Multi-Objective Problems in Terms of Relational Algebra

Florian Diedrich[1,⋆], Britta Kehden[1], and Frank Neumann[2]

[1] Institut für Informatik, Christian-Albrechts-Universität zu Kiel,
Olshausenstr. 40, 24098 Kiel, Germany
{fdi,bk}@informatik.uni-kiel.de
[2] Algorithms and Complexity,
Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany
fne@mpi-inf.mpg.de

Abstract. Relational algebra has been shown to be a powerful tool for solving a wide range of combinatorial optimization problems with small computational and programming effort. The problems considered in recent years are single- objective ones where one single objective function has to be optimized. With this paper we start considerations on the use of relational algebra for multi-objective problems. In contrast to single-objective optimization multiple objective functions have to be optimized at the same time usually resulting in a set of different trade-offs with respect to the different functions. On the one hand, we examine how to solve the mentioned problem exactly by using relational algebraic programs. On the other hand, we address the problem of objective reduction that has recently been shown to be NP-hard. We propose an exact algorithm for this problem based on relational algebra. Our experimental results show that this algorithm drastically outperforms the currently best one.

1 Introduction

Many real-world problems involve optimization of several objective functions simultaneously. For such multi-objective optimization problems usually there is not a single optimal function value for which a corresponding solution should be computed but a set of different trade- offs with respect to the different functions. This set of objective vectors is called the Pareto front of the given problem. Even for two objective functions the Pareto front may be exponential in the problem dimension. This is one reason for the assumption that multi-objective problems are in most cases harder to solve than single-objective ones. Other results from complexity theory support this claim as simple single- objective combinatorial optimization problems such as minimum spanning trees or shortest path become

⋆ Research supported in part by a grant "DAAD Doktorandenstipendium" of the German Academic Exchange Service and in part by EU research project AEOLUS, "Algorithmic Principles for Building Efficient Overlay Computers", EU contract number 015964.

NP-hard when two functions on the edges should be optimized at the same time [9]. Often optimizing just one of the given objective functions is an NP-hard task. Such problems occur frequently in network design problems where e. g. one task is to minimize the maximum degree of a spanning tree [10, 13, 16, 18]. Another well-known example is the multi-objective knapsack problem [20] where the task is to solve different knapsack problems simultaneously. This problem is a generalization of the classical knapsack problem which belongs to the oldest problems in combinatorial optimization; see the textbooks by Martello & Toth [14] and Kellerer et al. [12] for surveys.

The aim of this paper is to investigate the use of relation-algebraic methods for dealing with multi-objective optimization problems. Relational algebra provides a powerful framework for solving various optimization problems with small programming effort [2, 6, 17]. Computer programs based on relational algebra are in particular short and easy to implement and there are several tools that are able to execute relational programs in a quite efficient way. Tools like RelView [5] or CrocoPat [7] represent relations implicitly by Ordered Binary Decision Diagrams (OBDDs) [3] which enables practitioners even to deal with very large relations.

The advantage of relational programs has been pointed out for many single-objective combinatorial optimization problems [4, 5]. Computing an optimal solution for the considered problems often implicitly involves the consideration of the whole search space in this case. As pointed out previously, the task in multi-objective optimization is to compute a set of solutions which may in the worst case increase exponentially with the problem dimension. Using OBDDs in such cases may in particular result in a compact implicit representation of this set of solutions.

First we examine how to formulate the computation of the Pareto front for a given problem in terms of relational algebra. As this mainly relies on the intersection of quasi-orders, a relational algebraic formulation for this problem can be given in a straightforward way. The problem when computing the Pareto front in this way has to deal with the task to compute the dominance relations of the given objective functions with respect to the considered search space. As the search space is usually exponential in the input size we can only hope to be successful for problems where the relations between the different solutions are represented by OBDDs of moderate size. Later on we consider the problem of reducing the number of objectives for a given problem. Here the task is to compute a minimal subset of the given objective functions that represents the same weak dominance relation as the one implied by the set of all objectives. This problem has recently been shown to be NP- hard [8] by a reduction to the set covering problem. In the same paper an exact algorithm with worst-case exponential runtime has been proposed. We develop an algorithm on the basis of relational algebra for this problem which outperforms the one of Brockhoff & Zitzler [8] drastically in our experimental studies. The investigations show

that our algorithm is able to deal with large sets of objective functions which further shows the advantage of the relation-algebraic approach.
The outline of the paper is as follows. In Sect. 2 we introduce basic preliminaries on relational algebra and multi-objective optimization. Sect. 3 gives a relation-algebraic formulation for computing the Pareto optimal search points of a given problem and Sect. 4 shows how relational algebra can be used to reduce the number of necessary objectives. The results of our experimental studies are presented in Sect. 5 and finally we finish with some conclusions.

2 Multi-Objective Optimization and Relational Algebra

In this section we describe relation-algebraic preliminaries which are necessary to understand the development of algorithms. A more comprehensive presentation on the use of relational algebra can be found in [17]. Afterwards we give an introduction to the field of multi-objective optimization using the terminology of relational algebra.

2.1 Basic principles of Relational Algebra

A concrete relation is a subset of a cartesian product $X \times Y$ of two sets. We write $R : X \leftrightarrow Y$ and denote the set of all relations of the *type* $X \leftrightarrow Y$ by $[X \leftrightarrow Y]$. In the case of finite supports, we may consider a relation as a Boolean matrix and use matrix terminology and matrix notation in the following. Especially, we speak of the rows, columns and entries of R and write R_{ij} instead of $(i, j) \in R$. In some cases, especially if the relation is an order or preorder $\leq$, we also use the infix notation $i \leq j$ to increase readability. We assume the reader to be familiar with the basic operations on relations, viz. $R^\top$ (transposition), $\overline{R}$ (negation), $R \cup S$ (union), $R \cap S$ (intersection), RS (composition), and the special relations O (empty relation), L (universal relation), and I (identity relation). A relation R is called *vector*, if $R\mathsf{L} = R$ holds. As for a vector therefore the range is irrelevant, we consider in the following vectors $v : X \leftrightarrow \mathbf{1}$ with a specific singleton set $\mathbf{1} = \{\perp\}$ as range and write v_i instead of $v_{i\perp}$. Such a vector can be considered as a Boolean matrix with exactly one column, i.e. as a Boolean column vector, and describes the subset $\{x \in X : v_x\}$ of X. A vector v is called a *point* if it is injective and surjective. For $v : X \leftrightarrow \mathbf{1}$ these properties mean that it describes a singleton set, i.e. an element of X. In the matrix model, a point is a Boolean column vector in which exactly one component is true. A relation $R : X \leftrightarrow Y$ can be considered as a list of $|Y|$ vectors, the columns of R. We denote the y-th column of R with $R^{(y)}$, i.e. $R^{(y)}$ is a vector of type $X \leftrightarrow \mathbf{1}$ and for all $x \in X$ the expressions $R^{(y)}_x$ and R_{xy} are equivalent.

For all sets X and Y there exist a pair (π, ρ) of *natural projections* of $X \times Y$, i.e. two relations $\pi : X \times Y \leftrightarrow X$ and $\rho : X \times Y \leftrightarrow Y$ with

$$\pi_{\langle x,y \rangle x'} \iff x = x' \text{ and } \rho_{\langle x,y \rangle y'} \iff y = y'$$

As discussed in [17], the natural projections permit the definition of a Boolean lattice isomorphism $vec : [X \leftrightarrow Y] \rightarrow [X \times Y \leftrightarrow \mathbf{1}]$ by $vec(R) = (\pi R \cap \rho)\mathsf{L}$. With this mapping each relation R can be represented by a vector $r = vec(R)$ in the sense that

$$r_{\langle x,y\rangle} \iff R_{xy}.$$

The inverse mapping *rel* is given by $rel(r) = \pi^{\top}(\rho \cap r\mathsf{L})$. The mapping *vec* allows to establish the following representation of sets of relations. A subset $\mathcal{S} = \{R_1, \ldots, R_n\}$ of $[X \leftrightarrow Y]$ can be modelled by a relation $S : X \times Y \leftrightarrow [1..n]$ such that for each $i \in [1..n]$ the equation $S^{(i)} = vec(R_i)$ is satisfied, i.e, every column of S is the vector representation of a relation in $\mathcal{S}$.

2.2 Multi-Objective Optimization

Many problems in computer science deal with the optimization of one single objective function which should be optimized under a given set of constraints. In this case there is a linear preorder on the set of search points and an optimal solution can be defined as a smallest (or greatest) element with respect to this preorder depending on whether we consider minimization or maximization problems. The goal is to compute exactly one smallest element with respect to the given preorder.

In the case of multi-objective optimization (see, e.g. Ehrgott [9]), several objective functions are given. These functions define a partial preference on the given set of search points. Most of the best known single-objective polynomially solvable problems like shortest path or minimum spanning tree become NP-hard when at least two weight functions have to be optimized at the same time. In this sense, multi-objective optimization is considered at least as difficult as single-objective optimization.

For multi-objective optimization problems the objective function $f = (f_1, \ldots, f_k)$ is vector-valued, i.e., $f\colon \mathcal{S} \rightarrow \mathbb{R}^k$. Since there is no canonical complete order on $\mathbb{R}^k$, one compares the quality of search points with respect to the canonical partial order on $\mathbb{R}^k$, namely $f(x) \leq f(x')$ iff $f_i(x) \leq f_i(x')$ for all $i \in [1..k]$. A Pareto optimal search point $s \in \mathcal{S}$ is a point such that (in the case of minimization problems) $f(x)$ is minimal with respect to this partial order and all $f(s'), s' \in \mathcal{S}$. In terms of relational algebra the problem can be stated as follows.

Definition 1. *Given a minimization problem in a search space $\mathcal{S}$ and a set $\mathcal{F} = \{f_1, \ldots, f_k\}$ of functions $f_i : \mathcal{S} \rightarrow \mathbb{R}$, we define a set $\mathcal{R} = \{\preceq_1, \ldots, \preceq_k\}$ of k relations of type $\mathcal{S} \leftrightarrow \mathcal{S}$ by $x \preceq_i x' \iff f_i(x) \leq f_i(x')$. The weak dominance relation $\preceq\colon \mathcal{S} \leftrightarrow \mathcal{S}$ is defined by $x \preceq x' \iff \forall i \in [1..k] : f_i(x) \leq f_i(x')$. The strong dominance relation $\prec$ is defined by $x \prec x' \iff x \preceq x' \wedge \exists i \in [1..k] : f_i(x) < f_i(x')$. We say that x dominates x', if $x \prec x'$ holds. A search point x is called Pareto optimal if there exist no search point x' that dominates x.*

Again there can be many Pareto optimal search points but they do not necessarily have the same objective vector. The Pareto front, denoted by F, consists of all objective vectors $y = (y_1, \dots, y_k)$ such that there exists a search point s where $f(s) = y$ and $f(s') \leq f(s)$ implies $f(s') = f(s)$ for all $s' \in \mathcal{S}$. The Pareto set consists of all solutions whose objective vector belongs to the Pareto front. The problem is to compute the Pareto front and for each element y of the Pareto front one search point s such that $f(s) = y$. We sometimes say that a search point s belongs to the Pareto front which means that its objective vector belongs to the Pareto front. The goal is to present such a set of trade-offs to a decision maker who often has to choose one single solution out of this set based on his personal preference.

Especially in the case of multi-objective optimization, evolutionary algorithms seem to be a good heuristic approach to obtain a good set of solutions. Evolutionary algorithms have the advantage that they work at each time step with a set of solutions called the population. This population is evolved to obtain a good approximation of the Pareto front. The final set of solutions presented to a decision maker should represent the different trade-offs with respect to the given objective functions. It has been pointed out in [8] that often not all objectives are necessary to represent the different trade-offs. Reducing the number of objectives that have to be examined by a decision maker may simplify the decision which of the presented solutions should be finally chosen.

3 Computing the Pareto-Optimal Set

The classical problem that arises in multi-objective optimization is to compute for each objective vector belonging to the Pareto front a corresponding solution of the Pareto optimal set. In the following we show how this problem can easily be solved for small problem instances where the weak dominance relation can be expressed for each function as an OBDD of moderate size.

We consider the set $\mathcal{R}$ of relations introduced in Definition 1. Every relation $\preceq_i$ in $\mathcal{R}$ is a linear preorder, i.e reflexive and transitive, and $x \preceq_i x' \vee x' \preceq_i x$ holds for each two search points x and x'. From the definition immediately follows the equation

$$\preceq = \bigcap_{i \in [1..k]} \preceq_i$$

to describe the weak dominance relation. Hence, the relation $\preceq$, as an intersection of preorders, is also a preorder, but not necessarily linear. As discussed above, we model the set $\mathcal{R} = \{\preceq_1, \dots, \preceq_k\}$ by the relation $R : \mathcal{S} \times \mathcal{S} \leftrightarrow [1..k]$, such that $R^{(i)} = vec(\preceq_i)$ holds for each $i \in [1..k]$. In other words, each preorder $\preceq_i$ is modeled by a column of the relation R, and $R_{\langle x,x' \rangle i}$ is equivalent to $x \preceq_i x'$ for all search points x, x' and all $i \in [1..k]$. With this representation of the set $\mathcal{R}$ it is quite simple to compute the weak dominance relation, modeled by a vector w

of type $\mathcal{S} \times \mathcal{S} \leftrightarrow \mathbf{1}$. It holds

$$w := vec(\preceq) = \overline{\overline{R}\mathsf{L}},$$

where L is the universal vector of type $[1..k] \leftrightarrow \mathbf{1}$. This equation is a special case of Theorem 1 in the next section, therefore we do not prove the equation now. We obtain the RelView function

```
weakDom(R) = - (-R * L1n(R)^).
```

to determine the weak dominance relation in vector representation. Given the weak dominance relation $\preceq$, the strong dominance relation $\prec$ can be computed by

$$\prec = \preceq \cap \overline{\preceq}^\top$$

because for two search points x' and x it holds

$$\begin{aligned} x' \prec x &\iff \forall i \in [1..k] : f_i(x') \leq f_i(x) \land \exists i \in [1..k] : f_i(x') < f_i(x) \\ &\iff x' \preceq x \land \neg \forall i \in [1..k] : f_i(x) \leq f_i(x') \\ &\iff x' \preceq x \land \neg(x \preceq x') \\ &\iff x' \preceq x \land x' \overline{\preceq}^\top x \\ &\iff x' (\preceq \cap \overline{\preceq}^\top) x. \end{aligned}$$

This leads to the following RelView program `strongDom`, where the second parameter is an arbitrary relation of type $\mathcal{S} \leftrightarrow \mathcal{S}$ which is necessary to compute the relation representation of the weak dominance relation.

```
strongDom(R,Q)
  DECL w,W,S
  BEG w = weakDom(R);
      W = rel(w,Q);
      S = W & -W^
      RETURN S
  END.
```

Based on the strong dominance relation we can compute the set of all Pareto optimal search points. An element $x \in \mathcal{S}$ is Pareto optimal if there exist no $x' \in \mathcal{S}$ with $x' \prec x$. It follows that

$$\begin{aligned} x \text{ is Pareto optimal} &\iff \neg \exists x' : x' \prec x \\ &\iff \neg \exists x' : x \prec^\top x' \\ &\iff \neg(\prec^\top \mathsf{L})_x \\ &\iff \overline{(\prec^\top \mathsf{L})}_x. \end{aligned}$$

Hence, the set of Pareto optimal search points is represented by the vector o of type $\mathcal{S} \leftrightarrow \mathbf{1}$ defined by

$$o = \overline{\prec^\top \mathsf{L}}$$

and we obtain the RelView function

```
ParetoOpt(R,Q) = -(strongDom(R,Q)^ * Ln1(Q)).
```

In the set of Pareto optimal search points there can exist elements with the same fitness vector. In most cases one is interested in obtaining only one Pareto optimal search point for each fitness vector of the Pareto front. With the equivalence relation

$$\approx := \preceq \cap \preceq^\top$$

we have $x \approx x' \iff f(x) = f(x')$ for each $x, x' \in \mathcal{S}$. Obviously, the whole equivalence class $[x]_\approx$ is Pareto optimal if x is Pareto optimal. To determine a vector $r \subseteq o$ of representatives of the equivalence classes which are Pareto optimal, we use a linear order O and adopt the smallest element of each Pareto optimal equivalence class w. r. t. O. It holds

$$\begin{aligned} r_x &\iff o_x \land \forall x' : x \approx x' \to O_{xx'} \\ &\iff o_x \land \neg \exists x' : x \approx x' \land \overline{O}_{xx'} \\ &\iff o_x \land \neg \exists x' : (\approx \cap \overline{O})_{xx'} \\ &\iff o_x \land \overline{(\approx \cap \overline{O})\mathsf{L}}_x \\ &\iff (o \cap \overline{(\approx \cap \overline{O})\mathsf{L}})_x. \end{aligned}$$

We obtain the vector

$$r = o \cap \overline{(\approx \cap \overline{O})\mathsf{L}}$$

which contains exactly one representative of each Pareto optimal equivalence class with the following RelView program, where `O` is a linear order.

```
ParetoOptRep(R,O)
  DECL W,o,r
  BEG W = rel(weakdom(R),O);
      o = ParetoOpt(R,O);
      r = o & -((W & W^ & -O)*L(o))
      RETURN r
  END.
```

4 Reducing the Number of Objectives

Often multi-objective problems involve a large set of objectives for which the task is to compute a good approximation of the Pareto front. Often not all objectives are necessary to describe the approximation found by running some heuristic method such as an evolutionary algorithm [8]. In this case we are faced with the problem of computing a cardinality-wise minimal subset of objectives that preserves the same preference relation of the original set of objectives. Dealing with such a smaller set of objectives may make the decision easier for a decision maker which of the possible alternatives finally to choose.

In the following we deal with a given subset $X \subseteq \mathcal{S}$ instead of the whole search space. Therefore, we assume the introduced preorders $\preceq_1, \ldots, \preceq_k$ and $\preceq$ as relations of type $X \leftrightarrow X$. We consider the MINIMUM OBJECTIVE SUBSET PROBLEM introduced in [8] which can be defined as follows.

Definition 2. MINIMUM OBJECTIVE SUBSET PROBLEM
Given a set of solutions, the weak Pareto dominance relation $\preceq$ and for all objective functions $f_i \in \mathcal{F}$ the single relations $\preceq_i$ where

$$\preceq = \bigcap_{i \in [1..k]} \preceq_i \ .$$

Compute a subset $T \subseteq [1..k]$ of minimum size with

$$\preceq = \bigcap_{i \in T} \preceq_i \ .$$

As described in Sect. 2.2, we model the set $\mathcal{R} = \{\preceq_1, \ldots, \preceq_k\}$ by a relation $R : X \times X \leftrightarrow [1..k]$. Based on this relation R and the representation of subsets of $[1..k]$ by vectors $[1..k] \leftrightarrow \mathbf{1}$ (see Sect. 2.1) the following theorem states a relational expression to describe intersections of subsets of $\mathcal{R}$.

Theorem 1. *For every subset $T \subseteq [1..k]$ it holds*

$$vec(\bigcap_{i \in T} \preceq_i) = \overline{\overline{R}t},$$

where t is the vector of type $[1..k] \leftrightarrow \mathbf{1}$ that models the set T.

Proof. Using the definition of R and the fact that vec is a lattice isomorphism, we obtain $vec(\bigcap_{i \in T} \preceq_i) = \bigcap_{i \in T} vec(\preceq_i) = \bigcap_{i \in T} R^{(i)}$. For $y = \langle x, x' \rangle \in X \times X$ it follows

$$\begin{aligned} vec(\bigcap_{i \in T} \preceq_i)_y &\iff (\bigcap_{i \in T} R^{(i)})_y \\ &\iff \forall i \in T : R^{(i)}_y \\ &\iff \forall i \in T : R_{yi} \\ &\iff \forall i : t_i \rightarrow R_{yi} \\ &\iff \neg \exists i : t_i \wedge \overline{R}_{yi} \\ &\iff \overline{\overline{R}t}_y . \end{aligned}$$

$\square$

As an immediate consequence, with the set $[1..k]$ modeled by the universal vector $\mathsf{L} : [1..k] \leftrightarrow \mathbf{1}$, we obtain the vector representation of the weak dominance relation by $w = vec(\preceq) = \overline{\overline{R}\mathsf{L}}$ as stated in Sect. 3.

Using the equation of Theorem 1 we can now develop a relational expression to decide if a given subset $T \subseteq [1..k]$ is feasible in the sense that the intersection $\bigcap_{i \in T} \preceq_i$ equals the weak dominace relation $\preceq$.

Theorem 2. *For $T \subseteq [1..k]$ it holds*

$$\preceq \, = \bigcap_{i \in T} \preceq_i \iff \overline{\mathsf{L}\overline{(\overline{R}t \cup w)}} = \mathsf{L}.$$

Proof. For every subset $T \subseteq [1..k]$ it holds $\preceq \, \subseteq \bigcap_{i \in T} \preceq_i$. Using Theorem 1 we obtain

$$\begin{aligned}
\preceq \, = \bigcap_{i \in T} \preceq_i &\iff \bigcap_{i \in T} \preceq_i \, \subseteq \, \preceq \\
&\iff vec(\bigcap_{i \in T} \preceq_i) \subseteq vec(\preceq) \\
&\iff \overline{\overline{R}t} \subseteq w \\
&\iff \overline{\overline{R}t} \cap \overline{w} = \mathsf{O} \\
&\iff \overline{\overline{R}t \cup w} = \mathsf{O} \\
&\iff \mathsf{L}(\overline{\overline{R}t \cup w}) = \mathsf{O} \\
&\iff \overline{\mathsf{L}(\overline{\overline{R}t \cup w})} = \mathsf{L}.
\end{aligned}$$

□

Theorem 2 leads to a mapping $\varphi_{cut} : [[1..k] \leftrightarrow \mathbf{1}] \to [\mathbf{1} \leftrightarrow \mathbf{1}]$ defined by

$$\varphi_{cut}(t) = \overline{\mathsf{L}(\overline{\overline{R}t \cup w})}$$

to test if the vector t models a suitable subset to reduce the number of objectives, i.e. it holds

$$\varphi_{cut}(t) = \mathsf{L} \iff \preceq \, = \bigcap \{ \preceq_i \mid t_i \}.$$

Since φ_{cut} is a vector predicate in the sense of [11], it can be generalized to a test-mapping φ^Z_{cut} for evaluating the columns of relations of type $[1..k] \leftrightarrow Z$. More formally, we obtain for every set Z a mapping $\varphi^Z_{cut} : [[1..k] \leftrightarrow Z] \to [\mathbf{1} \leftrightarrow Z]$ by defining

$$\varphi^Z_{cut}(M) = \overline{\mathsf{L}(\overline{\overline{R}M \cup w\mathsf{L}})},$$

where L is the universal relation of type $\mathbf{1} \leftrightarrow Z$. For every relation $M : [1..k] \leftrightarrow Z$, the row vector $\varphi^Z_{cut}(M)$ represents the columns of M which model the subsets of $[1..k]$ which can be used to reduce the number of objectives, i.e it holds

$$\varphi^Z_{cut}(M)_{\perp j} \iff \varphi_{cut}(M^{(j)}) = \mathsf{L} \iff \preceq \, = \bigcap \{ \preceq_i \mid M^{(j)}_i \}.$$

By applying this approach to the membership relation $\mathsf{M} : [1..k] \leftrightarrow 2^{[1..k]}$ which models the power set of $[1..k]$, we are able to compute all suitable subsets. M is defined by

$$\mathsf{M}_{xY} \iff x \in Y$$

and lists all subsets of $[1..k]$ columnwise. With $\varphi_{cut}^{2^{[1..k]}}(\mathsf{M})$ we obtain a row vector $c : \mathbf{1} \leftrightarrow 2^{[1..k]}$ that specifies all subsets $T \subseteq [1..k]$ with $\preceq = \bigcap_{i \in T} \preceq_i$.

The test mapping $\varphi_{cut}^{2^{[1..k]}}$ leads to the following RelView program where `epsi(L1n(R^))` generates the membership relation of type $[1..k] \leftrightarrow 2^{[1..k]}$.

```
cut(R)
  DECL w, M, c
  BEG w  = weakDom(R);
      M  = epsi(L1n(R^));
      c  = -(Ln1(R)^ * -(-R * M | w * L1n(M)))
      RETURN c
  END.
```

The next step is to find the smallest subsets with this property. To this end, we use the size-comparison relation $\mathsf{C} : 2^{[1..k]} \leftrightarrow 2^{[1..k]}$, defined by $C_{AB} \iff |A| \leq |B|$ and define a mapping se, which computes for a given linear preorder relation Q and a vector v the smallest elements in v w. r. t. Q. More formally, with $se(Q, v) = v \cap \overline{\overline{Q}v}$ we obtain a vector such that

$$se(Q, v)_x \iff v_x \wedge \forall y : v_y \rightarrow Q_{xy}$$

holds. The immediate consequence is the following RelView function `se` to compute smallest elements.

```
se(Q,v)= v & -(-Q * v).
```

With $s = se(\mathsf{C}, c^\top)$ we obtain all subsets $T \subseteq [1..k]$ with the smallest cardinality that satisfy the property $\preceq = \bigcap_{i \in T} \preceq_i$. More formally, s is a vector of type $2^{[1..k]} \leftrightarrow \mathbf{1}$ with $s \subseteq c^\top$ and it holds

$$s_j \iff \preceq = \bigcap\{\preceq_i \mid \mathsf{M}_i^{(j)}\} \wedge \forall \ell : |\mathsf{M}^{(\ell)}| < |\mathsf{M}^{(j)}| \rightarrow \preceq \neq \bigcap\{\preceq_i \mid \mathsf{M}_i^{(\ell)}\}.$$

Hence each entry of s specifies a column of M that represents a suitable subset of $[1..k]$ with the smallest cardinality. By using the vector predicate φ_{cut} we can express the equivalence above as follows.

$$s_j \iff \varphi_{cut}(\mathsf{M}^{(j)}) = \mathsf{L} \wedge \forall \ell : |\mathsf{M}^{(\ell)}| < |\mathsf{M}^{(j)}| \rightarrow \varphi_{cut}(\mathsf{M}^{(\ell)}) = \mathsf{O}$$

The following RelView program computes the vector s. The size comparison relation on the power set $2^{[1..k]}$ is generated by `cardrel(L1n(R)^)` .

Fig. 1. The 5×5 successor relation

```
smallCuts(R)
  DECL c,C,s
  BEG c = cut(R);
      C = cardrel(L1n(R)^);
      s = se(C,c^)
      RETURN s
  END.
```

5 Experimental Results

In this section, we present the experimental results obtained for the objective reduction approach described in the previous section. We have carried out all of these computations using the RelView system which permits the evaluation of relation-algebraic terms and programs. All our computations were executed on a Sun Blade 1500 running Solaris 9 at 1000 MHz.

5.1 Results for Random Preorders

We have tested our program with instances of up to 145 randomly generated preorders computed by the RelView system. Generating a random total order relation of the type $X \leftrightarrow X$ is rather simple. Based on a given total hasse relation S and a randomly generated permutation P, both of the type $X \leftrightarrow X$, we obtain a random linear order by $O = (PSP^{\top})^*$, the reflexive-transitive closure of the hasse relation $PSP^{\top}$.
The following RelView program generates a random total order in this way, where the input Q is an arbitrary relation of type $X \leftrightarrow X$, `succ(Ln1(Q))` gives the successor relation (see Fig. 1 as an example) of the same type and `randomperm(Ln1(Q))` computes a random permutation.

```
randomOrder(Q)
  DECL S,P,O
  BEG S = succ(Ln1(Q));
      P = randomperm(Ln1(Q));
      O = refl(trans(P*S*P^))
      RETURN O
  END.
```

To obtain a preorder, we have to include some additional entries in the random order relation. To this end, we generate a random relation A and add $A \cup A^\top$ to $PSP^\top$ before computing the reflexive-transitive closure. Hence, the preorder is given by $(PSP^\top \cup A \cup A^\top)^*$. We use $A \cup A^\top$ instead of A to ensure that we get new entries which are not contained in the order relation $(PSP^\top)^*$ and therefore obtain a preorder instead of an order relation. The following RelView program generates a random preorder in this way, where the input is a nonempty relation which determines the type and influences the number of entries of the determined preorder. With `random(Q,Q)` a random relation $A : X \leftrightarrow X$ is generated such that for all $i, j \in X$ the probability of A_{ij} being true is $|Q|/|X|^2$.

```
randomPreOrder(Q)
  DECL S,P,A,PreO
  BEG S = succ(Ln1(Q));
      P = randomperm(Ln1(Q));
      A = random(Q,Q);
      PreO = refl(trans(P*S*P^ | A | A^))
      RETURN PreO
  END.
```

Using this program it is simple to produce random inputs consisting of k randomly generated preorders, modelled as a relation $R : X \times X \leftrightarrow [1..k]$. The following program successively determines k preorders `PreO` of type $X \leftrightarrow X$ and their vector representation `preO`. With `R = R | preO*p^`, where `p` is a point representing an element $i \in [1..k]$, the vector `preO` is inserted into `R` as the i-th column.

```
randomInput(Q,k)
  DECL R,z,PreO,preO,p
  BEG R = O(vec(Q)*k^);
      z = k
      WHILE -empty(z) DO
        PreO = randomPreOrder(Q);
        preO = vec(PreO);
        p = point(z);
        R = R | preO*p^;
        z = z & -p
      OD
      RETURN R
  END.
```

Our experimental results with respect to random preorders are given in Tab. 1. Depending on the probability used in our random function (which includes additional entries into the preorder relation) the results are shown. Note that such entries imply that solutions become indifferent which means that they have the same objective value with respect to the considered function. Tab. 1 shows that problems become easier with increasing this probability. The reason for that is that the number of different trade-offs becomes smaller when making solu-

Table 1. Results for random preorders with different values of p, where runtimes are given in seconds and the respective second columns give the reduced number of objectives

p	1/2500		1/500		1/250		3/500		1/125		1/50	
# obj	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.	Time	Obj.
5	0.04	5	0.01	5	0.01	5	0.01	4	0.01	4	0.16	2
15	0.63	8	0.51	15	0.48	14	0.46	12	0.47	9	0.38	3
25	5.71	7	0.25	11	0.07	14	0.07	14	0.06	14	0.05	8
35			1.22	13	0.20	22	0.16	23	0.16	14	0.15	5
45					0.49	26	0.41	28	0.41	25	0.38	19
55					39.06	17	0.87	30	0.80	30	0.79	18
65					5.29	23	2.44	29	3.19	32	1.54	17
75							2.91	34	2.72	38	3.52	23
85							7.86	27	5.61	33	4.43	20
95									9.24	40	7.87	27
105									11.60	40	23.86	21
115									16.80	42	24.43	26
125									27.37	44	23.19	29
135											32.52	32
145											77.77	30

tions indifferent. Depending on the choice of this probability RelView is able to deal with problems that involve 50 solutions and up to 145 objectives. The computation time for each instance is always less than 80 seconds.

5.2 Results for knapsack problems

A well-known problem in combinatorial optimization is the knapsack problem [12, 14] where a set of n items is given. With each item $j \in [1..n]$, a profit p_j and a weight w_j is associated. In addition a weight bound W is given and the goal is to select items such that the profit is maximized under the given weight constraint W. Omitted the weight constraints and optimizing both the profit and the weight simultaneously, Beier & Vöcking [1] have shown that for various input distributions the size of the Pareto front is polynomially bounded in the number of items. Their results imply that the well-known dynamic programming approach due to Nemhauser & Ullman [15] is able to enumerate these solutions in expected polynomial time. In the multi-objective knapsack problem [20], k knapsack problems are considered simultaneously. In this case we are faced with k knapsacks where knapsack i has capacity W_i. The weight of item j in knapsack i is denoted by w_{ij} and its profit by p_{ij}. The goal is to maximize for each knapsack i the function $f_i(x) = \sum_{j=1}^{n} p_{ij}x_j$ such that $w_i(x) = \sum_{j=1}^{n} w_{ij}x_j \leq W_i$ holds. Hence, the problem is given by the function $f = (f_1, \ldots, f_k)$ which should be optimized under the different weight constraints of the k knapsacks.

Table 2. Comparison of the relational approach with the exact one given in [8] where runtimes are given in milliseconds

Objectives	Runtime RelView	Runtime Exact Approach [8]
5	40	178
10	70	4369
15	590	166343
20	170	197690
25	430	5135040
30	1360	3203227
35	5990	—

We also investigated this problem in the same setting as done in [8]. The different solutions on which the objective reduction algorithms are executed are computed by running a multi-objective evolutionary algorithm called SPEA2 [19] on random instances with different number of objective functions. To compare the relation- algebraic approach with respect to efficiency we used the implementation of Brockhoff & Zitzler [8]. The results are given in Tab. 2 and show that the RelView program outperforms the previous approach drastically. RelView is able to compute for each instance an optimal solution within 6 seconds while the approach of Brockhoff and Zitzler needs large computation times and is unable to deal with instances which have more than 30 objectives.

6 Conclusions

In contrast to single-objective problems where one single optimal solution should be computed, the aim in multi-objective optimization is to compute solutions that represent the different trade-offs with respect to the objective functions. We have done a first step in examining such problems in terms of relational algebra and considered two important issues when dealing with multi-objective optimization. For the classical problem of computing the Pareto optimal solutions we have given a relation-algebraic approach that leads to a short RelView program which is at least able to deal with instances of moderate size. We have also examined the problem of reducing the number of objectives to be presented to a decision maker. It turns out that the relation-algebraic approach is very efficient for this problem and can deal with a large number of objectives. The comparison for the multi-objective knapsack problem shows that our algorithm outperforms the previous one drastically.

Acknowledgement

We thank Dimo Brockhoff and Eckart Zitzler for providing the implementation of their algorithms and the test instances for the multi- objective knapsack problem.

References

1. R. Beier and B. Vöcking. Random knapsack in expected polynomial time. *J. Comput. Syst. Sci.*, 69(3):306–329, 2004.
2. R. Berghammer. Solving algorithmic problems on orders and lattices by relation algebra and RelView. In *Proc. of CASC'06*, volume 4194 of *LNCS*, pages 49–63. Springer, 2006.
3. R. Berghammer, B. Leoniuk, and U. Milanese. Implementation of relational algebra using binary decision diagrams. In *Proc. of RelMiCS'01*, volume 2561 of *LNCS*, pages 241–257. Springer, 2002.
4. R. Berghammer and U. Milanese. Relational approach to boolean logic problems. In *Proc. of RelMiCS'05*, volume 3929 of *LNCS*, pages 48–59. Springer, 2006.
5. R. Berghammer and F. Neumann. RelView – an OBDD-based computer algebra system for relations. In *Proc. of CASC '05*, volume 3718 of *LNCS*, pages 40–51. Springer, 2005.
6. R. Berghammer, A. Rusinowska, and H. C. M. de Swart. Applying relational algebra and RelView to coalition formation. *European Journal of Operational Research*, 178(2):530–542, 2007.
7. D. Beyer, A. Noack, and C. Lewerentz. Efficient relational calculation for software analysis. *IEEE Transactions on Software Engineering*, 31(2):137–149, 2005.
8. D. Brockhoff and E. Zitzler. Dimensionality reduction in multiobjective optimization: The minimum objective subset problem. In K. H. Waldmann and U. M. Stocker, editors, *Operations Research Proceedings 2006*, pages 423–430. Springer, 2007.
9. M. Ehrgott. *Multicriteria Optimization*. Berlin, Springer, 2nd edition, 2005.
10. M. X. Goemans. Minimum bounded degree spanning trees. In *Proc. of FOCS'06*, pages 273–282. IEEE Computer Society, 2006.
11. B. Kehden. Evaluating sets of search points using relational algebra. In *Proc. of RelMiCS '06*, volume 4136 of *LNCS*, pages 266–280. Springer, 2006.
12. H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.
13. J. Könemann and R. Ravi. Primal-dual meets local search: Approximating MSTs with nonuniform degree bounds. *SIAM J. Comput.*, 34(3):763–773, 2005.
14. S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, 1990.
15. G. Nemhauser and Z. Ullmann. Discrete dynamic programming and capital allocation. *Management Sci.*, 15(9):494–505, 1969.
16. R. Ravi, Madhav V. Marathe, S. S. Ravi, Daniel J. Rosenkrantz, and Harry B. Hunt III. Many birds with one stone: multi-objective approximation algorithms. In *Proc. of STOC'93*, pages 438–447, 1993.
17. G. Schmidt and T. Ströhlein. *Relations and Graphs – Discrete Mathematics for Computer Scientists*. Springer, 1993.
18. M. Singh and L. C. Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *Proc. of STOC'07*, pages 661–670. ACM, 2007.
19. E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In K.C. Giannakoglou et al., editors, *Proc. of EUROGEN'01*, pages 95–100. CIMNE, 2002.
20. E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.