Computational Aspects of Helly's Theorem and its Relatives

David Avis* and Michael E. Houle**

*School of Computer Science, McGill University

Montréal, Québec, Canada H3A 2A7

**Department of Computer Science, University of Newcastle,

Newcastle, New South Wales 2308, Australia

Abstract

This paper investigates computational aspects of the well-known convexity theorem due to Helly, which states that the existence of a point in the common intersection of n convex sets is guaranteed by the existence of points in the common intersection of each combination of $d+1$ of these sets. Given an oracle which accepts $d+1$ convex sets and either returns a point in their common intersection, or reports its non-existence, we give two algorithms which compute a point in the common intersection of n such sets. The first algorithm runs in $O(n^{d+1}T)$ time and $O(n^d)$ space, where T is the time required for a single call to the oracle. The second algorithm is a multi-stage variant of the first by which the space complexity may be reduced to $O(n)$ at the expense of an increase in the time complexity by a factor independent of n.

We also show how these algorithms may be adapted to construct linear and spherical separators of a collection of sets, and to construct a translate of a given object which either contains, is contained by, or intersects a collection of convex sets.

1 Introduction

Of the known results in the field of convexity theory, perhaps the most famous is the existence theorem due to Helly which relates the intersection of a collection of convex sets with the intersection of its subcollections [2, 3, 8]:

Theorem 1 (Helly) *Let $\mathcal{C}$ be a collection of convex sets in $\Re^d$, $\mathcal{C}$ finite or all members of $\mathcal{C}$ compact. Then if every $d+1$ sets intersect, there exists a point common to all the sets.*

This result has had numerous applications in proving other combinatorial statements, in particular those of the general form: *If a certain type of collection is such that each of its subfamilies of k members has a certain property, then the whole collection has that property* (see [2, 8] for a partial survey).

For the case where the collection is finite, an algorithmic implication of such statements is that to determine whether the collection has the property, it suffices to determine whether every subfamily of k members has the property. In the case of Helly's theorem itself, if we had an algorithm which determined whether any $d+1$ members of $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ have a common intersection, then by applying this algorithm to all $\binom{n}{d+1}$ subfamilies of $d+1$ members, we could determine whether all sets of $\mathcal{C}$ intersect. However, until recently no *constructive* applications had been discovered for Helly's theorem or its relatives.

In the next section, we outline a general method by which a point in the intersection of n convex sets in $\Re^d$ may be computed, assuming the existence of an algorithm for finding a point in the intersection of any $d+1$ of these sets. The overall algorithm will be seen to require time proportional to $\binom{n}{d+1} \cdot f(d+1)$, where $f(d+1)$ is the time required to find a point in the intersection of $d+1$ sets.

2 Computing an Intersection Point Using Radon Partitions

A *Radon partition* of a set of n points in $\Re^d$ is a partition into two sets S and T such that their convex hulls $CH(S)$ and $CH(T)$ intersect. We call a point in the intersection of $CH(S)$ and $CH(T)$ a *Radon point*. Radon's theorem [10] guarantees the existence of a Radon partition whenever n is at least $d+2$. From this, the following lemma can be proved:

Lemma 2 *Let $\{K_1, K_2, \ldots, K_{d+2}\}$ be a collection of convex sets in $\Re^d$, and let $P = \{p_1, p_2, \ldots, p_{d+2}\}$ be a set of points such that $p_i \in \bigcap_{j \neq i} K_j$ for $0 \leq i \leq d+2$. If S and T form a Radon partition of P, then $CH(S) \cap CH(T) \subseteq \bigcap_j K_j$.*

This lemma may be used to prove Helly's theorem by induction. Indeed, Radon first used these arguments to this end in 1921 [10]. Implicit in the lemma, moreover, is the basis of a recursive algorithm for finding the common intersection of convex sets.

2.1 The Basic Algorithm

Given a collection $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ of convex sets in $\Re^d$, let us assume that we have available a "black box" or "oracle" which accepts as input the indices of $d+1$ sets of $\mathcal{C}$, and which gives as its output a point in the intersection of these sets, or reports that the sets do not intersect, as the case may be. In reality, the oracle is simply an intersection algorithm tailored specifically to the sets of the collection; to handle other collections of convex sets, we would expect that other oracles tailored to those collections be provided.

Since the size of the input provided to the oracle is dependent upon d but not n, we shall assume that each call to the oracle (and each computation of a Radon partition) takes unit time.

For each $k = 1, 2, \ldots, n-d$, we will compute an intersection point for each collection of the form

$$\{C_1, C_2, \ldots, C_k, C_{x_1}, C_{x_2}, \ldots, C_{x_d}\}, \ \{x_1, x_2, \ldots, x_d\} \subseteq \{k+1, \ldots, n\}. \tag{1}$$

Note that when $k = n-d$, this is the required intersection point. If at any time a call to the oracle reveals that some subcollection has no intersection, then the algorithm terminates.

For $k = 1$, an intersection point of each of the $\binom{n-1}{d}$ families $\{C_1, C_{x_1}, C_{x_2}, \ldots, C_{x_d}\}$ may be obtained directly from the oracle. In general, having found all of the intersection points for the families in (1) up to $k-1 \geq 1$, we compute the values for k by taking a Radon partition of the following $d+2$ intersection points:

- p_0, an intersection point of the family $\{C_1, C_2, \ldots, C_{k-1}, C_{x_1}, C_{x_2}, \ldots, C_{x_d}\}$,
- p_i, an intersection point of the family $\{C_1, C_2, \ldots, C_k, C_{x_1}, C_{x_2}, \ldots, C_{x_d}\} \setminus C_{x_i}$, for $1 \leq i \leq d$, and
- p_{d+1}, an intersection point of the family $\{C_k, C_{x_1}, C_{x_2}, \ldots, C_{x_d}\}$.

Note that $p_0 \ldots p_d$ have already been computed and can be looked up. The intersection point p_{d+1} is obtained by a call to the oracle.

Set $K_0 = C_k$, $K_i = C_{x_i}, i = 1 \ldots d$, and $K_{d+1} = \bigcap_{1 \leq i \leq k-1} C_i$. It is easy to verify that the conditions of Lemma 2 are satisfied by $K_0 \ldots K_{d+1}$ and $p_0 \ldots p_{d+1}$. Let p be a Radon point for the set $\{p_0, p_1, \ldots, p_{d+1}\}$. Then from the definitions and Lemma 2 we see that p is a point common to all the sets $C_1, C_2, \ldots, C_k, C_{x_1}, C_{x_2}, \ldots, C_{x_d}$. Note that even in the case where the points $p_0 \ldots p_{d+1}$ are not all distinct, a Radon partition can be trivially computed.

Therefore, computing a point in all combinations of sets of (1) for each value of $k \geq 2$ involves finding $\binom{n-k}{d}$ Radon partitions and $\binom{n-k}{d}$ calls to the oracle. The total number of Radon partitions computed is

$$\sum_{k=2}^{n-d} \binom{n-k}{d} = \sum_{i=d}^{n-2} \binom{i}{d} = \binom{n-1}{d+1}.$$

Taking the case $k = 1$ into consideration, the number of calls to the oracle is thus

$$\binom{n-1}{d} + \binom{n-1}{d+1} = \binom{n}{d+1},$$

one for each subfamily of $\mathcal{C}$ consisting of $d+1$ members.

2.2 The Multistage Algorithm

The algorithm just described is optimal in the number of calls to the oracle, since one can construct families of n convex sets for which all but one of its subfamilies of $d+1$ members has a non-empty intersection. On the other hand, it is certainly not optimal in terms of space, since at iteration $k = k'$ we keep on hand the $\binom{n-k'+1}{d}$ intersection points generated at iteration $k = k' - 1$. In the remainder of this section, we present a multistage variant of this algorithm which allows storage to be drastically reduced at the expense of a constant factor in the time complexity.

Conceptually, let us divide the collection $\mathcal{C}$ into $m \geq d+2$ groups $G_1, G_2, \ldots, G_m$ of (roughly) $\frac{n}{m}$ member sets. For each group G_i, consider the convex set I_i obtained by intersecting all sets of G_i. If the sets $\mathcal{I} = \{I_1, I_2, \ldots, I_m\}$ were available, the oracle could be used to obtain a point in the common intersection of any choice of $d+1$ sets of $\mathcal{I}$. We could then apply the time-optimal algorithm to generate a point in the common intersection of all sets of $\mathcal{I}$ — which is also the common intersection of all sets of $\mathcal{C}$.

Since the sets of $\mathcal{I}$ are generally unavailable, an alternative to the oracle is needed. Finding a point in the common intersection of the sets $\{I_{x_1}, I_{x_2}, \ldots, I_{x_{d+1}}\}$ is equivalent to finding a point in the common intersection of the member sets of $G = G_{x_1} \cup G_{x_2} \cup \ldots \cup G_{x_{d+1}}$. Such a point may be found by applying the time-optimal algorithm directly to G, or by subdividing G into smaller subgroups and applying recursion.

The following recursive algorithm accepts as its input a collection of convex sets $\mathcal{Q}$, the value m as defined above, and a value l corresponding to the maximum depth of the recursion. If $l = 0$, the number of recursive levels is zero, and the algorithm is equivalent to the more straightforward algorithm described earlier. If the convex sets have a non-empty common intersection, the recursive algorithm returns a point in this intersection; otherwise, it reports that such a point does not exist.

MRadon$(\mathcal{Q}, l, m)$

(1) If $|\mathcal{Q}| \leq d+1$ (or $l = -1$), compute an intersection point of $\mathcal{Q}$ using the oracle. If such a point exists, return it; otherwise, report that the collection has no common intersection point, and abort the algorithm.

(2) ($|\mathcal{Q}| > d+1$) If $l = 0$, set $m \longleftarrow |\mathcal{Q}|$.

(3) Divide $\mathcal{Q}$ into m groups $G_1, G_2, \ldots, G_m$ of roughly equal numbers of convex sets.

(4) (Iteration $k = 1$) For each choice of $1 < x_1 < x_2 < \ldots < x_d \leq m$, recursively compute the

intersection point $p_0 = \mathrm{MRadon}(G, l-1, m)$, where $G = G_1 \cup G_{x_1} \cup G_{x_2} \cup \ldots \cup G_{x_d}$. Set $A_l[x_1, x_2, \ldots, x_d] \longleftarrow p_0$.

(5) For $k = 2$ to $m - d$, do:

(5a) For each choice of $k < x_1 < x_2 < \ldots < x_d \leq m$, do:

(5a1) Recursively compute $p_0 = \mathrm{MRadon}(G, l-1, m)$, where $G = G_k \cup G_{x_1} \cup G_{x_2} \cup \ldots \cup G_{x_d}$.

(5a2) Let X be the ordered index set $\{x_1, x_2, \ldots, x_d\}$, and let X_i be the ordered index set obtained by omitting x_i from X. Find a Radon point p for the $d+2$ points p_0, $A_l[X]$, and $A_l[k, X_i]$, for $1 \leq i \leq d$. Set $A_l[X] \longleftarrow p$.

(6) Return the point $A_l[m-d+1, m-d+2, \ldots, m]$.

The overall correctness of Algorithm MRadon was established in the preceding discussion. However, it should be noted that with each iteration of the inner loop (Step 5a), an entry of the array A_l is overwritten. This use of storage is nevertheless permissible, since each overwritten entry is accessed only once per iteration of the outer loop (Step 5). In the inner loop, entries of the form $A_l[k, X_i]$ are never overwritten.

Algorithm MRadon as stated above makes no mention of how the various groups of sets are maintained, or of how Step 3 may be accomplished. The most straightforward way is to maintain a full list of pointers to the convex sets of the current group; at each recursive call, the pointers of the subgroups under consideration would be passed along. Perhaps a more efficient way is to treat the indices of sets in a subgroup as if they were a collection of disjoint intervals drawn from the index set $\{1, 2, \ldots, n\}$; only the endpoints of these intervals need be passed. For example, if the subgroup were $\{C_2, C_3, C_4, C_5, C_8\}$, then the index intervals would be $\{[2,5], [8,8]\}$, and pointers to C_2, C_5, C_8 and (again) C_8 would be passed. In the next section, we shall see that if l and m are chosen properly, neither strategy has much effect on the overall complexity of the algorithm.

3 Time and Space Analysis of Algorithm MRadon

As one might expect, the complexity of Algorithm MRadon depends largely on the specific choices of l and m. According to the statement of the algorithm, these choices are more or less free: the number of sets processed at the lowest level of the recursion may be larger than m, the number of groups processed at intermediate levels. In this section, we shall analyze the time and space performance of the algorithm by investigating the effects of various choices of l and m, subject to the restriction that the number of sets processed at any level be at most m.

As a measure of the time and space complexities of the algorithm, we shall consider the following quantities: the number of calls made to the oracle, denoted by $\omega_l(n)$; the number of Radon partitions computed, denoted by $\rho_l(n)$; and the total number of Radon intersection points which need be accommodated in the arrays $A_0, A_1, A_2, \ldots, A_l$ at any given time, denoted by $\sigma_l(n)$. ¿From the discussion of Section 2.1,

$$\begin{aligned}\omega_0(n) &= \binom{n}{d+1},\\ \rho_0(n) &= \binom{n-1}{d+1}, \text{ and}\\ \sigma_0(n) &= \binom{n-1}{d}.\end{aligned}$$

3.1 Space

Let us assume that l is taken to be fixed, and that the choice of m is to depend on that of l. At each of the first l levels of the recursion, the array which holds the Radon intersection points has at most $\binom{m-1}{d}$ entries. Noting that at every recursive level the size of the input is cut by a factor of $\frac{d+1}{m}$, if one wishes that the array at the bottom level also have at most $\binom{m-1}{d}$ entries, m must obey the restriction

$$\left(\frac{d+1}{m}\right)^l n \leq m.$$

Clearly then, to minimize the total storage used, the minimum value of m satisfying the above inequality should be chosen. The total storage required for Radon intersection points would then be

$$\sigma_l(n) \leq (l+1)\binom{m-1}{d},$$

subject to

$$(m-1)^{l+1} < (d+1)^l n \leq m^{l+1}.$$

In terms of n and l, the bound on the storage becomes

$$\begin{aligned}\sigma_l(n) &\leq (l+1)\frac{(m-1)^d}{d!}\\ \sigma_l(n) &\leq (l+1)\frac{\left[(d+1)^l n\right]^{\frac{d}{l+1}}}{d!}\\ \sigma_l(n) &\leq (l+1)\frac{(d+1)^{\frac{ld}{l+1}}}{d!}n^{\frac{d}{l+1}}.\end{aligned}$$

In this manner, by picking an appropriate value of l, the space required by Algorithm MRadon can be drastically reduced; in particular, if $l = d - 1$, linear space suffices for the Radon intersection points.

3.2 Time

In the case where $l > 0$, the set is partitioned into m groups, and each combination of $d+1$ of these groups is treated in a recursive manner. Since the size of each combination is $\frac{d+1}{m}n$, the number of calls to the oracle is simply

$$\omega_l(n) = \binom{m}{d+1}\omega_{l-1}\left(\frac{d+1}{m}n\right).$$

Since m and l have been chosen such that no more than m sets are handled at any given stage in the recursive algorithm,

$$\binom{m}{d+1}^l \le \omega_l(n) \le \binom{m}{d+1}^{l+1}. \tag{2}$$

A similar relation holds for the number of Radon partitions computed. In addition to those computed at deeper levels of the recursion, $\binom{m-1}{d+1}$ partitions are computed in the loops of Step 5: the analysis is the same as that for $\rho_0(n)$ in Section 2.1, with m substituted for n. This leads to the recurrence

$$\rho_l(n) = \binom{m}{d+1}\rho_{l-1}\left(\frac{d+1}{m}n\right) + \binom{m-1}{d+1},$$

and the closed-form equation

$$\rho_l(n) \quad \le \quad \binom{m-1}{d+1}\frac{\binom{m}{d+1}^{l+1} - 1}{\binom{m}{d+1} - 1}.$$

¿From this, it is quite clear that $\rho_l(n)$ and $\omega_l(n)$ are of the same order of magnitude, and as the bound for $\omega_l(n)$ is the looser of the two in absolute terms, we shall focus our attention on this quantity.

The bound on $\sigma_l(n)$ suggests that m should be made as small as possible, if one wishes to reduce the storage requirements of the algorithm. If m is greedily chosen to be a small constant value (greater than $d+1$), one can readily show that the time required would no longer be $O(n^{d+1})$. On the other hand, if the number of levels l is fixed, and m is allowed to vary, the situation is quite different.

For a fixed l, we choose m such that

$$(m-1)^{l+1} < (d+1)^l n \le m^{l+1}.$$

Under these assumptions, and using the fact that $m-2 \geq d \geq 2$ and that $m(m-2) < (m-1)^2$, we obtain

$$
\begin{aligned}
\omega_l\,(n) &\leq \binom{m}{d+1}^{l+1} \\
&\leq \frac{\prod_{i=0}^{d}(m-i)^{l+1}}{(d+1)!^{l+1}} \\
&\leq \frac{\prod_{i=0}^{d}(m-i)(m-1)^{l}}{(d+1)!^{l+1}} \\
&\leq \frac{(m-1)^{l+1}\left[(m-1)^{2l+2}-(m-1)^{2l}\right]\prod_{i=3}^{d}\left[(m-1)^{l+1}-(i-1)(m-1)^{l}\right]}{(d+1)!^{l+1}} \\
&\leq \frac{(m-1)^{l+1}\left[(m-1)^{2l+2}-(d+1)^{2l}\right]\prod_{i=3}^{d}\left[(m-1)^{l+1}-(i-1)(d+1)^{l}\right]}{(d+1)!^{l+1}} \\
&\leq \frac{\prod_{i=0}^{d}\left[(m-1)^{l+1}-(i-1)(d+1)^{l}\right]}{(d+1)!^{l+1}} \\
&\leq \frac{\prod_{i=0}^{d}\left[(d+1)^{l}n-(i-1)(d+1)^{l}\right]}{(d+1)!^{l+1}} \\
&\leq \left[\frac{(d+1)^{d}}{d!}\right]^{l}\binom{n+1}{d+1} \\
&\leq \left[\frac{(d+1)^{d}}{d!}\right]^{l}\omega_0\,(n+1)\,.
\end{aligned}
$$

The significance of this bound is that if the number of levels of the recursion is fixed at l, then the number of calls to the oracle (and the number of Radon partitions computed) increases by only a constant. In fact, a similar (but more tedious) argument shows that even if m is chosen independently of l, or if it is allowed to vary from one level of the recursion to another, the same constant-time increase is evidenced.

3.3 Summary

Before summarizing the analysis in the form of a theorem, the overhead associated with each recursive call must be taken into account. A very loose bound on the amount of storage required for pointers to sets is $O(ln)$, which is linear if l is constant. An even looser bound on the time required to pass these pointers is $l \cdot \omega_l\,(n)$, since a pointer can be passed down through at most l recursive levels before finally reaching the oracle. As the cost of passing a pointer is negligible compared to that of accessing the oracle, one may ignore the overhead associated with either of two parameter-passing mechanisms described at the end of Section 2.2.

Theorem 3 *For any choice of l between 0 and $d-1$, Algorithm MRadon can be made to run in*

$$\left[\frac{(d+1)^d}{d!}\right]^l \binom{n+1}{d+1}$$

time and $O(n^{\frac{d}{l+1}})$ space. The time is measured in terms of the number of calls to the oracle, and the number of Radon partitions computed.

In practice, the best strategy is of course to choose the minimum possible value of l for which the available space suffices. In higher-dimensional settings this may seem absurd, since the time complexity may be much too high to allow the program to run regardless of how much space is available. Even in lower dimensions, the time bound is more of a restriction than the space bound, and setting l to be any greater than 1 is of little or no benefit. Yet in the two-dimensional setting, setting l to 1 instead of 0 allows the space complexity to be reduced from $O(n^2)$ to $O(n)$, with an increase in time by a factor of only slightly more than $\frac{9}{2}$.

4 Applications

Just as many combinatorial results follow from Helly's theorem, the method presented in the previous section has many applications. The first follows from an application of Helly's theorem due to Vincensini [11] and Klee [6]:

Theorem 4 (Vincensini, Klee) *Let $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ be a collection of convex sets in $\Re^d$, and let C_0 be another convex set. Then the existence of some translate of C_0 which intersects (is contained in; contains) all members of $\mathcal{C}$ is guaranteed by the the existence of such a translate for each $d+1$ members of $\mathcal{C}$.*

The proof of this theorem is simply stated, and provides the key to a constructive algorithm:

Proof [2] For each $C_i \in \mathcal{C}$, let C_i' be the convex set $\{x \in \Re^d | (x + C_0) \mathbin{?} C_i\}$, where ? means "intersects" or "is contained in" or "contains". The hypotheses of the theorem imply that every $d+1$ of the sets C_i' have a common point. By Helly's theorem, all of the sets C_i' have a common point z, and thus the translate $(z + C_0) \mathbin{?} C_i$ for all $C_i \in \mathcal{C}$. □

Since $d+1$ of the sets C_i' have a common point t, then the corresponding sets of $\mathcal{C}$ intersect, contain, or are contained by $t + C_0$ (as appropriate). The value z can be obtained from the values t using Algorithm MRadon.

Theorem 5 *Let $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ be a collection of convex sets in $\Re^d$, and let C_0 be another set. Given an oracle that accepts as input $d+1$ sets of $\mathcal{C}$, and returns a translate of C_0 intersecting (contained in; containing) these sets (or reports its non-existence), then a translate of C_0 intersecting (contained in; containing) all members of $\mathcal{C}$ may be found (or its non-existence reported) in time proportional to $\binom{n}{d+1} \cdot T$, where T is the time taken by the oracle.*

Next, let $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ be a family of n sets in $\Re^d$, some labeled *red* and the others labeled *green*. A *linear separator* of $\mathcal{S}$ is a hyperplane h such that all the red sets are contained in one open half-space bounded by h, and all the green sets are contained in the other. A *spherical separator* of $\mathcal{S}$ is a hypersphere s (possibly degenerate) such that all the red sets are contained in one connected region of $\Re^d \setminus s$, and all the green sets are contained in the other.

In a refinement of a theorem due to Kirchberger [5, 8], Houle [4] showed a reduction of the problem of determining whether the sets of $\mathcal{S}$ can be linearly separated to the problem of determining whether n convex sets of the same dimension have a common intersection point. The proof relied on a simple transformation mapping red sets of points S_i into convex sets

$$C_i = \bigcap_{a \in S_i} \{x \in \Re^d |\, a \cdot x > 1\},$$

and green sets of points S_j into convex sets

$$C_i = \bigcap_{a \in S_i} \{x \in \Re^d |\, a \cdot x < 1\}.$$

Hyperplanes of the form $h_b = \{x \in \Re^d |\, b \cdot x = 1\}$ were mapped to the point b. Under this transformation, if the collection of sets $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ have a common intersection point b, then the hyperplane h_b is a linear separator for $\mathcal{S}$ — so long as b is guaranteed not to be the origin. This can be assured by translating the sets of $\mathcal{S}$ such that a point from one of the green sets (say, S_1) coincides with the origin.

The immediate implication of the transformation is that any oracle which

- accepts as input S_1 together with $d+1$ other sets of $\mathcal{S}$, and
- outputs a linear separator for the sets (or reports its non-existence)

is equivalent under the transformation to an oracle which

- accepts C_1 together with $d+1$ other sets of $\mathcal{C}$, and
- outputs a point in the common intersection of the sets (or reports its non-existence).

By applying Algorithm MRadon to the dual points associated with hyperplanes returned by the oracle for linear separation, the dual point associated with a linear separator of S may be found.

Theorem 6 *Given an oracle that accepts as input $d+2$ sets of $\mathcal{S}$, and returns a linear separator of this subset (or reports its non-existence), a linear separator for the entire collection may be found (or its non-existence reported) in time proportional to $\binom{n}{d+1} \cdot T$, where T is the time taken by the oracle.*

In the same paper [4], a similar refinement of a theorem due to Lay [7] is given, in which a correspondance is shown between the problem of determining whether the sets of $\mathcal{S}$ can be spherically separated and the problem of determining whether n convex sets in $\Re^{d+1}$ have a common intersection point. The former problem is transformed to the latter by means of a stereographic projection of the points of $\Re^d$ onto a hypersphere in $\Re^{d+1}$. As in the previous examples, Algorithm MRadon may be applied in the dual setting:

Theorem 7 *Given an oracle that accepts as input $d+3$ sets of $\mathcal{S}$, and returns a spherical separator of this subset (or reports its non-existence), a spherical separator for the entire collection may be found (or its non-existence reported) in time proportional to $\binom{n}{d+2} \cdot T$, where T is the time taken by the oracle.*

5 Concluding Remarks

Although this method perhaps cannot compete against fast algorithms for collections of special sets such as half-spaces, it is nevertheless the first algorithmic application of Helly's Theorem.

In many cases, a more practical alternative to the algorithm outlined in this paper may be applied. Recently, Amenta [1] showed that certain Helly-type theorems can be related to classes of General Linear Programming (GLP) problems, as defined by Matoušek, Sharir, and Welzl [9]. When the dimension d is fixed, the algorithm outlined in [9] constructs a solution in $O(n)$ expected time. If d is not considered to be fixed, the time complexity is subexponential in d.

Amente's approach requires that an objective function satisfying certain locality and monotonicity conditions be defined on the sets of the collection. If the collection of convex sets does not admit such a function, or if such a function cannot be found, the GLP algorithm cannot be applied. The advantage of our approach is that a solution is guaranteed for all possible collections of convex sets.

References

[1] Amente, N. Helly Theorems and Generalized Linear Programming, in *Proc. 9th ACM Symposium on Comput. Geom.*, San Diego, 1993, pp. 63-72.

[2] Danzer, L., Grünbaum, B. and Klee, V. Helly's Theorem and its Relatives, in *Convexity, AMS Proc. Symp. Pure Math.* **7**, 1963, pp. 101-181.

[3] Helly, E. Über Mengen konvexer Körper mit gemeinschaftlichen Punkten, *Jber. Deutsch. Math.-Verein.* **32**, 1923, pp. 175-176.

[4] Houle, M. E. Theorems on the Existence of Separating Surfaces, *Discrete Comput. Geom.* **6**, 1991, pp. 49-56.

[5] Kirchberger, P. Über Tchebychefsche Annäherungsmethoden, *Math. Ann.* **57**, 1903, pp. 509-540.

[6] Klee, V. The Critical Set of a Convex Body, *Amer. J. Math.* **75**, 1953, 178-188.

[7] Lay, S. R. On Separation by Spherical Surfaces, *Amer. Math. Monthly* **78**, 1971, pp. 1112-1113.

[8] Lay, S. R. *Convex Sets and Their Applications*, John Wiley and Sons, New York, 1982.

[9] Matoušek, J., Sharir, M. and Welzl, E. A Subexponential Bound for Linear Programming, in *Proc. 8th ACM Symposium on Comput. Geom.*, Berlin, 1992, pp. 1-8.

[10] Radon, J. Mengen konvexer Körper, die einen gemeinsamen Punkt enthalten, *Math. Ann.* **83**, 1921, pp. 113-115.

[11] Vincensini, P. Sur une extension d'un théorème de M. J. Radon sur les ensembles de corps convexes, *Bull. Soc. Math. France* **67**, 1939, pp. 115-119.