Journal of Logical and Algebraic Methods in Programming 83 (2014) 103–119

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Computing minimal extending sets by relation-algebraic modeling and development

Rudolf Berghammer *

Institut für Informatik, Christian-Albrechts-Universität zu Kiel, Olshausenstraße 40, 24098 Kiel, Germany

ARTICLE INFO

Article history:
Available online 10 February 2014

Keywords:
Social choice theory
Tournament
Minimal extending set
Relation algebra
Relation-algebraic modeling
RelView tool

ABSTRACT

In 2009 F. Brandt introduced minimal extending sets as a new tournament solution. Until now no efficient algorithm is known for their computation and, in fact, the NP-hardness of the corresponding decision problem has been proved quite recently by F. Brandt, P. Harrenstein and H.G. Seedig in a working paper. We develop a relation-algebraic specification of minimal extending sets. It is algorithmic and can be directly translated into the programming language of the ROBDD-based computer algebra system RelView. By this general and model-oriented approach we obtain almost the same efficiency as the specifically tailored program for minimal extending sets mentioned in another recent working paper of F. Brandt, A. Dau and H.G. Seedig. We also discuss an alternative approach that is based on testing the extending set property with relation-algebraic means, and a greedy strategy. Under favorable conditions it allows to solve much larger problem instances than our first solution and that of F. Brandt, A. Dau and H.G. Seedig.

1. Introduction

One of the most elementary questions in social choice theory is the aggregation of the preferences of certain individuals (voters, agents) in view of given alternatives (candidates) to a collective so-called *dominance relation*. For instance, in case of approval voting (see [9]) the individual preferences are the sets of alternatives the single voters approve and then collectively an alternative a dominates an alternative b if the number of voters which approve a is greater than the number of voters which approve b. Frequently it is assumed that there are no ties. Then the dominance relation is asymmetric (hence, irreflexive) and complete, i.e., the relation of a *tournament*. A tournament is acyclic iff it is a linear strict-order. Since finiteness is a general assumption in this context to ensure the existence of certain extremal sets and elements, acyclic tournaments possess exactly one winner that (strictly) dominates all other alternatives—the greatest element. But in case of cyclic tournaments it may happen that no alternative exists which dominates all other ones. To overcome this problem, a series of so-called *tournament solutions* has been proposed which define the sets of winners in such cases. For an overview see, for example, [20], and for computational issues see, for example, [10,19].

In [10,11] F. Brandt introduces minimal extending sets as a new tournament solution. Compared to other well-known tournament solutions its computational complexity has been an open problem for a long time. Quite recently it has been shown in the working paper [15] that deciding whether an alternative is contained in it is NP-hard. Even on small instances the computation of minimal extending sets seems to be rather difficult. In another working paper [14] F. Brandt, A. Dau

* Tel.: +49 431 7272; fax: +49 431 7613.
E-mail address: rub@informatik.uni-kiel.de.

http://dx.doi.org/10.1016/j.jlap.2014.02.002
2352-2208/

and H.G. Seedig report on their implementations of algorithms for the computation of the most important tournament solutions. Doing so, they also mention the cases where, despite the NP-hardness of the general problem, larger instances with specific properties can be treated successfully. Especially, this holds for the computation of the Banks set by means of the enumeration of all minimal feedback vertex sets via the elaborated algorithm of S. Gaspers and M. Mnich (see [18]). It is sufficiently fast on all tournaments with a not too large number of Banks trajectories (i.e., maximal transitive sets). But, although their program for the computation of minimal extending sets also bases on the algorithm of [18], the authors comment in [14] on it that it "already takes about 3 minutes on instances of 25 alternatives". In the original version of [14] that the author used when preparing the first version of the present paper they still write that "it is only feasible for instances of at most 20 alternatives".

In [7] we describe a simple computing technique for a series of tournament solutions. It rests upon relation algebra in the sense of [23,24] as the methodical tool, a goal-directed development of relation-algebraic specifications of tournament solutions from their formal logical descriptions, and the ROBDD-based computer algebra system RelView (see [4,21,22,27]) for the evaluation of the developed specifications and the visualization of the computed results. Because of the very positive results of our practical experiments, the above-mentioned difficulties when computing minimal extending sets, and since this tournament solution is not treated in [7], we have applied our technique to it and want to present the solutions and their performances in this paper.

Its remainder is organized as follows. In Section 2 we summarize the relation-algebraic preliminaries. The introduction of the minimal extending sets of a tournament and the development of an executable relation-algebraic specification from the formal logical description is done in Section 3. Section 4 shows how the result of Section 3 can be translated into RelView-code and presents results of our practical experiments with the tool. They demonstrate that, despite the very general and model-oriented approach, the use of ROBDDs to implement relations in RelView leads to a solution that is almost as efficient as the specifically tailored program mentioned in [14]. Section 5 describes an alternative method for solving the minimal extending sets problem that is based on a relation-algebraic model of truth values, the testing of the extending set property with relation-algebraic means and a greedy strategy. It assumes as pre-condition that the input has a unique minimal extending set. Our experiments together with those mentioned in [15] show that this seems to be true for all practical matters. In Section 6 we again present results of practical applications. They demonstrate that under favorable conditions the alternative approach allows to solve much larger problem instances than our first approach and the solution of F. Brandt, A. Dau and H.G. Seedig. The last Section 7 contains some concluding remarks.

2. Relation-algebraic preliminaries

In this section we recall the basics of heterogeneous relation algebra that are needed in the remainder of the paper. For more details, see [23,24] for example.

Given sets X and Y we write $R : X \leftrightarrow Y$ if R is a (typed, binary) relation with source X and target Y, i.e., a subset of the direct product $X \times Y$. If the sets of R's *type* $X \leftrightarrow Y$ are finite, then we may consider R as a Boolean matrix with $|X|$ rows and $|Y|$ columns. Since a Boolean matrix interpretation of relations is well suited for many purposes and also used by the RelView tool as the main possibility to depict them, in this paper we will frequently use matrix terminology and notation. Especially, we speak about the entries, rows and columns of a relation/matrix and write $R_{x,y}$ instead of $(x, y) \in R$ or $x\,R\,y$.

We assume the reader to be familiar with the five relation-algebraic basic operations, viz. R^T (*transposition*), $\overline{R}$ (*complement*), $R \cup S$ (*union*), $R \cap S$ (*intersection*) and $R;S$ (*composition*), the three special relations O (*empty relation*), L (*universal relation*) and I (*identity relation*), and the two relation-algebraic predicates $R \subseteq S$ (*inclusion*) and $R = S$ (*equality*). In case of the relations O, L and I we overload the symbols, i.e., avoid the binding of types to them. Furthermore, we assume that composition binds stronger than union and intersection.

For $R : X \leftrightarrow Y$ and $S : X \leftrightarrow Z$, by $syq(R, S) = \overline{R^\mathsf{T};\overline{S}} \cap \overline{\overline{R^\mathsf{T}};S}$ their *symmetric quotient* $syq(R, S) : Y \leftrightarrow Z$ is defined. We will only use its point-wise description saying that for all $y \in Y$ and $z \in Z$ it holds

$$syq(R, S)_{y,z} \iff \forall x : R_{x,y} \leftrightarrow S_{x,z}, \tag{1}$$

where the variable x ranges over X. In logical formulae the ranges of the variables are usually expressed via set-memberships. When we will translate logical formulae into relation-algebraic expressions, some of such memberships will be important for reaching the desired result, some however not, like $x \in X$ in case of (1). To improve readability, in the logical formulae of this paper we will explicitly state only the important membership-relationships that are transformed into relation-algebraic constructions. Those which are not used within a calculation are mentioned in the surrounding text.

Vectors are a well-known relation-algebraic means to model subsets of a given set. For our applications it suffices to define vectors as specific relations with the target being the singleton set $\mathbf{1} = \{\perp\}$. In the Boolean matrix interpretation a vector is a Boolean column vector and, as in linear algebra, we prefer in this context lower case letters. Furthermore, we omit in case of a vector $r : X \leftrightarrow \mathbf{1}$ always the second subscript, i.e., write r_x instead of $r_{x,\perp}$. Then r *describes*, by definition, the subset Y of X if for all $x \in X$ it holds r_x iff $x \in Y$.

If $r : X \leftrightarrow \mathbf{1}$ is a vector and Y is the subset of the set X that the vector r describes, then $inj(r) : Y \leftrightarrow X$ denotes the *embedding relation* (in [24] called *natural injection*) of the set Y into the superset X, which is induced by r. In Boolean