Algorithms for Combinatorial Optimization and Games Adapted from Linear Programming

Henrik Björklund, Sven Sandberg

Department of Information Technology
Uppsala University
Box 337, SE-751 05 Uppsala, Sweden

Technical report 2003-015
March 2003
ISSN 1404-3203

Abstract

The problem of maximizing functions $f : \{0,1\}^d \to \mathbb{R}$ from the boolean hypercube to real numbers arises naturally in a wide range of applications. This paper studies an even more general setting, in which the function to maximize is defined on what we call a *hyperstructure*. A hyperstructure is a Cartesian product of finite sets with possibly more than two elements. We also relax the codomain to any partially ordered set. Well-behaved such functions arise in game theoretic contexts, in particular from parity games (equivalent to the modal μ-calculus model checking) and simple stochastic games [4]. We show how several subexponential algorithms for linear programming [10, 12, 7] can be adapted to hyperstructures and give a reduction to the abstract optimization problems introduced in [7].

Contents

1 Introduction

We investigate the problem of optimizing certain well-behaved functions defined on combinatorial hyperstructures, generalizations of the boolean hypercube $\{0,1\}^d$. Such functions arise naturally in game-theoretic contexts. Their local maxima can be found with the following general local improvement scheme. Start at a vertex v of the hyperstructure. If v has a value greater or equal to all its neighbors, then return v; otherwise restart from one of the better neighbors. Generally, this algorithm can take exponential time [16]. Since the result is a local maximum, it is natural to require every local maximum to be global. This condition, called *local-global* or *LG* [15], holds in the game-theoretic setting. Additionally, games satisfy the important property that local maxima are global *on every substructure*. We call such functions *recursively local-global* or *RLG* for short. As we will show, the extra structure of RLG-functions allows us to optimize them in subexponential time, roughly $2^{O(\sqrt{d})}$, where d is the dimension.

The main motivation for our study is the application to parity and simple stochastic games. Parity games are two-player games played by moving a pebble along edges of a directed graph. The associated computational problem consists in finding the optimal strategies of both players. It is interesting from a complexity viewpoint as one of the few natural problems in NP $\cap$ coNP not known to be polynomial, especially after PRIMES was shown to be polynomial [1]. It is practically significant, being polynomial time equivalent to model-checking for the modal μ-calculus [6], one of the most expressive temporal logics of programs, subsuming logics like LTL, CTL, CTL*, etc. It is also polynomial time equivalent to Rabin chain tree automata nonemptiness [6].

In a *binary* parity game, where the players have at most two choices in each move, a strategy of one player can be viewed as a corner of a hypercube. A strategy improvement algorithm finds the winner of a parity game by assigning values to each strategy of one player and then moving to better and better strategies until the best one is reached. The strategy evaluation has to be carefully chosen so that the resulting function is LG with any maximum corresponding to an optimal strategy of the game. This guarantees that the algorithm is correct. This was first done in [17] and we later showed how to compress the value space to allow for a better upper bound on the number of iterations [3]. These functions are automatically also RLG, since any subgame is also a game. The same value functions also work for non-binary games, but the domain now becomes a more general *hyperstructure*. A hyperstructure is a Cartesian product of finite sets with possibly more than two elements. Although non-binary parity games can be transformed to binary games, keeping them non-binary allows for better upper bounds because the transformation increases the number of vertices. Simple stochastic games is a related class of games where iterative improvement algorithms are easier to understand and have been known for a longer time; see, e.g., [5]. The values of strategies are real numbers and the resulting functions are also RLG.

Linear programming (LP) is the problem of finding the maximum of a linear functional on a convex polyhedron in $\mathbb{R}^d$. Although the problem statement looks very different from that of optimizing an RLG-function on a hyperstructure, we show how to adapt several LP algorithms to RLG optimization. These algorithms are *subexponential* in the dimension of the polyhedron. Translated to our terms, they are subexponential in the dimension of the hyperstructure (defined as the number of sets in the Cartesian product), and in terms of parity games they are subexponential in the number of vertices that the first player owns. The first algorithm shown to be subexponential was Kalai's [10], adapted by us to parity games in [3]. The algorithm by Sharir and Welzl [14] was invented before Kalai's, but they (together with Matoušek) proved it subexponential later [12]; see also [13]. It is easier to understand than Kalai's and the upper bound is similar. For a survey of these and other related algorithms, see [8]. Ludwig was first to adapt these algorithms to games; he shows how Matoušek–Sharir–Welzl's algorithm works on binary simple stochastic games [11].

The success of transforming linear programming algorithms to hyperstructures and games can be understood through problem reductions. First, the strategy evaluation functions for parity games and simple stochastic games are RLG, as already discussed. Second, RLG-functions can be directly reduced to the *LP-type problems* introduced in [14]. LP-type problems is an abstract framework capturing the combinatorial structure of linear programming and numerous other problems in computational geometry [13]. The algorithms discussed here and several others work on such problems. The reduction from RLG-functions to LP-type problems was shown for the more restricted class of CLG-functions in [4], but works without modification on RLG-functions. The games actually have more structure, captured by CLG-functions, which allows for optimization algorithms that are intuitively 'more aggressive' (can take longer steps in the hyperstructure) but the best known upper bounds remain exponential.

The paper is organized as follows. In Section 2 we define the functions we optimize and the structures they are defined on. As an introduction to the concepts underlying later sections, we present a Ludwig-style optimization algorithm for hypercubes in Section 3. Sections 4 and 5 adapt the linear programming algorithms from [13] and [10], respectively. Finally, we show in Section 6 how our problem can be reduced to what Gärtner [7] calls an abstract optimization problem (AOP).

2 Hyperstructures

The set of all positional strategies in a parity game is isomorphic to a product of finite sets, each representing the choices in a vertex. Functions from strategies to values in such games motivate our study of functions on the *hyperstructures* defined here. They are generalizations of the boolean hypercube $\{0,1\}^d$. Optimization of well-behaved functions on hypercubes has been extensively studied in, e.g., [9, 18, 19, 16, 2].

Definition 2.1 (Hyperstructure) *For each $j \in \{1,\ldots,d\}$ let $\mathcal{P}_j = \{e_{j,1},\ldots, e_{j,\delta_j}\}$ be a finite nonempty set. Call $\mathcal{P} = \prod_{j=1}^d \mathcal{P}_j$ a d-dimensional* hyperstructure*, or* structure *for short.* □

A *substructure* of $\mathcal{P}$ is a product $\mathcal{P}' = \prod_{j=1}^d \mathcal{P}'_j$, where $\emptyset \neq \mathcal{P}'_j \subseteq \mathcal{P}_j$ for all j. A *facet* of $\mathcal{P}$ is a substructure obtained by fixing the choice in exactly one coordinate. Thus $\mathcal{P}'$ is a facet of $\mathcal{P}$ if there is a $j \in \{1,\ldots,d\}$ such that $|\mathcal{P}'_j| = 1$ and $\mathcal{P}'_k = \mathcal{P}_k$ for all $k \neq j$. We will sometimes identify a facet with its defining element $e_{j,i} \in \mathcal{P}_j$, assuming all sets P_k to be disjoint. If $|\mathcal{P}'_j| \leq 2$ for all j, then $\mathcal{P}'$ is called a *subcube* since it is isomorphic to a boolean hypercube. We will consistently use the letter d for the dimension and $n := \sum_{j=1}^d |\mathcal{P}_j|$ for the number of facets. If $|P_j| = 1$ for some j, this coordinate can be disregarded since it is constant, and we will consider such structures as having smaller dimension.

An element of $\mathcal{P}$ is called a *vertex*. Two vertices $x, y \in \mathcal{P}$ are *neighbors* if they differ in only one coordinate. The neighbor relation induces a graph with elements of $\mathcal{P}$ as nodes, and allows us to talk about paths and distances in $\mathcal{P}$.

For a set $E \subseteq \bigcup_{j=1}^d \mathcal{P}_j$, define *struct*$(E)$ to be the substructure $\mathcal{P}' = \prod_{j=1}^d (\mathcal{P}_j \cap E)$. Thus, if E does not have elements from each $\mathcal{P}_j$, then *struct*$(E) = \emptyset$. For a set $\mathcal{F}$ of facets, we use *struct*$(\mathcal{F})$ for *struct*(E) where E is the set of defining elements of the facets in $\mathcal{F}$.

Throughout this paper, let $\mathcal{D}$ be some partially ordered set. We are interested in finding optima of functions from $\mathcal{P}$ to $\mathcal{D}$, such that any two neighbors have *comparable* values. A *local maximum* of a function $f : \mathcal{P} \to \mathcal{D}$ is a vertex v such that $f(v) \geq f(u)$ for all neighbors u of v. If $f(v) \geq f(u)$ for all vertices $u \in \mathcal{P}$, then v is a *global maximum* of f on $\mathcal{P}$. Local and global *minima* are defined symmetrically.

Arbitrary functions on $\mathcal{P}$ cannot be efficiently optimized; see, e.g., Corollary 19 of [16], but the prospects are better for some nontrivial and very interesting subclasses. The class of *completely unimodal* (CU) functions on hypercubes $\mathcal{H} = \{0,1\}^d$ has been studied in, e.g, [9, 19, 18, 2], and can be optimized in subexponential time [2]. In [4] we defined the class of *completely local-global* (CLG) functions, a generalization of CU-functions, defined on hyperstructures and capturing the properties of value-functions obtained from parity games. To demonstrate the full applicability of the algorithms in later sections, we here define the even wider class of *recursively local-global* (RLG) functions. The inclusions CU $\subset$ CLG $\subset$ RLG follow from the definitions. The strictness of these inclusions is easy to establish.

Definition 2.2 (Recursively Local-Global) *A function $f : \mathcal{P} \to \mathcal{D}$ for which all neighbors have comparable values is called* recursively local-global *(RLG) if for every substructure $\mathcal{P}'$ of $\mathcal{P}$, all local maxima of the restriction of f to $\mathcal{P}'$ are also global.* □

In the sequel, we discuss algorithms for maximizing RLG-functions. Such algorithms can also be used to maximize CLG-functions and CU-functions. We use *RLG-structure* to denote an RLG-function together with its underlying hyperstructure, and we use *RLG-cube* if the hyperstructure is a hypercube. An RLG-structure can be thought of as a hyperstructure with its vertices labeled by function values. Given an RLG-function $f : \mathcal{P} \to \mathcal{D}$, and a substructure $\mathcal{P}'$ of $\mathcal{P}$, denote by $w_f(\mathcal{P}')$ the maximum value of f on $\mathcal{P}'$.

3 Ludwig-Style Algorithm for Hypercubes

The first subexponential randomized algorithm for solving simple stochastic games was presented in [11]. It applies the ideas of the linear programming algorithms in [12, 13] and [10], which we will discuss in later sections. Unfortunately, the algorithm only applies to binary games (with vertex outdegree at most two), and reduction to such games may increase the number of vertices, undermining the subexponential analysis. Binary games give rise to RLG-cubes [4].

As an introduction to the ideas underlying the algorithms presented later, and their adaptations to RLG-functions, the Ludwig-style Algorithm 1 maximizes any RLG-function $f : \mathcal{H} \to \mathcal{D}$, where $\mathcal{H} = \{0,1\}^d$.

Algorithm 1: Ludwig's Algorithm for RLG-functions
LUDWIG(RLG-cube $\mathcal{H}$, initial vertex v_0)
(1) **if** $\dim(\mathcal{H}) = 0$
(2)   **return** v_0
(3) choose a random facet F of $\mathcal{H}$ containing v_0
(4) $v^* \leftarrow$ Ludwig(F, v_0)
(5) **if** neighbor u of v^* on $\mathcal{H} \setminus F$ is better than v^*
(6)   **return** Ludwig($\mathcal{H} \setminus F$, u)
(7) **else**
(8)   **return** v^*

The key observation in the analysis of this algorithm is that, depending on the choice in line 3, the actual dimension of the remaining problem solved on line 6 may decrease with more than one. There are d equally likely choices; call them $F_1, F_2, \ldots, F_d$. Assume the set $\{w_f(F_1), \ldots, w_f(F_d)\}$ is linearly ordered so that

$$w_f(F_1) \leq \cdots \leq w_f(F_d).$$

If F_i is chosen for the first recursive call, none of the facets $F_1, \ldots, F_i$ will ever be visited by the algorithm again, since the value of the current vertex v^* is the biggest value on any of them. Let $T(d)$ be the expected number of times that the second recursive call will be made. Then $T(0) = 0$ and

$$T(d) \leq T(d-1) + 1 + \frac{1}{d} \sum_{j=0}^{d-1} T(j).$$

Solving the recurrence gives $T(d) = 2^{O(\sqrt{n})}$ [11]. What if $\{w_f(F_1), \ldots, w_f(F_d)\}$ is not linearly ordered? This is no disadvantage to the algorithm, since it only visits vertices with *comparable and strictly bigger* values than the current vertex. Therefore, linearly ordered maximal values on the facets is the worst case.

4 Matoušek–Sharir–Welzl-Style Algorithm

In this section, we show how the LP-algorithm by Matoušek, Sharir, and Welzl [12, 13] can be modified for RLG-functions, yielding a simple subexponential randomized algorithm for the problem (and thus also for parity games; see also [4]). It has expected running time at most $2^{O(\sqrt{d \log(n/\sqrt{d})}+\log n)}$, where $n = \sum_{j=1}^{d} |\mathcal{P}_j|$ is the number of facets of $\mathcal{P}$.

```
Algorithm 2: MSW-Style Optimization Algorithm
MSW(RLG-structure P, initial vertex v0)
(1)   if dim(P) = 0
(2)      return v0
(3)   choose a random facet F of P, not containing v0
(4)   v* ← MSW(P \ F, v0)
(5)   if neighbor u of v* on F is better than v*
(6)      return MSW(F, u)
(7)   else
(8)      return v*
```

Note that this algorithm generalizes Ludwig's algorithm, discussed in Section 3. If $\mathcal{P}$ is a hypercube, the two algorithms are the same.

The algorithm always terminates since the recursive calls in lines (4) and (6) are made on strictly smaller substructures. It is correct since any vertex without better neighbors in an RLG-structure is globally optimal.

It remains to show that the bound from [13] holds in this setting. This is done by translating into the concepts of [13], thus showing that the same recurrence for the running time holds. The following definition will be useful.

Definition 4.1 *A facet of an RLG-structure is* extreme *if it contains all local maxima on $\mathcal{P}$.* □

At most one facet in each coordinate can contain all maxima, and thus there are at most d extreme facets. The second recursive call on line (6) of the algorithm will be performed only when the chosen facet F is extreme on $\mathcal{P}$. This means that the second call will be made with probability at most $\frac{d}{n-d}$. If the current vertex belongs to $d-k$ extreme facets, the probability is at most $\frac{k}{n-d}$, hence $\frac{\min(k,n-d)}{n-d}$.

Definition 4.2 (Hidden Dimension.) *Given an RLG-structure $\mathcal{P}$ and a vertex $v \in \mathcal{P}$, the* hidden dimension *of the pair $(\mathcal{P}, v)$ is d minus the number of facets of $\mathcal{P}$ containing v and every $u \in \mathcal{P}$ with $f(u) > f(v)$.* □

Intuitively, if $(\mathcal{P}, v)$ has hidden dimension k, then v and all vertices with better values than v belong to some k-dimensional substructure $\mathcal{P}'$ of $\mathcal{P}$. In particular, if the hidden dimension is 0, then v is a maximum on $\mathcal{P}$. The algorithm will only visit vertices that belong to $\mathcal{P}'$. There are $d-k$ facets, $F_1, F_2, \ldots, F_{d-k}$, that contain $\mathcal{P}'$. In the worst case there are k more extreme facets. For each such facet, consider the best value that does not belong to it. Assume first that these values are totally ordered. Enumerate the facets so that

$$w_f(\mathcal{P} \setminus F_{d-k+1}) \leq \cdots \leq w_f(\mathcal{P} \setminus F_{d-k+i}) \leq \cdots \leq w_f(\mathcal{P} \setminus F_d).$$

Suppose the algorithm chooses facet F_{d-k+i}. Then $f(v^*) = w_f(\mathcal{P} \setminus F_{d-k+i}) < f(u)$. Every vertex with a better value than v^* that belongs to F_{d-k+i} must also belong to F_{d-k+j} for all $1 \leq j < i$. Thus u belongs to $(d-k+i)$ facets that contain all vertices with better values, and the hidden dimension of the pair (F_{d-k+i}, u) for the second call is at most $(k-i)$.

If the best values outside the remaining extreme facets are not totally ordered, this only benefits the algorithm. The values are partially ordered, and if facet F_{d-k+i} is chosen, u will belong to all facets that do not have a strictly bigger value for $w_f(\mathcal{P} \setminus F_j)$, and they will all contain all vertices with better values. This is because if $w_f(\mathcal{P} \setminus F_{d-k+i})$ and $w_f(\mathcal{P} \setminus F_j)$ are incomparable, then any value better than $w_f(\mathcal{P} \setminus F_{d-k+i})$ will be better than or incomparable to $w_f(\mathcal{P} \setminus F_j)$, since the order on $\mathcal{D}$ is transitive.

This discussion is enough to get the same recurrences for the numbers of tests on line (5) and jumps to u on line (6) as [13] gets for the numbers of violation tests and basis computations, respectively, in the linear programming setting. They are both bounded by the recurrence $t_k(d) = 0$ (where $0 \leq k \leq d$) and

$$t_k(n) \leq t_k(n-1) + 1 + \frac{1}{n-d} \sum_{i=1}^{\min(k, n-d)} t_{k-i}(n), \qquad \text{for } n > d.$$

The recurrences are solved in [13], and from the solutions it is easy to infer that the expected running time of the algorithm on RLG-structures is at most $2^{O(\sqrt{d \log(n/\sqrt{d})} + \log n)}$, as long as function evaluation and comparison of function values can be performed in polynomial time. As soon as n can be assumed to be polynomially bounded by d, this bound is subexponential in d. The reduction from parity games with d vertices to RLG-functions gives $n = O(d^2)$, so we can solve parity games in expected time $2^{O(\sqrt{d \log d})}$.

5 Kalai-Style Algorithm

We now describe another algorithm for optimizing RLG-structures, originally invented for linear programming by Kalai [10, 8] and later adapted by us to parity games [3]. It is also subexponential with complexity similar to the MSW-style algorithm of the previous section (but it is slightly more complicated).

Let $\mathcal{P}(d,n)$ denote the class of RLG-structures with dimension d and $n = \sum_{j=1}^{d} |\mathcal{P}_j|$. If v is a vertex in $\mathcal{P}$ then a facet F is *v-improving* if some *witness vertex* $v' \in F$ has a better value than v.

The Algorithm takes an RLG-structure $\mathcal{P} \in \mathcal{P}(d,n)$ and an initial vertex v_0 as inputs and returns the optimal vertex on $\mathcal{P}$. It uses the subroutine Collect-Improving-Facets, described later, that collects a set of pairs (F, v) of v_0-improving facets F and corresponding witness vertices v.

Algorithm 3: Kalai-Style Optimization Algorithm
Kalai(RLG-structure $\mathcal{P}$, initial vertex v_0)
(1) **if** $\dim(\mathcal{P}) = 0$
(2) **return** v_0
(3) $M \leftarrow$ Collect-Improving-Facets$(\mathcal{P}, v_0)$
(4) choose a random pair $(F, v_1) \in M$
(5) $v^* \leftarrow$ Kalai(F, v_1)
(6) **if** some neighbor u of v^* on $\mathcal{P} \setminus F$ is better than v^*
(7) **return** Kalai$(\mathcal{P}, u)$
(8) **else**
(9) **return** v^*

How to Find Many Improving Facets. Now we describe the subroutine Collect-Improving-Facets. The goal is to find r v_0-improving facets, where r is a parameter (Kalai uses $r = \max(d, n/2)$ to get the best complexity analysis). To this end we construct a sequence $(\mathcal{P}^0, \mathcal{P}^1, \ldots, \mathcal{P}^{r-d})$ of substructures of $\mathcal{P}$, with $\mathcal{P}^i \in \mathcal{P}(d, d+i)$ and $\mathcal{P}^i \subset \mathcal{P}^{i+1}$. All the $d+i$ facets of $\mathcal{P}^i$ are v_0-improving; we simultaneously determine the corresponding witness vertices v^j optimal in $\mathcal{P}^j$. The subroutine returns r facets of $\mathcal{P}$, each one obtained by fixing one of the r choices in $\mathcal{P}^{r-d} \in \mathcal{P}(d,r)$. All these are v_0-improving by construction.

Let v^0 be a better neighbor of v_0 on $\mathcal{P}$. (If no better neighbor exists then v_0 is optimal in $\mathcal{P}$ and we are done.) Set $\mathcal{P}^0$ to the RLG-structure containing only v^0. Fixing any of the d coordinates of $\mathcal{P}$ as in v^0 defines a v_0-improving facet of $\mathcal{P}$ with v^0 as a witness.

To construct $\mathcal{P}^{i+1}$ from $\mathcal{P}^i$, let v' be a better neighbor of v^i. (Note that v^i is optimal in $\mathcal{P}^i$ but not necessarily in the full structure $\mathcal{P}$. If it is, we terminate.) Let $\mathcal{P}^{i+1}$ be the smallest substructure of $\mathcal{P}$ containing v' and all vertices in $\mathcal{P}^i$. Recursively apply the algorithm to find the optimal v^{i+1} in $\mathcal{P}^{i+1}$. Note that fixing a coordinate in $\mathcal{P}$ to any of the $d+i$ choices in $\mathcal{P}^i$ defines a v_0-improving facet; therefore, the final $\mathcal{P}^{r-d}$ has r improving facets.

Analysis. First note that we can pick the random number before line 3, pass it as parameter to COLLECT-IMPROVING-FACETS, and only find that many facets. Now each solved subproblem starts from a strictly better vertex, and it is clear that the algorithm terminates. It is correct because it can only terminate by returning an optimal vertex.

This algorithm yields a recurrence similar to those in previous sections. Kalai solves it and gets the subexponential running time $2^{O((\log n)\sqrt{d/\log d})}$.

It is interesting to note the similarities of Ludwig's, Matoušek–Sharir–Welzl's, and Kalai's subexponential algorithms. They all take a structure $\mathcal{P}$ and a vertex v_0 as parameters and recursively find the optimum v^* on a substructure $\mathcal{P}'$ of $\mathcal{P}$. The choice of $\mathcal{P}'$ is random, but it is guaranteed to contain vertices at least as good as v_0, and the recursive call starts from a witness of this fact. If the result is not an optimum on the entire structure, then we optimize the rest of the structure, starting from a better neighbor of v^*. Intuitively, the subexponential upper bound relies on the substructure being taken randomly from some set. Thus it is expected that the optima of many structures in the set are no better than v^*, and those structures will be ignored in the remainder of the algorithm.

6 Abstract Optimization Problems

The class of *abstract optimization problems* (AOPs) was introduced by Gärtner [7]. He uses this generalization to show how the ideas from [13, 10] can be used to obtain a subexponential randomized algorithm for a wider class of optimization problems. In this section we show that optimizing an RLG-function with totally ordered codomain can be reduced to solving an AOP.

Definition 6.1 (AOP [7]) *An AOP is a triple $(A, <, \Phi)$, where A is a finite set, $<$ is a total order on 2^A, and $\Phi : \{(C, B) | C \subseteq B \subseteq A\} \to 2^A$ satisfies*

$$\Phi(C, B) = \begin{cases} C, & \text{iff } C = \max_<\{C' | C' \subseteq B\}; \\ C', \text{ for some } C' \text{ s.t. } C < C' \subseteq B, & \text{otherwise.} \end{cases}$$

For $B \subseteq A$, let $opt(B)$ denote $\max_<\{C | C \subseteq B\}$. Solving an AOP means finding $opt(A)$. □

Intuitively, solving an AOP corresponds to maximizing some function on a boolean hypercube. The function may be unstructured, but there is an oracle capable of finding some better vertex than the current one on any subcube containing the origin. The algorithm in [7] solves any AOP with $|A| = n$ using expected $e^{2\sqrt{n}+O(\sqrt[4]{n}\ln n)}$ calls to Φ. Maximizing an RLG-function f with n facets can be reduced to solving an AOP with $|A| = n$, if its codomain is totally ordered. This restriction is often satisfied, as for, e.g., parity games [4].

Let $\mathcal{P}$ be a d-dimensional hyperstructure with n facets, and let $f : \mathcal{P} \to \mathcal{D}$ be an RLG-function with totally ordered codomain. Define an AOP $(\mathcal{F}, <, \Psi)$ as follows. Let $-\infty$ be an artificial value, smaller than all values of vertices of $\mathcal{P}$. Let $\mathcal{F}$ be the set of all facets of p, and define $val : 2^{\mathcal{F}} \to \mathcal{D} \cup \{-\infty\}$ by

$$val(F) = \begin{cases} w_f(struct(F)), & \text{if } struct(F) \neq \emptyset; \\ -\infty, & \text{otherwise.} \end{cases}$$

Take $\prec$ to be any total order on $2^{\mathcal{F}}$ such that $F \prec F'$ whenever $|F| > |F'|$. For $F, F' \in \mathcal{F}$ let $F < F'$ iff 1) $val(F) < val(F')$, or 2) $val(F) = val(F')$ and $F \prec F'$. Now define $\Phi(F, G)$ as follows.

- Suppose $val(F) \neq -\infty$ and $struct(F)$ contains only one vertex v. If v is a local maximum on $struct(G)$, then $\Phi(F, G) = F$. Otherwise $\Phi(F, G)$ is a set defining a 0-dimensional substructure containing only a better neighbor of v on $struct(G)$.
- If $val(F) \neq -\infty$ but $|F| > d$ then $\Phi(F, G) = F'$ where F' defines a substructure containing only one vertex with maximal value on $struct(F)$.

- If $val(F) = -\infty$ and $struct(G) = \emptyset$ then $\Phi(F, G) = \emptyset$.
- If $val(F) = -\infty$ and $struct(G) \neq \emptyset$ then $\Phi(F, G) = F'$ for some $F' \in struct(G)$.

Now $(\mathcal{F}, <, \Phi)$ defines an AOP, and applying Gärtner's algorithm will yield a solution from which a maximal vertex of the original RLG-structure can be recovered.

Unfortunately, the bound this gives for maximizing RLG-functions is not very good. The previously discussed algorithms give bounds subexponential in d as long as n is polynomial in d, which is not the case here. If for example, $n = \Theta(d^2)$, which is the worst case when reducing games to RLG-functions, we only get a $2^{O(d)}$ bound on the expected calls to Φ. The reason seems to be that the original dimension is lost in the reduction, and all facets are treated as independent.

Gärtner's algorithm, called with the appropriate parameters, will never call Φ with any set of size bigger than d as the first parameter. For such sets, Φ can be computed in polynomial time.

7 Conclusions

In [3, 4] we showed how the well-known linear programming algorithms from [10, 13] can be adapted to solving parity games. In this paper we showed that these algorithms work for maximizing recursively local-global functions on hyper-structures. Together with the reductions from games to RLG-functions from [4], this stresses the combinatorial similarities between linear programming and solving games. It also provides a simple setting, stripped of cumbersome details, convenient for investigation of such similarities and developing new algorithms for parity and simple stochastic games.

References

[1] M. Agrawal, N. Kayal, and N. Saxena. Primes is in P. Unpublished manuscript, `http://www.cse.iitk.ac.in/news/primality.html`, August 2002.

[2] H. Björklund, S. Sandberg, and S. Vorobyov. Optimization on completely unimodal hypercubes. Technical Report 018, Uppsala University / Information Technology, May 2002. `http://www.it.uu.se/research/reports/`.

[3] H. Björklund, S. Sandberg, and S. Vorobyov. A discrete subexponential algorithm for parity games. In H. Alt and M. Habib, editors, *20th International Symposium on Theoretical Aspects of Computer Science, STACS'2003*, volume 2607 of *Lecture Notes in Computer Science*, pages 663–674, Berlin, 2003. Springer-Verlag. Full preliminary version: TR-2002-026, Department of Information Technology, Uppsala University, September 2002.

[4] H. Björklund, S. Sandberg, and S. Vorobyov. On combinatorial structure and algorithms for parity games. Technical Report 2003-002, Department of Information Technology, Uppsala University, January 2003. `http://www.it.uu.se/research/reports/`.

[5] A. Condon. On algorithms for simple stochastic games. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 13:51–71, 1993.

[6] E. A. Emerson, C. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. In *Computer Aided Verification, Proc. 5th Int. Conference*, volume 697, pages 385–396. Lect. Notes Comput. Sci., 1993.

[7] B. Gärtner. A subexponential algorithm for abstract optimization problems. *SIAM Journal on Computing*, 24:1018–1035, 1995.

[8] Goldwasser. A survey of linear programming in randomized subexponential time. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 26:96–104, 1995.

[9] P. L. Hammer, B. Simeone, Th. M. Liebling, and D. De Werra. From linear separability to unimodality: a hierarchy of pseudo-boolean functions. *SIAM J. Disc. Math.*, 1(2):174–184, 1988.

[10] G. Kalai. A subexponential randomized simplex algorithm. In *24th ACM STOC*, pages 475–482, 1992.

[11] W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117:151–155, 1995.

[12] J. Matoušek, M. Sharir, and M. Welzl. A subexponential bound for linear programming. In *8th ACM Symp. on Computational Geometry*, pages 1–8, 1992.

[13] J. Matoušek, M. Sharir, and M. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16:498–516, 1996.

[14] M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In *9th Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 577 of *Lecture Notes in Computer Science*, pages 569–579, Berlin, 1992. Springer-Verlag.

[15] C. A. Tovey. Low order polynomial bounds on the expected performance of local improvement algorithms. *Mathematical Programming*, 35:193–224, 1986.

[16] C. A. Tovey. Local improvement on discrete structures. In E. Aarts and Lenstra J. K., editors, *Local Search in Combinatorial Optimization*, pages 57–89. John Wiley & Sons, 1997.

[17] J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *CAV'00: Computer-Aided Verification*, volume 1855 of *Lect. Notes Comput. Sci.*, pages 202–215. Springer-Verlag, 2000.

[18] D. Wiedemann. Unimodal set-functions. *Congressus Numerantium*, 50:165–169, 1985.

[19] K. Williamson Hoke. Completely unimodal numberings of a simple polytope. *Discrete Applied Mathematics*, 20:69–81, 1988.

Recent technical reports from the Department of Information Technology

2002-036	Bernhard Müller: *Control Errors in CFD!*
2002-037	Bob Melander and Mats Björkman: *Trace-Driven Network Path Emulation*
2002-038	Parosh Aziz Abdulla and Alexander Rabinovich: *Verification of Probabilistic Systems with Faulty Communication*
2002-039	R. Blaheta, S. Margenov, and M. Neytcheva: *Uniform estimate of the constant in the strengthened CBS inequality for anisotropic non-conforming FEM systems*
2002-040	Torsten Söderström: *Why are errors-in-variables problems often tricky?*
2002-041	Per Lötstedt and Martin Nilsson: *A Minimum Residual Interpolation Method for Linear Equations with Multiple Right Hand Sides*
2003-001	Parosh Abdulla, Johann Deneux, Pritha Mahata, and Aletta Nylén: *Downward Closed Language Generators*
2003-002	Henrik Björklund, Sven Sandberg, and Sergei Vorobyov: *On Combinatorial Structure and Algorithms for Parity Games*
2003-003	Magnus Svärd and Jan Nordström: *A Stable and Accurate Summation-by-Parts Finite Volume Formulation of the Laplacian Operator*
2003-004	Kaushik Mahata and Torsten Söderström: *Subspace estimation of real-valued sine wave frequencies*
2003-005	Samuel Sundberg: *Solving the linearized Navier-Stokes equations using semi-Toeplitz preconditioning*
2003-006	Henrik Brandén and Per Sundqvist: *An Algorithm for Computing Fundamental Solutions of Difference Operators*
2003-007	Henrik Brandén, Sverker Holmgren, and Per Sundqvist: *Discrete Fundamental Solution Preconditioning for Hyperbolic Systems of PDE*
2003-008	Julian Richardson and Pierre Flener: *Program Schemas as Proof Methods*
2003-009	Alexandre David, M. Oliver Möller, and Wang Yi: *Verification of UML Statecharts with Real-Time Extensions*
2003-010	Alexandre David, Johann Deneux, and Julien d'Orso: *A Formal Semantics for UML Statecharts*
2003-011	Alexandre David, Gerd Behrmann, Kim G. Larsen, and Wang Yi: *A Tool Architecture for the Next Generation of UPPAAL*
2003-012	Ken Mattsson and Jan Nordström: *Finite Difference Approximations of Second Derivatives on Summation by Parts*
2003-013	Ken Mattsson, Magnus Svärd, and Jan Nordström: *Stable Artificial Dissipation Operators for High Order Finite Difference Schemes*
2003-014	Brahim Hnich, Julian Richardson, and Pierre Flener: *Towards Automatic Generation and Evaluation of Implied Constraints*

UPPSALA
UNIVERSITET

March 2003
ISSN 1404-3203
http://www.it.uu.se/