Available online at www.sciencedirect.com

ScienceDirect

Discrete Applied Mathematics 156 (2008) 2124–2141

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

Violator spaces: Structure and algorithms[☆]

B. Gärtner[a], J. Matoušek[b, c], L. Rüst[a], P. Škovroň[b]

[a] *Institute of Theoretical Computer Science, ETH Zürich, 8092 Zürich, Switzerland*
[b] *Department of Applied Mathematics, Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic*
[c] *Institute of Theoretical Computer Science, Charles University, 118 00 Praha 1, Czech Republic*

Received 10 November 2005; received in revised form 25 May 2007; accepted 12 August 2007
Available online 22 October 2007

Abstract

Sharir and Welzl introduced an abstract framework for optimization problems, called *LP-type problems* or also *generalized linear programming problems*, which proved useful in algorithm design. We define a new, and as we believe, simpler and more natural framework: *violator spaces*, which constitute a proper generalization of LP-type problems. We show that Clarkson's randomized algorithms for low-dimensional linear programming work in the context of violator spaces. For example, in this way we obtain the fastest known algorithm for the P-*matrix generalized linear complementarity problem* with a constant number of blocks. We also give two new characterizations of LP-type problems: they are equivalent to *acyclic* violator spaces, as well as to *concrete* LP-type problems (informally, the constraints in a concrete LP-type problem are subsets of a linearly ordered ground set, and the value of a set of constraints is the minimum of its intersection).
© 2007 Elsevier B.V. All rights reserved.

Keywords: LP-type problem; Generalized linear programming; Violator space; Clarkson's algorithms; Unique sink orientation; Generalized linear complementarity problem

1. Introduction

The framework of LP-type problems, invented by Sharir and Welzl in 1992 [27], has become a well-established tool in the field of geometric optimization. Its origins are in linear programming: Sharir and Welzl developed a randomized variant of the dual simplex algorithm for linear programming and showed that this algorithm actually works for a more general class of problems they called LP-type problems.

For the theory of linear programming, this algorithm constituted an important progress, since it was later shown to be *subexponential* in the RAM model [20]. Together with a similar result independently obtained by Kalai [17], this was the first linear programming algorithm provably requiring a number of arithmetic operations subexponential in the dimension and number of constraints (independent of the precision of the input numbers).

For many other geometric optimization problems in fixed dimension, the algorithm by Sharir and Welzl was the first to achieve expected linear runtime, simply because these problems could be formulated as LP-type problems. The class

[☆] The first and the third author acknowledge support from the Swiss Science Foundation (SNF), Project no. 200021-100316/1. The fourth author acknowledges support from the Czech Science Foundation (GACR), Grant no. 201/05/H014. An extended abstract of this paper has appeared in *Proceedings of the 14th Annual European Symposium on Algorithms* (*ESA*), pp. 387–398.

E-mail addresses: gaertner@inf.ethz.ch (B. Gärtner), matousek@kam.mff.cuni.cz (J. Matoušek), ruestle@inf.ethz.ch (L. Rüst), xofon@kam.mff.cuni.cz (P. Škovroň).

0166-218X/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2007.08.048

of LP-type problems for example includes the problem of computing the minimum-volume ball or ellipsoid enclosing a given point set in $\mathbf{R}^d$, and the problem of finding the distance of two convex polytopes in $\mathbf{R}^d$. Many other problems have been identified as LP-type problems over the years [20,2,3,5,16].

Once it is shown that a particular optimization problem is an LP-type problem, and certain algorithmic primitives are implemented for it, several efficient algorithms are immediately at our disposal: the Sharir–Welzl algorithm, two other randomized optimization algorithms due to Clarkson [8] (see [14,7] for a discussion of how it fits the LP-type framework), a deterministic version of it [7], an algorithm for computing the minimum solution that violates at most k of the given n constraints [18], and probably more are to come in the future.

The framework of LP-type problems is not only a prototype for concrete optimization problems; it also serves as a mathematical tool by itself, in algorithmic [15,6] and nonalgorithmic contexts [4].

An (abstract) LP-type problem is given by a finite set H of *constraints* and a *value* $w(G)$ for every subset $G \subseteq H$. The values can be real numbers or, for technical convenience, elements of any other linearly ordered set. Intuitively, $w(G)$ is the minimum value of a solution that satisfies all constraints in G. The assignment $G \mapsto w(G)$ has to obey the axioms in the following definition.

Definition 1. An *abstract LP-type problem* is a quadruple $(H, w, W, \leqslant)$, where H is a finite set, W is a set linearly ordered by $\leqslant$, and $w : 2^H \to W$ is a mapping satisfying the following two conditions:
Monotonicity: For all $F \subseteq G \subseteq H$ we have $w(F) \leqslant w(G)$, and
Locality: For all $F \subseteq G \subseteq H$ and all $h \in H$ with $w(F) = w(G)$ and $w(G) < w(G \cup \{h\})$, we have $w(F) < w(F \cup \{h\})$.

As our running example, we will use the smallest enclosing ball problem, where H is a finite point set in $\mathbf{R}^d$ and $w(G)$ is the radius of the smallest ball that encloses all points of G. In this case monotonicity is obvious, while verifying locality requires the nontrivial but well-known geometric result that the smallest enclosing ball is unique for every set.

It seems that the order $\leqslant$ of subsets is crucial; after all, LP-type problems model *optimization problems*, and indeed, the subexponential algorithm for linear programming and other LP-type problems [20] heavily relies on such an order.

A somewhat deeper look reveals that often, we only care whether two subsets have the *same* value, but not how they compare under the order $\leqslant$. The following definition is taken from [27]:

Definition 2. Consider an abstract LP-type problem $(H, w, W, \leqslant)$. We say that $B \subseteq H$ is a *basis* if for all proper subsets $F \subset B$ we have $w(F) \neq w(B)$. For $G \subseteq H$, a *basis of* G is an inclusion-minimal subset B of G with $w(B) = w(G)$.

We observe that a minimal subset $B \subseteq G$ with $w(B) = w(G)$ is indeed a basis.

Solving an abstract LP-type problem $(H, w, W, \leqslant)$ means to find a basis of H. In the smallest enclosing ball problem, a basis of H is a minimal set B of points such that the smallest enclosing ball of B has the same radius (and is in fact the same) as the smallest enclosing ball of H, $w(B) = w(H)$.

In defining bases, and in saying what it means to solve an LP-type problem, we therefore do not need the order $\leqslant$. The main contribution of this paper is that many of the things one can prove about LP-type problems do not require a concept of order.

We formalize this by defining the new framework of *violator spaces*. Intuitively, a violator space is an LP-type problem without order. This generalization of LP-type problems is proper, and we can exactly characterize the violator spaces that “are” LP-type problems. In doing so, we also establish yet another equivalent characterization of LP-type problems that is closer to the applications than the abstract formulation of Definition 1. In a concrete LP-type problem, the constraints are not just elements of a set, but they are associated with subsets of some linearly ordered ground set X, with the minimal elements in the intersections of such subsets corresponding to “solutions”. The framework of concrete LP-type problems is similar to the model presented in [3] as a mathematical programming problem, with a few technical differences. Additional results concerning violator spaces can be found in [28,29].

These are our main findings on the structural side. Probably the most surprising insight on the algorithmic side is that Clarkson’s algorithms [8] work for violator spaces of fixed dimension, leading to an expected linear-time algorithm for “solving” the violator space. Clarkson’s algorithms were originally developed for linear programs with small dimension. They can be generalized for LP-type problems [14,7]. The fact that the scheme also works for violator spaces may come as a surprise since the structure of violator spaces is not acyclic in general (in contrast to LP-type problems).

Actually, it turns out that problems effectively solvable by Clarkson's algorithms are in a well-defined sense exactly the violator spaces [29]; see Proposition 23 for a precise formulation.

We give an application of Clarkson's algorithms in this more general setting by linking our new violator space framework to well-known abstract and concrete frameworks in combinatorial optimization. For this, we show that any *unique sink orientation* (USO) of the cube [30,12,23,19,24,11,21,25,26,13] or more generally, of the grid [12] gives rise to a violator space, but not to an LP-type problem in general. Grid USO capture some important problems like linear programming over products of simplices or *generalized linear complementarity problems* (GLCPs) over P-matrices [12].

We show that we can find the sink in a USO by solving the violator space, for example with Clarkson's algorithms. A concrete new result is obtained by applying this to P-matrix GLCPs. These problems are not known to be polynomial-time solvable, but NP-hardness would imply NP = co-NP [22,12]. Since any P-matrix GLCP gives rise to a USO [12], we may use violator spaces and Clarkson's algorithms to solve the problem in expected linear time in the (polynomially solvable) case of a *fixed* number of *blocks*. This is optimal and beats all previous algorithms; see Section 5. We are confident that more applications of violator spaces that are not subsumed by the LP-type framework will be discovered in the future.

The rest of the paper is organized as follows. In Section 2, we formally define the frameworks of concrete LP-type problems and violator spaces, along with their essential terminology. Then we state our main structural result.

In Section 3, we prove this result by deriving the equivalence of abstract and concrete LP-type problems, and of *acyclic* violator spaces.

Section 4 shows that Clarkson's algorithms work for (possibly cyclic) violator spaces. Section 5, finally, shows how USOs induce violator spaces. An USO can be cyclic, and a cyclic orientation gives rise to a cyclic violator space. USOs are therefore nontrivial examples of possibly cyclic violator spaces.

2. Structural results

2.1. Concrete LP-type problems

Although intuitively one thinks about $w(G)$ as the value of an optimal solution of an optimization problem, the solution itself is not explicitly represented in Definition 1. In specific geometric examples, the constraints can usually be interpreted as a subset of some ground set X of points, and the optimal solution for G is the point with the smallest value in the intersection of all constraints in G. For example, in linear programming, the constraints are halfspaces, the value is given by the objective function, and the optimum is the point with minimum value in the admissible region, i.e., the intersection of the halfspaces. In order to have a unique optimum for every set of constraints (which is needed for w to define an LP-type problem because the set W needs to be linearly ordered), one assumes that the points are linearly ordered by the value; for linear programming, we can always take the lexicographically smallest optimal solution, for instance.

Such an interpretation is possible for the smallest enclosing ball problem too, although it looks a bit artificial. Namely, the "points" of X are all balls in $\mathbf{R}^d$, where the ordering can be an arbitrary linear extension of the partial ordering of balls by radius. The "constraint" for a point $h \in H$ is the set of all balls containing h.

The following definition captures this approach to LP-type problems.

Definition 3. A *concrete LP-type problem* is a triple $(X, \preccurlyeq, \mathscr{H})$, where X is a set linearly ordered by $\preccurlyeq$, $\mathscr{H}$ is a finite multiset whose elements are subsets of X, and for any $\mathscr{G} \subseteq \mathscr{H}$, if the intersection $\bigcap \mathscr{G} := \bigcap_{G \in \mathscr{G}} G$ is nonempty, then it has a minimum element with respect to $\preccurlyeq$ (for $\mathscr{G} = \emptyset$ we define $\bigcap \mathscr{G} := X$).

The definition allows $\mathscr{H}$ to be a multiset, i.e., a constraint set $A \subseteq X$ may be included several times. For example, in an instance of linear programming, some constraints can be the same, which we can reflect by this. In Section 3.4 we provide an example of an abstract LP-type problem, where the multiplicity is useful for representing it as a concrete LP-type problem.

Essentially the same model was introduced earlier by Amenta [3,1], who calls it a mathematical programming problem. The slight difference is that a mathematical programming problem allows several points to have the same value and the constraints form a set rather than a multiset.

Bases in concrete LP-type problems are defined in analogy to Definition 2.

Definition 4. Consider a concrete LP-type problem $(X, \preccurlyeq, \mathscr{H})$. We say that $\mathscr{B} \subseteq \mathscr{H}$ is a *basis* if for all proper submultisets $\mathscr{F} \subset \mathscr{B}$ we have $\min(\bigcap \mathscr{F}) \prec \min(\bigcap \mathscr{B})$. For $\mathscr{G} \subseteq \mathscr{H}$, a *basis of* $\mathscr{G}$ is a minimal $\mathscr{B} \subseteq \mathscr{G}$ with $\min(\bigcap \mathscr{B}) = \min(\bigcap \mathscr{G})$.

As before, a minimal $\mathscr{B} \subseteq \mathscr{G}$ with $\min(\bigcap \mathscr{B}) = \min(\bigcap \mathscr{G})$ is indeed a basis.

Given any concrete LP-type problem $\mathscr{P} = (X, \preccurlyeq, \mathscr{H})$, we obtain an abstract LP-type problem $P = (\mathscr{H}, w, X, \preccurlyeq)$ according to Definition 1 by putting $w(\mathscr{G}) = \min(\bigcap \mathscr{G})$ (or $w(\mathscr{G}) = +\infty$ if $\bigcap \mathscr{G}$ is empty), as is easy to check (some care is needed if $\mathscr{H}$ is a multiset; we come back to this issue shortly). It is clear that $\mathscr{B} \subseteq \mathscr{G}$ is a basis of $\mathscr{G}$ in P if and only if $\mathscr{B}$ is a basis of $\mathscr{G}$ in $\mathscr{P}$. We say that P is *basis-equivalent* to $\mathscr{P}$.

We find somewhat surprising the converse, which we prove below in Theorem 8: Any abstract LP-type problem $(H, w, W, \leqslant)$ has a “concrete representation”, that is, a concrete LP-type problem that is basis-equivalent to $(H, w, W, \leqslant)$. Amenta in her thesis [1] shows a method for converting a class of abstract LP-type problems into concrete LP-type problems: First she constructs from an abstract LP-type problem an object she calls a Helly-type theorem, and then she shows that every Helly-type theorem possessing a particular *intersectional representation* can be converted to a concrete LP-type problem. Our proof can be regarded as completing her programme by finding a suitable intersectional representation for *all* abstract LP-type problems.

Strictly speaking, if the multiset $\mathscr{H}$ in the concrete LP-type problem has elements with multiplicity bigger than 1, then $\mathscr{H}$ cannot be used as the set of constraints for the abstract LP-type problem (since it is not a set). However, we can bijectively map $\mathscr{H}$ to a set, i.e., we take any set H with $|H| = |\mathscr{H}|$ and a mapping $f: H \to \mathscr{H}$ such that for any $\bar{h} \in \mathscr{H}$, the number of elements $h \in H$ that map to $\bar{h}$ is equal to the multiplicity of $\bar{h}$. For $G \subseteq H$ we then define $w(G) = \min(\bigcap_{g \in G} f(g))$ which gives us a fair abstract LP-type problem $P = (H, w, X, \preccurlyeq)$ basis-equivalent to $\mathscr{P}$. In this case, by basis-equivalence we mean the existence of a suitable mapping f together with the condition that $B \subseteq G$ is a basis of G in P if and only if the multiset $\{f(b): b \in B\}$ is a basis of $\{f(g): g \in G\}$ in $\mathscr{P}$.

2.2. Violator spaces

Let $(H, w, W, \leqslant)$ be an abstract LP-type problem. It is natural to define that a constraint $h \in H$ *violates* a set $G \subseteq H$ of constraints if $w(G \cup \{h\}) > w(G)$. For example, in the smallest enclosing ball problem, a point h violates a set G if it lies outside of the smallest ball enclosing G (which is unique).

Definition 5. The *violator mapping* of $(H, w, W, \leqslant)$ is defined by $\mathsf{V}(G) = \{h \in H: w(G \cup \{h\}) > w(G)\}$. Thus, $\mathsf{V}(G)$ is the set of all constraints violating G.

It turns out that the knowledge of $\mathsf{V}(G)$ for all $G \subseteq H$ is sufficient to describe the “structure” of an LP-type problem. That is, while we cannot reconstruct W, $\leqslant$, and w from this knowledge, it is natural to regard two LP-type problems with the same mapping $\mathsf{V}: 2^H \to 2^H$ as isomorphic. Indeed, the algorithmic primitives needed for implementing the Sharir–Welzl algorithm and the other algorithms for LP-type problems mentioned above can be phrased in terms of testing violation (Does $h \in \mathsf{V}(G)$ hold for a certain set $G \subseteq H$?), and they never deal explicitly with the values of w.

We now introduce the notion of *violator space*:

Definition 6. A *violator space* is a pair (H, V), where H is a finite set and V is a mapping $2^H \to 2^H$ such that
Consistency: $G \cap \mathsf{V}(G) = \emptyset$ holds for all $G \subseteq H$, and
Locality: For all $F \subseteq G \subseteq H$, where $G \cap \mathsf{V}(F) = \emptyset$, we have $\mathsf{V}(G) = \mathsf{V}(F)$.

A basis of a violator space is defined in analogy to a basis of an LP-type problem.

Definition 7. Consider a violator space (H, V). We say that $B \subseteq H$ is a *basis* if for all proper subsets $F \subset B$ we have $B \cap \mathsf{V}(F) \neq \emptyset$. For $G \subseteq H$, a *basis of* G is a minimal subset B of G with $\mathsf{V}(B) = \mathsf{V}(G)$.

Observe that a minimal subset $B \subseteq G$ with $\mathsf{V}(B) = \mathsf{V}(G)$ is indeed a basis: Assume for contradiction that there is a set $F \subset B$ such that $B \cap \mathsf{V}(F) = \emptyset$. Locality then yields $\mathsf{V}(B) = \mathsf{V}(F) = \mathsf{V}(G)$, which contradicts minimality of B.

We will check in Section 3.2 that the violator mapping of an abstract LP-type problem satisfies the two axioms above. Consistency is immediate: since $w(G) = w(G \cup \{h\})$ for $h \in G$, no element in G violates G. The locality condition has the following intuitive interpretation: adding only nonviolators to a set does not change the value.

We actually show more: given an abstract LP-type problem $(H, w, W, \leqslant)$, the pair (H, V), with V being the violator mapping, is an *acyclic* violator space. (Acyclicity of a violator space will be defined later in Definition 10.) It turns out in Section 3.3 that acyclicity already characterizes the violator spaces obtained from LP-type problems, and thus any acyclic violator space can be represented as an LP-type problem (abstract or concrete). These equivalences are stated in our main theorem.

Theorem 8. *The axioms of abstract LP-type problems, of concrete LP-type problems, and of acyclic violator spaces are equivalent. More precisely, every problem in one of the three classes has a basis-equivalent problem in each of the other two classes.*

The construction is illustrated on simple instances of problems of linear programming and the smallest enclosing ball in Section 3.4.

3. Equivalence of LP-type problems and acyclic violator spaces

In this section we prove Theorem 8.

3.1. Preliminaries on violator spaces

To show that every acyclic violator space (H, V) originates from some concrete LP-type problem, we need an appropriate linearly ordered set X of "points", and then we will identify the elements of H with certain subsets of X.

What set X will we take? Recall that for smallest enclosing balls, X is the set of all balls, and the subset for $h \in H$ is the subset of balls containing h. It is not hard to see that we may restrict X to smallest enclosing balls of *bases*; in fact, we may choose X as the set of bases, in which case the subset for h becomes the set of bases not violated by h.

This also works for general acyclic violator spaces, with bases suitably ordered. The only blemish is that we may get several minimal bases for $G \subseteq H$; for smallest enclosing balls, this corresponds to the situation in which several bases define the same smallest enclosing ball. To address this, we will declare such bases as equivalent and choose X as the set of all equivalence classes instead.

In the following, we fix a violator space (H, V). The set of all bases in (H, V) will be denoted by $\mathscr{B}$.

Definition 9. $B, C \in \mathscr{B}$ are *equivalent*, $B \sim C$, if $\mathsf{V}(B) = \mathsf{V}(C)$.

Clearly, the relation $\sim$ defined on $\mathscr{B}$ is an equivalence relation. The equivalence class containing a basis B will be denoted by $[B]$.

Now we are going to define an ordering of the bases, and we derive from this an ordering of the equivalence classes as well as the notion of acyclicity in violator spaces.

Definition 10. For $F, G \subseteq H$ in a violator space (H, V), we say that $F \leqslant_0 G$ (F is *locally smaller* than G) if $F \cap \mathsf{V}(G) = \emptyset$.

For equivalence classes $[B], [C] \in \mathscr{B}/\sim$, we say that $[B] \leqslant_0 [C]$ if there exist $B' \in [B]$ and $C' \in [C]$ such that $B' \leqslant_0 C'$.

We define the relation $\leqslant_1$ on the equivalence classes as the transitive closure of $\leqslant_0$. The relation $\leqslant_1$ is clearly reflexive and transitive. If it is antisymmetric, we say that the violator space is *acyclic*, and we define the relation $\leqslant$ as an arbitrary linear extension of $\leqslant_1$.

The intuition of the *locally smaller* notion comes from LP-type problems: if no element of F violates G, then $G \cup F$ has the same value as G (this is formally proved in Lemma 11), and monotonicity yields that valuewise, F is smaller than or equal to G.

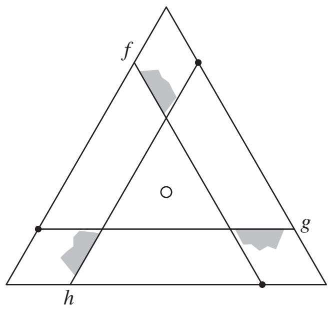

Fig. 1. A cyclic violator space.

Note that in the definition of $[B] \leqslant_0 [C]$ we do not require $B' \leqslant_0 C'$ to hold for *every* B' and C'. It may happen that $B' \nleqslant_0 C'$ for some bases B' and C'; however, $C' \leqslant_0 B'$ cannot hold, as can be easily checked. To show that acyclicity need not always hold, we conclude this section with an example of a cyclic violator space.

We begin with an intuitive geometric description; see Fig. 1. We consider a triangle without the center point. We say that one point of the triangle is "locally smaller" than another if it is farther clockwise with respect to the center. The constraints in our violator space are the three halfplanes f, g, h.

The locally smallest point within each halfplane is marked, and a halfplane violates a set of halfplanes if it does not contain the locally smallest point in their intersection.

Now we specify the corresponding violator space formally. We have $H = \{f, g, h\}$, and V is given by the following table:

G	$\emptyset$	f	g	h	f, g	f, h	g, h	f, g, h
$\mathsf{V}(G)$	f, g, h	h	f	g	h	g	f	$\emptyset$

This (H, V) is really a violator space, since we can easily check both consistency and locality. The bases are $\emptyset$, one-element sets, and H. We have $\{f\} \leqslant_0 \{h\} \leqslant_0 \{g\} \leqslant_0 \{f\}$, but none of the one-element bases are equivalent; i.e., $\leqslant_1$ is not antisymmetric.

3.2. Abstract LP-type problems yield acyclic violator spaces

In this subsection, we show that the violator mapping of an abstract LP-type problem is an acyclic violator space. To this end, we need the following two lemmas.

Lemma 11. *Consider an abstract LP-type problem $(H, w, W, \leqslant)$ with violator mapping V. Let $A, B \subseteq H$, where B is not violated by any $h \in A$ (i.e., $A \cap \mathsf{V}(B) = \emptyset$). Then $w(A \cup B) = w(B)$.*

Proof. From monotonicity, we immediately obtain the inequality "$\geqslant$". The inequality "$\leqslant$" can be shown by induction on $|A|$. If $|A| = 1$, i.e., $A = \{h\}$, then $w(B \cup \{h\}) > w(B)$ would imply that B is violated by $h \in A$, a contradiction.

Let $|A| > 1$ and $A = A_0 \dot{\cup} \{h\}$ (disjoint union). From the induction hypothesis we have $w(B \cup A_0) = w(B)$. Now, if $w(B \cup A_0) < w(B \cup A_0 \cup \{h\})$, then by locality (for $B \cup A_0$, B and h) we get $w(B) < w(B \cup \{h\})$. This means that $h \in \mathsf{V}(B)$, and since $h \in A$ we have $h \in A \cap \mathsf{V}(B)$, a contradiction. So $w(B) = w(B \cup A_0) \geqslant w(B \cup A_0 \cup \{h\}) = w(B \cup A)$. We have proved $w(A \cup B) \leqslant w(B)$. $\square$

Lemma 12. *Consider an abstract LP-type problem $(H, w, W, \leqslant)$ with violator mapping V. Then for any $A, B \subseteq H$ with $\mathsf{V}(A) = \mathsf{V}(B)$ we have $w(A) = w(B)$. Conversely, $w(A) = w(B) = w(A \cup B)$ implies $\mathsf{V}(A) = \mathsf{V}(B)$. In particular, if $A \subseteq B$ and $w(A) = w(B)$, then $\mathsf{V}(A) = \mathsf{V}(B)$.*

Note that the condition $w(A) = w(B)$ generally does not suffice for $\mathsf{V}(A) = \mathsf{V}(B)$. For example, having any H, we can define w by $w(G) = |G|$ for all $G \subseteq H$ (it can be checked that it is an abstract LP-type problem). Then any G's of

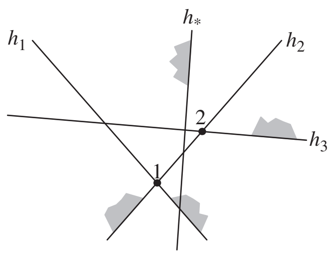

Fig. 2. A linear programming example ($F = \{h_1, h_2\} \subseteq G = \{h_1, h_2, h_3\}$ with $\mathsf{V}(G) \not\subseteq \mathsf{V}(F)$).

the same size have the same w, however, $\mathsf{V}(G) = H \backslash G$, and so no distinct G's share the value of V. Roughly speaking, the equality $w(A) = w(B)$ may hold just "by accident". This is one way in which we can see that w by itself does not reflect the combinatorial structure of the problem in a natural way.

Proof of Lemma 12. Let $w(A) \neq w(B)$. Without loss of generality we assume $w(A) > w(B)$ (note that here we use the linearity of the ordering $\leqslant$). If $A \cap \mathsf{V}(B) = \emptyset$, from Lemma 11 we would get $w(A \cup B) = w(B)$, which contradicts $w(A \cup B) \geqslant w(A) > w(B)$. So there necessarily exists $h \in A \cap \mathsf{V}(B)$, but since $h \in A$, we have $h \notin \mathsf{V}(A)$. So $\mathsf{V}(A) \neq \mathsf{V}(B)$.

Conversely, suppose $w(A) = w(B) = w(A \cup B)$. We want to show $\mathsf{V}(A) = \mathsf{V}(B)$, i.e., that $w(A) < w(A \cup \{h\})$ holds if and only if $w(B) < w(B \cup \{h\})$ holds. By symmetry, it suffices to show only one of the implications. We assume $w(A) < w(A \cup \{h\})$. Then $w(A \cup B) = w(A) < w(A \cup \{h\}) \leqslant w(A \cup B \cup \{h\})$. Since $B \subseteq A \cup B$ and $w(B) = w(A \cup B)$, we may use locality, which gives $w(B) < w(B \cup \{h\})$. So the desired equivalence holds. □

Proposition 13. *Consider an abstract LP-type problem* $(H, w, W, \leqslant)$, *and let* V *be its violator mapping. Then* (H, V) *is an acyclic violator space. Moreover*, (H, V) *is basis-equivalent to* $(H, w, W, \leqslant)$.

Proof. Clearly $G \cap \mathsf{V}(G) = \emptyset$, since $w(G \cup \{g\}) = w(G)$ for any $g \in G$, so consistency holds. If $G \cap \mathsf{V}(F) = \emptyset$ for $F \subseteq G$, then by Lemma 11 we get $w(F \cup G) = w(F)$. Since $F \subseteq G$, we have $F \cup G = G$ and so $w(G) = w(F)$. Lemma 12 then yields $\mathsf{V}(G) = \mathsf{V}(F)$, so locality holds.

We proceed to prove acyclicity of (H, V). Fix $[B]$ and $[C]$, $[B] \neq [C]$, with $[B] \leqslant_0 [C]$, that is $B' \cap \mathsf{V}(C') = \emptyset$ for some $B' \in [B]$ and $C' \in [C]$. Lemma 11 implies $w(C') = w(B' \cup C')$. For contradiction, assume $w(B') \geqslant w(C')$; then $w(B') \geqslant w(B' \cup C')$ which with monotonicity yields $w(B') = w(B' \cup C') = w(C')$. Lemma 12 gives $\mathsf{V}(B') = \mathsf{V}(C')$, a contradiction to $[B] \neq [C]$. Thus $[B] \leqslant_0 [C]$ for $[B] \neq [C]$ implies $w(B') < w(C')$ for some bases B' and C' out of the respective equivalence classes. By Lemma 12, $w(B')$ is the same for all $B' \in [B]$ (because all bases in $[B]$ have the same violators). Therefore, by chaining several $\leqslant_0$'s we also get $w(B') < w(C')$ for $[B] \leqslant_1 [C]$. This proves that $\leqslant_1$ is necessarily antisymmetric (since $\leqslant$ is an ordering of W).

Finally, observe that by Lemma 12, $B \subseteq G$ is an inclusion-minimal subset of G with $w(B) = w(G)$ if and only if B is an inclusion-minimal subset of G with $\mathsf{V}(B) = \mathsf{V}(G)$. So, $B \subseteq G$ is a basis of G in $(H, w, W, \leqslant)$ if and only if B is a basis of G in (H, V). Thus (H, V) is basis-equivalent to $(H, w, W, \leqslant)$. □

At first glance, one might think that for $F \subseteq G$ we should have $\mathsf{V}(F) \supseteq \mathsf{V}(G)$. Unfortunately, this is not the case, as the linear programming example in Fig. 2 shows (the y-coordinate is to be minimized).

We put $F = \{h_1, h_2\}$ and $G = \{h_1, h_2, h_3\} \supseteq F$. Point 1 is minimum in the intersection of F, and 2 is minimum in the intersection of G. We have $1 \in h^*$, $2 \notin h^*$, and so $h^* \notin \mathsf{V}(F)$ and $h^* \in \mathsf{V}(G)$.

3.3. Acyclic violator spaces yield concrete LP-type problems

The following proposition is the last ingredient for Theorem 8.

Proposition 14. *Every acyclic violator space* (H, V) *can be represented as a concrete LP-type problem that is basis-equivalent to* (H, V).

Proof. We are given an acyclic violator space (H, V) and we define the mapping $S\colon H \to 2^{\mathscr{B}/\sim}$ that will act as a "concretization" of the constraints in H:

$$S(h) = \{[B]\colon B \in \mathscr{B},\ h \notin \mathsf{V}(B)\}.$$

Further, let $\mathscr{H}$ be the image of the mapping S taken as a multiset, i.e.,

$$\mathscr{H} = \{S(h)\colon h \in H\}.$$

Thus, S is a bijection between H and $\mathscr{H}$. By saying that a mapping S is a bijection between a set and a multiset we mean that for any $\bar{h} \in \mathscr{H}$, the number of $h \in H$ that map to $\bar{h}$ is equal to the multiplicity of $\bar{h}$. Note that we cannot use some of the common properties of set bijections; for instance we have to avoid using the inverse mapping S^{-1}.

Additionally, let σ be the induced bijection of 2^H and $2^{\mathscr{H}}$ defined by $\sigma(G) = \{S(h)\colon h \in G\}$, for $G \subseteq H$.

Now, consider the triple $(\mathscr{B}/\sim, \leqslant, \mathscr{H})$, where $\leqslant$ is an arbitrary linear extension of $\leqslant_1$ (such an extension exists since (H, V) is acyclic and so $\leqslant_1$ is antisymmetric). This is a concrete LP-type problem: The only thing to check is the existence of a minimal element of every nonempty intersection $\bigcap \mathscr{G}$ $(\mathscr{G} \subseteq \mathscr{H})$, which is guaranteed by the linearity of $\leqslant$ (remember from Definition 3 that $\bigcap \mathscr{G} := \bigcap_{G\in\mathscr{G}} G$).

It remains to prove basis-equivalence, which we do with the following two lemmas.

Lemma 15. *If B is an inclusion-minimal subset of G with $\mathsf{V}(B) = \mathsf{V}(G)$ in (H, V) (that is, B is a basis of G), then $\min(\bigcap \sigma(B)) = \min(\bigcap \sigma(G))$ in $(\mathscr{B}/\sim, \leqslant, \mathscr{H})$.*

Proof. It is clear that $[B] \in \bigcap \sigma(G)$. Therefore, showing that there is no other basis in $\bigcap \sigma(G)$ that is locally smaller than $[B]$ proves the lemma, because then $\min(\bigcap \sigma(G)) = [B] = \min(\bigcap \sigma(B))$ (the second equality holds since B is a basis of B; just replace G by B in the following proof). Assume for contradiction that a C with $[C] \neq [B]$, $[C] \in \bigcap \sigma(G)$ and $C \cap \mathsf{V}(B) = \emptyset$ exists. By $[C] \in \bigcap \sigma(G)$ we have $G \cap \mathsf{V}(C) = \emptyset$, which is equivalent to

$$(G \cup C) \cap \mathsf{V}(C) = \emptyset,$$

and $C \cap \mathsf{V}(B) = \emptyset$ is equivalent to (because B is a basis of G)

$$(G \cup C) \cap \mathsf{V}(B) = \emptyset.$$

Applying locality in (H, V) to these two equations tells us that $\mathsf{V}(C) = \mathsf{V}(B)$, a contradiction to $[C] \neq [B]$. □

Lemma 16. *If $\sigma(B)$ is an inclusion-minimal submultiset of $\sigma(G)$ with $\min(\bigcap \sigma(B)) = \min(\bigcap \sigma(G))$ in $(\mathscr{B}/\sim, \leqslant, \mathscr{H})$ (that is, $\sigma(B)$ is a basis of $\sigma(G)$), then $\mathsf{V}(B) = \mathsf{V}(G)$ in (H, V).*

Proof. Let A be a basis of B, so $\mathsf{V}(A) = \mathsf{V}(B)$. Note that $[A] \in \bigcap \sigma(B)$. Let $[C] = \min(\bigcap \sigma(B))$, thus $B \cap \mathsf{V}(C) = \emptyset$ and therefore also $A \cap \mathsf{V}(C) = \emptyset$. This means that $[A] \leqslant_0 [C]$ from which we conclude that $[A] = [C]$. From $\min(\bigcap \sigma(G)) = [C]$ we get $G \cap \mathsf{V}(C) = \emptyset$ which is equivalent to

$$G \cap \mathsf{V}(B) = \emptyset.$$

As $\sigma(B) \subseteq \sigma(G)$ if and only if $B \subseteq G$, we can apply locality and derive $\mathsf{V}(B) = \mathsf{V}(G)$ as needed. □

Lemmas 15 and 16 prove that (H, V) and $(\mathscr{B}/\sim, \leqslant, \mathscr{H})$ are basis-equivalent, in the sense that B is a basis of G in (H, V) if and only if $\sigma(B)$ is a basis of $\sigma(G)$ in $(\mathscr{B}/\sim, \leqslant, \mathscr{H})$: Starting with a basis B of G in (H, V), Lemma 15 yields $\min(\bigcap \sigma(B)) = \min(\bigcap \sigma(G))$. This $\sigma(B)$ is inclusion-minimal w.r.t. $\sigma(G)$, since otherwise Lemma 16 would yield a contradiction to the inclusion-minimality of B w.r.t. G (where we again use that $\sigma(B) \subseteq \sigma(G)$ if and only if $B \subseteq G$). The reasoning for inclusion-minimality in the other direction is analogous. This concludes the proof of Proposition 14. □

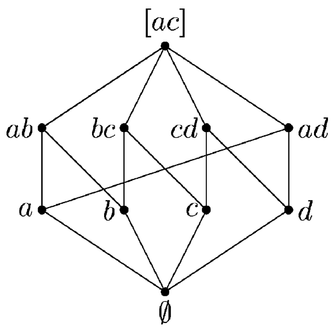

Fig. 3. The Hasse diagram of $\leqslant_1$ for the smallest enclosing circle problem on the vertices of a square.

Propositions 13 and 14, together with the fact that every concrete LP-type problem can be transformed into an abstract one (as described in Definition 4), yield Theorem 8.

3.4. Examples

Here we present some particular abstract LP-type problems and we demonstrate the construction (via acyclic violator spaces) of their concrete representations.

Let a, b, c and d be the vertices of a unit square (in the counterclockwise order); let $H = \{a, b, c, d\}$. For $G \subseteq H$ let $w(G)$ be the radius of the smallest circle enclosing all the points of G (for $G = \emptyset$ put $w(G) = -\infty$). The corresponding acyclic violator space is described by the following table:

G	$\emptyset$	a	b	c	d	ab	ac	ad
$\mathsf{V}(G)$	$abcd$	bcd	acd	abd	abc	cd	$\emptyset$	bc
G	bc	bd	cd	abc	abd	acd	bcd	$abcd$
$\mathsf{V}(G)$	ad	$\emptyset$	ab	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

The bases are $\emptyset$, a, b, c, d, ab, ac, ad, bc, bd, cd; the only equivalent pair is $ac \sim bd$. There is no inconvenience concerning differences between $\leqslant_0$ on sets and equivalence classes and $\leqslant_1$; the ordering $\leqslant_1$ is given by the Hasse diagram in Fig. 3.

As the linear extension $\leqslant$ of $\leqslant_1$ we may choose $\emptyset < a < b < c < d < ab < bc < cd < ad < [ac]$. Finally, the concrete representation S is as follows:

h	a	b	c	d
$S(h)$	$a, ab, ad, [ac]$	$b, ab, bc, [ac]$	$c, bc, cd, [ac]$	$d, cd, ad, [ac]$

In the geometric view that we have mentioned earlier, $S(a)$ corresponds to the set of all "canonical" (i.e., basic) balls that contain the point a (inside or on the boundary). The same holds for the other points.

As an additional example, consider the following LP problem in the positive orthant (rotated by 45° for convenience). Beside the restriction to the positive orthant, the constraints are the four halfplanes depicted in Fig. 4. The optimization direction is given by the arrow.

Here the violator space bases are $\emptyset$, a, b, c, d, ac, ad, bc, bd; the equivalence classes are $O = \emptyset$, $A = \{a\}$, $B = \{b\}$, $C = \{c\}$, $D = \{d\}$ and $Q = \{ac\} \sim \{ad\} \sim \{bc\} \sim \{bd\}$. Note that the equivalence classes correspond to points in the plane. We have $O \leqslant_1 B \leqslant_1 A \leqslant_1 Q$ and $O \leqslant_1 C \leqslant_1 D \leqslant_1 Q$; we choose $\leqslant$ to be $O < B < A < C < D < Q$. The concrete representation is

h	a	b	c	d
$S(h)$	A, Q	A, B, Q	C, D, Q	D, Q

Here we may interpret $S(a)$ as the set of all "canonical" points lying in the halfplane a.

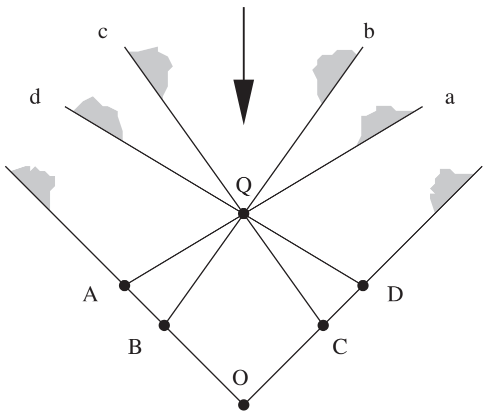

Fig. 4. Illustration example—linear programming.

To see why we allow $\mathscr{H}$ in the definition of a concrete LP-type problem to be a multiset, consider the abstract LP-type problem with $H = \{a, b\}$ and $w(G) = 0$ for every $G \subseteq H$. The only basis is $\emptyset$ and it is not violated by any $h \in H$. Thus we have $S(a) = S(b) = \{[\emptyset]\}$. If we do not allow $\mathscr{H}$ to be a multiset, we have $\mathscr{H} = \{S(a), S(b)\} = \{\{[\emptyset]\}\}$ with only one constraint; it seems improper to define this to be basis-equivalent to H. We could alter the construction of $\mathscr{H}$ and get $S(a) = \{0\}$, $S(b) = \{0, 1\}$, which does represent the original abstract LP-type problem with $\mathscr{H}$ being a set; however, we believe that our definition catches the structure in a more straightforward way, although it may seem unusual at first glance.

4. Clarkson's algorithms

We show that Clarkson's randomized reduction scheme, originally developed for linear programs with many constraints and few variables, actually works for general (possibly cyclic) violator spaces. The two algorithms of Clarkson involved in the reduction have been analyzed for LP and LP-type problems before [8,14,7]; the analysis we give below is almost identical on the abstract level. Our new contribution is that the combinatorial properties underlying Clarkson's algorithms also hold for violator spaces.

We start off by deriving these combinatorial properties; the analysis of Clarkson's reduction scheme is included for completeness.

4.1. Violator spaces revisited

We recall that an abstract LP-type problem is of the form $(H, w, W, \leqslant)$. In this subsection we will view a violator space as an "LP-type problem without the order $\leqslant$", i.e., we will only care whether two subsets F and G, $F \subseteq G \subseteq H$, have the same value (and therefore the same violators, see Lemma 12), but not how they compare under the order $\leqslant$. It turns out that the order is irrelevant for Clarkson's algorithms.

Even without an order, we can talk about monotonicity in violator spaces:

Lemma 17. *Any violator space* (H, V) *satisfies* monotonicity *defined as follows*:
Monotonicity: $\mathsf{V}(F) = \mathsf{V}(G)$ *implies* $\mathsf{V}(E) = \mathsf{V}(F) = \mathsf{V}(G)$ *for all sets* $F \subseteq E \subseteq G \subseteq H$.

Proof. Assume $\mathsf{V}(E) \neq \mathsf{V}(F), \mathsf{V}(G)$. Then locality yields $\emptyset \neq E \cap \mathsf{V}(F) = E \cap \mathsf{V}(G)$ which contradicts consistency. $\square$

Recall Definition 7: A basis is a set B satisfying $B \cap \mathsf{V}(F) \neq \emptyset$ for all proper subsets F of B. A basis of G is an inclusion-minimal subset of G with the same violators. This can be used to prove Observation 18, well-known to hold for LP-type problems [14].

Observation 18. *Let (H, V) be a violator space. For $R \subseteq H$ and all $h \in H$, we have*

(i) $\mathsf{V}(R) \neq \mathsf{V}(R \cup \{h\})$ *if and only if* $h \in \mathsf{V}(R)$, *and*
(ii) $\mathsf{V}(R) \neq \mathsf{V}(R \backslash \{h\})$ *if and only if* h *is contained in every basis of* R.

An element h such that (ii) *holds is called* extreme *in R.*

Proof. (i) If $h \notin \mathsf{V}(R)$, we get $\mathsf{V}(R) = \mathsf{V}(R \cup \{h\})$ by locality. If $h \in \mathsf{V}(R)$, then $\mathsf{V}(R) \neq \mathsf{V}(R \cup \{h\})$ is a consequence of consistency applied to $G = R \cup \{h\}$. (ii) if $\mathsf{V}(R) = \mathsf{V}(R\backslash\{h\})$, there is a basis B of $R\backslash\{h\}$, and this basis is also a basis of R not containing h. Conversely, if there is some basis B of R not containing h, then $\mathsf{V}(R) = \mathsf{V}(R\backslash\{h\})$ follows from monotonicity (Lemma 17). □

We are particularly interested in violator spaces with small bases.

Definition 19. Let (H, V) be a violator space. The size of a largest basis is called the *combinatorial dimension* $\delta = \delta(H, \mathsf{V})$ of (H, V).

Observation 18 implies that in a violator space of combinatorial dimension δ, every set has at most δ extreme elements. This in turn yields a bound for the *expected* number of violators of a random subset of constraints:

Lemma 20. *Let (H, V) be a violator space of combinatorial dimension δ and $W \subseteq H$ some fixed set. Let v_r be the expected number of violators of the set $W \cup R$, where $R \subseteq H\backslash W$ is a random subset of size $r < n = |H|$. Then*

$$v_r \leqslant \delta \frac{n-r}{r+1}.$$

The lemma can be proved using the *Sampling Lemma* of [15]. As suggested by a referee, we give an independent proof here.

Proof. By definition of v_r and Observation 18

$$\begin{aligned}\binom{|H\backslash W|}{r} v_r &= \sum_{\substack{R\subseteq H\backslash W\\|R|=r}} \sum_{h\in(H\backslash W)\backslash R} [h \text{ violates } W \cup R] \\ &= \sum_{\substack{R\subseteq H\backslash W\\|R|=r}} \sum_{h\in(H\backslash W)\backslash R} [h \text{ is extreme in } W \cup (R \cup \{h\})].\end{aligned}$$

Here $[A]$ is the *indicator variable* for the event A that has value 1 if A holds and 0 otherwise.

Substituting Q for $R \cup \{h\}$ and using the fact that any set has at most δ extreme elements, we can rewrite this as

$$\begin{aligned}\binom{|H\backslash W|}{r} v_r &= \sum_{\substack{Q\subseteq H\backslash W\\|Q|=r+1}} \sum_{h\in Q} [h \text{ is extreme in } W \cup Q] \\ &\leqslant \sum_{\substack{Q\subseteq H\backslash W\\|Q|=r+1}} \delta = \binom{|H\backslash W|}{r+1}\delta.\end{aligned}$$

This yields $v_r \leqslant \delta(|H\backslash W| - r)/(r+1)$, and the lemma follows. □

4.2. The trivial algorithm

Given a violator space (H, V) of combinatorial dimension δ, the goal is to find a basis of H. For this, we assume availability of the following primitive, which we use for sets G of size at most δ.

Primitive 21. *Given $G \subseteq H$ and $h \in H \setminus G$, decide whether $h \in \mathsf{V}(G)$.*

Given this primitive, a basis of H can be found in a brute-force manner by going through all sets of size at most δ, testing each of them for being a basis of H. More generally, $B \subseteq G$ is a basis of G if and only if

$$h \in \mathsf{V}(B \setminus \{h\}) \quad \text{for all } h \in B$$

and

$$h \notin \mathsf{V}(B) \quad \text{for all } h \in G \setminus B.$$

Consequently, the number of times the primitive needs to be invoked in order to find a basis of H ($|H| = n$) is at most

$$n \sum_{i=0}^{\delta} \binom{n}{i} = \mathrm{O}(n^{\delta+1}).$$

The next two subsections show that this can be substantially improved.

4.3. Clarkson's first algorithm

Fix a violator space (H, V) of combinatorial dimension δ, implicitly specified through Primitive 21. Clarkson's first algorithm calls Clarkson's second algorithm (`Basis2`) as a subroutine. Given $G \subseteq H$, both algorithms compute a basis B of G.

```
Basis1 ($G$):
  (* computes a basis $B$ of $G$ *)
  IF $|G| \leqslant 9\delta^2$ THEN
    RETURN Basis2($G$)
  ELSE
    $r := \lfloor \delta\sqrt{|G|} \rfloor$
    $W := \emptyset$
    REPEAT
      choose $R$ to be a random $r$-element subset of $G \setminus W$
      $C :=$ Basis2($W \cup R$)
      $V := \{h \in G \setminus C : h \in \mathsf{V}(C)\}$
      IF $|\mathsf{V}| \leqslant 2\sqrt{|G|}$ THEN
        $W := W \cup V$
      END
    UNTIL $V = \emptyset$
    RETURN $C$
  END
```

Assuming `Basis2` is correct, this algorithm is correct as well: if B is a basis of $W \cup R \subseteq G$ that in addition has no violators in G, B is a basis of G. Moreover, the algorithm augments the working set W at most δ times, which is guaranteed by the following observation.

Observation 22. *If $C \subseteq G$ and $G \cap \mathsf{V}(C) \neq \emptyset$, then $G \cap \mathsf{V}(C)$ contains at least one element from every basis of G.*

We remark that the condition resulting from Observation 22 can be used instead of locality in the definition of violator spaces. More precisely, the following is proved in [29].

Proposition 23. *Consider a set H and a mapping $\mathsf{V}: 2^H \to 2^H$, where*

(i) $\mathsf{V}(G) \cap G = \emptyset$ *for every* $G \subseteq H$, *and*
(ii) *if a set* $C \subseteq G$ *satisfies* $G \cap \mathsf{V}(C) \neq \emptyset$, *then* $G \cap \mathsf{V}(C)$ *contains at least one element from every inclusion-minimal subset* B *of* G *with* $\mathsf{V}(B) = \mathsf{V}(G)$.

Then (H, V) *is a violator space.*

Proof of Observation 22. Let B be a basis of G. Assuming

$$\emptyset = B \cap G \cap \mathsf{V}(C) = B \cap \mathsf{V}(C),$$

consistency yields $C \cap \mathsf{V}(C) = \emptyset$, implying $(B \cup C) \cap \mathsf{V}(C) = \emptyset$. From locality and monotonicity (Lemma 17), we get

$$\mathsf{V}(C) = \mathsf{V}(B \cup C) = \mathsf{V}(G),$$

meaning that $G \cap \mathsf{V}(G) = G \cap \mathsf{V}(C) = \emptyset$, a contradiction. □

It is also clear that `Basis2` is called only with sets of size at most $3\delta\sqrt{|G|}$. Finally, the expected number of iterations through the `REPEAT` loop is bounded by 2δ: by Lemma 20 (applied to $(G, \mathsf{V}|_G)$) and the Markov inequality, the expected number of calls to `Basis2` before we next augment W is bounded by 2.

Lemma 24. *Algorithm* `Basis1` *computes a basis of G with an expected number of at most $2\delta|G|$ calls to Primitive* 21, *and an expected number of at most 2δ calls to* `Basis2`, *with sets of size at most* $3\delta\sqrt{|G|}$.

4.4. Clarkson's second algorithm

This algorithm calls the trivial algorithm as a subroutine. Instead of adding violated constraints to a working set, it gives them larger probability of being selected in further iterations. Technically, this is done by maintaining G as a multiset, where $\mu(h)$ denotes the multiplicity of h (we set $\mu(F) := \sum_{h\in F} \mu(h)$). Sampling from G is done as before, imagining that G contains $\mu(h)$ copies of the element h.

```
Basis2($G$):
    (* computes a basis $B$ of $G$ *)
    IF $|G| \leqslant 6\delta^2$ THEN
        RETURN Trivial($G$)
    ELSE
        $r := 6\delta^2$
        REPEAT
            choose random $R \in \binom{G}{r}$
            $C :=$ Trivial($R$)
            $V := \{h \in G\backslash C : h \in \mathsf{V}(C)\}$
            IF $\mu(V) \leqslant \mu(G)/3\delta$ THEN
                $\mu(h) := 2\mu(h), \quad h \in V$
            END
        UNTIL $V = \emptyset$
        RETURN $C$
    END
```

Here `Trivial`(G) refers to calling the brute-force search algorithm described in Primitive 21.

Invoking Lemma 20 again (which also applies to multisets as we use them), we see that the expected number of calls to `Trivial` before we next reweight elements (a *successful* iteration), is bounded by 2. It remains to bound the number of successful iterations.

Lemma 25. *Let k be a positive integer. After $k\delta$ successful iterations, we have*

$$2^k \leqslant \mu(B) \leqslant |G|e^{k/3},$$

for every basis B of G. In particular, $k < 3\ln|G|$.

Proof. Every successful iteration multiplies the total weight of elements in G by at most $(1 + 1/3\delta)$, which gives the upper bound (not only for $\mu(B)$ but actually for $\mu(G)$). For the lower bound, we use Observation 22 again to argue that each successful iteration doubles the weight of some element in B, meaning that after $k\delta$ iterations, one element has been doubled at least k times. Because the lower bound exceeds the upper bound for $k \geqslant 3\ln|G|$, the bound on k follows. □

Summarizing, we get the following lemma.

Lemma 26. *Algorithm* `Basis2` *computes a basis of G with an expected number of at most $6\delta|G|\ln|G|$ calls to Primitive* 21, *and expected number of at most $6\delta\ln|G|$ calls to* `Trivial`, *with sets of size $6\delta^2$.*

4.5. Combining the algorithms

Theorem 27. *Using a combination of the above two algorithms, a basis of H in a violator space (H, V) can be found by calling Primitive* 21 *expected*

$$\mathrm{O}(\delta n + \delta^{\mathrm{O}(\delta)})$$

many times.

Proof. Using the above bound for the trivial algorithm, `Basis2` can be implemented to require an expected number of at most

$$\mathrm{O}\left(\delta \log|G|(|G| + \delta^{\mathrm{O}(\delta)})\right)$$

calls to the primitive. Applying this as a subroutine in `Basis1`(H) with $|H| = n$, $|G|$ is bounded by $3\delta\sqrt{n}$, and we get an overall expected complexity of

$$\mathrm{O}\left(\delta n + \delta^2(\log n(\delta\sqrt{n} + \delta^{\mathrm{O}(\delta)}))\right)$$

in terms of the number of calls to Primitive 21. The terms $\delta^2 \log n\delta\sqrt{n}$ and $\delta^2 \log n\delta^{\mathrm{O}(\delta)}$ are asymptotically dominated by either δn or $\delta^{\mathrm{O}(\delta)}$, and we get the simplified bound of $\mathrm{O}(\delta n + \delta^{\mathrm{O}(\delta)})$. □

5. Grid USO as models for violator spaces

We show in this section that the problem of finding the sink in a δ-dimensional *grid USO* [12] can be reduced to the problem of finding the (unique) basis of a violator space of combinatorial dimension δ.

USOs of grids arise from various problems, including linear programming over products of simplices and GLCPs over P-matrices [12]. The GLCP has been introduced by Cottle and Dantzig [9] as a generalization of the well-known LCP [10].

5.1. Grid USO

Fix a partition

$$\Pi = (\Pi_1, \ldots, \Pi_\delta)$$

of the set $H := \{1, \ldots, n\}$ into δ nonempty subsets, where we refer to Π_i as the *block* i. A subset $J \subseteq H$ is called a *vertex* if $|J \cap \Pi_i| = 1$ for all i. The vertices naturally correspond to the Cartesian product of the Π_i. Let $\mathscr{V}$ be the set of all vertices.

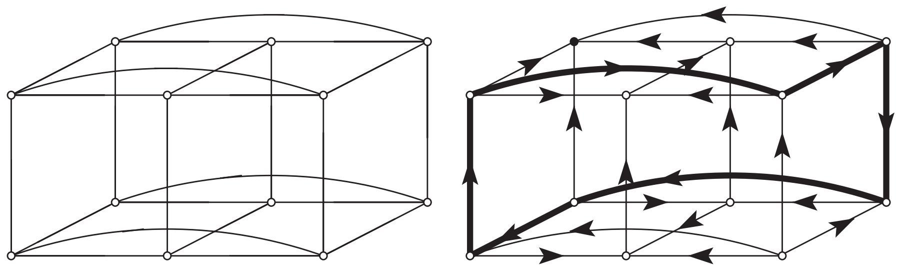

Fig. 5. A three-dimensional grid $\mathscr{G}(H)$ with $H = \{1, \dots, 7\}$ where $\Pi = (\{1, 2, 3\}, \{4, 5\}, \{6, 7\})$ and a USO of it.

In the following definition, we introduce the *grid spanned* by subsets Π'_i whose union is $G \subseteq H$. The vertex set of this grid contains all vertices $J \subseteq G$ $(J \in \mathscr{V})$, with two vertices being adjacent whenever they differ in exactly two elements.

Definition 28. The δ-dimensional *grid* spanned by $G \subseteq H$ is the undirected graph $\mathscr{G}(G) = (\mathscr{V}(G), \mathscr{E}(G))$, with

$$\mathscr{V}(G) := \{J \in \mathscr{V} : J \subseteq G\}, \quad \mathscr{E}(G) := \{\{J, J'\} \subseteq \mathscr{V}(G) : |J \oplus J'| = 2\}.$$

Here $\oplus$ is the symmetric difference of sets.

$\mathscr{V}(G)$ is in one-to-one correspondence with the Cartesian product

$$\prod_{i=1}^{\delta} G_i, \quad G_i := G \cap \Pi_i,$$

and the edges in $\mathscr{E}(G)$ connect vertices in $\mathscr{V}(G)$ whose corresponding tuples differ in exactly one coordinate. See Fig. 5 left for an example of a grid.

Note that $\mathscr{G}(G)$ is the empty graph whenever $G_i = G \cap \Pi_i = \emptyset$ for some i. We say that such a G is *not Π-valid*, and it is *Π-valid* otherwise.

A *subgrid* of $\mathscr{G}(G)$ is any graph of the form $\mathscr{G}(G')$, for $G' \subseteq G$.

Definition 29. An orientation ψ of the graph $\mathscr{G} := \mathscr{G}(H)$ is called a *USO* if all nonempty subgrids of $\mathscr{G}$ have unique sinks w.r.t. ψ.

We are interested in finding the sink in a USO of $\mathscr{G}$ as fast as possible, since the sink corresponds to the solution of the underlying problem (the P-matrix GLCP, for example). Our measure of complexity will be the expected number of *edge evaluations*, see [12]. An edge evaluation returns the orientation of the considered edge and can typically be implemented to run in polynomial time (depending on the underlying problem). In the remainder of this paper, we derive the following theorem.

Theorem 30. *The sink of a unique sink grid orientation can be found by evaluating expected* $\mathrm{O}(\delta n + \delta^{\mathrm{O}(\delta)})$ *edges.*

Note that a USO ψ can be cyclic (see the thick edges in Fig. 5 right). If ψ induces the directed edge (J, J'), we also write $J \overset{\psi}{\to} J'$. Any USO can be specified by associating each vertex J with its outgoing edges. Given J and $j \in H \backslash J$, we define $J \rhd j$ to be the unique vertex $J' \subseteq J \cup \{j\}$ that is different from J, and we call J' the *neighbor* of J *in direction* j. Note that J is a neighbor of J' in some direction different from j.

Definition 31. Given an orientation ψ of $\mathscr{G}$, the function $s_\psi : \mathscr{V} \to 2^H$, defined by

$$s_\psi(J) := \{j \in H \backslash J : J \overset{\psi}{\to} J \rhd j\} \tag{1}$$

is called the *outmap* of ψ.

By this definition, any sink w.r.t. ψ has empty outmap value.

5.2. Reduction to violator spaces

Let us fix a USO ψ of $\mathcal{G}$. Given a Π-valid subset $G \subseteq H$, we define $\text{sink}(G) \in \mathcal{V}(G)$ to be the unique sink vertex in $\mathcal{G}(G)$. For a subset G that is not Π-valid, let

$$\bar{G} := \bigcup_{i:G_i=\emptyset} \Pi_i.$$

Thus $\bar{G}$ is the set of elements occurring in blocks of Π disjoint from G.

Definition 32. For $G \subseteq H$, define

$$\mathsf{V}(G) = \begin{cases} s_\psi(\text{sink}(G)) & \text{if } G \text{ is } \Pi\text{-valid}, \\ \bar{G} & \text{if } G \text{ is not } \Pi\text{-valid}. \end{cases}$$

Theorem 33. *The pair (H, V) from Definition 32 is a violator space of combinatorial dimension δ. Moreover, for all Π-valid $G \subseteq H$, the unique sink of the subgrid $\mathcal{G}(G)$ corresponds to the unique basis of G in (H, V).*

Proof. For every $G \subseteq H$, consistency holds by definition of $\text{sink}(G)$, $s_\psi(J)$ and $\bar{G}$. In order to prove locality for $F \subseteq G \subseteq H$, we look at three different cases.

G is not Π-valid: Then, $F \subseteq G$ is not Π-valid either. The condition $\emptyset = G \cap \mathsf{V}(F) = G \cap \bar{F}$ means that F is disjoint from the same blocks as G. This implies $\bar{G} = \bar{F}$, hence $\mathsf{V}(G) = \mathsf{V}(F)$.

G and F are both Π-valid: Then $\mathcal{G}(F)$ is a nonempty subgrid of $\mathcal{G}(G)$, and $G \cap \mathsf{V}(F) = \emptyset$ means that the sink of $\mathcal{G}(F)$ has no outgoing edges into $\mathcal{G}(G)$. Thus the unique sink of $\mathcal{G}(F)$ is also a sink of $\mathcal{G}(G)$ and therefore the unique one. This means that $\text{sink}(G) = \text{sink}(F)$, from which $\mathsf{V}(G) = \mathsf{V}(F)$ follows.

G is Π-valid, F is not Π-valid: Then the condition $G \cap \mathsf{V}(F) = \emptyset$ can never be satisfied since $\mathsf{V}(F) = \bar{F}$ contains at least one full block Π_i, and $G_i = G \cap \Pi_i \neq \emptyset$.

Next we prove that a largest basis in (H, V) has at most δ elements. For this, let $G \subseteq H$ be a set of size larger than δ. If G is Π-valid, we have

$$\mathsf{V}(G) := s_\psi\left(\text{sink}(G)\right) = s_\psi\left(\text{sink}(\text{sink}(G))\right) =: \mathsf{V}\left(\text{sink}(G)\right),$$

since $J = \text{sink}(J)$ for any vertex J. This means that G has a subset of size δ with the same violators, so G is not a basis.

If G is not Π-valid, we consider some subset B that contains exactly one element from every block intersected by G. By definition, we have $\bar{G} = \bar{B}$ and $\mathsf{V}(G) = \mathsf{V}(B)$. Since B has less than δ elements, G cannot be a basis in this case, either.

It remains to prove that for G being Π-valid, the vertex $\text{sink}(G)$ is the unique basis of G in (H, V). We have already shown that $\mathsf{V}(G) = \mathsf{V}(\text{sink}(G))$ must hold in this case. Moreover, $\mathsf{V}(\text{sink}(G))$ contains no full block Π_i. On the other hand, any proper subset F of $\text{sink}(G)$ is not Π-valid, so its violator set *does* contain at least one full block. It follows that $V(F) \neq V(\text{sink}(G))$, so $\text{sink}(G)$ is a basis of G. The argument is complete when we can prove that no other vertex $J \subseteq G$ is a basis of G. Indeed, such a vertex J is not a sink in $\mathcal{G}(G)$, meaning that $G \cap \mathsf{V}(J) \neq \emptyset$. This implies $\mathsf{V}(J) \neq \mathsf{V}(G)$. □

Note that the global sink of the grid USO corresponds to the unique δ-element (and Π-valid) set B with $\mathsf{V}(B) = \emptyset$. This is exactly the set output by the call `Basis1`(H) of Clarkson's algorithms, when we apply it to the violator space constructed in Definition 32.

Primitive 21 corresponds to one edge evaluation in the USO setting. With Theorem 27, we therefore have proved Theorem 30. For small δ, the running time given in the theorem is faster than the one from the *Product Algorithm* [12] which needs expected $\mathrm{O}(\delta! n + H_n^\delta)$ edge evaluations, where H_n is the nth harmonic number.

6. Conclusions

We introduced violator spaces as a new framework for optimization problems and showed that acyclic violator spaces are equivalent to abstract and concrete LP-type problems. It turned out that the explicit ordering inherent to LP-type problems is not necessary in order to capture the structure of the underlying optimization problem. Violator spaces are more general than LP-type problems, yet Clarkson's algorithms still work on them.

The Sharir–Welzl algorithm is also applicable for violator spaces in a straightforward way. However, the most obvious translation of this algorithm to the setting of violator spaces is not even guaranteed to finish, since for a general violator space it may run in a cycle and the subexponential analysis thus breaks down.

We have seen that USOs are models for possibly cyclic violator spaces, and with Clarkson's algorithms we therefore have a fast scheme to solve fixed dimensional USO problems like the GLCP with a P-matrix. The GLCP with a P-matrix has in general a cyclic structure and therefore gives rise to a cyclic USO. A violator space obtained from a cyclic USO is again cyclic. It is interesting that there are no cycles in a two-dimensional grid USO [12]. Whether the same is true for violator spaces of combinatorial dimension 2 is an open question.

Acknowledgments

We thank an anonymous referee for many useful comments. The second author would like to thank Nina Amenta for discussions concerning LP-type problems, possibly already forgotten by her as they took place many years ago, but nevertheless helpful for reaching the results in this paper.

References

[1] N. Amenta, Helly theorems and generalized linear programming, Ph.D. Thesis, U.C. Berkeley, 1993.
[2] N. Amenta, Bounded boxes, Hausdorff distance, and a new proof of an interesting Helly-type theorem, in: Proceedings of the 10th Annual Symposium on Computational Geometry (SCG), ACM Press, New York, 1994, pp. 340–347.
[3] N. Amenta, Helly theorems and generalized linear programming, Discrete Comput. Geom. 12 (1994) 241–261.
[4] N. Amenta, A short proof of an interesting Helly-type theorem, Discrete Comput. Geom. 15 (1996) 423–427.
[5] H. Björklund, S. Sandberg, S. Vorobyov, A discrete subexponential algorithm for parity games, in: Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS), Springer, Berlin, 2003, pp. 663–674.
[6] T. Chan, An optimal randomized algorithm for maximum Tukey depth, in: Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2004, pp. 423–429.
[7] B. Chazelle, J. Matoušek, On linear-time deterministic algorithms for optimization problems in fixed dimension, J. Algorithms 21 (1996) 579–597.
[8] K.L. Clarkson, Las Vegas algorithms for linear and integer programming, J. ACM 42 (1995) 488–499.
[9] R.W. Cottle, G.B. Dantzig, A generalization of the linear complementarity problem, J. Combin. Theory 8 (1970) 79–90.
[10] R.W. Cottle, J. Pang, R.E. Stone, The Linear Complementarity Problem, Academic Press, New York, 1992.
[11] M. Develin, LP-orientations of cubes and crosspolytopes, Adv. Geom. 4 (2004) 459–468.
[12] B. Gärtner, W.D. Morris Jr., L. Rüst, Unique sink orientations of grids, in: Proceedings of the 11th Conference on Integer Programming and Combinatorial Optimization (IPCO), Lecture Notes in Computer Science, vol. 3509, Springer, Berlin, 2005, pp. 210–224.
[13] B. Gärtner, I. Schurr, Linear programming and unique sink orientations, in: Proceedings of the 17th Annual Symposium on Discrete Algorithms (SODA), 2006, pp. 749–757.
[14] B. Gärtner, E. Welzl, Linear programming—randomization and abstract frameworks, in: Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science (STACS), Springer, London, UK, 1996, pp. 669–687.
[15] B. Gärtner, E. Welzl, A simple sampling lemma—analysis and applications in geometric optimization, Discrete Comput. Geom. 25 (4) (2001) 569–590.
[16] N. Halman, Discrete and lexicographic Helly theorems and their relations to LP-type problems, Ph.D. Thesis, Tel-Aviv University, 2004.
[17] G. Kalai, A subexponential randomized simplex algorithm, in: Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC), 1992, pp. 475–482.
[18] J. Matoušek, On geometric optimization with few violated constraints, Discrete Comput. Geom. 14 (1995) 365–384.
[19] J. Matoušek, The number of unique sink orientations of the hypercube, Combinatorica 26 (1) (2006) 91–99.
[20] J. Matoušek, M. Sharir, E. Welzl, A subexponential bound for linear programming, Algorithmica 16 (1996) 498–516.
[21] J. Matoušek, T. Szabó, Random edge can be exponential on abstract cubes, in: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2004, pp. 92–100.
[22] N. Megiddo, A note on the complexity of P-matrix LCP and computing an equilibrium, Technical Report, IBM Almaden Research Center, San Jose, 1988.
[23] W.D. Morris Jr., Randomized principal pivot algorithms for P-matrix linear complementarity problems, Math. Programming Ser. A 92 (2002) 285–296.

[24] W.D. Morris Jr., Distinguishing cube orientations arising from linear programs, Manuscript, 2002.
[25] I. Schurr, T. Szabó, Finding the sink takes some time, Discrete Comput. Geom. 31 (2004) 627–642.
[26] I. Schurr, T. Szabó, Jumping doesn’t help in abstract cubes, in: Proceedings of the 11th Conference on Integer Programming and Combinatorial Optimization (IPCO), Lecture Notes in Computer Science, vol. 3509, Springer, Berlin, 2005, pp. 225–235.
[27] M. Sharir, E. Welzl, A combinatorial bound for linear programming and related problems, in: Proceedings of the Ninth Symposium on Theoretical Aspects of Computer Science (STACS), Lecture Notes in Computer Science, vol. 577, Springer, Berlin, 1992, pp. 569–579.
[28] P. Škovroň, Generalized linear programming, Master’s Thesis, Charles University, Prague, 2002. URL ⟨http://kam.mff.cuni.cz/∼xofon/diplomka/⟩.
[29] P. Škovroň, Abstract models of optimization problems, Ph.D. Thesis, Charles University, Prague, 2007. URL ⟨http://kam.mff.cuni.cz/∼xofon/thesis/⟩.
[30] T. Szabó, E. Welzl, Unique sink orientations of cubes, in: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS), 2000, pp. 547–555.