

Dominator search for skylines

Patrick Rooks, Markus Endres

Angaben zur Veröffentlichung / Publication details:

Rooks, Patrick, and Markus Endres. 2016. "Dominator search for skylines."
Augsburg: Universität Augsburg.

Nutzungsbedingungen / Terms of use:

licgercopyright

Dieses Dokument wird unter folgenden Bedingungen zur Verfügung gestellt: / This document is made available under the following conditions:

Deutsches Urheberrecht

Weitere Informationen finden Sie unter: / For more information see:

<https://www.uni-augsburg.de/de/organisation/bibliothek/publizieren-zitieren-archivieren/publizieren>



UNIVERSITÄT AUGSBURG

Dominator Search for Skylines

P. Roocks M. Endres

Report 2016-05

August 2016

INSTITUT FÜR INFORMATIK
D-86135 AUGSBURG

Copyright © P. Rooks M. Endres
Institut für Informatik
Universität Augsburg
D-86135 Augsburg, Germany
<http://www.Informatik.Uni-Augsburg.DE>
— all rights reserved —

Dominator Search for Skylines

Patrick Rooks, Markus Endres

Institut für Informatik, Universität Augsburg, D-86135 Augsburg, Germany
{rooks,endres}@informatik.uni-augsburg.de

Abstract. For a given set of Skyline points, we consider the problem of finding a minimal set of new points which dominates all given Skyline points. The placement of the new points is subject to a given feasibility condition. We propose an optimal and efficient algorithm for the 2-dimensional case. Regarding the generalized d -dimensional ($d \geq 3$) problem we present different efficient heuristics approximating the minimal set of dominators.

Keywords Skyline, dominator, optimization

1 Motivation

Assume a business use case where company A has a set of products which are Pareto-optimal [Kie02,Cho03] w.r.t. some given customer preference. This preference is assumed to be a Pareto order, which implies that the set of products forms a Skyline. Now company B wants to find a set of products which commonly Pareto-dominates all products from A . More precisely, for every A -product there exists some B -product dominating that A -product. As the developing of new products is expensive, the portfolio of the new B -products should be minimal. Clearly, new products cannot be placed arbitrarily within the domain. For example, there are technical limitations for hardware products or regulatory limitations for financial products. We assume that an explicit feasibility function is given, specifying the limitations where new products can be placed. The problem of finding a minimal set of new products, dominating the old ones under the given feasibility conditions, is abstractly described by the *dominator search problem for Skylines*.

In the following we will formalize the problem in Section 2 and present an efficient and exact solution for the 2-dimensional case in Section 3. For the d -dimensional problem (with $d \geq 3$) there is an exact solution with exponential complexity (Section 4), hence we show different efficient approximating heuristics (which are not exact) in Section 5. Finally we present some benchmarks in Section 6, related work in Section 7, and a conclusion in Section 8.

2 The Problem

First we describe the general d -dimensional problem. In the following, the domain will be restricted to $\mathbb{R}_+^d =_{df} \{x \in \mathbb{R}^d \mid x \geq 0\}$. We give a formal description of the problem together with some useful definitions for the following algorithms.

2.1 Skylines

At first we define the Skyline $\mathcal{S}(M)$ of a set $M \subseteq \mathbb{R}_+^d$ (cp. [BKS01,Kie02]). Assume a set of vectors $M \subseteq \mathbb{R}_+^d$. We define the Pareto ordering $<_{\otimes}$ for all $x = (x_1, \dots, x_d), y = (y_1, \dots, y_d) \in M$:

$$x <_{\otimes} y \iff \forall j \in \{1, \dots, d\} : x_j \leq y_j \wedge \exists i \in \{1, \dots, d\} : x_i < y_i . \quad (1)$$

The *Skyline* of M is defined by the minima in M according to the ordering $<_{\otimes}$, or explicitly by the set

$$\mathcal{S}(M) = \{t \in M \mid \nexists u \in M : u <_{\otimes} t\} .$$

We call a set M a *Skyline set* if it is invariant under \mathcal{S} , i.e., $\mathcal{S}(M) = M$. We say that x *Pareto-dominates* y iff $x <_{\otimes} y$. This means that finding the Pareto-optimal tuples simultaneously minimizes all d dimensions. In consequence, the tuple $(0, \dots, 0) \in \mathbb{R}_+^d$ Pareto-dominates all other tuples in \mathbb{R}_+^d .

2.2 Dominator Search

In the following we define the general problem of searching dominators in the d -dimensional case. Assume a feasibility area $F \subseteq \mathbb{R}_+^d$ and a feasible Skyline set $S \subseteq F$ which shall be dominated by some new points in F . The problem of searching the smallest possible set of dominators $D \subseteq F$ dominating all points in S is formally given by the minimization problem

$$\begin{aligned} & |D| \rightarrow \min \\ \text{under } & D \subseteq F, \\ & \mathcal{S}(D \cup S) = D . \end{aligned}$$

This means, we search for a minimal set D containing a dominating (or equivalent) point for each point in S . The second condition of the problem is formally equivalent to

$$\forall s \in S : \exists p \in D : p <_{\otimes} s \vee p = s ,$$

where the latter predicate equals the usual product order $p \leq s$. In most cases we will have $D \cap S = \emptyset$. If $|D|$ is minimal *and* some Skyline point $s \in S$ has no Pareto-dominating point, then this implies $s \in D$ and s is also called a dominator in the following (a dominator of its own). This can only occur when s is on the border of the feasibility area F . Regardless of the minimization goal, the boundary conditions of this minimization problem can be trivially fulfilled for $D = S$. Next we give some definitions which will turn out to be helpful for our algorithmic approach to the problem.

Definition 2.1 (Pareto Hull). *The Pareto hull for a set $M \subseteq \mathbb{R}_+^d$ contains all points from the domain contained in M or dominated by some point in M , formally*

$$\mathcal{H}(M) = \{x \in \mathbb{R}_+^d \mid x \in M \vee \exists y \in M : y <_{\otimes} x\} .$$

2.3 Bounding Boxes

The main idea to find all possible dominator candidates to dominate points in S is to consider subsets of S and to find bounding boxes for these subsets. Any lower bound for a subset of points is a dominator candidate for the points in the respective subset. As the dominator search is restricted to feasible points, all candidates have to be feasible, i.e., contained in F . In d dimensions we need all subsets of d points to get all possible bounding boxes. In the following we define the *bounding box limits* formally and show that it is sufficient to search for dominators within them.

Definition 2.2 (Bounding Box Limits). *Let $S \subseteq \mathbb{R}_+^d$ be a Skyline set, and F the feasibility set, i.e. $S \subseteq F$. We define the set of lower bounds of each d -subset of S w.r.t. F by:*

$$\text{bbl}(S, F, d) := \bigcup_{Q \subseteq S, |Q| \leq d} \text{pick}(\{r \in F \mid \forall q \in Q : r \leq q\}) ,$$

where $\text{pick}(\cdot)$ chooses one arbitrary element of the set. Semi-formally we can denote $\text{pick}(\cdot)$ as a non-deterministic function by

$$\begin{aligned} \text{pick}(\{m_1, \dots, m_k\}) &= \{m_1\} , \\ \text{pick}(\emptyset) &= \emptyset . \end{aligned}$$

When searching for dominators we will consider only the feasible bounding box limits as candidates. In the following lemma we will state that this is sufficient.

Lemma 2.3. *Let $S \subseteq \mathbb{R}_+^d$ some Skyline set and $F \subseteq \mathbb{R}_+^d$ a feasibility area. Let $p \in F$ some point dominating or being equivalent to all points in $Q \subseteq S$, i.e., $\forall q \in Q : p \leq q$, i.e., p is a dominator for Q . Then there exists some point $p' \in \text{bbl}(S, F, d)$ which is also a dominator for Q .*

Proof. We first consider the arguments of the lower bounds of the set Q in each dimension, i.e., the points $r^{(i)} \in \mathbb{R}_+^d$ specifying the size of the bounding box in dimension i ,

$$r^{(i)} = \arg \min \{q_i \mid (q_1, \dots, q_d) \in Q\} \text{ for all } i \in \{1, \dots, d\} .$$

Now we consider a point p' which is a feasible lower bound for all points in $\{r^{(1)}, \dots, r^{(d)}\}$, i.e.,

$$p' \in F \quad \wedge \quad \forall q \in \{r^{(1)}, \dots, r^{(d)}\} : p' \leq q .$$

As $|\{r^{(1)}, \dots, r^{(d)}\}| \leq d$, any point chosen by $\text{pick}(\{r \in F \mid \forall q \in Q : r \leq q\})$ in Definition 2.2 fulfills this condition, i.e., is a lower bound of a d -subset of S . We see that p' is smaller than any point of Q in each dimension, hence it is clear that such a p' (if it exists) is a dominator for Q .

To see that p' exists, i.e., that there is such a feasible point, consider

$$\tilde{p} = (\min Q_1, \dots, \min Q_d) \quad \text{where } Q_i = \{q_i \mid (q_1, \dots, q_d) \in Q\} \text{ for all } i \in \{1, \dots, d\} .$$

The point \tilde{p} is the greatest lower bound of all points in Q . As p is required to be a dominator for Q , we have $p \leq \tilde{p}$ by definition of the dominance property. Hence $p' = p$ is a possible feasible candidate dominating all points in Q and p' is a lower bound of a d -subset as shown above. This implies $p' \in \text{bbl}(S, F, d)$ and shows the claim. \square

The complexity of the dominator search mainly depends on the size of the candidate set. Hence we are interested in an upper estimate of $|\text{bbl}(S, F, d)|$. For number of subsets in S having a cardinality $\leq d$ we get that

$$|\text{bbl}(S, F, d)| \leq |\{Q \subseteq S \mid |Q| \leq d\}| = \sum_{i=1}^d \binom{|S|}{i} \leq |S|^d \quad (2)$$

holds, where $\binom{|S|}{i}$ over i is the binomial coefficient. This is a very rough upper estimate, as all bounding box limits not belonging to the feasibility area F are not contained in $\text{bbl}(S, F, d)$ but counted in this estimate.

3 Solution for the 2-Dimensional Case

Now we will describe an algorithm solving the problem for the 2-dimensional case. In contrast to the higher dimensional case, we can show an efficient, i.e. polynomial, algorithm. We specify the algorithm, show its correctness and present some examples.

3.1 The Algorithm

Our approach starts with finding the Pareto hull of the feasibility area F . We pick the left most point $(x_m, y) \in S$ which shall be dominated and project it in y -direction to the y -maximal point of Pareto hull. This yields a new dominator p , dominating or being equivalent to (x_m, y) , and potentially dominating some more points. All dominated points $q \in S$ fulfill $p \leq q$. We remove them, i.e., we proceed with the set $S := S \setminus \{q \in S \mid p \leq q\}$ and iterate this process to find the next dominator. The iteration stops if there are no more points left in S , i.e., all points of S are dominated.

We formalize this description in the pseudocode given in Algorithm 1. A visual example of searching a new dominator is given in the Figures 3 and 4.

Note that the set of points which is removed from S contains the Pareto-dominated points and the dominator p itself, i.e., $\{q \in S \mid p \leq q\} = \{q \in S \mid p <_{\otimes} q\} \cup \{p\}$. This equality follows directly from the definition of \otimes , as the non-strict product order \leq is the relation-algebraic union of $<_{\otimes}$ and the identity.

At first we analyze the time complexity of the algorithm. The while loop (Line 4) adds one new dominator to the result set in every loop run. Within the loop we have a complexity of $O(|S|)$ for removing the points in S , Line 10. Hence the complexity of the entire algorithm is $O(|S| \cdot |D|)$. By $|D| \leq |S|$ we get the upper estimate $O(|S|^2)$.

We proceed by stating the correctness of the algorithm, i.e., we state that our method minimizes the set of dominators.

Algorithm 1 2-Dimensional Dominator Search, Exact Solution**Input:** Skyline set $S \subseteq \mathbb{R}_+^2$ to dominate, feasibility area $F \subseteq \mathbb{R}_+^2$ **Output:** Optimal dominator set D

```

1: function DOMINATOR_SEARCH_2DIM( $S, F$ )
2:    $H \leftarrow \mathcal{H}(F)$  // Get Pareto hull of feasibility area
3:    $D \leftarrow \emptyset$  // Initial dominator set
4:   while  $S \neq \emptyset$  do
5:      $x_m \leftarrow \min\{u \mid (u, v) \in S\}$  // Find minimal  $x$ -value of remaining points
6:      $y \leftarrow \min\{v \mid (u, v) \in H \wedge u = x_m\}$  // Calculate  $y$ -projection of  $x_m$  to Pareto hull
7:      $x \leftarrow \max\{u \mid (u, v) \in F \wedge v = y\}$  // Find according feasible point to hull point  $h = (x_m, y)$ 
8:      $p \leftarrow (x, y)$ 
9:      $D \leftarrow D \cup \{p\}$  // Add new dominator  $p$ 
10:     $S \leftarrow S \setminus \{q \in S \mid p \leq q\}$  // Remove dominated points
11:  end while
12:  return  $D$ 
13: end function

```

Lemma 3.1. *Algorithm 1 finds the minimal set of dominators.*

Proof. (Idea) Order all points by ascending x -coordinates. In any optimal dominator set, there must be some dominator which dominates the Skyline point $s_{\min} \in S$ with lowest x -coordinate. By construction, Algorithm 1 picks the right-most dominating point p for s_{\min} , which is obtained by the y -projection of s_{\min} . Hence p dominates the greatest possible set of points in S , given that $p \leq s_{\min}$ holds. Any algorithm placing the dominator for s_{\min} somewhere else cannot be better (but perhaps equally good). After removing the dominated points in Line 10 we iteratively apply the argument to the remainder of the points. Hence one can show inductively that the greedy strategy chooses a minimal set of dominators. \square

We omit a more detailed proof, as Algorithm 1 makes use of the usual *dynamic programming* principle which is a standard principle used by many greedy approaches.

Note that there is a symmetric version of the algorithm ordering the points by x -coordinates descendingly and projecting them in x -direction. There we can apply the same argument; this is also an optimal solution to the problem. This insight offers an approach for a multi-core implementation: We can split the “left \rightarrow right” and “right \rightarrow left” searches in two threads. The search terminates when the sets of covered points in S from both threads overlaps.

3.2 Implementation and Examples

To offer a convenient way to specify the feasibility area and to simplify the calculation of $\mathcal{H}(F)$ we make some assumptions on the feasibility area. Assume a continuous function $\gamma : [0, 1] \rightarrow \mathbb{R}_+^d$ fulfilling the requirements

$$\begin{aligned}\gamma(0) &= (0, y_0) , \\ \gamma(1) &= (x_0, 0) .\end{aligned}$$

With such a function we specify the feasibility area in our implemented use cases and in the following examples. The feasibility area is derived from γ as explained in the following. We define the complement of the feasibility area, $\mathbb{R}_+^d \setminus F$ as the area surrounded by the following segments:

1. The line from $(0, 0)$ to $(0, y_0)$,
2. the curve of $\gamma(t)$ for $t \in [0, 1]$,
3. the line from $(x_0, 0)$ to $(0, 0)$.

Example 3.2. Assume the following definition for the function restricting the feasibility area

$$\gamma(t) = (3 \cdot t, 9/(4 \cdot (t + 0.5)) - 3/2) .$$

We have $\gamma(0) = (0, 3)$ and $\gamma(1) = (3, 0)$ hence this function fulfills the requirements given above. The Pareto Hull of the induced feasibility F area is identical to the feasibility area, i.e., $\mathcal{H}(F) = F$.

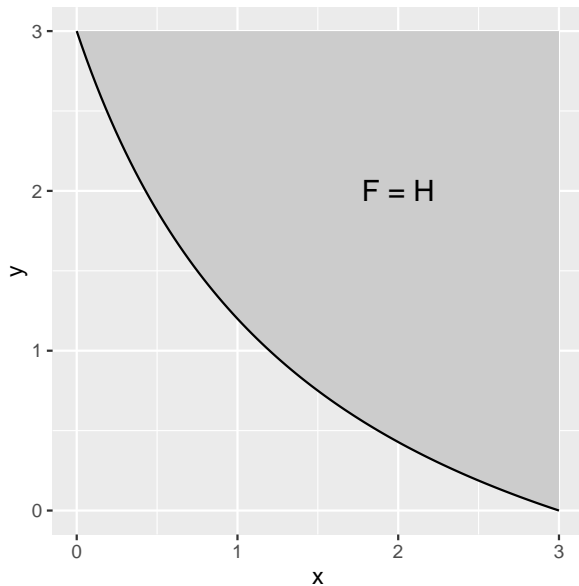


Fig. 1: Feasibility area F and Pareto hull H of Example 3.2.

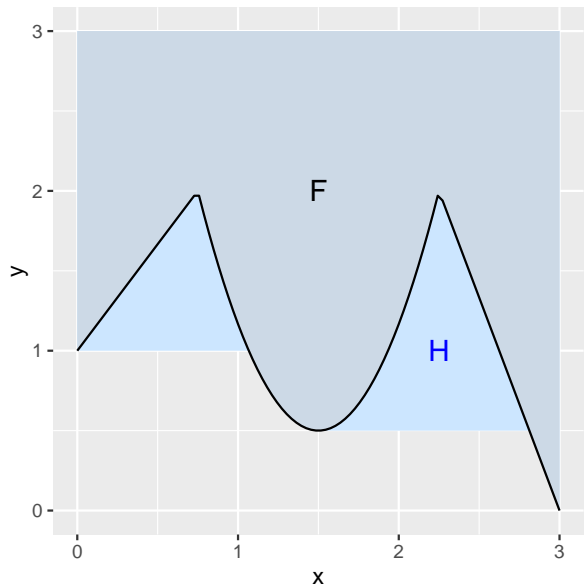


Fig. 2: Feasibility area F and Pareto Hull H (superset of F) of Example 3.3.

As another example we consider a piecewise defined function, where the feasibility set is a proper subset of its Pareto hull.

Example 3.3. We define γ via

$$\gamma(t) = (3 \cdot t, y(t)) \text{ where } y(t) = \begin{cases} 1 + 4 \cdot t & \text{for } 0 \leq t \leq 0.25 \\ 1.5 \cdot (4 \cdot (t - 0.5))^2 + 0.5 & \text{for } 0.25 < t \leq 0.75 \\ 2 - 8 \cdot (t - 0.75) & \text{for } 0.75 < t \leq 1 \end{cases}$$

For the induced feasibility area of this γ function we have $F \subsetneq \mathcal{H}(F)$.

For the implementation, we use γ to derive a function $\gamma_H : [0, 1] \rightarrow \mathbb{R}_+^2$ which is the border of the induced Pareto hull of γ . As we do for γ , we also assume that $\gamma_H(t)$ lies on the y -axis for $t = 0$ and on the x -axis for $t = 1$. This function γ_H is implicitly defined by:

$$\begin{aligned} \forall t \in [0, 1] : \exists t' \in [0, 1] : \gamma(t') \leq \gamma_H(t) \\ \forall t \in [0, 1] : \exists t' \in [0, 1] : \gamma_H(t') \leq \gamma(t) \\ \forall t \in [0, 1] : \nexists t' \in [0, 1] : \gamma_H(t') < \gamma_H(t) \end{aligned}$$

The first two conditions ensure that the induced Pareto hulls of the function graphs are identical. The third condition of γ_H is the actual hull property of γ_H , ensuring that no two points, where one is strictly better than the other one, are contained on the function graph of γ_H . We show both function graphs for Example 3.3 in Figure 3.

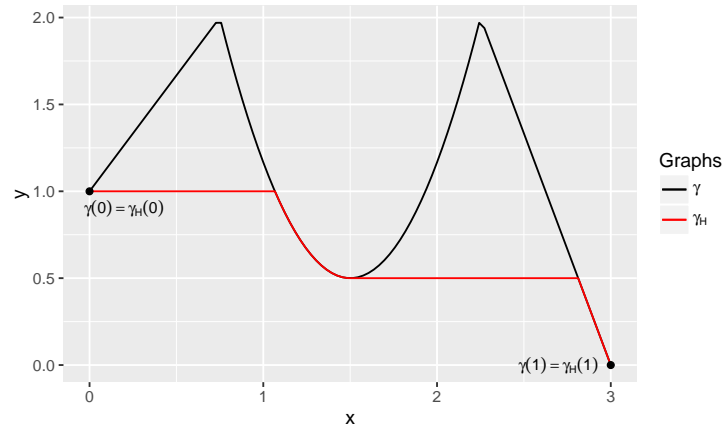


Fig. 3: The function graphs of γ (inducing feasibility area) and γ_H (inducing Pareto hull).

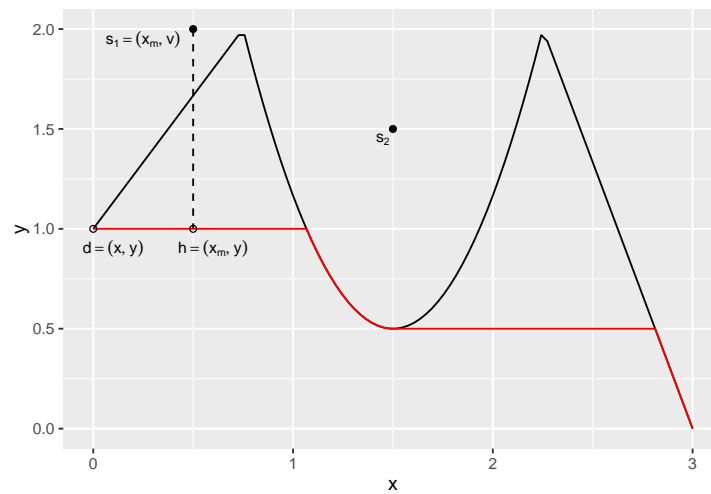


Fig. 4: Process of finding a dominator for $\{s_1, s_2\}$ where the feasibility area is the top right space of the black line. Symbols are according to Algorithm 1.

In the implementation of the 2-dimensional dominator search, γ_H is represented as a list containing segments from γ (i.e., $\gamma(t) = \gamma_H(t)$ holds) and additionally some straight segments, where γ_H is strictly below γ . In Figure 3 we have 4 segments, where 2 of them are parts from the original γ function.

Using γ_H , we can calculate the projection to the Pareto hull in Line 6 of Algorithm 1 efficiently by a binary search. Finding the next feasible point as done in Line 7 of the algorithm can be easily implemented when every straight segment is annotated with its left bottom point, dominating all other points of the segment. For instance, the first segment in Figure 3 has the left bottom point $(0, 1)$.

Figure 4 illustrates, how a new dominator is found for $S = \{s_1, s_2\}$. At first we take the left most point, then we project it to the Pareto hull and finally we find the according feasible point.

4 Exact Solution for the d -Dimensional Case

The main idea of the 2-dimensional solution is based on the fact that it is sufficient to consider x -projections of the Skyline points as dominator candidates. This idea does not translate to the 3-dimensional case as we will show in Remark 4.1. First, we consider an exponentially complex exact solution for the d -dimensional case.

4.1 Naive Solution for the d -Dimensional Case

To find the minimal set of dominators in d dimensions, we have to consider the lower limits of all bounding boxes of d points as given in Definition 2.2. In the following we show the algorithm. We start with an initial set $D = P$ containing all bound box limits $P = \text{bbl}(S, F, d)$, which is clearly a valid (but probably not minimal) dominator set. The set D is iteratively compared to all subsets of P to check if there is a subset which is both smaller and fulfills also the dominator set property. If this is the case, it serves as a new candidate set. This process is iterated for all subsets of P and finally the candidate set is returned.

We define the dimensionality of a set as $\dim(S) =_{df} d$ iff $S \subseteq \mathbb{R}_+^d$ in the following algorithms.

Algorithm 2 d -Dimensional Dominator Search, Exact Solution

Input: Skyline set $S \subseteq \mathbb{R}_+^d$ to dominate, feasibility area $F \subseteq \mathbb{R}_+^d$

Output: Optimal dominator set D

```

1: function DOMINATOR_SEARCH( $S, F$ )
2:    $d \leftarrow \dim(S)$  // Dimensionality
3:    $P \leftarrow \text{bbl}(S, F, d)$  // Calculate feasible lower bounds for  $d$ -subsets (Definition 2.2)
4:    $D \leftarrow P$  // Initial (trivial) candidate of dominator set
5:   for  $C \in \mathcal{P}(P) \setminus \{\emptyset, P\}$  do // Cycle through the power set of the candidates
6:     if  $(|C| < |D|) \wedge (\forall q \in S : \exists p \in C : p \leq q)$  then // Check if  $C$  is smaller than  $D$  and a valid dominator set
7:        $D \leftarrow C$  //  $C$  is a better candidate for  $D$ 
8:     end if
9:   end for
10:  return  $D$ 
11: end function

```

The correctness of this algorithm follows directly from Lemma 2.3 stating that is sufficient to consider bounding box limits as dominator candidates. In the algorithm all possible subsets of bounding box limits are tested, and the smallest one which is a valid dominator set, is returned. Hence this method minimizes the set of dominators.

When implementing this algorithm, one should make use of short-circuit evaluation in Line 6. This means, the relatively expensive check if a candidate set C is a valid dominator set should be omitted if the cardinality of C is not an improvement (i.e., is not lower) over D .

The upper cost estimate of for this algorithms is $O(2^{|\text{bbl}(S, F, d)|})$, as in the worst case all subsets of the bounding box limits have to be checked. Using the rough upper estimate for $|\text{bbl}(S, F, d)| \leq |S|^d$ from Eq. (2) we get $O(2^{|S|^d})$. This means we are faced with an extremely costly problem. Hence we will consider some optimizations of this solution and we will also suggest some approximative heuristics later on.

4.2 Minimizing the Candidate Set

In the 2-dimensional case it suffices to consider just projections (in one dimension) as dominator candidates. For $d \geq 3$ dimensions this is not sufficient as the following counterexample shows.

Remark 4.1. Assume $d = 3$ dimensions and the complement of the standard 3-simplex as feasibility area, i.e.

$$F = \{x \in \mathbb{R}_+^3 \mid x_1 + x_2 + x_3 \geq 1\}.$$

Assume the Skyline set $S = \{s_1, s_2, s_3\}$ where

$$\begin{aligned} s_1 &= (1, 1, 0.5) , \\ s_2 &= (1, 0.5, 1) , \\ s_3 &= (0.5, 1, 1) . \end{aligned}$$

Here the optimal dominator set is $D = \{p\}$ with $p = (0.5, 0.5, 0.5)$. The set of projections of S to F in all 3 dimensions is (the i -th row contains the projections of point s_i):

$$\begin{aligned} P &= \{(0, 1, 0.5), (1, 0, 0.5), (1, 1, 0), \\ &\quad (0, 0.5, 1), (1, 0, 1), (1, 0.5, 0), \\ &\quad (0, 1, 1), (0.5, 0, 1), (0.5, 1, 0)\} \end{aligned}$$

We have $p \notin P$, i.e., the only optimal dominator p is not contained in the projection set. Hence it is not sufficient to consider only projection points for finding the minimal dominator set. \square

Nevertheless many of the bounding box limits in $\text{bbl}(S, F, d)$ are useless in the sense that they do not have a better dominance behavior w.r.t. the Skyline points S as other bounding box limits. They can be eliminated before doing the main dominator search.

In the following definition we formalize the dominance of each candidate point, by comparing it to all points from the Skyline S which shall be dominated.

Definition 4.2 (Dominance Relation). Let $S \subseteq \mathbb{R}_+^d$ be the Skyline to be dominated. We define

$$\begin{aligned} \text{dominance}_S : \mathbb{R}_+^d &\rightarrow \{0, 1\}^{|S|} , \\ q &\mapsto (\alpha_1, \dots, \alpha_n) \text{ with } \alpha_i = 1 \text{ for } q \leq s_i \text{ and } \alpha_i = 0 \text{ otherwise ,} \\ &\text{where } S = \{s_1, \dots, s_n\}, n = |S| . \end{aligned}$$

The result of $\text{dominance}_S(\cdot)$ are n -dimensional Boolean vectors which can be compared using the non-strict product order \leq . For two projection points, i.e. dominator candidates p and q , we say that p is a better or equivalent dominator candidate than q iff $\text{dominance}_S(q) \leq \text{dominance}_S(p)$. This means that p dominates all Skyline points which are dominated by q and perhaps some more. Note that “better” means maximizing all the n (Boolean) dimensions. Here “better” is used in contrast to the Skyline definition, where we try to minimize all dimensions. In the following example we show the dominance behavior of different candidates and the implication on dominance_S .

Example 4.3. Let F and $S = \{s_1, s_2, s_3\}$ be the same as in Remark 4.1. We get 7 bounding box points $\text{bbl}(S, F, d) = \{q_1, \dots, q_7\}$, where $q_i = s_i$ for $i \leq 3$ (i.e., trivial bounding box of one point). The points $\{q_4, q_5, q_6\}$ stem from bounding boxes with 2 points and q_7 is the only one greatest lower bound for all 3 Skyline points. We consider the dominance_S relation for these points. This Boolean matrix shows if a projection point dominates one of the Skyline points (indicated with 1) or does not (indicated with 0).

$\text{dominance}_S(q)$	s_1	s_2	s_3
$q_1 = (1, 1, 0.5)$	1	0	0
$q_2 = (1, 0.5, 1)$	0	1	0
$q_3 = (0.5, 1, 1)$	0	0	1
$q_4 = (1, 0.5, 0.5)$	0	1	1
$q_5 = (0.5, 1, 0.5)$	1	0	1
$q_6 = (0.5, 0.5, 1)$	1	1	0
$q_7 = (0.5, 0.5, 0.5)$	1	1	1

We can see that all points q_i with $i \leq 6$ have a strict lower dominance than q_7 . Hence we can drop them before running the dominator search. In consequence the optimal dominator set is simply $\{q_7\}$.

Algorithm 3 BNL-style Bounding Box Limits

Input: Skyline set $S \in \mathbb{R}_+^d$ to project, feasibility area $F \subseteq \mathbb{R}_+^d$
Output: Optimal projection set (dominator candidates)

```

1: function BNL_BBL( $S, F$ )
2:    $d \leftarrow \dim(S)$  // Dimensionality
3:    $P \leftarrow \emptyset$  // Initial projection set (window)
4:   for  $p \in \text{bbl}(S, F, d)$  do // Cycle through all bounding box limits
5:     if  $\nexists q \in P : \text{dominance}_S(p) \leq \text{dominance}_S(q)$  then // Check if no better/equal candidates than  $p$  exist
6:        $P \leftarrow P \cup \{p\}$  // Then add  $p$  as dominator candidate
7:        $P \leftarrow P \setminus \{q \in P : \text{dominance}_S(q) \leq \text{dominance}_S(p)\}$  // Remove all dominator candidates worse than  $p$ 
8:     end if
9:   end for
10:  return  $P$ 
11: end function

```

We formalize the idea of dropping worse dominator candidates in the following algorithm. The concept of keeping a window of optimal elements is equivalent to *BNL*, which is used in [BKS01] for the Skyline calculation. Hence we call our suggested method *BNL-style bounding box limits*.

For the implementation it does not make sense to calculate $\text{dominance}_S(q)$ each time when it is required. Instead, this vector should be calculated once and then each point q is annotated its $\text{dominance}_S(q)$ vector.

In the case of Example 4.3 we get $\text{BNL_BBL}(S, F) = \{(0, 0.5, 0.5)\}$ which is the only dominator. In the following we give another example where the algorithm does not return a singleton set.

Example 4.4. Let F be again the simplex complement in 3 dimensions. We assume the Skyline set $S = \{s_1, s_2, s_3\}$ with

$$s_1 = (1, 1, 0), \quad s_2 = (0, 1, 1), \quad s_3 = (1, 0, 1) .$$

The feasible bounding box limits $\text{bbl}(S, F, 3) = \{q_1, \dots, q_6\}$ are given, together with their dominance relation, by

dominates_S	s_1	s_2	s_3
$q_1 = (1, 1, 0)$	1	0	0
$q_2 = (0, 1, 1)$	0	1	0
$q_3 = (1, 0, 1)$	0	0	1
$q_4 = (1, 0, 0)$	1	0	1
$q_5 = (0, 1, 0)$	1	1	0
$q_6 = (0, 0, 1)$	0	1	1

As the q_i are dominated by q_{i+3} for all $i \leq 3$, the optimal bounding box limits selected by the BNL-style algorithm are $\text{BNL_BBL}(S, F) = \{q_4, q_5, q_6\}$. Note that the overall minimum $(0, 0, 0)$ is not contained in H , i.e., $(0, 0, 0)$ is not feasible.

Here each subset $D \subset \text{BNL_BBL}(S, F)$ having cardinality $|D| = 2$ is a valid and optimal dominator set. There is no single candidate point dominating all Skyline points. The dominator set which is returned depends on the first set which is chosen in the for-loop in Algorithm 2, Line 5. For example $D = \{q_4, q_5\}$ is a valid and minimal dominator set.

To optimize Algorithm 2 we can replace Line 3 by $P \leftarrow \text{BNL_BBL}(S, F)$, i.e., we restrict our attention to the optimal bounding box limits. This reduces the costs of the exact dominator search in the average case, but not in the worst case. Algorithm 2 still returns the minimal set of dominators.

5 Approximating Heuristics for the d -Dimensional case

The exact dominator search has exponential costs, making it infeasible for large input sizes. In our experiments, Skylines S with $|S| > 100$ are usually too large to compute the minimal dominator set in reasonable time.

At first we propose a very simple but fast strategy adapting our 2-dimensional solution to the d -dimensional case. Next we propose another approach based on the dominance relation and the greedy strategy approximating the set covering problem [Fei98].

5.1 Simple Ordering and Projection Approach

The idea of this heuristic is to use the idea of the 2-dimensional algorithm in the d -dimensional case. We pick an arbitrary sorting dimension $k \in \{1, \dots, d\}$ and a projection dimension $l \in \{1, \dots, d\} \setminus \{k\}$. In the 2-dimensional case we had $k = 1$ and $l = 2$ (but $k = 2$ and $l = 1$ is also optimal for 2 dimensions). We also see in the 2-dimensional case, that $l = k$ leads to a non-optimal solution, hence we exclude such a parametrization for our d -dimensional heuristic. Finding new dominator candidates and removing points from the remaining set S is done analogously to the 2-dimensional case. This idea is formalized in Algorithm 4.

Algorithm 4 d -Dimensional Dominator Search, Approximating Heuristic

Input: Skyline set $S \subseteq \mathbb{R}_+^d$ to dominate, feasibility area $F \subseteq \mathbb{R}_+^d$,
 sorting dimension k , projection dimension l (with $k \neq l$ and $k, l \in \{1, \dots, d\}$)

Output: Approximated optimal dominator set D

```

1: function DOMINATOR_SEARCH_ORDER_PROJECTION( $S, F, k, l$ )
2:    $H \leftarrow \mathcal{H}(F)$  // Get Pareto hull of feasibility area
3:    $D \leftarrow \emptyset$  // Initial dominator set
4:   while  $S \neq \emptyset$  do
5:      $m \leftarrow \arg \min\{x_k \mid (x_1, \dots, x_d) \in S\}$  // Find remaining point with minimal  $k$ -th component value
6:      $p \leftarrow \min\{q \in H \mid \forall j \neq l : q_j = p_j\}$  // Calculate projection in  $l$ -th component to Pareto hull
7:      $f \leftarrow \text{pick}(\{q \in F \mid q \leq p\})$  // Pick a feasible point similar to  $p$ 
8:      $D \leftarrow D \cup \{f\}$  // Add  $f$  as new dominator
9:      $S \leftarrow S \setminus \{q \in S \mid f \leq q\}$  // Remove dominated points
10:  end while
11:  return  $D$ 
12: end function

```

Note that the set $\{q \in F \mid q \leq p\}$ (in Line 7) cannot be empty, as $p \in H$ is in the Pareto hull of the feasibility area. Hence by definition of the Pareto hull there must be some feasible point $q \in F$ dominating p .

Regarding the cost analysis of this algorithm consider that the main loop is processed $|D|$ times (where $|D| \leq |S|$ as $D = S$ is the trivial dominator set), i.e., depends on the number of dominators found. For each loop run the costs to find the minimal k -th component value and to remove the dominated points are proportional (or less than) $|S|$. The cost of picking a feasible point depends on the implementation of this (geometric) operation. In our examples this could be done in constant time, especially in the trivial case $F = H$ (like it is the case for the simplex complement). In summary, the costs for this heuristic algorithm are $O(|D| \cdot |S|)$. Using $|D| \leq |S|$, we get an upper bound of $O(|S|^2)$. This heuristic turned out to be very fast, but the dominator sets are usually largely overestimated, as we will see in the benchmarks in Section 6.

5.2 Greedy Approach on Bounding Boxes

By making use of the dominance relation, i.e., dominance_S as defined in Section 4.2, the dominator search problem can be reduced to the *Set Coverage problem*. We search for a minimal set within the dominator candidates, covering all points which shall be dominated. The set coverage problem has been studied extensively in the Operations Research community and we use the greedy approach from [Joh74] in the following algorithm. We use all bounding box limits $\text{bbl}(S, F, d)$ from Definition 2.2 and greedily search for the points with the most 1-entries in dominance_S . We do not apply Algorithm 3 (BNL-style) to minimize the bounding box limits, as this is much more costly than applying the Greedy approach to all bounding box limits.

By convention, in the following the sum of a vector is the sum of its components, i.e., $\sum(v_1, \dots, v_k) =_{df} \sum_{i=1}^k v_i$. Applied to the values returned by $\text{dominance}_S(\cdot)$, this sum coincides with the cardinality of a set in the Set Covering problem. As a further convention, the “arg max” function in following algorithm non-deterministically selects one possible maximum argument if there are more than one.

Algorithm 5 d -Dimensional Dominator Search, Greedy Set Covering

Input: Skyline set to dominate $S \subseteq \mathbb{R}_+^d$, feasibility area $F \subseteq \mathbb{R}_+^n$

Output: approximated optimal dominator set D

```

1: function DOMINATOR_SEARCH_GREEDY( $S, F$ )
2:    $d \leftarrow \text{dim}(S)$  // Dimensionality
3:    $P \leftarrow \text{bbl}(S, F, d)$  // Calculate feasible lower bounds for  $d$ -subsets (Definition 2.2)
4:    $D \leftarrow \emptyset$  // Dominator set
5:   while  $S \neq \emptyset$  do // Loop until all points are dominated (removed) from S
6:      $p \leftarrow \arg \max_{q \in P} \sum \text{dominance}_S(q)$  // Greedily select most dominant point from candidate set
7:      $S \leftarrow \{q \in S \mid \neg(p \leq q)\}$  // Remove dominated points
8:      $D \leftarrow D \cup \{p\}$  // Add  $p$  to dominators
9:   end while
10:  return  $D$ 
11: end function

```

Note that the dominance_S relation in the above algorithm uses the current value of S which is modified within the algorithm. In the implementation, one should not recalculate dominance_S but just restrict the Boolean matrix to the columns (i.e., points in S) which are not yet dominated.

Regarding the cost analysis, we see that every run of the while loop causes $O(|P|)$ costs as the most dominant candidate is searched within $P = \text{bbl}(S, F, d)$ (Line 6 of the algorithm). The while loop is repeated $|D|$ times. Using the rough upper estimate for $|P| = |\text{bbl}(S, F, d)| \leq |S|^d$ from Eq. (2) we get an upper bound of $O(|S|^d \cdot |D|)$ for the algorithm. By $|D| \leq |S|$ this has the upper estimate $O(|S|^{d+1})$. This means that this heuristic solution has still exponential complexity.

To see that this greedy approach does not return the optimal result, consider the following example.

Example 5.1. Let $S = \{s_1, \dots, s_4\} \subset \mathbb{R}_+^3$ with the points

$$\begin{aligned} s_1 &= (0.5, 0.5, 0.5) \\ s_2 &= (0, 0.5, 1) \\ s_3 &= (1, 0, 0.5) \\ s_4 &= (0, 0, 1.5) . \end{aligned}$$

The result of DOMINATOR_SEARCH_GREEDY is $D = \{p_1, p_2, p_3\}$ with

$$\begin{aligned} p_1 &= (0, 0.5, 0.5) \\ p_2 &= (0.5, 0, 0.5) \\ p_3 &= (0, 0, 1) . \end{aligned}$$

The dominance relation of these three points is:

$\text{dominates}_S(p)$	s_1	s_2	s_3	s_4
p_1	(1, 1, 0, 0)			
p_2	(1, 0, 1, 0)			
p_3	(0, 1, 0, 1)			

Here the optimal dominator set is $D = \{p_2, p_3\}$. The greedy algorithm first picks one of the p_i . All p_i have two 1-entries and hence are equivalent good for the greedy strategy. Let us assume, the algorithm chooses p_1 first. The order in which the remaining tuples are picked does not matter. If p_1 was accidentally picked first, the dominator set returned by the greedy strategy is $D = \{p_1, p_2, p_3\}$. But this is not optimal as p_1 is contained in the result set which is not necessary to dominate S .

According to the theorem in [?] the size of the dominator set returned by the greedy strategy has the following upper bound, where D_{exact} is the result from the exact search:

$$|\text{DOMINATOR_SEARCH_GREEDY}(S, F)| \leq |D_{\text{exact}}| \cdot \sum_{i=1}^m \frac{1}{i} \text{ where } m = \max_{q \in S} \sum \text{dominance}_S(q) .$$

As $m = 2$ in Example 5.1, we get $(1 + 0.5) \cdot |D_{\text{exact}}| = 1.5 \cdot 2 = 3$ as an upper estimate for the dominator set found by the greedy strategy. This is exactly the cardinality of the heuristic result.

5.3 Heuristic Candidate Sets

The greedy approach itself is quite fast, but the candidate set $\text{bbl}(S, F, d)$, having an upper bound of $|S|^d$, is too large. This makes Algorithm 5 too expensive for input sets S with $|S| > 100$. Hence we consider smaller candidate sets, which may not contain all dominators required for the exact solution. But a smaller candidate set may still be good enough for a quite good approximative solution.

Given a Skyline set $S \subseteq \mathbb{R}_+^d$ and a feasibility area F , we consider the projection of each point $s \in S$ to the Pareto hull $\mathcal{H}(F)$ in each of the d dimensions. Formally we define

$$\text{proj}(S, F, d) := \bigcup_{i=1}^d \min\{q \in \mathcal{H}(F) \mid \exists s \in S : \forall j \neq i : q_j = s_j\} .$$

Analogously to the 2-dimensional case such a projection to the Pareto hull may not be feasible. By definition of the Pareto hull there exists a smaller or equivalent point for each point in the Pareto hull, which is feasible. Using this we define the feasible projection

$$\text{proj_feas}(S, F, d) = \bigcup_{q \in \text{proj}(S, F, d)} \text{pick}\{p \in F \mid p \leq q\} .$$

In Remark 4.1 an example of these projections was given. Note that we have $F = H$ on the simplex complement, i.e., the Pareto hull lets the feasibility area invariant in this special case. In that remark we see that choosing the minimal set of dominators from the projection set does not lead to the optimum in general. The big advantage of $\text{proj_feas}(S, F, d)$ over $\text{bbl}(S, F, d)$, where we had an upper bound of $|S|^d$ candidates, is the much smaller candidate set. We get maximally $|S| \cdot d$ candidates, perhaps less if some projection points coincide.

For a modified greedy approach using the feasible projections, we have to replace Line 3 of Algorithm 5 by $P \leftarrow \text{proj_feas}(S, F, d)$. The complexity of the greedy approach using projections is given by $O(|S| \cdot |D|)$ which has the upper bound $O(|S|^2)$, as $|D| \leq |S|$ holds.

5.4 Restricting the Exact Search Using Heuristics

To speed up the exact dominator search, we can replace $D \leftarrow P$ in Algorithm 2, Line 4, by

$$D \leftarrow \text{DOMINATOR_SEARCH_GREEDY}(S, F) .$$

This initial dominator set avoids testing candidate sets having a higher cardinality than already found by the greedy heuristic.

Additionally we can avoid testing those sets which are too small. A simple lower bound for the cardinality of a dominator set for Skyline S is given by the minimum number of candidates, which have in summary $|S|$ entries with 1 in the dominance relation. Formally, let P be the projection set and dominates_S the dominance relation for dominating the Skyline S . Then for every valid dominator set D holds

$$|D| \geq \min \left\{ |Q| \mid Q \subseteq P, \sum_{q \in Q} \text{dominates}_S(q) \geq |S| \right\} .$$

All these optimizations lead to a better average case performance, but not to a better worst case performance of the exact search. The exact dominator search is still an exponentially complex problem and we assume that it is NP-hard.

6 Benchmarks

Now we present our empirical results of the complexity. For the 2-dimensional search (Section 3.1) and for the heuristic search using projections as candidate set (Section 5.3) we both have an upper bound for the time complexity of $O(|S| \cdot |D|)$. There $|S|$ is the size of Skyline points which shall be dominated and $|D|$ the cardinality of the returned dominator set. Hence the difficulty of the problems increases with the number of returned dominators. In the following we construct Skyline sets S and feasibility areas F where we can easily vary the minimal number of dominators.

6.1 Benchmark Setting

As in the examples in the previous sections we use the simplex complement as a feasibility area. For dimensionality d our feasibility area is given by

$$F_d = \left\{ x \in \mathbb{R}_+^d \mid \sum x \geq 1 \right\} .$$

We place the Skyline points randomly on the border of the d -simplex, multiplied by some constant $c \in [1, \infty)$:

$$S_{d,c} \subset \left\{ c \cdot x \mid x \in \mathbb{R}_+^d, \sum x = c \right\} .$$

To generate n points which are uniformly distributed on $S_{d,c}$, we first generate points $x_{i,j}$ which are uniformly distributed on $[0, 1]$, where $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, d\}$. The Skyline points $s_i \in \mathbb{R}_+^d$ are then calculated by

$$s_i := \frac{c}{\sum_j \ln(x_{i,j})} \cdot (\ln(x_{i,1}), \dots, \ln(x_{i,d})) \quad \text{for } i \in \{1, \dots, n\} .$$

By definition of $<_{\otimes}$ all points in $S_{d,c}$ are Pareto-incomparable. As $\sum s_i = c \geq 1$ we have $s_i \in F$, i.e., all these points are feasible. By varying c we can adjust the number of the smallest possible dominator set. For $c = 1$ every s_i is on the border of the feasibility area. Assuming that there are no duplicate points in $S_{d,1}$, the set $D = S_{d,1}$ is the optimal dominator set in this special case.

On the other hand, consider a large value of c where $c \geq d$. Then d dominators

$$D = \{(1, 0, \dots, 0), \dots, (0, \dots, 0, 1)\} \quad \text{with } D \subset \mathbb{R}_+^d$$

are sufficient. To see this, consider $p = (p_1, \dots, p_d) \in S_{d,c}$. To fulfill $p_1 + \dots + p_d \geq d$ and $p_j \geq 0$ for all j there must be at least one p_{j^*} fulfilling $p_{j^*} \geq 1$. To generate settings with a large number of dominators we are interested in Skyline sets $S_{d,c}$ with $c = 1 + \epsilon$ for a small $\epsilon > 0$.

In the Figures 5 and 6 we illustrate our benchmark setting for the 2-dimensional case. The area above the $x + y = 1$ line is the feasibility area. The points to be dominated are placed on the $x + y = c$ line. With smaller values for $c \geq 1$ the minimum number of dominators needed is increasing. For $c \rightarrow 1$ we have $|D| \rightarrow |S|$.

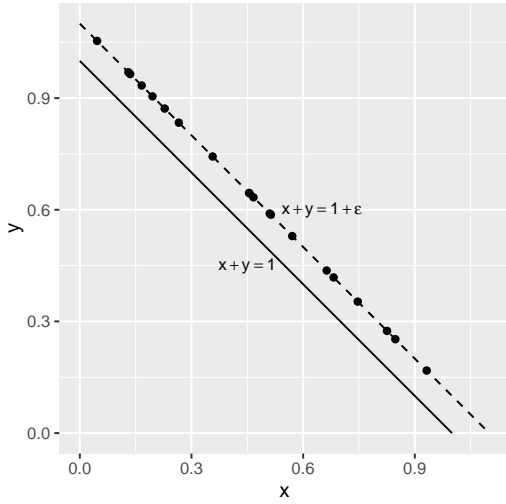


Fig. 5: Feasibility area (simplex complement) and randomly simulated points on $x + y = 1 + \epsilon$ with $\epsilon = 0.1$.

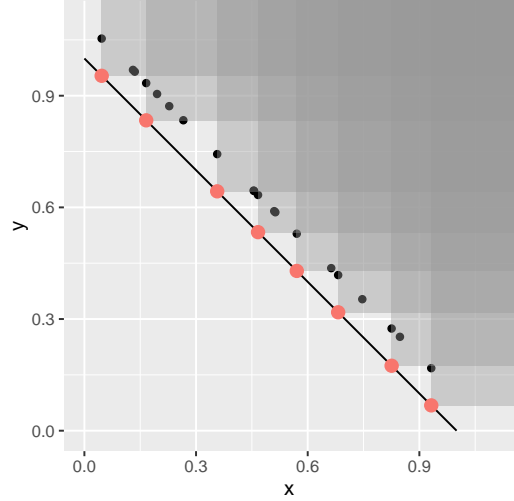


Fig. 6: Minimal set of dominators for given example, the dominance area is grey-shaded.

6.2 Results for the 2-Dimensional Case

In Table 1 we show some results from an implementation of Algorithm 1, i.e., the 2-dimensional dominator search. For a number of 10^5 tuples, which shall be dominated, and $c = 1 + \epsilon$ with $\epsilon = 10^{-4}$ the search for a minimal set of dominators takes less than 30 seconds. All runtimes were measured on an off-the-shelf personal computer.

$n (= S)$	c	$ D $	runtime (sec)
10^4	1.001	907	0.14
10^4	1.0001	5030	0.59
10^5	1.001	992	2.51
10^5	1.0001	9086	24.56

Table 1: Selected runtimes for 2-dimensional dominator search.

Next, we empirically underpin the theory that the runtime scales with the number of tuples, multiplied with the number of dominators, as indicated by the performance analysis in Section 3.1. Therefore we consider a log-log diagram of $|S| \cdot |D|$ on one axis and the runtime on the other axis. The regression line in Figure 7 shows that the runtime grows linearly with $|S| \cdot |D|$.

6.3 Results for the d-Dimensional Case

Regarding the d -dimensional dominator search we have a set of different methods. The exact solution has exponential costs but determines the provably minimal set of dominators. Next to this, we presented different approximating heuristics in Section 5. In the following we want to compare them regarding costs and accuracy. We compare the following four methods:

- “Exact”: Algorithm 2 using the BNL-style bounding box limits from Algorithm 3 and the heuristic pre-calculation from Section 5.4. Upper cost estimation: $O(2^{|S|^d})$.

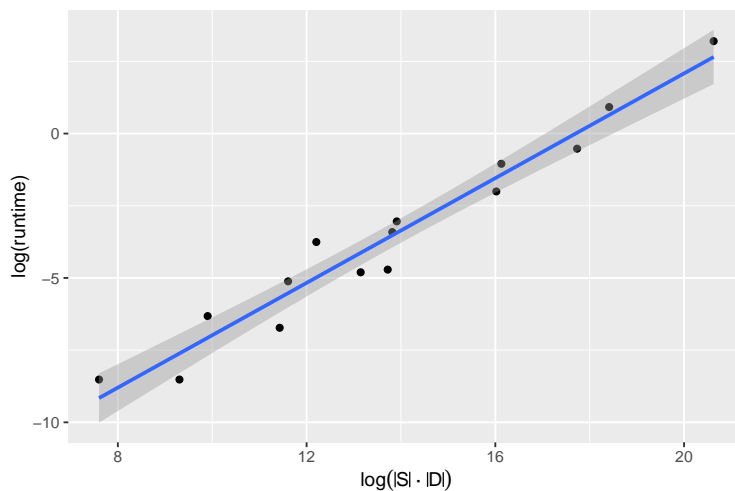


Fig. 7: A log-log diagram showing how the runtime of the 2-dimensional dominator search scales with $|S| \cdot |D|$.

- “Greedy BBL”: The greedy approach (Algorithm 5) using the bounding box limits (BBL) as candidates. Upper cost estimation: $O(|S|^d \cdot |D|)$.
- “Greedy Projection”: The greedy approach using the projections as candidate set, i.e., Algorithm 5 with the modifications given in Section 5.3. Upper cost estimation: $O(|D| \cdot |S|)$.
- “Sorting/Projection”: Algorithm 4. Upper cost estimation: $O(|D| \cdot |S|)$.

We run all benchmarks for $d = 3$ dimensions. We randomly generated 5 different data sets for each configuration of n and c . We picked $n \in [20, 30]$ and $c \in [1.5, 2]$. For significantly larger values of n and smaller values of c we got a timeout for the exact search in some instances (more than 60 seconds). There was one data set with $n = 30$ and $c = 1.5$ where the exact dominator search took 12 seconds (overall maximum).

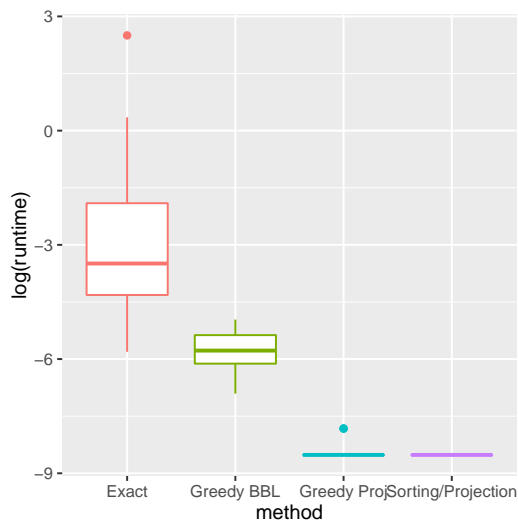


Fig. 8: Logarithmic runtimes for different search methods including exact search.

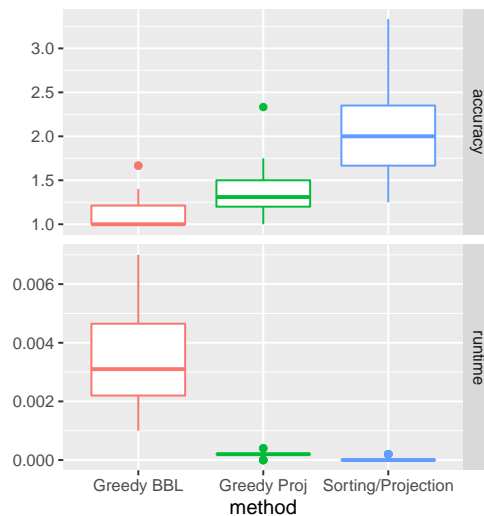


Fig. 9: Accuracy (exact method has accuracy 1 per definition) and runtime (not logarithmic, in seconds) of different methods.

In Figure 8 we show the runtimes of the different methods on a logarithmic scale. We see that the exact search is much more costly than all other methods. In Figure 9 we take a closer look at the accuracy and runtime. There accuracy is defined by $|D_{\text{heuristic}}|/|D_{\text{exact}}|$, i.e., the factor specifying how the cardinality of the found dominator set is overestimated for a given heuristic method. Note that all test cases for this study have to be that simple, such that the very expensive exact method can be executed in reasonable time. We need the result of the exact search to calculate the accuracy. Hence the runtimes of all heuristic methods are very short.

As indicated by the upper cost estimates, the ‘‘Greedy BBL’’ algorithm is very expensive (exponentially) while the two other methods are quite cheap (both have an upper estimate of $O(|D| \cdot |S|)$). Considering the accuracy, we see that that Greedy BBL is slightly more accurate than Greedy Projection. As we can see from Figure 9, in most cases the accuracy for Greedy Projection is lower than 1.5, i.e., the overestimation of the dominator set cardinality $|D|$ is mostly less than 50%. The accuracy of the Sorting/Projection approach is clearly worse, with an overestimation of more than 100% in half of the cases.

Hence the Greedy Projection method can be considered as a good compromise of speed and accuracy. Similar to the 2-dimensional dominator search we take a closer look on its performance. In Table 2 we consider some runtimes for $d = 3$ dimensions and different values of n and c .

$n (= S)$	c	$ D $	runtime (sec)
500	1.1	98	0.60
500	1.01	423	2.03
1000	1.1	132	2.51
1000	1.01	809	16.03

Table 2: Selected runtimes for d -dimensional dominator search with $d = 3$ using the greedy projection method.

As the upper time complexity of the greedy projection method is $O(|S| \cdot |D|)$ we consider a log-log diagram (Figure 10) showing how the runtime depends on $|S| \cdot |D|$. This analysis is analogously to the analysis of the 2-dimensional dominator search.

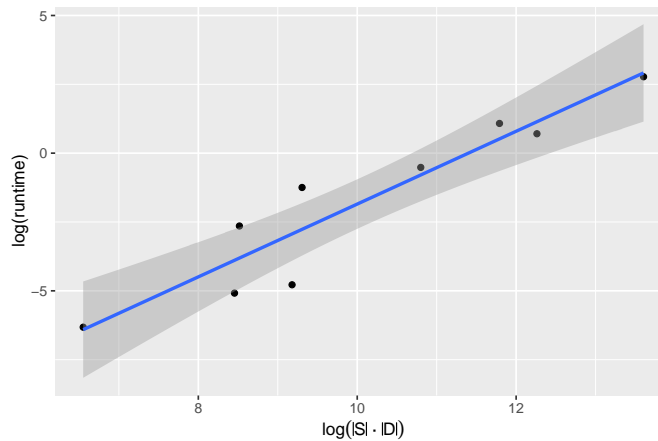


Fig. 10: A log-log diagram showing how the runtime of the 3-dimensional dominator search using the Greedy Projection method scales with $|S| \cdot |D|$.

Again we can see that the runtime scales linearly with $|S| \cdot |D|$. But note that the constant factor in the time complexity is much higher than for the 2-dimensional search for similar values of n and c , cf. Table 1 vs. Table 2.

7 Related Work

Skyline queries have been introduced in database systems by Börzsönyi et al. [BKS01] in 2001. Since then many algorithms for computing the Skyline have been developed, cp. [CCM13] for an overview. To the best of our knowledge the problem of finding dominators for Skyline objects within a given feasibility area is new and has never been considered in the past. Nevertheless, there is some work which deal with similar tasks and hence we want to discuss them in this section.

When talking about *dominators* the concept of *k-dominant Skylines* [CJT⁺06] comes in ones mind. However, the target of *k-dominant Skylines* relax the idea of *dominance* to *k-dominance*. That means a point p is said to *k-dominate* another point q if there are k ($\leq d$) dimensions in which p is better than or equal to q and is better in at least one of these k dimensions. A point that is not *k-dominated* by any other point is in the *k-dominant Skyline*. There also exists the approach of *top-k dominating Skylines* [YM07], where a query returns k data objects which dominate the highest number of objects in the dataset. Such queries are important for decision support since they provide data analysts an intuitive way for finding significant objects. Both concepts do not consider dominators in the sense of this paper.

A related class of problems are the many different approaches of determining *representative Skylines* [LYZZ07,TDLP09]. The goal is to select a subset of k tuples from a Skyline S (with $|S| > k$), which are somehow representative for all tuples in the Skyline. If one allows that representative tuples may reside outside of the Skyline, we can use the concept of *near dominators* to represent a Skyline. For a given Skyline S we want to find those k dominators (without a given feasibility area), whose maximal distance to the Skyline is minimized. There the notion of distance is arguable. For example, the Euclidean distance to the Pareto front line could be an appropriate distance function [TDLP09]. It is an open question to find efficient algorithms for determining a dominator-based representative Skyline. Perhaps some of the methods presented in the mentioned papers can be used for this problem.

Koltun and Papadimitriou [KP07] consider *approximately dominating representatives* (ADR) The ADR concept is similar to ours and was inspired by the work of [PY00,PY01]. However, they focus on combinatorial optimization problems, where instead of a database of objects we have an implicitly represented exponential set of feasible solutions, evaluated by multiple cost functions. Similar to ADRs is the work of Vasilvitskii and Yannakakis [VY05] which focus on multi-objective optimization problems, as opposed to static multi-dimensional Skyline queries. Their goal is to obtain approximate ϵ -Pareto curves with as few points as possible for a given ϵ -environment.

8 Conclusion and Outlook

We have presented an efficient solution for the 2-dimensional dominator search problem based on a greedy strategy. The d -dimensional problem is closely related to the NP-hard set covering problem [Fei98,CTF00]: Given a collection F of subsets of $S = \{1, \dots, n\}$, *set cover* is the problem of selecting as few as possible subsets from F such that their union covers S . Hence we assume that for $d \geq 3$ the dominator search problem is NP-hard. According to the results in [Fei98] the greedy strategy for the set covering problem is essentially the best polynomial-time approximation to the set covering problem, unless $N=NP$. Hence it seems reasonable to adapt the greedy set covering approach to our problem.

The naive greedy approach to the d -dimensional ($d \geq 3$) problem does not lead to a polyomial-time algorithm, as the candidate set, i.e., the set of bounding box limits, is exponentially large. Hence we suggested a smaller candidate set, obtained by all the orthogonal projections in d dimensions. Our empirical results show that the approximation ratio using the projections as candidate set is only slightly worse in comparison to the bounding box limits. A much more simpler method is the sorting/projection approach, which is a naive translation of the 2-dimensional solution into the d -dimensional space. This method is much faster but empirical results show that the approximation ratio is much worse and probably not appropriate for real world use cases.

For future research, it is an open question how better heuristic candidate sets for the d -dimensional dominator search could be found. The goal is to find the best compromise of speed and approximation ratio.

References

- [BKS01] S. Börzsönyi, D. Kossmann, and K. Stocker. The Skyline Operator. In *Proceedings of ICDE '01*, pages 421–430, Washington, DC, USA, 2001. IEEE.
- [CCM13] J. Chomicki, P. Ciaccia, and N. Meneghetti. Skyline Queries, Front and Back. *SIGMOD*, 42(3):6–18, 2013.
- [Cho03] J. Chomicki. Preference Formulas in Relational Queries. In *TODS '03: ACM Transactions on Database Systems*, volume 28, pages 427–466, New York, NY, USA, 2003. ACM Press.
- [CJT⁺06] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang. Finding k-Dominant Skylines in High Dimensional Space. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 503–514, New York, NY, USA, 2006. ACM.
- [CTF00] A. Caprara, P. Toth, and M. Fischetti. Algorithms for the Set Covering Problem. *Annals of Operations Research*, 98(1):353–371, 2000.
- [Fei98] U. Feige. A Threshold of $\ln N$ for Approximating Set Cover. *J. ACM*, 45(4):634–652, July 1998.
- [Joh74] S. D. Johnson. Approximation Algorithms for Combinatorial Problems. *Journal of computer and system sciences*, 9(3):256–278, 1974.
- [Kie02] W. Kießling. Foundations of Preferences in Database Systems. In *Proceedings of VLDB '02*, pages 311–322, Hong Kong, China, 2002. VLDB.
- [KP07] V. Koltun and C. H. Papadimitriou. Approximately Dominating Representatives. *Theor. Comput. Sci.*, 371(3):148–154, 2007.
- [LYZZ07] Xuemin Lin, Yidong Yuan, Qing Zhang, and Ying Zhang. Selecting Stars: The k Most Representative Skyline Operator. In *ICDE '07: Proceedings of the 23rd International Conference on Data Engineering*, pages 86–95. IEEE, April 2007.
- [PY00] C. H. Papadimitriou and M. Yannakakis. On the Approximability of Trade-offs and Optimal Access of Web Sources. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science, FOCS '00*, Washington, DC, USA, 2000. IEEE Computer Society.
- [PY01] C. H. Papadimitriou and M. Yannakakis. Multiobjective Query Optimization. In *PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 52–59, New York, NY, USA, 2001. ACM.
- [TDLP09] Y. Tao, L. Ding, X. Lin, and J. Pei. Distance-Based Representative Skyline. In *ICDE '09: Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 892–903, Washington, DC, USA, 2009. IEEE Computer Society.
- [VY05] S. Vassilvitskii and M. Yannakakis. Efficiently Computing Succinct Trade-off Curves. *Theor. Comput. Sci.*, 348(2):334–356, December 2005.
- [YM07] M. L. Yiu and N. Mamoulis. Efficient Processing of Top-k Dominating Queries on Multi-dimensional Data. In *Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB '07*, pages 483–494. VLDB Endowment, 2007.
-