An Experimental Study of Algorithms for Completely Unimodal Optimization

Henrik Björklund, Sven Sandberg, and Sergei Vorobyov

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

Technical report 2002-030
October 2002
ISSN 1404-3203

Abstract

The paper presents an account of the experimental study of five different algorithms for the Completely Unimodal Pseudoboolean Function (CUPBF) Optimization. CUPBFs satisfy the Hirsch's conjecture, but are not known (although conjectured) to be polynomial. We summarize known and new upper and lower bounds, describe methods of random CUPBFs generation, and use them to compare the algorithms.

*) Supported by Swedish Research Council Grants "*Infinite Games: Algorithms and Complexity*", "*Interior-Point Methods for Infinite Games*".

Contents

1 Introduction

Pseudo-boolean functions (PBFs), i.e., real-valued functions of a finite set of 0-1 variables, have numerous applications in computer science, optimization, management sciences, economics, discrete mathematics, VLSI design, etc., and have been extensively studied during the last half-century (Boros & Hammer 2001). Several authors have investigated special classes of functions possessing 'nice' structural properties relevant for practical problems and possibly amenable to a more efficient optimization, as compared to the general PBFs (Tovey 1986, Hammer, Simeone, Liebling & De Werra 1988, Williamson Hoke 1988, Tovey 1997).

In this paper we are primarily concerned with algorithms optimizing *completely unimodal pseudo-boolean functions* (CUPBFs), possessing a *unique local maximum* on every subcube of $\{0,1\}^n$, introduced and studied in (Wiedemann 1985, Hammer et al. 1988, Williamson Hoke 1988); see also survey (Tovey 1997).

Complete unimodality is a useful abstract property shared by linear programming, stochastic and other games, certain linear complementarity, and geometric optimization problems. There is a long-standing conjecture that any CUPBF can be optimized in polynomial time by the standard randomized local search algorithm (Williamson Hoke 1988). For a CUPBF there is a value-increasing path following the edges of the hypercube $\{0,1\}^n$ from every point to the global maximum of length at most n, i.e., the Hirsch conjecture holds, and the existence of a polynomial time optimization algorithms is not excluded. Note however that the comprehensive surveys (Tovey 1997, Boros & Hammer 2001) do not mention any *subexponential* algorithms for the CUPBF optimization (first reported in (Björklund, Petersson & Vorobyov 2001, Björklund, Sandberg & Vorobyov 2002*b*)).

Our interest in the CUPBF optimization is mainly motivated by applications to infinite *parity games* pertinent to Rabin chain tree automata on infinite trees, program verification, and model checking (Emerson 1997). Determining a winner in such games is known to be in **NP** $\cap$ **coNP** with a long-standing open and widely conjectured **PTIME**-membership. Meanwhile, all previous algorithms for parity games and CUPBF optimization were invariably exponential, until we recently developed *randomized subexponential* ones for both problems (Björklund et al. 2001, Petersson & Vorobyov 2001, Björklund et al. 2002*b*, Björklund, Sandberg & Vorobyov 2002*a*), and also improved previous bounds for other known algorithms.

For the purposes of rapid prototyping, experimenting, and comparison of different algorithms we needed a general abstract framework where we could easily implement and compare 'high-level' algorithms without being overwhelmed by tedious implementation details specific to games (like solving LP instances, performing graph optimization with high-precision representation of weights, etc.), and CUPBF optimization appeared the right choice. We were able to abstract away new algorithms appropriate for CUPBF optimization (see the next section on the multiple switch algorithms APSA, RMSA, such algorithms are not even mentioned in (Tovey 1997)), test them experimentally, and improve their lower and upper bounds (Björklund et al. 2001, Björklund et al. 2002*b*). This approach played a crucial role in our development of subexponential algorithms for parity games (Petersson & Vorobyov 2001, Björklund et al. 2002*a*).

The best currently known upper bound for a (randomized) algorithm optimizing a CUPBF function is $2^{O(\sqrt{n})}$ expected function evaluations, provided in (Björklund et al. 2002*b*). It has been conjectured (Williamson Hoke 1988) that only polynomially many evaluations are necessary, but this claim has never been proved. On the other hand, no nontrivial lower bounds have been shown. One of the main motivations of the current work was to investigate both experimentally and theoretically the gap between the best known upper and lower bounds of algorithms for the problem. Finding problem instances on which an algorithm fares poorly guides the search for families of instances that provide new lower bounds for that algorithm. On the other hand, the absence of hard instances for an algorithm encourages further investigations towards improving upper bounds, maybe for special classes of functions.

We now briefly explain how optimizing CUPBFs is related to solving games, such as, e.g., parity or simple stochastic games. In a game where a player has n binary choices, a strategy can be seen as a vertex of the n-dimensional hypercube. If one can assign values to strategies, then solving the game amounts to finding a vertex with the best value. Also, fixing one of the choices creates a new game, corresponding to an $(n-1)$-dimensional face of the hypercube. Thus if each game in a class (closed wrt subgames) has a unique optimal strategy, then this is true for subgames as well, making the hypercubes of strategy-values completely unimodal. Strictly speaking, in parity and simple stochastic games we do not have uniqueness of optimal strategies, and the class of the *Completely LG* PBFs (on every face a local maximum is global) would be more appropriate. However, 1) our subexponential algorithm applies to the Completely LG PBFs, 2) there is no standard definition of such PBFs, 3) we do not know how to generate such PBFs. Two random distributions of LG PBFs (every local maximum is global, but not recursively on each face) are described in (Tovey 1986) and also expected polynomial upper bounds on such distributions for standard local search algorithms are proved. It would be desirable to extend these results to the classes of random distributions of CUPBFs and Completely LG PBFs.

It should be mentioned that generating 'hard' instances of CUPBFs is a nontrivial problem (after all, the problem may turn out polynomial!) We describe two such methods, one of which is random, in Sections 4 and 6, and the corresponding experimental results for different algorithms in Sections 5 and 7. Trying different methods of generating CUPBFs and testing the algorithms on them 1) suggests that the expected running time of the algorithms we consider on 'natural' random distributions is low degree polynomial, similar to the conclusions of (Tovey 1986); 2) has led us to discovering new lower and upper bounds for different algorithms and subclasses of CUPBFs.

The results we obtained are promising and already provided useful consequences and applications for games (Björklund et al. 2002*a*). CUPBF optimization has numerous remaining inspiring open problems and deserves further theoretical and experimental investigations.

2 Completely Unimodal Basics

Definition 2.1 *Let $n \in \mathbb{N}$ and $\mathcal{H} = \{0,1\}^n$ be a boolean hypercube. An n-dimensional* completely unimodal pseudo-boolean function *(CUPBF) is an injective mapping $\mathcal{H} \to \mathbb{R}$ with a unique local maximum on every face of $\mathcal{H}$.*

There are several equivalent ways to define/characterize CUPBFs:

Theorem 2.2 (Hammer et al. (1988)) *For an injective pseudo-boolean function f on the n-dimensional boolean hypercube $\mathcal{H}$, the following are equivalent:*

- *f is completely unimodal,*
- *f has a unique local minimum on every face of $\mathcal{H}$,*
- *f has a unique local minimum on every 2-dimensional face of $\mathcal{H}$,*
- *f has a unique local maximum on every 2-dimensional face of $\mathcal{H}$.* □

Definition 2.3 *Two vertices of the n-dimensional hypercube (binary vectors) are* neighbors *if they differ in only one coordinate. For a CUPBF f, a move from a vertex v to one of its neighbors v' such that $f(v) < f(v')$ is called a* profitable switch*.*

Definition 2.4 *Given a PBF f and a vertex $v \in \mathcal{H}$, define $VID_f(v)$, the* vector of f-improving directions *from v, in the following way. If $f(v) < f(v_i)$ for the neighbor v_i of v in dimension i, then $VID_f(v)$ has 1 in position i, and 0 otherwise.*

We use the following remarkable property in generating CUPBFs.

Theorem 2.5 (Williamson Hoke (1988)) *An injective PBF f is completely unimodal if and only if its corresponding function $VID_f : \{0,1\}^n \to \{0,1\}^n$ is bijective.* □

3 Algorithms for Completely Unimodal Optimization

In this section we briefly describe five algorithms for completely unimodal optimization.

3.1 Greedy and Random Local Search

The two most obvious and well-investigated algorithms for the CUPBF optimization are: **1)** *the Greedy Single Switch Algorithm* (GSSA) and **2)** *the Random Single Switch Algorithm (RSSA)*. Both are local search iterative improvement algorithms proceeding by taking 1) the best and 2) a random better neighbor of the current iterate.

Both algorithms perform excellently in practice, in particular, on the randomly generated CUPBFs; see Section 5. However, the best known upper bounds are not very strong. Even worse, the GSSA *requires exponential time* optimizing some CUPBFs.

Theorem 3.1 *For optimizing an n-dimensional CUPBF:*

1. *the RSSA and the GSSA make $O(2^n/n)$ steps in the worst case (Mansour & Singh 1999, Björklund et al. 2002*b*);*

2. *the GSSA makes at least $\Omega(3^{n/3}) = \Omega(1.442^n)$ steps in the worst case (Wiedemann 1985).* □

Different deterministic single switch variations of the GSSA also require exponential time on simple stochastic games, as shown in (Melekopoglou & Condon 1994).

On some subclasses of CUPBFs the RSSA is polynomial. Call a CUPBF *fully absorbing* if it is one-dimensional, or has a fully absorbing facet with all values larger than their neighbors on the opposite facet.

Theorem 3.2 (Williamson Hoke (1988)) *The RSSA makes at most $O(n^2)$ expected steps on any fully absorbing CUPBF.* □

This led (Williamson Hoke 1988) to conjecture that the same holds for all CUPBFs.

3.2 Multiple Switch, Non-Local Search Algorithms

Complete unimodality allows for 'more aggressive' (compared with greedy or random single) optimization steps. Consider any vertex v_0 of the hypercube and a subcube generated by edges leading from v_0 to all its better neighbors. In this subcube v_0 is the unique minimum, so jumping from v_0 to any other vertex of the subcube guarantees an improvement. This gives rise to the following two *multiple switch* algorithms:

3) *All Profitable Switches Algorithm* (APSA) iteratively proceeds from a current iterate v_0 by simultaneously switching *all* coordinates corresponding to better neighbors of v_0 (i.e., jumping along the big diagonal in the subcube generated by all improving directions). In the context of simple stochastic games this algorithm is known as Hoffman-Karp's (Hoffman & Karp 1966, Condon 1993).

4) *Random Multiple Switches Algorithm* (RMSA) iteratively proceeds from a current iterate v_0 by simultaneously switching in *some* coordinates corresponding to better neighbors of v_0, each improving direction being chosen with probability 1/2 (i.e., jumping to some uniformly and randomly chosen vertex in the subcube generated by all improving directions).

Both algorithms behave excellently in practice, in particular, on randomly generated CUPBFs (Section 5). Only quite poor upper bounds are known for the APSA, but for the RMSA the best bound is somewhat better.

Theorem 3.3 (Björklund et al. (2002*b*), Mansour & Singh (1999)) *The RMSA makes $O(2^{0.775n}) = O(1.71^n)$ expected steps in the worst case, for optimizing an n-dimensional CUPBF. The APSA makes at most $2^n/n$ steps.*

3.3 The Folklore Algorithm

Randomly select a facet F of $\mathcal{H}$ and recursively find a maximum v^* in F. If the switch from v^* to the opposite facet is not profitable, then v^* is local (hence global) maximum, stop. Otherwise, find the optimum on the opposite facet. Obviously, this algorithms optimizes a CUPBF in expected $O((3/2)^n) = O(2^{0.58n})$ iterations.

3.4 Random Sampling

The APSA and the RMSA can be combined with the initial random sampling of the starting point for optimization, by selecting the best in a set of vertices chosen uniformly at random. The following is the first bound below $O(2^{n/2})$ for this type of algorithms; see (Tovey 1997).

Theorem 3.4 (Björklund et al. (2002*b*)) *The RMSA combined with random sampling makes $O(2^{0.46n}) = O(1.38^n)$ expected steps, for optimizing an n-dimensional CUPBF.* □

The following result (sketched in Section 6) is the first nontrivial lower bound for the APSA.

Theorem 3.5 *The APSA requires $\Omega(n^2)$ steps in the worst case.* □

The were no previously known nontrivial lower bounds for the APSA (Melekopoglou & Condon 1994, Mansour & Singh 1999). Actually, this algorithm performs excellently in practice, and it is the best on our random CUPBFs; see Section 5.

3.5 Kalai-style Randomized Algorithm

We were the first to observe (Björklund et al. 2001, Björklund et al. 2002*b*) that the following algorithm (similar to the folklore algorithm above, and also to a subexponential algorithm of (Kalai 1992) for linear programming, later adapted for simple stochastic games in (Ludwig 1995)) also gives a subexponential algorithm for the CUPBF optimization.

5) *Kalai-Ludwig's Randomized Algorithm* (KLRA) starts from a random vertex v_0 and proceeds as follows. A) Choose uniformly at random an $(n-1)$-dimensional face of the cube containing v_0 and recursively find an optimum v^* on this face. B) If v^* is better than its neighbor v' on the opposite face, the global optimum is found, terminate. Otherwise, switch to v' (i.e., $v_0 := v'$) decrease n by one, and repeat from A.

Theorem 3.6 (Björklund et al. (2001), Björklund et al. (2002*b*)) *The KLRA makes expected $2^{O(\sqrt{n})}$ steps to optimize a CUPBF.* □

The subexponential bound in Theorem 3.6 follows from a more involved (compared with the folklore algorithm) analysis of the KLRA, coupled with the CUPBF structure, and using the solution of a probabilistic recurrence from (Kalai 1992, Ludwig 1995). Thus KLRA is the first subexponential algorithm for the CUPBF problem. Another one, less efficient, but with the same upper bound may be derived from (Gärtner 1995). To improve the bound in Theorem 3.6 one should somehow 'exploit' the CUPBF structure more extensively.

A quadratic lower bound for an algorithm similar to the KLRA is settled in (Gärtner & Ziegler 1994), on Klee-Minty cubes. Our APSA and RMSA optimize such cubes just in $O(n)$ steps, and the RSSA does it in $O(n^2)$ expected steps.

Thus, no (sub)exponential, even superquadratic lower bounds are known for the APSA, RSSA, RMSA, KLRA on CUPBFs. Note however that for some *superclasses* of the class of all CUPBFs, providing for less structure, exponential lower bounds are known. For instance, any randomized algorithm for finding the maximum of a *unimax function* (possessing a unique local maximum), requires an expected $\Theta(n)2^{n/2}$ number of function evaluations in the worst case (Tovey 1997).

The following table summarizes the known bounds (cited above) on the number of steps for the different algorithms.

Algorithm	Lower bound	Upper bound
Greedy Single Switch, GSSA	$\Omega(3^{n/3})$	$2^{n-(1-\epsilon)\log n}$
Random Single Switch, RSSA	$\Omega(n^2)$	$2^{n-(1-\epsilon)\log n}$ $O(n^2)$ on fully absorbing cubes
All Profitable Switches, APSA	$\Omega(n^2)$	$2^{n-(1-\epsilon)\log n}$ $O(n)$ on Klee-Minty cubes $O(n)$ on fully absorbing cubes
Random Multiple Switches, RMSA	$\Omega(n)$	$2^{0.775n} = 1.71^n$ $2^{0.46n} = 1.38^n$ with random sampling $O(n)$ on fully absorbing cubes
Kalai-Ludwig's Randomized, KLRA	$\Omega(n^2)$	$2^{O(\sqrt{n})}$ $O(n^2)$ on Klee-Minty cubes

4 Generating Random Instances of CUPBFs

It was appealing to investigate the big discrepancies between known lower and upper bounds for algorithms described in Section 3, by running them on 'presumably hard' instances of CUPBFs. CUPBFs are complicated combinatorial objects. To our knowledge, there are no canonical ways of representing all CUPBFs in implicit form, and even their asymptotic number is unknown. All 'regular' ways of construction seemingly produce instances that are easy for one or all algorithms. We decided to create random generation schemes producing CUPBFs in explicit form, as enumerations of values for all hypercube vertices, and exhaustively run all algorithms from all points. Although this is very memory and time consuming (exponential in the number of dimensions), we feel that this approach is productive, because already in a few dimensions the diversity and sophistication of the CUPBF structure is astounding. Also, finding hard random instances in low dimensions is more probable.

In (Björklund et al. 2001) we used random CUPBF generation based on the following idea, similar to (Tovey 1986), but taking complete unimodality into account. Take a cube and assign the maximum value 2^n to one of its vertices. To avoid local maxima, the preceding biggest value $2^n - 1$ should be a random neighbor of the maximum. In general, if the k biggest values are already assigned to vertices V', the biggest remaining value $2^n - k$ should be given to a (random) neighbor v of some vertex in V'. Moreover, in every subcube containing v, it is either the first vertex assigned a value, or has a neighbor from V'. This algorithm was able to generate random CUPBFs of dimension up to 16. In this paper we describe a more powerful method, allowing us to go up to 27 dimensions with currently available memory.

Our CUPBF generation (see details in (Björklund et al. 2002*b*)) is based on Theorem 4.1, the converse of Theorem 2.5. Take a bijective function $g : \{0,1\}^n \to \{0,1\}^n$. In general, there may be no CUPBF f for which $g = VID_f$, because g may generate cycles (we have examples even in 3 dimensions). Let the binary 'improvement' relation I on $\{0,1\}^n$ generated by g be defined as follows: $I(u,v)$ iff u, v differ only in their i-th coordinate and the i-th bit of $g(u)$ is 1.

Theorem 4.1 *If $g : \{0,1\}^n \to \{0,1\}^n$ is bijective and the improvement relation I generated by g is acyclic then $g = VID_f$ for some CUPBF f. The latter can be obtained by topologically sorting I.* □

It follows that if u and v differ only in the i-th coordinate, then the i-th bit of $g(u)$ is 0 iff the i-th bit of $g(v)$ is 1. In particular, exactly one of $I(u, v)$ and $I(v, u)$ holds for all neighbors u and v.

This allows for a recursive 'divide-and-conquer' generation. Take two functions g_0, $g_1 : \{0, 1\}^{n-1} \to \{0, 1\}^{n-1}$, each bijective with acyclic improvement relation. From these create a function $g : \{0, 1\}^n \to \{0, 1\}^n$ with the same properties, as follows: for every $x \in \{0, 1\}^{n-1}$ define $g(0x) = b_0(x)g_0(x)$, $g(1x) = b_1(x)g_1(x)$ for some bits $b_{0,1}(x)$ that keep g satisfying the premises of Theorem 4.1. Selecting the bits $b_{0,1}(x)$ corresponds to linking the respective vertices of $(n - 1)$-dimensional cubes by directed edges, called *connecting edges*. We now describe two methods for drastically reducing the set of valid connections by partitioning edges into *classes of dependent edges* or simply *classes* for brevity. All edges in the same class must be given the same direction.

Define the relation R on the connecting edges as follows: $R((0x, 1x), (0y, 1y))$ iff $g_0(x) = g_1(y) \vee g_1(x) = g_0(y)$. The transitive closure R^+ of R can be proved to be an equivalence relation. To make g bijective, we need to guarantee $g_0(x) = g_1(y) \Rightarrow b_i(x) = b_i(y)$. Thus in every equivalence class C of R^+ we have to assign the same bit $b_0(u)$ to all vertices $0u$ such that $(0u, 1u) \in C$, and assign the opposite bit to $1u$.

We can further reduce the number of classes. Take $u, v \in \{0, 1\}^{n-1}$ so that $I(0u, 0v)$ and $I(1v, 1u)$, i.e., the edges between u and v are opposite in g_0 and g_1. This forces $b_i(u) = b_i(v)$. Otherwise, the two-dimensional face consisting of $0u, 0v, 1u, 1v$ is either cyclic or has two minima. We can thus merge classes containing $(0u, 1u)$ and $(0v, 1v)$.

As a result, the number of feasible solutions is exponential in the number of classes rather than in the number of vertices. Some of these possibilities give cycles and do not correspond to any CUPBF.

The algorithm for orienting classes iteratively modifies an assignment A of directions to classes. Initially A orients all classes except one (chosen uniformly at random) leftwards. Then the recursive backtracking Algorithm 1 looks for cycles and breaks them by flipping classes containing connecting edges which the cycle passes through. To avoid generating trivial test data, it never accepts an absorbing assignment (in which all vertices on one facet have outgoing edges to their neighbors on the opposite facet). It fails if the only possible assignment is absorbing.

The algorithm described above generates coordinates in order. Since the i-th bits of the left and right subcube were generated at the same depth of

Algorithm 1:

BREAK-CYCLES(A : assignment of directions to classes)

(1) **if** (A is absorbing) **then return** FAILURE

(2) **if** (A is acyclic) **then return** SUCCESS

(3) **foreach** (class c where the cycle goes from right to left)

(4) **if** (BREAK-CYCLES(A with c flipped) = SUCCESS)

(5) **return** SUCCESS

(6) **return** FAILURE

recursion, one might suspect some kind of dependency between them. To avoid this, we randomly 'scramble' generated cubes by randomly permuting the order of coordinates. This does not affect complete unimodality, which does not depend on the order of coordinates. Full details can be found in (Björklund et al. 2002*b*).

The running time of this algorithms is in the worst case exponential in the number of classes (because of backtracking in Algorithm 1), times linear in the number of vertices (the time to look for cycles). Classes are generated in a preprocessing step taking time linear in the size of the hypercube. However, the amount of backtracking in all our experiments has been negligible, and the number of classes has never exceeded 100. Thus the running time is practically linear in the size of the hypercube. It takes our program 7 hours to generate one 27-dimensional CUPBF on a modern computer and another 18 hours to run all algorithms from all starting points.

Although the suggested scheme was designed to create possibly hard instances of CUPBFs without giving advantages to any algorithms, it nevertheless revealed considerable differences in their worst-case running times. A generally good behavior of the algorithms on the generated cubes may be explained in several ways, in particular, by the polynomial time conjecture for CUPBFs (Williamson Hoke 1988, Tovey 1997).

Our program is available online at http://peg.it.uu.se/~cubes/.

5 Experimental Results on Randomly Generated Instances

All the algorithms were run starting from all vertices on hundreds CUPBFs of dimensions 3 to 27, generated randomly as described in Section 4. The randomized algorithms were run four times from each vertex on the CUPBFs of dimensions 3 through 20, twice from each vertex for dimensions 21 through 25, and once from each vertex for dimensions 26 and 27. We gathered the maximal number of iterations for each algorithm on each CUPBF, as well as the average over the different starting points on all CUPBFs.

In our experiments, the deterministic algorithms outperform the randomized ones. This is to be expected, since the main motivation for randomization lies in evening out the running time over the set of all inputs. It is, however, somewhat surprising that the GSSA is the algorithm with the best exhibited behavior (together with the APSA). This to some extent contradicts the results presented in (Björklund et al. 2001), which indicated that both the multiple switching algorithms (the APSA and the RMSA) performed better than all the single switching algorithms. This could be explained by different methods of random CUPBF generation.

The average and worst exhibited behaviors of the algorithms over all vertices and generated CUPBFs are summarized in Figure 1 and Figure 2, respectively. The numbers are in Appendix A. The general picture is that the APSA and the GSSA showed the best performance, followed by the RMSA. The RSSA performed significantly worse, while the KLRA has a worst exhibited behavior that is very much worse than that of all the other algorithms.

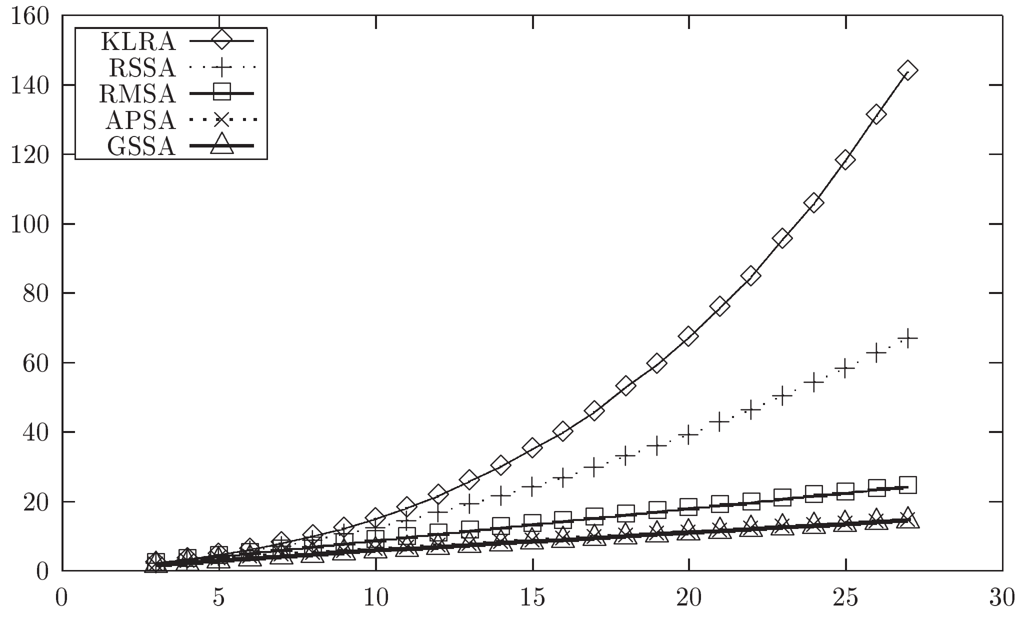

Figure 1: Average number of iterations for five algorithms run from each vertex of CUPBFs of dimensions 3–27. Results are shown for more than 400 CUPBFs of dimensions 3–24, 277 of dimension 25, and more than 195 of dimensions 26 and 27. Randomized algorithms started four times from each vertex in dimensions 3–20, twice per vertex in dimensions 21–25, and once per vertex in dimensions 26 and 27.

6 APSA-Quadratic Instances

There were no previously known nontrivial lower bounds on the worst-case behavior of the APSA. We briefly present a new class of CUPBFs (called APSA-cubes) where APSA requires $\Theta(n^2)$ iterations, both from the worst starting point and averaged over all starting points.

Theorem 6.1 *There is a class of CUPBFs for which APSA requires* $\binom{n}{2}+1$ *iterations in the worst case.*

Proof. We will construct a partial order $\prec$ on $\{0,1\}^n$ such that any $f : \{0,1\}^n \to \mathbb{R}$ satisfying $x \prec y \Rightarrow f(x) < f(y)$ is a CUPBF that foils APSA

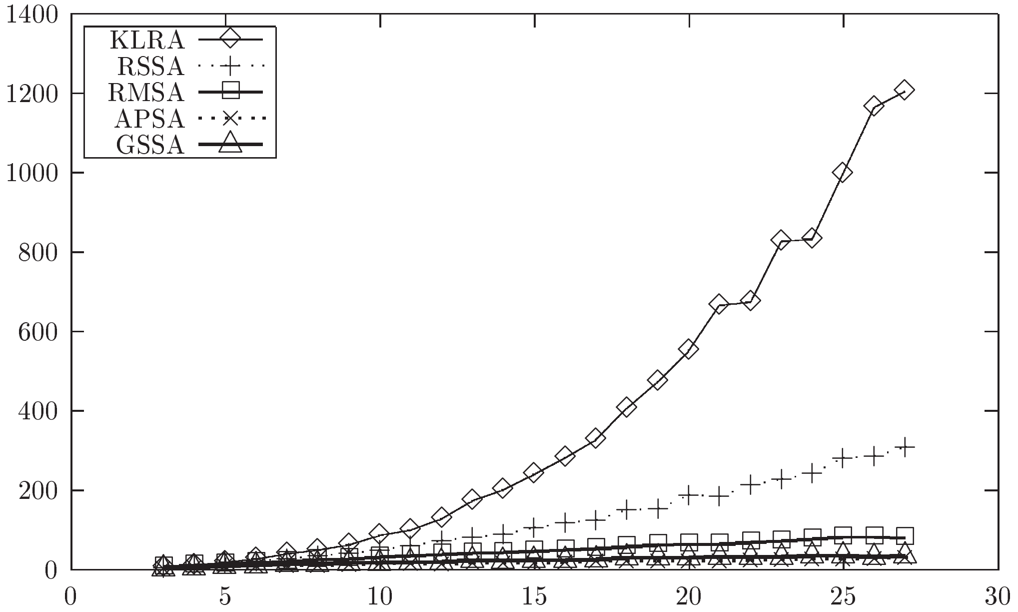

Figure 2: Worst number of iterations in the same experiments as in Figure 1.

forcing it to make quadratically many steps.

Partition the coordinate vectors into classes G_i for $i \in \{0, \ldots, n\}$ as follows. A *b-block*, $b \in \{0, 1\}$, in a coordinate vector x is a maximal sequence of consecutive b's. Denote by $\beta_b(x)$ the number of b-blocks in x.

- If $i \equiv 0 \pmod 4$ then $G_i = \{x \in \{0,1\}^n : \beta_1(x) = i/2 \wedge x_1 = 0\}$,
- if $i \equiv 1 \pmod 4$ then $G_i = \{x \in \{0,1\}^n : \beta_1(x) = \lceil i/2 \rceil \wedge x_n = 1\}$,
- if $i \equiv 2 \pmod 4$ then $G_i = \{x \in \{0,1\}^n : \beta_0(x) = i/2 \wedge x_1 = 1\}$,
- if $i \equiv 3 \pmod 4$ then $G_i = \{x \in \{0,1\}^n : \beta_0(x) = \lceil i/2 \rceil \wedge x_n = 0\}$.

It can be shown that $\{G_i\}_{i=0}^n$ partitions $\{0,1\}^n$ and $|G_i| = \binom{n}{i}$.

Now we define the partial order $\prec$ on $\{0,1\}^n$ $x \in G_i, y \in G_j$. If $i < j$ then $x \succ y$. If $i = j$ then let $S : \{0,1\}^n \to \mathcal{P}\{1, \ldots, n-1\}$, $S(x_1 \ldots x_n) = \{k : x_k \neq x_{k+1}\}$ and set $L(x) = \sum_{k \in S(x)} k$. Define $x \succ y$ if $L(x)(-1)^i - n|S(x)| < L(y)(-1)^i - n|S(y)|$.

Take any $f : \{0,1\}^n \to \mathbb{R}$ satisfying $x \prec y \Rightarrow f(x) < f(y)$. It can be proved that for $x \in G_i$, $VID_f(x)$ has i 1's, located adjacent to borders of 0-blocks and 1-blocks. VID_f does not depend on the choice of f. Using this characterization of VID_f, Theorem 2.5 implies that f is a CUPBF. It also follows that APSA, after starting from the unique vertex in G_n, visits $n-i$ vertices in G_i for $i = n-1, n-2, \ldots, 1$, and then visits the unique vertex in G_0. Thus the number of steps is $1 + \sum_{i=1}^{n}(n-i) = n(n-1)/2 + 1$. □

6.1 Further Bounds on APSA-cubes.

All our algorithms are *switching algorithms*, proceeding by flipping bits in the coordinate vector x corresponding to 1's in $VID_f(x)$. Using the characterization of VID_f from Theorem 6.1 we can show that the maximum total number of switches taken by any switching algorithm is $O(ni)$ in G_i. Thus not only APSA but also RMSA (expected to take a number of switches linear in i each step) needs at most $O(n^2)$ iterations. The same reasoning gives a cubic upper bound for any single-switch algorithm, in particular RMSA, RSSA and KLRA.

It can also be shown that GSSA, on a subclass of the APSA-cubes, needs $\Omega(n^3)$ iterations in the worst case. Although this is worse than the quadratic bound for APSA, we have not emphasized it because GSSA is already known exponential on another class of CUPBFs (Wiedemann 1985).

By exhaustively generating all CUPBFs, we have also found that APSA needs strictly more iterations on APSA-cubes than on any other CUPBF of dimension ≤ 4.

7 Experimental Results on APSA-Quadratic Instances

As explained in Section 6.1, we have tight quadratic and cubic bounds for APSA and GSSA, respectively, on the APSA-cubes. However, for RMSA we only have a quadratic upper bound and for KLRA and RSSA a cubic one. Since the algorithms appear better than these bounds in our experiments on randomly generated CUPBFs (see Section 5), it is relevant to investigate how they behave on APSA-cubes.

We ran RMSA, KLRA, and RSSA at least 40 times from each vertex of APSA-cubes of dimensions 3–24 and 9 times from each vertex of APSA-cubes of dimensions 25–27. We also ran them from at least 2^{29} random vertices on APSA-cubes of dimensions 28–32. For each dimension, we computed the maximal as well as average number of steps over all runs from all starting points. The results are summarized in Figures 3 and 4, respectively. The numbers are in Appendix A.

All three algorithms behave surprisingly well in experiments. As in the experiments on random CUPBFs, RMSA is the best of the randomized algorithms. Unlike in those experiments, however, KLRA is about as efficient as RSSA. Also, the randomized algorithms now outperform the deterministic ones.

The decrease in worst number of iterations for KLRA and RSSA from dimension 24 to 25 can possibly be explained by the fact that we made fewer experiments on 25-dimensional APSA-cubes, thus decreasing the chance of "bad luck" with random numbers. The same cannot be said about the decrease for RSSA from dimension 30 to 31 since the number of experiments was constant. We do not have any explanation of this, except possibly random deviations (note that not all starting points were tested).

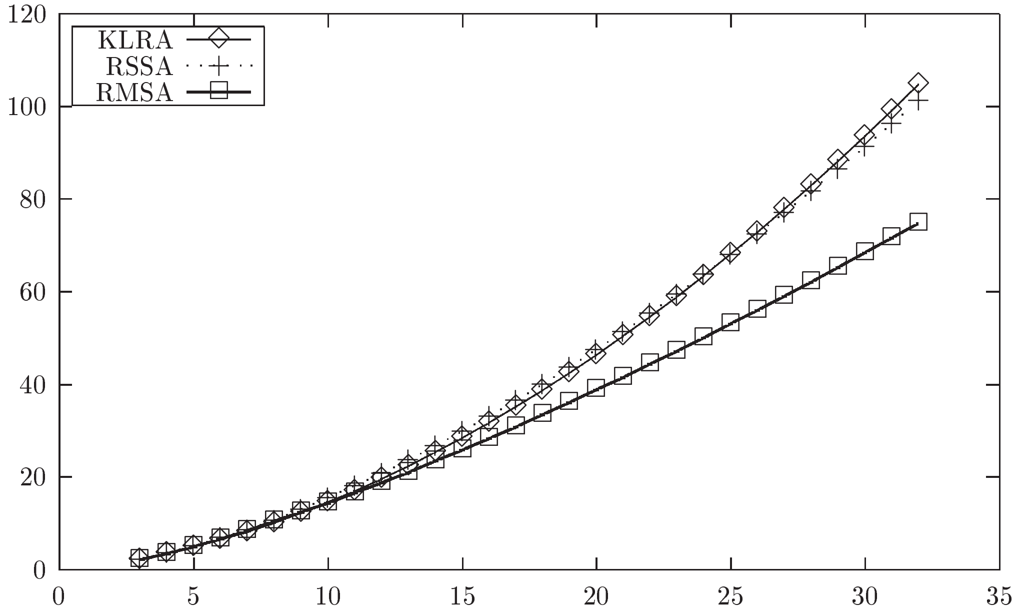

Figure 3: Average number of iterations for three algorithms running from each corner of APSA-cubes of dimensions 3–32. The algorithms started at least 40 times per vertex in dimensions 3–24 and 9 times per vertex in dimensions 25–27. In dimensions 28–32, they started from more than 2^{29} randomly chosen vertices.

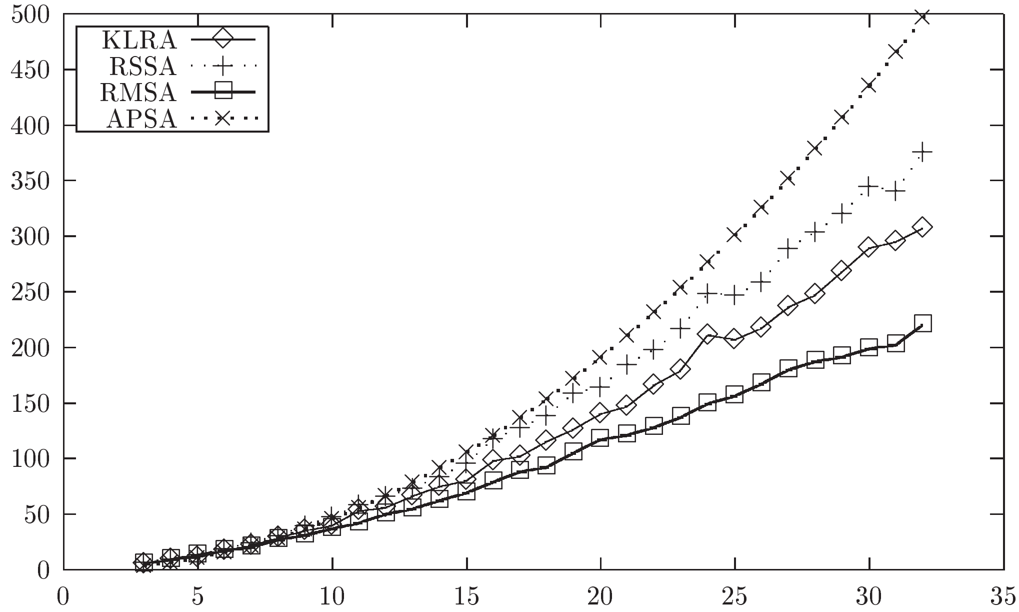

Figure 4: Worst numbers of iterations for the same experiments as in Figure 3. The results for APSA, included for reference, are exact calculations and not experimental data.

A Tables of Experimental Data

In this section we give the numerical data represented in Figures 1, 2, 3, and 4.

The table below contains the values used in Figure 1, where n is the dimension (x axis in the figure).

n	KLRA	RSSA	RMSA	APSA	GSSA
3	2.1	2.1	2.1	2.0	1.6
4	3.2	3.2	3.0	2.9	2.2
5	4.5	4.4	4.0	3.5	2.9
6	6.0	5.8	4.9	4.1	3.5
7	7.8	7.2	5.9	4.7	4.1
8	9.9	8.9	6.8	5.3	4.6
9	12.2	10.6	7.7	5.8	5.2
10	14.9	12.6	8.7	6.4	5.8
11	18.0	14.6	9.6	6.9	6.3
12	21.6	16.9	10.5	7.4	6.8
13	25.7	19.3	11.5	7.9	7.4
14	30.1	21.8	12.4	8.3	7.8
15	34.9	24.3	13.3	8.9	8.4
16	39.8	26.9	14.2	9.4	8.9
17	45.7	29.8	15.0	9.8	9.4
18	52.8	33.0	16.0	10.3	9.9
19	59.4	36.0	16.9	10.9	10.4
20	67.0	39.3	17.7	11.3	10.9
21	75.8	42.9	18.7	11.7	11.4
22	84.5	46.4	19.6	12.3	11.9
23	95.3	50.4	20.5	12.8	12.4
24	105.7	54.3	21.5	13.3	12.9
25	117.8	58.4	22.3	13.8	13.4
26	131.0	62.9	23.3	14.3	14.0
27	143.7	67.0	24.1	14.7	14.5

The table below contains the values used in Figure 2, where n is the dimension (x axis in the figure).

n	KLRA	RSSA	RMSA	APSA	GSSA
3	7	7	7	4	4
4	12	13	11	7	6
5	19	17	15	9	10
6	27	23	17	10	11
7	40	29	21	11	14
8	50	36	26	12	14
9	63	42	27	13	17
10	87	50	32	16	18
11	100	62	36	16	19
12	129	75	39	15	20
13	175	82	42	17	24
14	202	91	44	17	22
15	240	107	46	18	24
16	283	120	50	20	25
17	328	126	54	22	26
18	407	151	58	22	30
19	474	155	63	22	30
20	551	189	65	22	30
21	666	186	65	23	32
22	674	215	70	25	33
23	827	229	72	26	32
24	833	243	77	28	35
25	996	282	83	28	35
26	1165	287	82	29	33
27	1205	309	81	30	35

The tables below contain the average values used in Figure 3 and the maximum values used in Figure 4, respectively. Here n is the dimension (x-axis in the figures). The rightmost column in the right table, labelled APSA, contains exact calculations and not experimental data.

n	KLRA	RSSA	RMSA
3	2.2	2.4	2.2
4	3.6	3.7	3.4
5	4.9	5.2	4.9
6	6.4	7.0	6.7
7	8.1	8.7	8.4
8	10.0	10.7	10.4
9	12.3	13.1	12.4
10	14.6	15.6	14.3
11	16.9	18.1	16.6
12	19.5	20.8	18.8
13	22.4	23.7	21.0
14	25.3	26.7	23.4
15	28.5	29.9	25.8
16	31.7	33.2	28.2
17	35.1	36.6	30.8
18	38.7	40.1	33.4
19	42.4	43.8	36.0
20	46.3	47.5	38.7
21	50.4	51.4	41.5
22	54.6	55.4	44.3
23	58.9	59.5	47.1
24	63.4	63.8	50.0
25	68.1	68.1	53.0
26	72.9	72.5	56.0
27	77.8	77.1	59.0
28	82.9	81.7	62.1
29	88.2	86.5	65.2
30	93.6	91.3	68.3
31	99.1	96.3	71.5
32	104.8	101.3	74.7

n	KLRA	RSSA	RMSA	APSA
3	5	7	5	4
4	9	10	9	7
5	11	14	13	11
6	17	19	17	16
7	22	25	20	22
8	29	31	27	29
9	35	40	31	37
10	40	48	37	46
11	53	59	42	56
12	56	66	50	67
13	66	74	55	79
14	75	84	62	92
15	80	96	69	106
16	98	118	79	121
17	102	128	88	137
18	115	139	92	154
19	126	159	105	172
20	140	165	117	191
21	147	185	121	211
22	166	198	128	232
23	179	217	137	254
24	211	249	149	277
25	207	247	156	301
26	217	259	167	326
27	236	289	180	352
28	247	304	187	379
29	268	321	191	407
30	289	345	199	436
31	295	341	202	466
32	307	376	220	497

References

Björklund, H., Petersson, V. & Vorobyov, S. (2001), Experiments with iterative improvement algorithms on completely unimodal hypercubes, Technical Report 2001-017, Information Technology/Uppsala University. `http://www.it.uu.se/research/reports/`.

Björklund, H., Sandberg, S. & Vorobyov, S. (2002*a*), A discrete subexponential algorithm for parity games, Technical Report 2002-026, Department of Information Technology, Uppsala University. `http://www.it.uu.se/research/reports/`.

Björklund, H., Sandberg, S. & Vorobyov, S. (2002*b*), Optimization on completely unimodal hypercubes, Technical Report 018, Uppsala University / Information Technology. `http://www.it.uu.se/research/reports/`.

Boros, E. & Hammer, P. L. (2001), Pseudo-boolean optimization, Technical Report RRR 48-2001, RUTCOR Rutger Center for Operations Research.

Condon, A. (1993), 'On algorithms for simple stochastic games', *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **13**, 51–71.

Emerson, E. A. (1997), Model checking and the Mu-calculus, *in* N. Immerman & P. G. Kolaitis, eds, 'DIMACS Series in Discrete Mathematics', Vol. 31, pp. 185–214.

Gärtner, B. (1995), 'A subexponential algorithm for abstract optimization problems', *SIAM Journal on Computing* **24**, 1018–1035.

Gärtner, B. & Ziegler, G. M. (1994), Randomized simplex algorithms on Klee-Minty cubes, *in* 'FOCS '94'.

Hammer, P. L., Simeone, B., Liebling, T. M. & De Werra, D. (1988), 'From linear separability to unimodality: a hierarchy of pseudo-boolean functions', *SIAM J. Disc. Math.* **1**(2), 174–184.

Hoffman, A. J. & Karp, R. M. (1966), 'On nonterminating stochastic games', *Management Science* **12**(5), 359–370.

Kalai, G. (1992), A subexponential randomized simplex algorithm, *in* '24th ACM STOC', pp. 475–482.

Ludwig, W. (1995), 'A subexponential randomized algorithm for the simple stochastic game problem', *Information and Computation* **117**, 151–155.

Mansour, Y. & Singh, S. (1999), On the complexity of policy iteration, *in* 'Uncertainty in Artificial Intelligence'99'.

Melekopoglou, M. & Condon, A. (1994), 'On the complexity of the policy improvement algorithm for Markov decision processes', *ORSA Journal of Computing* **6**(2), 188–192.

Petersson, V. & Vorobyov, S. (2001), 'A randomized subexponential algorithm for parity games', *Nordic Journal of Computing* **8**, 324–345.

Tovey, C. A. (1986), 'Low order polynomial bounds on the expected performance of local improvement algorithms', *Mathematical Programming* **35**, 193–224.

Tovey, C. A. (1997), Local improvement on discrete structures, *in* E. Aarts & L. J. K., eds, 'Local Search in Combinatorial Optimization', John Wiley & Sons, pp. 57–89.

Wiedemann, D. (1985), 'Unimodal set-functions', *Congressus Numerantium* **50**, 165–169.

Williamson Hoke, K. (1988), 'Completely unimodal numberings of a simple polytope', *Discrete Applied Mathematics* **20**, 69–81.

Recent technical reports from the Department of Information Technology

2002-011	Sheayun Lee, Andreas Ermedahl, Sang Lyul Min and Naehyuck Chang: *Statistical Derivation of an Accurate Energy Consumption Model for Embedded Processors*
2002-012	Lars Ferm and Per Lötstedt: *Accurate and Stable Grid Interfaces for Finite Volume Methods*
2002-013	Jakob Engblom: *Effects of Branch Predictors on Execution Time*
2002-014	Samuel Sundberg and Lina von Sydow: *Analysis of a semi-Toeplitz preconditioner for a convection-diffusion problem*
2002-015	Andreas Johnsson: *Differentiated Security in Wireless Networks*
2002-016	Fredrik Edelvik: *A New Technique for Accurate and Stable Modeling of Arbitrarily Oriented Thin Wires in the FDTD Method*
2002-017	Eva Olsson, Stefan Seipel, Anders Jansson and Bengt Sandblad: *The Windscreen Used as a Display for Navigation Information. An introductory study*
2002-018	Henrik Björklund, Sven Sandberg and Sergei Vorobyov: *Optimization on Completely Unimodal Hypercubes*
2002-019	Erik Bängtsson, Daniel Noreland and Martin Berggren: *Shape optimization of an acoustic horn*
2002-020	Tobias Amnell, Elena Fersman, Paul Pettersson, Hongyan Sun and Wang Yi: *Code Synthesis for Timed Automata*
2002-021	Johan Runeson and Sven-Olof Nyström: *Generalizing Chaitin's Algorithm: Graph-Coloring Register Allocation for Irregular Architectures*
2002-022	Henrik Lundgren, Erik Nordström and Christian Tschudin: *Coping with Communication Gray Zones in IEEE 802.11b based Ad hoc Networks*
2002-023	Lars Ferm and Per Lötstedt: *Anisotropic grid adaptation for Navier-Stokes' equations*
2002-024	Jakob Engblom and Magnus Nilsson: *Time Accurate Simulation: Making a PC Behave Like a 8-Bit Embedded CPU*
2002-025	Paul Petterson and Wang Yi (eds.): *Workshop on Real-Time Tools (Proceedings)*
2002-026	Henrik Björklund, Sven Sandberg and Sergei Vorobyov: *A Discrete Subexponential Algorithm for Parity Games*
2002-027	Markus Nordén, Sverker Holmgren and Michael Thuné: *OpenMP versus MPI for PDE Solvers Based on Regular Sparse Numerical Operators*
2002-028	Bengt Eliasson: *Numerical Modelling of the Two-Dimensional Vlasov-Maxwell System*
2002-029	Bengt Eliasson: *Domain Decomposition of the Padé Scheme and Pseudo-Spectral Method, Used in Vlasov Simulations*
2002-030	Henrik Björklund, Sven Sandberg and Sergei Vorobyov: *An Experimental Study of Algorithms for Completely Unimodal Optimization*

UPPSALA
UNIVERSITET

October 2002
ISSN 1404-3203
http://www.it.uu.se/