime is dominated by the number of *containment tests* «$p \notin E$». For Algorithm 1, the expected number of such tests is O(n), where $n = |P|$. In contrast, we have no good bound for Rublev's algorithm.

One might be tempted to think that every iteration of the main loop in Rublev's algorithm adds one element of the final basis to the working basis S, but this is not true. In fact, experiments show that points from the final basis could be added and removed from the working basis several times. It remains an open problem whether there exists any nontrivial bound on the number of outer loop iterations.

2.2 The Convex Hull Heuristic

Like in many other geometric algorithms, correct results can only be guaranteed for Algorithms 1 and 2 if multiprecision arithmetic is used. In fact, the package `Min_ellipse_2` in the CGAL library implements the (faster) move-to-front variant of Algorithm 1 [10], using exact arithmetic. It turns out that the containment tests «$p \notin E$» are the major bottleneck. This is on the one hand due to the fact that containment tests are the most frequent operations, but on the other hand, exact containment tests are not easy because the involved ellipse E will in general have irrational coordinates [4].

Therefore, reducing the necessary number of containment tests will lead to an immediate speedup of the algorithm. Here is the simple but crucial observation:

Observation 1 *If during Algorithm 1 or 2, the considered point p is contained in the convex hull of the working basis S, then p can be removed from further consideration.*

The convex hull of S is the smallest convex set that contains all points from S. The ellipse $E(P\backslash\{p\}, R)$ (for Algorithm 2, $R = \varnothing$) is some convex set that contain S (because of $S \subseteq P \cup R$), and therefore it also contains p. It follows that $E(P\backslash\{p\}, R) = E(P, R)$, meaning that p can be ignored without changing the output of the algorithm.

The convex hull computation introduces at most constant overhead, since it involves at most five points; it has the potential, though, of removing many points. We will refer to resulting variant of Welzl's method as Algorithm 3.

Obviously, there are inputs for which Algorithm 3 will not improve over Algorithm 1, for example points in convex position. For points randomly distributed within a square or disk, through, major savings can be expected (see the benchmark section below).

In the same way, we can enhance Rublev's Algorithm 2, which will be refered as Algorithm 4 with the convex hull heuristic.

2.3 Benchmarks

Algorithms described above were implemented using the `Min_ellipse_2` package of CGAL library as a base. Hence all implementations share the same primitive operations, perform exact computations and only differ by the core algorithm itself. To ensure thorough testing several distributions of random points were chosen: uniform in unit square, uniform in unit disk, uniform on unit circle, uniform on the integer rectangular grid (lots of duplicates). Also small and large synthetic cocircular sets were used to check behavior of algorithms in extreme (mostly theoretical) cases.

In Table 1 we provide the averaged runtimes for over about 100 random sets of N = 10 000 points for each distribution. Because absolute runtimes don't provide a lot of useful information we organized table as following: the actual runtime of base implementation (`Min_ellipse_2`) only is provided, for other implementations we specify the relative speed up multiplier (the bigger is the multiplier, the faster is the implementation).

	in unit square	**in unit disk**	**on unit circle**	**on grid**	**small cocircular**	**large cocircular**
Algorithm 1	22021ms	8155ms	7205ms	2743ms	*272ms*	*597ms*
Algorithm 3	8.3x	2.3x	0.9x	*2.8x*	0.7x	0.7x
Algorithm 2	2.3x	1.8x	*1.7x*	0.6x	0.5x	0.5x
Algorithm 4	*10.7x*	*2.7x*	1.5x	1.5x	0.4x	0.4x

Table 1: **Smallest enclosing ellipse implementations**

3 An Abstract Framework

This section discusses how the convex hull heuristic described in the previous section can be generalized to the whole abstract class of *LP-type problems.* This class includes the smallest enclosing ellipse problem but also many other geometric optimization problems [6].

3.1 LP-type Problems

Let us consider abtract optimization problems specified by pairs (H, w), where H is a finite set, and w: $w^{H} \rightarrow W \cup \{-\infty\}$ is a function with values in a linearly ordered set $(W \cup \{-\infty\}, \leq)$, the value $-\infty$ (standing for 'undefined') preceding all values in *W*. The elements of H are called *constraints,* and for G $\subseteq$ H, w(G) is called the (optimal) *value* of G. The goal is to compute the value w(H) of H, using certain primitive operations to which we get below.

(H,w) is called an LP-type problem if the following two axioms are satisfied.

Axiom 1 *(Monotonicity) For any F, G with $F \subseteq G \subseteq H$, we have $w(F) \leq w(G)$.*

Axiom 2 *(Locality) For any $F \subseteq G \subseteq H$ with $-\infty \neq w(F)$ $w(G)$ and any $h \in H$, $w(G) < w(G \cup \{h\})$ implies that also $w(F) < w(F \cup \{h\})$.*

If w(G) < w(G $\cup$ {h}), we say that constraint h is *violated* by G. Monotonicity is a natural requirement when we are talking about minimization problems: adding more constraints cannot decrease the optimal value. Locality essentially says that there are no local optima: an equivalent formulation is that whenever w(F) < w(G) for F $\subseteq$ G, then there is h $\in$ G such that the value of F can locally be improved by switching to F$\cup${h}.

When we write the smallest enclosing ellipse problem as an LP-type problem, the set of constraints is the set of input points, and the value of a subset is the volume of its smallest enclosing ellipse (value $-\infty$ arises if the affine hull of the subset is not the whole plane). A constraint (point) is violated by a subset if it lies outside its smallest enclosing ellipse. Uniqueness of the smallest enclosing ellipse [10] is easily seen to imply the locality property.

The fact that makes Welzl's method efficient is that the smallest enclosing ellipse is determined by no more than five points. The abstract counterpart is the *combinatorial dimension* of an LP-type problem. A *basis* is a subset B $\subseteq$ H such that w(B\{h}) < w(B) for all h $\in$ B. This means, a basis is an inclusion-minimal set defining a certain value. Now, the maximum cardinality of any basis is called the combinatorial dimension of (H,w), and is denoted by $\boldsymbol{\delta} = \boldsymbol{\delta}_{(Hw)}$. For any smallest enclosing ellipse instance, we have $\boldsymbol{\delta} < 5$.

Solving an LP-type means to find a basis B such that w(B) = w(H). In general, a basis *of* G$\subseteq$ H is a basis B $\subseteq$ G with w(B) = w(G). From such a basis, the value w(G) is usually easy to compute. We assume that the following primitive operations are available:

Violation test
: Given a basis B and a constraint h $\notin$ B, decide whether $w(B) < w(B \cup \{h\})$

Basis computation
: Given a basis B and a violating constraint h, compute a basis of $B \cup \{h\}$.

3.2 LP-type Algorithms

In the abstract setting of LP-type problems, there is no notion of «fixing points on the boundary» like it is employed in Welzl's algorithm 1. Consequently, this algorithm does not generalize to the LP-type setting. But Rublev's Algorithm 2 has an immediate LP-type counterpart, see Algorithm 5.

```
Data: LP-type problem $(H, w)$ specified by primitive operations
Result: Basis $B$ such that $w(B) = w(H)$
begin
    $B \leftarrow \emptyset$;
    $done \leftarrow false$;
    while not done do
        $done \leftarrow true$;
        foreach $h \in H$ do
            if $w(B) < w(B \cup \{h\})$ then
                $B \leftarrow$ basis $(B \cup \{h\})$;
                $done \leftarrow false$;
            end
        end
    end
    return $B$
end
```

Algorithm 5 Rublev/Friedman algorithm for LP-type problems

But there is also an algorithm in the LP-type setting that comes with theoretical runtime guarantees, and this is Algorithm 6 due to Matousek, Sharir and Welzl [6]. The algorithm is randomized and recursive; given a pair (G, B), where B ⊆ G is a basis (not necessarily of G yet), the algorithm computes a basis of G. To start off, it requires some initial basis (B = ∅ will do).

```
Data: LP-type problem $(H, w)$ specified by primitive operations;
      $G \subseteq H$; $B \subseteq G$ a basis
Result: Basis $B$ such that $w(B) = w(G)$
MSW(G,B) begin
    $F \leftarrow \emptyset$;
    foreach $h \in G \setminus B$ in random order do
        $F \leftarrow F \cup \{h\}$;
        if $w(B) < w(B \cup \{h\})$ then
            $B \leftarrow$ basis $(B \cup \{h\})$;
            return MSW(F,B);
        end
    end
end
```

Algorithm 6 MSW(G, B) algorithm for LP-type problems

The following has been shown by Matousek, Sharir and Welzl[6]

Theorem 1 *Given an LP-type problem (H,w) with |H|=n and fixed combinatorial dimension δ, Algorithm 6 requires an expected number of O(n) primitive operations to solve it.*

If δ is fixed, each primitive operation takes constant time (even if done in a brute-force fashion), so that the algorithm takes expected linear time. For a number of concrete LP-type problems, this was the first known linear-time algorithm.

3.3 The Domination Heuristic

In this section, we want to generalize the convex hull heuristic of Section 2.2 to the abstract setting of LP-type problems. Assume that a point p is in the convex hull of a set S. The convex hull heuristic works for the following reason: whenever p is outside the smallest enclosing ellipse of some subset, then there is also some point from S that is outside (an immediate consequence of convexity of ellipses). This leads us to the abstract concept of *domination.*

Definition 1 *Let (H, w) be an LP-type problem, B ⊆ H and h ∈ H. h is called dominated by B if the following holds: for every set G ⊆ H such that w(G) < w(G∪{h}), there exists an element h′ ∈ B such that w(G) < w(G∪{h′}).*

Again, it easily follows that dominated elements can be removed without changing the result.

Lemma 1 *Let (H,w) be an LP-type problem, h ∈ G ⊆ H. If h is dominated by some set B ⊆ G\{h}, then w(G) = w(G\{h}).*

Since B ⊆ G\{h}, we have G\{h}∪{h′}=G\{h} for all h′ ∈ B, hence w(G\{h}∪{h′}) = w(G\{h}). This means that no element of B violates G\{h}, and since h is dominated by B, h cannot violate G\{h}, either.

This lemma implies the correctness of Algorithm 7 which is Algorithm 6, enhanced with the *domination heuristic.* Here, dom (h, B) is a shorthand for «h is dominated by B».

```
Data: LP-type problem $(H, w)$ specified by primitive operations;
      $G \subseteq H$; $B \subseteq G$ a basis
Result: Basis $B$ such that $w(B) = w(G)$
MSW_D(G,B) begin
    $F \leftarrow \emptyset$;
    foreach $h \in G \setminus B$ in random order do
        if not dom (h, B) then
            $F \leftarrow F \cup \{h\}$;
            if $w(B) < w(B \cup \{h\})$ then
                $B \leftarrow$ basis $(B \cup \{h\})$;
                return MSW_D(F,B);
            end
        end
    end
end
```

Algorithm 7 MSW_D(G,B) algorithm for LP-type problems, with domination heuristic

In order to be able to execute Algorithm 7, we need to stipulate a new primitive, namely dom (h, B). Having an exact such primitive may be difficult; even in the case of smallest enclosing ellipses, we don't get this: the fact that p is in the convex hull of S is a *sufficient* condition for p being dominated by S, but not a necessary condition. But the following conservative primitive can be implemented and is enough to ensure correctness of Algorithm 7.

Domination test Given a basis B and a constraint h ∉ B, dom (h, B) returns true only if h is dominated by B; if dom (h, B) returns false, h may or may not be dominated by B.

This primitive can of course always be implemented (simply return false), but it makes sense only if it can actually «recognize» some dominations.

3.4 Complexity Analysis

We want to argue that Algorithm 7 still requires only O(n) primitive operations for constant combinatorial dimension. This seems intuitively clear (how can the removal of a dominated constraint generate more work?), but we are not aware of any direct argument along these lines. After all, the removal of a constraint may have the effect that the algorithm «goes along a different path» in the future, and this path may take longer.

In order to argue formally, we have to go into the analysis of Algorithm 6 and show that this analysis still works for Algorithm 7. We only provide a sketch here.

Hidden dimension. Given a pair (G, B) where B ⊆ G is a basis, we call h ∈ B *enforced* in (G, B) if w(B) > w(G\{h}), i.e. if B already has higher value than G\{h}. If h is enforced in (G, B), this implies that h will be contained in *every basis* encountered during the call to MSW(G, B). The *hidden dimension* of (G, B) is δ (combinatorial dimension) minus the number of enforced elements in (G, B).

Let T(n, k) denote the maximum expected number of violation tests of a call to MSW(G, B), where G has size at most n, and (G, B) has hidden dimension at most k. Matousek, Sharir and Welzl prove that T(n, k) = O(n) for fixed k, by exhibiting a suitable recurrence relation for T(n, k). The nontrivial part is the analysis of the recursive call for which it is shown that on average, the hidden dimension goes down substantially (if the call takes place at all which happens with small probability only, just like in Welzl's algorithm).

Adding the domination heuristic. Consider the point in Algorithm 7 at which the last constraint h is being added. (Note that the distribution of h is uniformly random in G.) At this point, we have computed a basis of some set F ⊆ G\{h} with w(F) = w(G\{h}), because we have removed only dominated elements, see Lemma 1. (In Algorithm 6, we have F = G\{h}.)

The fact that w(F) = w(G\{h}) implies that the probability for a second recursive call is still at most k/n, where k is the hidden dimension of (G, B): the recursive call only happens for elements h that violate G\{h} (equivalently F, by locality), and by definition of hidden dimension, there are at most k of them in G\B.

A similar argument shows that the decrease in hidden dimension (when we move to (F, B) in the recursive call) is at least what it would be in Algorithm 6. Since also |F| ≤ |G| at this point, it follows that the recurrence relation that bounds T(n, k) for Algorithm 6 is also valid for Algorithm 7.

Summarizing, we obtain

Theorem 2 *Given an LP-type problem (H,w) with |H| = n and fixed combinatorial dimension δ, Algorithm 5 requires an expected number of O(n) primitive operations to solve it.*

4 Smallest Enclosing Sphere of Spheres

So far, we have shown (Section 2.3) that the domination heuristic can be very effective for smallest enclosing ellipses where it assumes the form of the convex hull heuristic. In this final section, we want to examine another LP-type problem, namely finding the smallest enclosing sphere of a set of *spheres.* This problem is relevant for bounding volume heuristics, and it is theoretically interesting because it can (maybe surprisingly) *not* be solved by Welzl's algorithm [1]. For this problem, we need the general LP-type techniques.

The CGAL library has code also for this problem, and we have integrated a suitable domination heuristic into this code. Ideally, we would like to test whether a given sphere is contained in the convex hull of a set of other spheres, but this is not a cheap operation. Since we are allowed to use a conservative (but possibly less effective) domination test: assume that sphere to check is actually a smallest box, that encloses that sphere, with sides oriented along the axes; and instead of the convex hull of the spheres we use a convex hull of the *centers* of spheres. Then if box (sphere) box actually passes containment test inside decreased convex hull, it is also guaranteed to lie inside the bigger convex hull of spheres. Thus we get the sufficient condition which can be used as domination test.

4.1 Benchmarks

Testing was done for planar case only. The same technique and parameters were used to gather runtimes of implementations for smallest enclosing sphere of spheres problem as in Section 2.3. Because new random variable (sphere radius) should be generated Table 2 represents running times of implementations on the tests where spheres radiuses were exponential distributed (i. e. lots of relatively small spheres and small number of large). On the other hand 3 represents running times of implementations on the tests where all spheres have equal very small radius.

	in unit square	**in unit disk**	**on unit circle**	**on grid**
CGAL	9551ms	17066ms	18687ms	8876ms
CGAL with heuristics	1.9x	2.2x	2.1x	*2.0x*
Rublev	1.7x	1.7x	1.6x	1.7x
Rublev with heuristics	*2.4x*	*2.3x*	*2.4x*	1.7x

Table 2: **Smallest enclosing sphere of spheres implementations (radiuses are exponentially distributed)**

	in unit square	**in unit disk**	**on unit circle**	**on grid**
CGAL	3569ms	6425ms	6108ms	823ms
CGAL with heuristics	1.2x	1.2x	0.8x	0.7x
Rublev	*1.8x*	*1.8x*	*1.7x*	0.6x
Rublev with heuristics	1.6x	1.6x	1.4x	0.5x

Table 3: **Smallest enclosing sphere of spheres implementations (radiuses are equal and very small)**

We didn't use the default geometric heuristic of the `Min_sphere_of_spheres` package during testing to show effect of domination heuristics.

Being quite effective that heuristics performs well on all tested algorithms, making their running times mostly equal.

5 Conclusion

In this paper, we have developed a heuristic that can speed up the existing linear-time algorithm for LP-type problems in practice, while it at the same time maintains the theoretical runtime guarantee of the algorithm. The heuristic is combinatorial, meaning that it can be formulated purely within the known abstract framework of LP-type problems.

We have presented two applications where the heuristic successfully improves the practical performance for many inputs.

1. Kaspar Fischer and Bernd Gärtner. The smallest enclosing ball of balls: Combinatorial structure and algorithms. *International Journal of Computational Geometry and Applications (IJCGA),* 14(4-5):341-387, 2004.
2. F. Friedman. Minimal enclosing circle and two and three point partitions of a plane. In *Proc. International Conference on Scientific Computing,* 2006.
3. B. Gärtner. Fast and robust smallest enclosing balls. In *Proc. 7th annu. European Symposium on Algorithms (ESA),* volume 1643 *of Lecture Notes in Computer Science,* pages 325-338. Springer-Verlag, 1999.
4. B. Gärtner and S. Schönherr. Exact primitives for smallest enclosing ellipses. *Information Processing Letters,* 68:33-38, 1998.
5. S. I. Lyashko and B. V. Rublyov. Minimal ellipsoids and maximal simplexes in 3d euclidean space. *Cybernetics and Systems Analysis,* 39(6):831-834, 2003.
6. J. Matousek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica,* 16:498-516, 1996.
7. B. V. Rublyov and Y. I. Petunin. Minimum-area ellipse containing a finite set of points. i. *Ukrainian Mathematical Journal,* 50(7):1115-1124, 1998.
8. B.V. Rublyov and Y. I. Petunin. Minimum-area ellipse containing a finite set of points. ii. *Ukrainian Mathematical Journal,* 50(8):1253-1261, 1998.
9. R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.,* 6:423-434, 1991.
10. E. Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer, editor, *New Results and New Trends in Computer Science,* volume 555 of *Lecture Notes in Computer Science,* pages 359-370. Springer-Verlag, 1991.

Галковський Т. О., Гартнер Б., Рубльов Б. В.

ЕВРИСТИКА ПЕРЕВАГИ ДЛЯ ЗАДАЧ ЛІНІЙНОГО ПРОГРАМУВАННЯ

Деякі задачі геометричної оптимізації, наприклад пошук найменшого покриваючого еліпса множини точок, можуть бути розв'язані за лінійний час, використовуючи нескладні випадкові (чи складні детерміновані) комбінаторні алгоритми. На практиці ці алгоритми поліпшуються чи заміняються варіантами евристик, що працюють швидше, але теоретичні оцінки часу роботи для них не доведені. У цій статті ми пропонуємо нову прискорюючу евристику, що може бути легко застосована до відомих лінійних алгоритмів, без зменшення їх швидкості у найгіршому випадку. Ми показуємо, що ця евристика може бути визначена для будь-якої задачі з добре відомого класу задач лінійного програмування. Її ефективність на практиці залежить від того, чи можлива, і якщо можлива, то наскільки швидкою виявиться реалізація предиката для конкретної задачі. Ми наводимо результати експериментів, які показують, що для двох задач нова евристика може значно прискорити існуючі реалізації алгоритмів (з бібліотеки геометричних алгоритмів CGAL).