The Number of Knight's Tours Equals 33,439,123,484,294 — Counting with Binary Decision Diagrams

Martin Löbbing* and Ingo Wegener*

FB Informatik, LS II, Univ. Dortmund, 44221 Dortmund, Germany
e-mail: loebbing/wegener@ls2.informatik.uni-dortmund.de

Submitted: October 25, 1995; Accepted: January 19, 1996.

AMS subject classification: 05-04, 05C38, 68R10.

Abstract

The number of knight's tours, i.e. Hamiltonian circuits, on an 8×8 chessboard is computed with decision diagrams which turn out to be a useful tool for counting problems.

1 Introduction

Binary decision diagrams are representations of Boolean functions with many applications in hardware verification and computer-aided design (Bryant (1992)). We believe that binary decision diagrams also have many applications in combinatorics and graph theory. To support this claim we determine the number of cycle coverings of the knight's graph on an 8×8 chessboard as well as the number of knight's tours with binary and slightly more general multi decision diagrams. We have chosen the knight's tour problem because of its long history (famous mathematicians like Euler, Legendre, and Vandermonde (see Rouse Ball and Coxeter (1987)) have worked on this problem) and since it is a combinatorial chess problem known to everybody. Our results are the following ones.

Theorem 1 *The number of cycle coverings of the directed knight's graph for* 8×8 *chessboards equals* α^2, *where* $\alpha = 2,849,759,680$, *i. e. it equals 8,121,130,233,753,702,400.*

Theorem 2 *The number of undirected knight's tours on an* 8×8 *chessboard equals 33,439,123,484,294. The number of directed knight's tours is twice the number of undirected ones.*

2 Binary and multi decision diagrams

Many counting problems can be modeled as the problem of counting the number of satisfying inputs for a function $f : A_1 \times \cdots \times A_n \to \{0,1\}$. An input $a = (a_1, \ldots, a_n)$ is called satisfying if $f(a) = 1$. Some types of representations of functions allow to solve this problem efficiently, if f has a short (especially not exponential) representation. For typical representations of finite functions, like circuits

*Supported in part by DFG grant We 1066/7–3.

or formulae, the problem of counting the satisfying inputs is #P-complete. Hence, we look for types of representations, where it is easy to count the satisfying inputs.

Definition 1 *A variable ordering is a permutation π on $\{1,\ldots,n\}$. An ordered multi decision diagram (OMDD) for a function $f : A_1 \times \cdots \times A_n \rightarrow \{0,1\}$, where $A_i = \{0,\ldots,k_i - 1\}$, and a variable ordering π is a rooted directed acyclic graph with at most two sinks labelled with 0 or 1 and inner nodes, each one labelled with an index $i \in \{1,\ldots,n\}$. Each inner node with label i has k_i outgoing edges labelled with $0,\ldots,k_i - 1$, each of these edges leads to one of the nodes with label j, where $\pi(j) > \pi(i)$, or to a sink. For the input $a = (a_1,\ldots,a_n) \in A_1 \times \cdots \times A_n$ the unique path leaving the root and following the edge with label a_i at nodes with label i has to reach the sink with label $f(a)$. If $k_i = 2$ for all i, we have OBDDs (B=binary).*

Typically, there are a lot of nodes with label i, but each directed path can contain at most one of them. It is also possible that no node with label i exists. In general it is difficult to find an appropriate variable ordering to obtain small OMDDs, but in our applications we define the functions in such a way that the variable ordering π, where $\pi(i) = i$ for all i, is appropriate. In the following we only use this variable ordering. We summarize known results on OMDDs. It is easy to count the number of satisfying inputs. A formal description, e. g. a circuit, can be transformed step by step into an OMDD. Let $f := g \otimes h$ for some binary operator $\otimes$. Then it is easy to obtain an OMDD for f from OMDDs for g and h. Moreover, the OMDD of minimal size for a given function and variable ordering is unique and called reduced. It is possible to create reduced OMDDs only. The whole approach is heuristic, since for some functions OMDDs are exponentially larger than, e. g., circuits. The practical usefulness of OMDDs relies on the uniqueness of reduced OMDDs. Therefore, we avoid typical problems which arise in backtracking algorithms. Let us consider all the different "situations" for the different values of $(a_1,\ldots,a_i)$, where $i < n$. A situation corresponds to the subfunction of f, where the first i variables are replaced by constants. In backtracking algorithms it is a problem to detect situations without any solutions (the subfunction is the constant 0) and to detect isomorphic situations (the subfunctions are equal). In OMDDs all situations without any solutions are represented automatically by the sink with label 0 and isomorphic cases are represented automatically by the same node (for details see Bryant (1992)).

For the knight's tour problem we have to count the number of satisfying inputs of the following function TOUR defined on 64 variables. We assume a fixed numbering of the squares. Then k_i is the number of knight's moves leaving the i-th square. Hence, $a = (a_1,\ldots,a_{64})$ describes for each square a knight's move leaving this square and TOUR$(a) = 1$ if and only if a describes a knight's tour.

3 Counting knight's tours with decision diagrams

The counting of cycle coverings is a quite direct application of OBDD techniques. It is easy to see that the number of cycle coverings on an 8×8 chessboard is equal to α^2, if α is the number of one-to-one mappings from the white to the black squares respecting the legal moves of a knight. The Boolean function deciding whether some choice of moves represents such a one-to-one mapping is described as a circuit and then translated gate by gate into an OBDD. We obtain α as the number of satisfying inputs. We got the same result for α with three independent approaches: OBDDs of size 598,472, ZBDDs (an OBDD variant introduced by Minato (1993)) of size 406,660 (6.5 CPU minutes on a SUN 670/140 with 128 MB storage), and with backtracking (the most clever approach took more than 30 days).

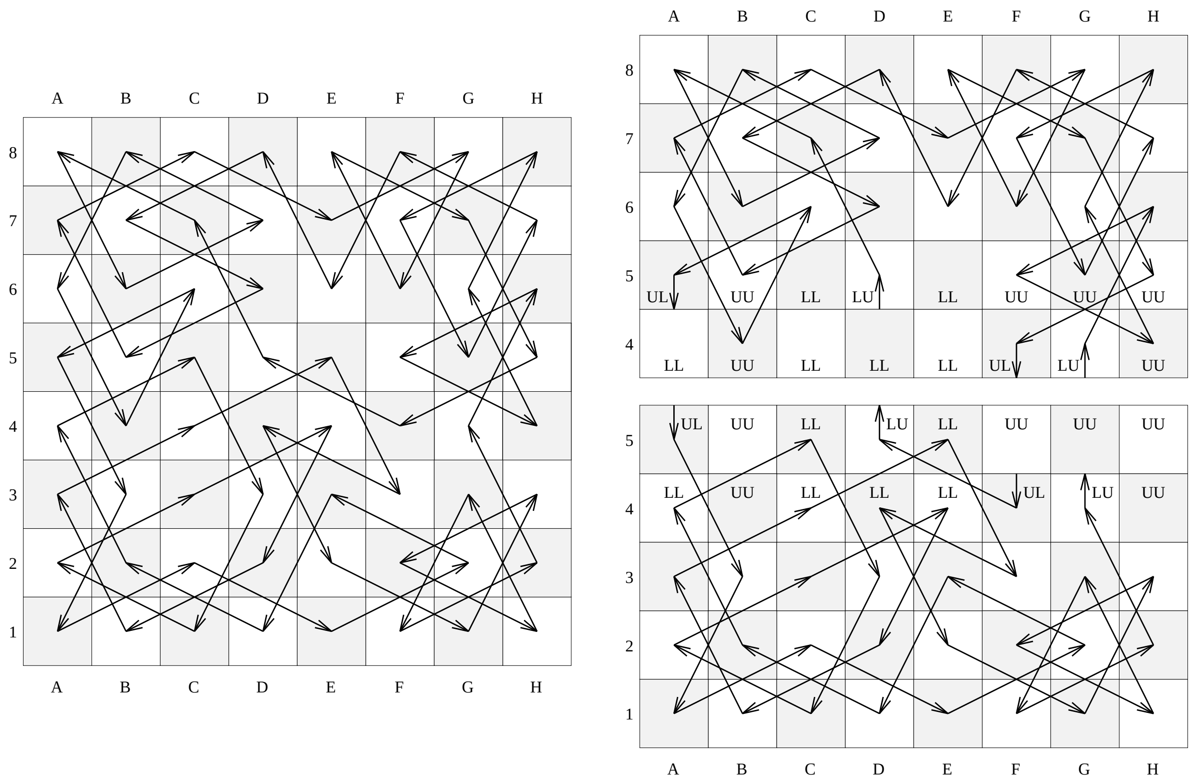

Figure 1: An example of a knight's tour

Counting knight's tours is much more difficult, since tours cannot be described by local properties. We use a hybrid approach of OMDD techniques, backtracking and divide-and-conquer. The ideas are illustrated in Fig. 1.

The chessboard is divided into L (row 1, 2, and 3), M (row 4 and 5) and U (row 6, 7, and 8). The lower "half board" LM consists of L and M and the upper half board UM consists of U and M. The overlapping of the half boards leads to a smaller number of cases in our divide-and-conquer approach. If we define that a move from row 4 to row 5 belongs to LM and a move from row 5 to row 4 belongs to UM, each move belongs to one half board.

Let us consider a fixed directed knight's tour. A square belongs to UL, if the move to this square belongs to the upper half board and the move leaving this square belongs to the lower half board. LU, LL and UU are defined similarly. Each square of U (or L) belongs to UU (resp. LL). We divide our problem according to the different partitions of M into UL, LU, LL and UU. It is sufficient to consider $\binom{16}{1}^2 + \binom{16}{2}^2 + \cdots + \binom{16}{8}^2 = 383,358,644$ cases. For each $i \in \{1, \ldots, 8\}$ we choose each pair (A, B) of subsets of M of size i and let $LL = A \cap B$, $UL = A - A \cap B$, $LU = B - A \cap B$ (ensuring the necessary condition $|UL| = |LU|$) and $UU = M - (A \cup B)$. We obtain all cases, where $|LL| \leq 8$. Because of symmetry it is sufficient to multiply the number of solutions of the cases where $|UU| \geq 9$ by 2.

Each directed knight's tour consists in $LM - UU$ of cycle free disjoint paths combining each node in UL with one node in LU. Let $f : UL \rightarrow LU$ be the one-to-one mapping such that $f(v)$ is the endpoint of the path starting at v, i.e. v points to $f(v)$. Similarly, we obtain for the paths in $UM - LL$ a one-to-one mapping $g : LU \rightarrow UL$. The pointers defined by f and g lead to a cycle on $UL \cup LU$. Such pairs (f, g) are called good. In our example $f(A5) = G4$, $f(F4) = D5$, $g(G4) = F4$ and $g(D5) = A5$ define the cycle $A5 \rightarrow G4 \rightarrow F4 \rightarrow D5 \rightarrow A5$. Disjoint path systems on $LM - UU$ and $UM - LL$ define a directed knight's tour iff (f, g) is good. Let $\#(UL, LU, LL, UU, f)$ be the number of disjoint path systems on $LM - UU$ respecting the partition of M and respecting f, and let $\#(UL, LU, LL, UU, g)$ be defined similarly for $UM - LL$. Then the number of directed knight's tours equals

$$\sum_{(UL,LU,LL,UU)} \sum_{(f,g) \text{ good}} \#(UL, LU, LL, UU, f) \cdot \#(UL, LU, LL, UU, g)$$

and the number of undirected tours is half this number.

To check the rather "global" property that a set of moves respects a given function $f : UL \rightarrow LU$ is difficult with OMDDs. They become too large to be stored in the storage space which is nowadays available. We create for each partition (UL, LU, LL, UU) two OMDDs for $LM - UU$. The first one checks whether each white square is left once and each black square is reached exactly once by the moves chosen for the white squares and whether the partition (UL, LU, LL, UU) is respected. The other OMDD does the same for the moves leaving the black squares. For each pair of inputs satisfying the two OMDDs we like to check whether they describe a cycle free path system. Each such pair has to be counted for the corresponding parameter $\#(UL, LU, LL, UU, f)$. We obtain the parameters $\#(UL, LU, LL, UU, g)$ by symmetry. Finally, the above formula is evaluated.

The use of OMDDs has two major advantages. It is easy to check whether a reduced OMDD represents the constant 0. Then nothing has to be done. Otherwise, the satisfying inputs can be enumerated by backtracking on the OMDD without considering any non satisfying input.

The computation can be easily distributed to many computers. We have performed the computation with 20 SUN work stations within approximately four months. The CPU time is much less, since we could use the computers only during their idle times.

Conclusion

BDD techniques known from hardware verification and computer-aided design have been applied to the solution of open combinatorial problems. The number of knight's tours on the 8×8 chessboard is determined.

References

Bryant, R. E. (1992). Symbolic Boolean manipulation with ordered binary-decision diagrams. ACM Computing Surveys 24(3), 293–318.

Minato, S.-I. (1993). Zero-suppressed BDDs for set manipulation in combinatorial problems. Proc. of the 30th ACM/IEEE Design Automation Conference, 272–277.

Rouse Ball, W. W. and Coxeter, H. S. M. (1987). Mathematical recreations and essays. Dover, New York.