Problem Solving on Simple Games via BDDs

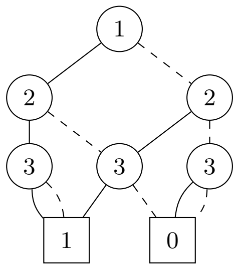
Rudolf Berghammer and Stefan Bolus

Abstract

Simple games are yes/no cooperative games which arise in many practical applications, especially in political life and the formation of alliances and coalitions. Binary decision diagrams (BDDs) can be used to represent, for instance, Boolean function, sets of subsets and relations. They are extensively studied and were applied to various research problems. In this extended abstract we'll give a motivation why it is a good idea to consider BDDs as another representation for simple games.

1 Motivation

A simple game (see e.g. [4]) is a pair $(N, \mathcal{W})$ where N is a set of so called *players* and $\mathcal{W} \subseteq 2^N$ is an up-set (with respect to set inclusion) of so called *winning coalitions*. Elements not in $\mathcal{W}$ are called *losing* and elements in 2^N are called *coalitions*. Binary decision diagrams (see e.g. [2]) are directed, labeled and acyclic graphs with a root and two designated sinks (1-/0-sink) such that each non-sink has two outgoing edges. As one can see from Fig. 1, they can be used to represent Boolean function in a very natural way. Each path corresponds to an assignment and the sink determines the outcome. Because simple games are technically a set of subsets they can easily be represented by their characteristic function $\chi : \{0,1\}^{|N|} \to \{0,1\}$ where the first player corresponds to the first Boolean variable and so one.



variable 1	2	3	f-value
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Figure 1: A BDD for a Boolean function. Numbers inside circles (labels) correspond to Boolean variables. The rectangular nodes are the 1- and 0-sink, respectively. Edges are directed downwards. Solid/dashed edges are 1-/0-edges.

So called *quasi-reduced and ordered binary decision diagrams* (QOBDDs) are BDDs that share sub-BDDs whenever possible. E.g., in Fig. 1 the center node with label 3 is shared. QOBDDs are often small in practice. In general, however, they can grow exponentially in the number of Boolean variables. The same holds for monotone Boolean functions and even threshold functions[1] where in the latter case the bounds for the number of nodes are $\mathcal{O}(2^{n/2})$ and $\mathcal{O}(|N|Q)$ if Q is the threshold (see [3, 1]), but even the latter bound is rarely reached in practice. A similar bound can be shown for multiple weighted voting games (MWVG; see again [1]). Additionally, different classes of QOBDDs (WVG, MWVG, any) can exhibit useful properties which perhaps can be exploited to derive efficient algorithms. For instance, building the QOBDD for the minimal winning coalitions of a WVG from the QOBDD of its winning coalition is a linear time algorithm in the number of QOBDD nodes.

[1] Threshold functions correspond exactly to characteristic functions of weighted voting games (WVGs).

The use of QOBDDs offers not only another representation of simple games, WVG and MWVG, but due to its relatively compact representation of simple games it also allows to solve problems for real world instances. A feature which is offered by other explicit representation just to a very limited degree. Moreover, QOBDDs can be manipulated like sets as long as they represent an up-set. For instance, constraints for the winning coalitions can be applied. Winning coalitions of multiple games can be combined not only using conjunction but also using other operations like disjunction to model multiple opportunities for a coalition to win. For instance, the US Federal Legal System and Taylor's and Zwicker's Magic Squares can be modeled using that.

Despite the very famous problem of computing different power indices for simple games, the computation of the desirability relation on the players and the test for dummy players are two basic problems which appear in some other more complex problems like the test to be a WVG or not . Here, one can profit from the fact that BDDs were already applied to many problems from different areas and many problems have been solved in a slightly different notion. For instance, dummy players in simple games correspond exactly to redundant variables in Boolean functions. Other problems can be solved using existing operations on QOBDDs and some simple algorithms like the following one to compute the QOBDD for the blocking coalitions (and thus the dual game) from the winning coalition of a simple game:

Compls(v) $\equiv$
  if v is a sink **then return** v
  elsif v was already visited with result r **then return** r
  else r := $\mathsf{ite}(i, \mathit{Compls}(\mathsf{else}(v)), \mathit{Compls}(\mathsf{then}(v)))$
    mark v as visited with result r and **return** r

Graphically, the algorithm just exchanges each node's 1- and 0-edge. Thus, it has a running time linear in the number of nodes. This allows to handle even larger real world problems like the International Monetary Fund with 186 players which has about 16 mil. nodes.

Our research in this direction has two main objectives. The first one is to study the complexity of known problems using the BDD representation. This is especially interesting since QOBDDs can have exponential size in general but have a bounded size for special classes like WVGs. The second objective is to develop and provide applicable methods which can be used not only by computer scientists and maybe serve as a foundation for new questions.

References

[1] S. Bolus. Power indices of simple games and vector-weighted majority games by means of binary decision diagrams. 2010. Submitted.

[2] R.E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3):293–318, 1992.

[3] K. Hosaka, Y. Takenaga, and S. Yajima. On the size of ordered binary decision diagrams representing threshold functions. *Algorithms and Computation*, pages 584–592, 1994.

[4] A.D. Taylor and W.S. Zwicker. *Simple Games: Desirability Relations, Trading, Pseudoweightings.* Princeton University Press, 1999.

Rudolf Berghammer and Stefan Bolus
Institut für Informatik, Universität Kiel
Olshausenstraße 40
24098 Kiel, Germany
Email: `{rub|stb}@informatik.uni-kiel.de`