March Madness is (NP-)Hard
(draft)

David Liben-Nowell, Moses Liskov, Chris Peikert, Abhi Shelat, Adam Smith, and Grant Wang

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA, 02139, USA
{dln,mliskov,cpeikert,abhi,asmith,gjw}@theory.lcs.mit.edu

Abstract. We formally define the MARCH-MADNESS decision problem (inspired by popular betting pools for the NCAA basketball tournament), and prove it NP-complete.

1 Introduction

The National Collegiate Athletic Association (NCAA) Basketball Tournament[3], held every March, is a tournament among the top 64 (or more recently, 65) collegiate basketball teams in the United States. The tournament is set up in single-elimination form: to start, each team occupies the leaf of a complete binary tree. Two teams occupying sibling nodes play against each other, and the winner moves up to occupy its parent node, while the loser is out of the tournament. This process continues until only one team (occupying the root node) remains, and that team is crowned the national champion.

Popular distractions during the tournament are so-called "March Madness" pools, often organized among friends and co-workers. Each participant in a pool predicts the winner of every game, by filling out the internal nodes of the tournament tree, prior to the start of the tournament. Once the tournament begins, everyone's predictions are made public. Participants get some number of points for each winner they correctly predict (note that they need not predict *which team* is defeated by the winner). After the entire tournament has been played, the participant with the most points is declared the winner of the pool. Most scoring methods give more points for correct predictions in later games, because it is hard to predict which teams will advance many rounds, and to predict the winner of a game between two closely-matched teams.

As the tournament progresses, many interesting phenomena can occur. For example, many participants may have predicted Team A to win many games, when in fact it may lose its first game. In this case, those participants lose the potential to gain many points later in the tournament, because Team A cannot possibly win any of the games it was predicted to win. For this reason, early in the tournament one participant may have many more points than all the others, but no hope of winning the pool (because the teams he picked to win many games have already lost).

As a participant of a March Madness pool, one might naturally ask: "Given the results of the tournament thus far, is there any way I can win the pool? Or is it the case that, no matter which teams win from now on, some other participant will have a higher score than me?" Of course, for the actual constant-sized NCAA tournament tree, this question can be answered in time linear in the size of the pool, by enumerating all tournament outcomes in constant time. However, we show that under an appropriate formalization (i.e., in the sizes of the pool and the tournament tree), answering this question is NP-complete. Our results hold for a wide class of scoring methods; see the remark at the end of Section 4.

This result is interesting because it leaves the participant with little choice but to keep his interest in tournament, because he cannot efficiently tell if he is eliminated from the pool until the end of the tournament (unless P=NP). Note that we never make any assumptions about how likely it is for a team to win (i.e., odds or probabilities); even if a participant were omnipotent in the sense that he could control the outcome of any remaining games, he still would not be able to efficiently determine whether doing so would allow him to win the pool.

This result is also interesting in contrast to similar problems. In Major League Baseball, for example, after the season has started, there is an efficient procedure (by a reduction to a flow problem [4]) for determining if a team can make it to the playoffs or not (even though this decision is based on the outcomes of all the other games in the same league).

2 Definitions

Before formally define the MARCH-MADNESS problem, we first need to define some related terms.

Definition 1 (March-Madness Vocabulary). *We define the following:*

- *A* tournament (T, λ) *is a complete binary tree* T *(having vertices* V*, leaves* L*, and internal nodes* I*), with a bijective labelling* $\lambda : L \to \{1, \ldots, |L|\}$.
- *A* bracket β *for a tournament* (T, λ) *is a complete set of predictions about the games of a tournament. That is,* β *is a labelling of* V*, where* $\beta(l) = \lambda(l)$ *for all* $l \in L$*, and* $\beta(u) \in \{\beta(s), \beta(t)\}$*, for each* $u \in I$ *having children* s, t.
- *A* partial result π *of a tournament* (T, λ) *is a labelling of some* $V' \subseteq V, V' \supseteq L$*, where* $\pi(l) = \lambda(l)$ *for all* $l \in L$*, and for every* $u \in I$ *having children* s, t*:* $u \in V' \Rightarrow s, t \in V'$ *and* $\pi(u) \in \{\pi(s), \pi(t)\}$.
- *A* tournament result *for a tournament* (T, λ) *is simply a partial result defined on all of* V*. A tournament result* ρ *is* consistent *with a partial result* π*, written as* $\rho \succeq \pi$*, if* $\rho(v) = \pi(v)$ *for all nodes* v *for which* π *is defined.*
- *The* score $\sigma_\pi(\beta)$ *of a bracket* β *relative to a partial result* π *(defined on* V'*) is* $|\{v \in V' \cap I \ : \ \beta(v) = \pi(v)\}|$.

Definition 2 (March-Madness). *The language*

$$\textsc{March-Madness} = \{(T, \lambda, \delta, \{\beta_j\}_{j=1}^m, \pi) \ : \\ \exists \text{ a tournament result } \rho \succeq \pi \text{ such that} \\ \sigma_\rho(\delta) \geq \max_{j=1,\ldots,m} \sigma_\rho(\beta_j)\},$$

where

- (T, λ) *is a tournament,*
- δ *is the* distinguished bracket *for* (T, λ)*,*
- $\{\beta_j\}_{j=1}^m$ *is a set of* m competing brackets *for* (T, λ)*, and*
- π *is a partial result of* (T, λ)*.*

3 Reduction from an NP-complete problem

We reduce the NP-complete problem 3SAT to March-Madness. We design a function Γ which, given a 3SAT instance ϕ having n variables $\{x_i\}_{i=1}^n$ and m clauses $\{c_j\}_{j=1}^m$, outputs an instance $(T, \lambda, \delta, \{\beta_j\}_{j=1}^m, \pi)$ of March-Madness, with one competing bracket β_j for each clause c_j in ϕ. The tree T is a complete binary tree having $8 \cdot 2^{\lceil \log_2 n \rceil}$ leaves. It is constructed by connecting n "widgets" of 8 leaves each, one corresponding to each variable x_i in ϕ, plus enough other widgets to complete the binary tree. The labelling λ of the leaves L is an arbitrary injective function.

We now describe the partial results π and the distinguished bracket δ. First, we arbitrarily assign the results of all the first-round games by labelling the parents of L. Then δ is constructed as follows: in each of n widgets corresponding to variables from ϕ, it makes the correct predictions in exactly three of the four first-round games. The predicted winner of the other first-round game is also predicted to win in the second round, but not in the third round. Finally, in the other second-round game, the winner is chosen arbitrarily. See Figure 1 for the actual predictions we will use, with the partial results from π included. Note the special vertex v_i, and that the distinguished bracket's prediction at vertex v_i cannot possibly be correct, given π. Also note that for each v_i, b_i is its "bottom" child (the lower of its two children in Figure 1), and u_i is its "upper" child.

We now describe how the competing brackets are constructed. For each clause c_j in ϕ involving distinct variables x_r, x_s, x_t we construct a competing bracket β_j, which is identical to δ, except in the widgets corresponding to variables x_r, x_s, x_t. In place of the predictions shown in Figure 1, we use the predictions from one of the two widgets pictured below. For each non-negated variable in the clause, we include the predictions shown in Figure 2. For each variable that appears negated in the clause, we include the predictions shown in Figure 3. Note that in each widget, there is one prediction which is not fixed. In two of these three widgets, β_j makes the incorrect prediction, and in the other widget β_j makes the correct prediction.

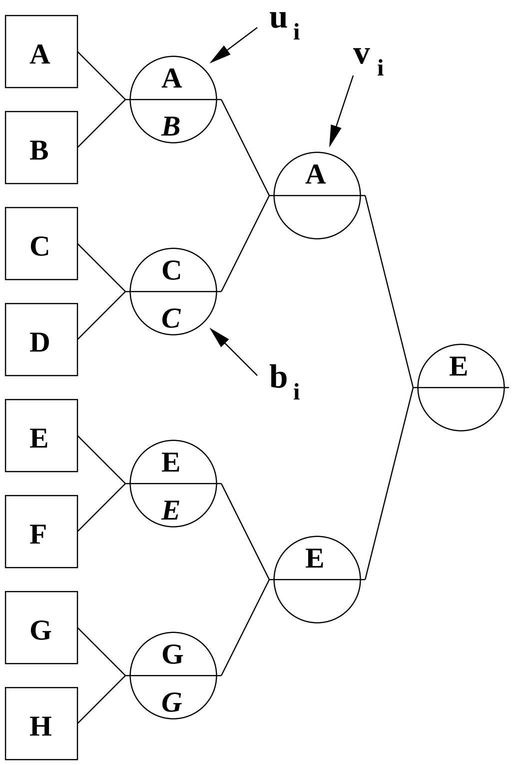

Fig. 1. A widget from T, corresponding to the variable x_i in ϕ, with distinguished draw δ and partial results π. The upper half of each internal node u contains $\delta(u)$, while the lower half contains $\pi(u)$ (if it is defined). Note that $\delta(v_i)$ is incorrect, regardless of the outcome of the game between B and C.

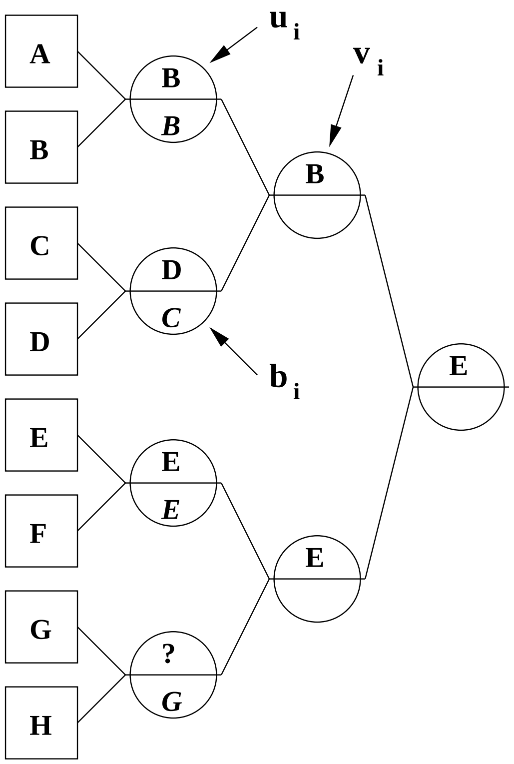

Fig. 2. Partial results π, and predictions from β_j for widget i, where clause c_j contains the literal x_i. Note that it is possible for $\beta_j(v_i)$ to be correct. Also note that the prediction for the game between G and H is not fixed.

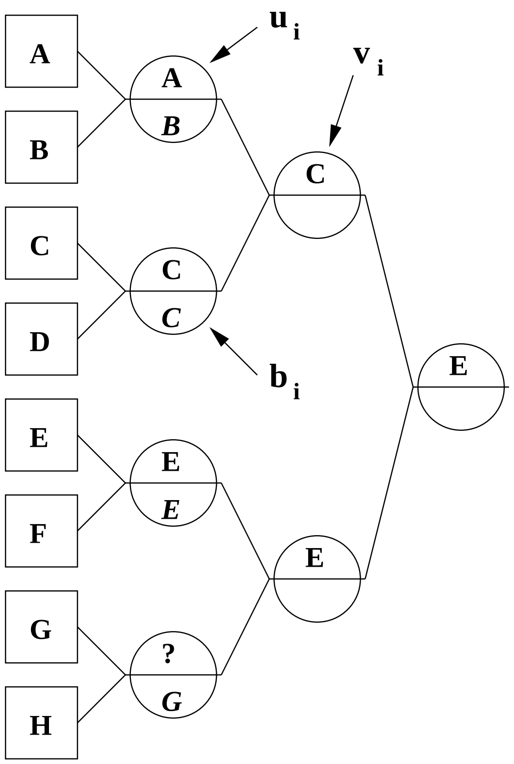

Fig. 3. Partial results π, and predictions from β_j for widget i, where clause c_j contains the literal $\overline{x_i}$. Note that it is possible for $\beta_j(v_i)$ to be correct. Also note that the prediction for the game between G and H is not fixed.

4 Correctness of the reduction

We are now ready to prove the following theorem.

Theorem 1. $\phi \in \text{3SAT} \iff \Gamma(\phi) = (T, \lambda, \delta, \{\beta_j\}_{j=1}^m, \pi) \in$ MARCH-MADNESS.

Proof: We first make a few observations about the brackets. First, note that for all j, $\sigma_\pi(\delta) = 2 + \sigma_\pi(\beta_j)$. Next, note that for all j, and for all (except three) nodes v for which π is undefined, $\delta(v) = \beta_j(v)$. The three exceptional nodes are precisely v_r, v_s, v_t, where variables x_r, x_s, x_t appear in clause j of ϕ. Combining these observations, we see that for any $\rho \succeq \pi$, $\sigma_\rho(\beta_j) > \sigma_\rho(\delta) \Rightarrow \beta_j(v_r) = \rho(v_r), \beta_j(v_s) = \rho(v_s), \beta_j(v_t) = \rho(v_t)$.

$\Rightarrow$: let $\boldsymbol{x}$ be a satisfying assignment of ϕ. Then we construct a tournament result ρ as follows: for each i, if x_i is true, let $\rho(v_i) = \rho(b_i)$; if x_i is false, let $\rho(v_i) = \rho(u_i)$. Assign ρ arbitrarily (but so that it is a tournament result) at all other nodes.

Because every clause in ϕ is satisfied by $\boldsymbol{x}$, each clause c_j contains some true literal, say either some variable x_i or its negation. Then $\beta_j(v_i) \neq \rho(v_i)$ by construction, so by the above observation, $\sigma_\rho(\delta) \geq \sigma_\rho(\beta_j)$ as desired.

$\Leftarrow$: given a tournament outcome ρ for which δ has the highest score, construct a satisfying assignment $\boldsymbol{x}$ for ϕ as follows: if $\rho(v_i) = \rho(b_i)$, let x_i be true, otherwise let x_i be false. Because the distinguished bracket δ has the highest score, each competing bracket β_j has $\beta_j(v_i) \neq \rho(v_i)$ for some variable x_i involved in clause c_j. By construction, the corresponding literal (either x_i or its negation) is true in c_j, so c_j is satisfied, therefore $\boldsymbol{x}$ satisfies ϕ. $\square$

Corollary 1. MARCH-MADNESS *is* NP*-complete.*

Proof: The running time of the reduction is polynomial in the size ϕ: building a widget requires $O(1)$ time, as does connecting two disconnected binary trees. At most $2n$ widgets are created, and are connected into one tree. Creating each of the $m+1$ draws require time linear in the size of tree, so the total running time of the reduction is $O(mn) = \text{poly}(|\phi|)$.

Finally, MARCH-MADNESS $\in$ NP: given any instance $(T, \lambda, \delta, \{\beta_j\}, \pi)$, a witness of membership in MARCH-MADNESS consists of any tournament result $\rho \succeq \pi$ such that $\sigma_\rho(\delta) \geq \max_{j=1,\ldots,m} \sigma_\rho(\beta_j)$. Checking the validity of the witness consists of counting the number of correct predictions from each bracket, and verifying that the distinguished bracket has the highest score. This again can be done in time polynomial in the number of teams in the tournament and the number of brackets in the pool.

By Theorem 1, 3SAT $\leq_P$ MARCH-MADNESS, and the result follows. $\square$

Remark: typically, actual betting pools offer more points for correctly predicting games in later rounds of the tournament. For example, each first-round game may be worth 1 point, each second-round game 2 points, and so on. Our results remain valid under these types of scoring systems, as long as all games in the same round are worth the same number of points, and the value of a second-round game is at most a polynomial (in the number of brackets) factor of the value of a first-round game. If the value of a second-round game is s, we only need the distinguished bracket's score to lead all other brackets' scores by between $2s$ and $3s - 1$ (inclusive) after the first round. Therefore we can modify the reduction by enlarging the size of the widgets so that each clause's bracket makes enough incorrect predictions in the first round.

5 Acknowledgements

We would like to thank Umesh Shankar for a helpful discussion, in which he conjectured that MARCH-MADNESS might be NP-complete.

References

1. Stephen A. Cook. The complexity of theorem-proving procedures. In *ACM Symposium on Theory of Computing*, pages 151–158, 1971.
2. Michael R. Garey and David S. Johnson. *Computers and Intractibility*. W. H. Freeman and Company, 1979.
3. NCAA final four tournament website. `http://www.finalfour.net`.
4. Kevin Wayne. A new property and a faster algorithm for baseball elimination. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1999.