A Tractability Gap Beyond Nim-Sums: It's Hard to Tell Whether a Bunch of Superstars Are Losers

Kyle Burke
Florida Southern College, Lakeland, FL, USA

Matthew Ferland
University of Southern California, Los Angeles, CA, USA

Svenja Huntemann
Mount Saint Vincent University, Halifax, Canada

Shanghua Teng
University of Southern California, Los Angeles, CA, USA

Abstract

In this paper, we address a natural question at the intersection of combinatorial game theory and computational complexity: "Can a sum of simple *tepid games* in canonical form be intractable?" To resolve this fundamental question, we consider *superstars*, positions first introduced in *Winning Ways* where all options are *nimbers*. Extending Morris' classic result with hot games to tepid games, we prove that disjunctive sums of superstars are intractable to solve. This is striking as sums of nimbers can be computed in linear time. Our analyses also lead to a family of elegant board games with intriguing complexity, for which we present web-playable versions of the rulesets described within.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Combinatorial Game Theory, NP-hardness, Superstars

Digital Object Identifier 10.4230/LIPIcs.FUN.2024.8

Funding *Shanghua Teng*: Supported in part by NSF Grant CCF-2308744 and the Simons Investigator Award from the Simons Foundation.

1 Introduction

"The whole is greater than the sum of their parts" is an ancient phrase that particularly exemplifies combinatorial game theory. As an area of mathematics dedicated to analyzing what happens when several games are combined, the field is rich with results both in isolation and with interdisciplinary connections. Indeed, even casually, games are often combined for enjoyment, such as Bughouse (2 simultaneous games of Chess) and Ultimate Tic Tac Toe (9 simultaneous games of Tic Tac Toe).

While several different ways to combine games are studied, the predominant one is the *disjunctive sum of games* following the *normal play* convention. In a disjunctive sum, players alternate turns choosing a single game component, making a move on it, and passing the turn over to their opponent, leaving the other components unchanged. Under normal play, when a player is unable to make a move (because there are no moves remaining for them in any game), then that player loses. In other words, the last player to make a move wins.

The modern forms of both combinatorial game theory and computational complexity theory were born around half a century ago, and there are several results of the latter about the former. Most relevant to this work, in 1981, Morris demonstrated that a sum of (individually polynomial-time solvable games) is PSPACE-complete [10]. The hard game sums in that reduction have several key requirements. First, they involve deeply asymmetric games (i.e., games where the moves available to the two players are very different). Second, the games have exponential length (the number of turns). Third, a polynomial number of games are included in the sums. Finally, most components of the game are *hot games*, meaning that players are incentivized to play first on most games in the sum.

Later results by Yedwab [16], Mowes [9], and eventually Wolfe [15] improved the reduction by eliminating the exponential length and reducing the branching factor size (to a smaller constant). However, the other two limitations remain.

A more recent result in 2021 [4] demonstrated that the sum of two tractable symmetric polynomial-length impartial games[1], when combined, are PSPACE-complete. This was accomplished using a pair of natural games known as UNDIRECTED GEOGRAPHY. Although UNDIRECTED GEOGRAPHY positions can be solved in polynomial time [6], it is PSPACE-complete to determine their values, as shown in [4].

Therefore, the hardness comes from finding that value rather than describing the difficulty of performing the mathematical operation. In fact, these impartial games have a simple polynomial algorithm to identify a winner in a sum if the game is already in its simplest form.

This paper continues the chain of results that Morris started, finding intractible summands with even more shallow game trees than were previously known. First, instead of hot games, we sum components from a family of *tepid* games. Tepid is a term based in temperature theory, where the temperature of a game is the number that approximates the incentive to move in a position. Hotter games have a higher value of potentially-earned moves in the favor of the first player to play on them. This can be by supplying moves to use later or denying your opponent later free plays.[2] Cold games use up these moves when a player plays on them; their temperature is negative. Tepid games all have a temperature of zero: playing on them doesn't earn either player any moves but can influence the parity of the current situation.

The second reason our work continues the chain of intractible sums is that, instead of the more deeply asymmetric games that Morris used, we use a family of *nearly symmetric* games, which become symmetric after a single move. Third, unlike the result with UNDIRECTED GEOGRAPHY, we use a family of games that are already in *canonical form*, which is to say, directly in the form of their values. And finally, this family of games is deeply related to one whose existence traces back to the birth of the modern theory.

The main family of games we consider are called *superstars*[3].These values naturally occur in the game PAINT CAN[13], which we discuss in Section 2. We show that for sums of superstars, it is computationally-intractable to determine which player has a winning strategy.

▶ **Theorem 1.** *A sum of superstars is* NP*-hard.*

The paper is structured as follows: In Section 2 we introduce the necessary concepts from combinatorial game theory. The proof of the main theorem is given in Section 3. The reduction used to prove our main theorem also leads to a nice new ruleset which we call BLACKOUT, introduced in Section 4.

[1] positions in which both players have the same options
[2] For a more thorough description of temperature theory in CGT, we recommend [12].
[3] There are some inconsistencies with this choice of name, which we discuss in full in Section 2.2.

2 Superstars: Theory and Paint Can

2.1 Rising Stars: From Stars to Superstars

In this section we will give a brief introduction to concepts from combinatorial game theory (CGT) required for this paper. For more information and a rigourous treatment of these topics, see [2], [1], and [12].

In the game Nim, played on piles of tokens, the two players take turns choosing a pile and removing any nonzero number of tokens. Under *normal play* the player to pick the last token(s) wins.

Nim is an *impartial game*, meaning one in which the two players have the same possible moves. An impartial game is denoted $G = \{G_1, \dots, G_n\}$, where $G_1, \dots, G_n$ are the options the players can move to.

When only a single pile remains in Nim, the current player will simply remove all tokens. But when several piles remain, the optimal move is often to only take some of the tokens. Thus all possible moves on a pile need to be considered when in a sum. To do so, we assign a *value* to each pile which represents the possible moves. An empty pile, thus one in which there are no moves, is given the value 0. The value of a pile with n tokens is the *nimber* $*n$. For a consistent recursive definition, we think of 0 as the nimber $*0$. The shorthand $*1 = *$ is also generally used.

▶ **Definition 2.** *The nimber $*n$ is recursively defined by its options as*
- $*0 = 0 = \emptyset$ *(no available moves);*
- $*1 = * = \{0\}$
- $*n = \{0, *, *2, \dots, *(n-1)\}$

The *(disjunctive) sum* $G_1 + \cdots + G_n$ of games $G_1, \dots, G_n$ is the game in which the players chose a summand (or component), then make a move in it. A Nim position is naturally the disjunctive sum of its separate piles. Many other games also naturally break into components, but we can consider sums of any games in general.

We say that two games are *equal* to each other when one can be replaced with the other in *any* disjunctive sum without changing the winnability. I.e., $A = B$ whenever who wins in $A + X$ is the same as in $B + X$ for any game X.

A sum of nimbers is always equal to a single nimber. Finding which nimber it is requires only an XOR sum (also known as *nim sum* in CGT) and therefore is in P (solveable in polynomial time).

In Nim, both players have the same available moves. For a game where the two players, which we call *Left* and *Right*, have differing moves, we use the notation

{Left's options | Right's options}.

Such games are called *partizan games*, and they have four *outcome (winnability) classes*. A game in
- $\mathcal{N}$ is won by the player that moves first, no matter whether they are Left or Right;
- $\mathcal{P}$ is won by the player that moves second, no matter whether they are Left or Right;
- $\mathcal{L}$ is won by Left no matter who goes first; and
- $\mathcal{R}$ is won by Right no matter who goes first.

▶ **Definition 3.** *A superstar is a game in which all options for Left and Right are nimbers, possibly not all the same.*

A superstar in which the options for both players are the same is a nimber. Even more in general we have the following:

▶ **Proposition 4** ([5]). *The superstar*

$$\{0, *, \ldots, *(n-1), *x_1, \ldots, *x_k \mid 0, *, \ldots, *(n-1), *y_1, \ldots, *y_l\},$$

where $x_i, y_j > n$ for all i and j, is equal to the nimber $*n$.

When a sum consists of only superstars of this form, the sum is reduced to a sum of nimbers, and is thus solvable in polynomial time. As we show in the main result of our paper in section 3, solving a sum of superstars in NP-hard.

2.2 Naming Superstars

There is some historical overloading of the term "superstar" in two foundational CGT texts, which share an author. In *Winning Ways*, first published in 1982, superstars are defined as we use them here[2]. In *On Numbers and Games*, first published in 1976, the same term is used to describe specific sums of (Winning Ways) superstars. In 2023, Silva et. al.[13] used another term, *quasi-nimbers*, because they were aware of the On Numbers and Games definition, but not the one from Winning Ways.

We do not make this choice lightly. The terminology collision was not known until parts of this paper was presented in the Virtual Combinatorial Games Seminar[4] in 2023. (No one at the seminar was aware of both definitions beforehand.) We solicited informal advice from the greater CGT community. Based on that, we chose to use the term "superstars", as in Winning Ways. Although this deviates from the first-published choice in On Numbers and Games, we are comfortable going forward with this because:

- We are still using historical terminology.
- As pointed out by Neil McKay, "superstar" is nice because these games are one move above stars (nimbers) in a game tree.
- Only one published paper uses the term superstar in either context since the two books have been published.[5]

This only solves the issue with superstars from Winning Ways. In order to handle the objects described as superstars in On Numbers and Games, we propose a new term, *comets*, and use that throughout. We like this term because comets are bright celestial objects like (super)stars, but have very little mass in comparison[6]. Additionally, the alliteration of "Conway Comets" works nicely.

We hope that our chosen terminology will continue to be used going forward.

2.3 The Game of Paint Can

PAINT CAN [7] is a pleasant combinatorial game ruleset that models superstars [13].

---

[4] `https://sites.google.com/view/virtual-cgt/seminar`

[5] Combinatorial Game Theory [12], a text published in 2013, references the Winning Ways version in an exercise.

[6] Although we do not provide the details of this property here, these comet positions have zero atomic weight.

[7] A playable version of PAINT CAN is available online at: `http://kyleburke.info/DB/combGames/paintCan.html`.

▶ **Definition 5** (Paint Can). *Paint Can is a combinatorial game played on stacks of bricks, each colored Red, Blue, Green, or Gray. Each turn a player chooses a brick in one stack either of their own color or Green. (No player can choose Gray bricks.) The chosen brick and all bricks above it are then removed from the stack. On top of each stack that has any non-Green bricks sits a can of green paint. When any brick is taken from that stack, the can of paint spills, coloring all the remaining bricks in the stack Green.*

Under Normal Play, the last person to move on a Paint Can position wins. Every superstar is equivalent to a Paint Can position with a single stack of bricks. Starting with an index of zero for the bottom brick: if brick i has color Blue, then $*i$ is only a Left option; if Red, $*i$ is only a Right option; if Green, $*i$ is both a Left and Right option; and if Gray, $*i$ is not an option for either player. For example, the game $\{ 0, *2, *4 \mid *, *2 \}$ is equal to the stack with bricks colored (from bottom to top) Blue, Red, Green, Gray, and Blue. See Figure 1 for an example of a position with that same stack. The entire position in the figure is equal to $\{ 0, *2, *4 \mid *, *2 \} + *4$.

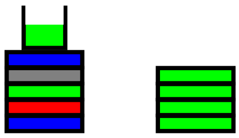

Figure 1 A Paint Can position consisting of two stacks of bricks with value $\{ 0, *2, *4 \mid *, *2 \} + *4$. In the leftmost stack, the Blue player may choose to remove either of the blue bricks or the green brick. The Red player may choose to remove either the red or green brick. Neither player may choose to remove the gray brick. In the rightmost stack, all bricks are already green, so no can of paint is necessary. If the Blue player removes the top brick from the left stack, the result will be a stack of four green bricks, as is in the right stack.

Any sum of superstars can be represented as an instance of Paint Can with each term equivalent to a single stack of bricks. For example, to create a position equivalent to $\{ 0, * \mid 0, *2 \} + \{ *2 \mid *3 \} + \{ 0, *, *2 \mid * \}$, we color bricks in each of three stacks corresponding to which player has the nimber option that matches the index of the brick (starting at the bottom with index 0). If both players have an option to $*i$, then brick i is green. If only Left has an option to $*i$, then brick i is blue. If only Right has an option to $*i$, then brick i is red. If neither player has an option to $*i$, then brick i is gray, though we do not include gray boxes for i higher than the nimbers in either option. See Figure 2 for a position equal to the prior sum of superstars. Thus, Paint Can is a ruleset where all superstars and sums of superstars occur; players need to evaluate them in order to determine which player can win.

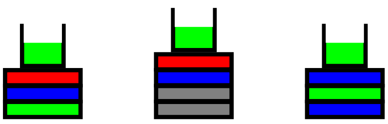

Figure 2 A Paint Can position equal to $\{ 0, *1 \mid 0, *2 \} + \{ *2 \mid *3 \} + \{ 0, *, *2 \mid * \}$.

3 From Bits to Superstars: Hardness Reduction

In order to show that sums of superstars (and Paint Can) are NP-hard, we need to introduce some additional computational problems.

XOR-SAT [11] is a classical logical satisfiability problem consisting of a conjuction of clauses of the XOR of boolean literals. That is to say, it takes this form: $(x_i \oplus x_j \oplus \cdots \oplus x_k) \wedge (x_l \oplus \cdots \oplus x_p) \wedge \cdots \wedge (x_q \oplus \cdots \oplus x_r)$. It is known that XOR-SAT is polynomial-time solvable [11].

Our next problem, which is NP-hard, is motivated by XOR-SAT. It uses *multi-state variables*, which can be assigned to one of many states instead of just True and False. Each literal of a variable is labelled with one of those states (e.g. x_{a,s_i}) and is only true if the variable is assigned to that state. More formally, let x_a be a *multi-state* variable with possible states $s_1, s_2, \ldots, s_i, \ldots, s_k$, then for all states s_i of x_a we have

$$x_{a,s_i} = \begin{cases} \text{True} & \text{if } x_a \text{ is set to } s_i, \\ \text{False} & \text{if } x_a \text{ is set to } s_j \text{ and } j \neq i. \end{cases}$$

Figure 3 displays a multi-state variable.

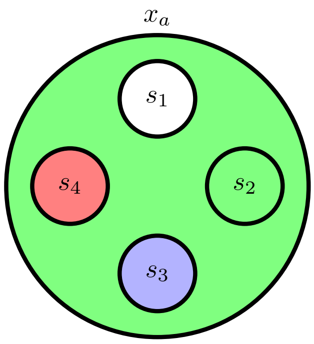

Figure 3 Multi-state variable x_a with four possible states: s_1, s_2, s_3, s_4. The overall color indicates that the chosen state is s_2.

▶ **Definition 6.** *Multistate XOR-SAT is a ruleset where a position is a conjunction of clauses consisting of the XOR of multi-state literals instead of boolean literals. In other words, the clauses are of the form $(x_{i,s_i} \oplus \cdots \oplus x_{j,s_j})$. Variables are divided between the two players, X, and Y, and clauses may contain variables from both players, e.g. $(x_{i,s_i} \oplus y_{j,s_j} \oplus \cdots)$. On their turn, the current player selects one of their unassigned variables and picks a state to assign it to. Once both players have assigned all variables, X wins if the formula is true, and Y wins if the formula is false. If both players have the same number of variables, then we call it Equal-Partitioned Multistate-XORSAT, or EPMX.*

We will show that EPMX is NP-hard after we show the reduction from EPMX to a sum of super stars (AKA Paint Can).

To reduce from EPMX, we must first discuss elementary strategies in a sum of superstars. Do aid in this, we partition superstars into six classes:

- nimbers,
- *no-0*: neither player has 0 as one of their options,
- *left-0*: only Left has a move to 0,
- *right-0*: only Right has a move to 0,
- *both-0*: both players have moves to 0, and
- *one-sided*: one player has no options while the other player does.

First, an observation that follows directly from Proposition 4:

▶ **Corollary 7** (No-0 games). *A No-0 game has value 0.*

Then, we will prove the following lemma:

▶ **Lemma 8** (0 game win). *Consider a sum of superstars with no both-0 games nor one-sided games. If at the start of the Left player's turn there are more left-0 games than right-0 games, then Left wins. Similarly, if at the start of the Right player's turn there are more right-0 games than left-0 games, then Right wins.*

It is possible to prove Lemma 8 using atomic weights and the two-ahead rule, as shown in [13]. We provide the following proof that avoids use of atomic weights.

Proof. We will call the winning player A and the losing player B, so there are more A-0 games than B-0. We will prescribe the following algorithm for A to win: they can "eliminate" 0s in the B-0 games by making a move on the game (by choosing one of their nimber options arbitrarily). They repeatedly do this until no B-0 games remain. Now, after B's following turn, the remaining games include at least one A-0 game, some nimbers (maybe none), and some no-0 games (maybe none). At this point, A should avoid playing A-0 games until there are only A-0 games remaining, or there is exactly one A-0 game left (along with the other types of games), whatever comes first.

If there are only A-0 games remaining, then for the first of those games, A can just bring a game to 0, and then if there are any games left, B has to make one into a nimber, which A can just bring to 0. This will repeat until the last A-0 game is taken this way, in which case A wins.

If there is exactly one A-0 game (along with potential no-0 games and nimbers), then A can identify the value of the sum of everything but the single game by XORing the nimbers (by observation 7, the others have value 0). If the nim-sum is 0, then A may take the move to 0 and thus wins the game. Otherwise, they can bring the nim-sum to 0 and inductively keep it so until B plays on A-0, bringing the game to a non-zero nim sum, which A can then win from. ◀

With this lemma, we can prove the following theorem:

▶ **Theorem 9.** *There exists a polynomial-time reduction from Equal-Partitioned Multistate-XORSAT (EPMX) to Sum of superstars, such that if True wins going first on EPMX, then the outcome class of Sum of superstars is $\mathcal{L}$ or $\mathcal{P}$ (i.e. Left wins going second).*

Proof. Let X be the player whose goal is to make the formula true in EPMX, Y be the player whose goal is to make it false, and m be the number of clauses. We assume that the EPMX formula contains literals for each state of each variable. (If it doesn't, we can create a dummy clause that will always be true for each missing variable missing a state. That clause contains one copy of each state that variable can have.)

We will use the following construction: First, we will assign the t^{th} clause an identity $z_t = 2^t$. In other words, we use power of twos $\{1, 2, 4, \ldots, 2^{m-1}\}$ to identify clauses. For the i^{th} variable assigned to X, we will create a Right-0 game which we will call x_i, and for the i^{th} variable assigned to Y, we will create a Left-0 game we will call y_i. Right's options for each x_i contain only a single option of 0, and similarly, Left's options for y_i contain only 0. Then, for each possible state of the i^{th} variable for X, there will be a Left option in x_i to a nimber whose value is, for each clause that contains the variable at that state, the sum of their corresponding identity values (i.e., if the t^{th} clause is involved, then z_t is included in the sum). Similarly, for each state of variable y_i for Y, there will be a Right option in y_i to a nimber with value equal to the sum of the corresponding identity values of the involved clauses. In addition, for each game x_j, there will be an additional option of $*2^{m+j}$. (The y_i positions do not have this extra option.)

The game we consider is then

$$G = x_0 + \ldots + x_k + y_0 + \ldots + y_\ell + *(2^m - 1).$$

For example, the position $(x_{0,a} \oplus x_{1,a} \oplus y_{0,a} \oplus y_{1,c}) \wedge (x_{0,b} \oplus x_{1,a} \oplus y_{0,b} \oplus y_{1,a}) \wedge (x_{1,a} \oplus x_{1,b}) \wedge (y_{1,a} \oplus y_{1,b} \oplus y_{1,c})$ with states $x_0 : \{a, b\}$, $x_1 : \{a, b\}$, $y_0 : \{a, b\}$, $y_1 : \{a, b, c\}$ reduces to $x_0 + x_1 + y_0 + y_1 + *15$, where

- $x_0 = \left\{ \underbrace{*}_{a}, \underbrace{*2}_{b}, \underbrace{*16}_{2^{m+0}} \middle| 0 \right\}$
- $x_1 = \left\{ \underbrace{*7}_{a}, \underbrace{*4}_{b}, \underbrace{*32}_{2^{m+1}} \middle| 0 \right\}$
- $y_0 = \left\{ 0 \middle| \underbrace{*}_{a}, \underbrace{*2}_{b} \right\}$
- $y_1 = \left\{ 0 \middle| \underbrace{*10}_{a}, \underbrace{*8}_{b}, \underbrace{*9}_{c} \right\}$

In the example above, the identity of these four clauses are respectively, 1, 2, 4, and 8.

Now we demonstrate the correctness of the reduction. As mentioned in the theorem statement, the union of $\mathcal{L}$ and $\mathcal{P}$ is equivalent to proving that Left wins going second. We will show that the game should progress by alternating moves of Right playing on left-0s and Left playing on right-0s. Since the options of those components are all nimbers, each of these plays changes the whole game by removing that component and modifying the nimber term. If this pattern is followed, then after Left plays on the final right-0, they win if and only if the nimber term has been reduced to zero. Otherwise, Right can bring the nin-sum to 0, and then win through following the nim strategy.

If both continue to hold to that pattern, at the beginning of each of Left's turns, there is one more right-0 component than left-0. Thus, Left must play on a right-0 component or they will lose by Lemma 8. Right starts each turn with balanced left-0s and right-0s, but they still need to follow the pattern. If Right deviates by playing on an x game, then they will lose by Lemma 8. If Right plays on the nimber term instead, this switches the roles of the players with respect to their ending conditions; now Right will win if and only if the nimber term is reduced to zero when they make their final move. Left, however, can avoid this by playing on one of the large nimber (with value at least 2^m) options included in any x. Right doesn't have any options that can cancel out that large nimber, so the final nimber term will always be non-zero and Right can no longer win.

Following the prescribed sequence of play, Left wins if the nimber term equals zero. Since it started at $*2^m - 1$, the sum of all nimber options chosen must also equal $*2^m - 1$. If Left has a winning strategy in EPMX, they can play on the nim values corresponding to each variable state in their winning strategy, which must then result in a final nimsum of 0. If they do not have a winning strategy, then note that playing on the 2^{m+j} values can't give them a chance to win, since Right can stick with their EPMX strategy and the nimsum can't equal 0. Thus, Right can follow Y's winning strategy of assignments to result in a non-zero value. ◀

Now we will show NP-hardness for EPMX.

▶ **Theorem 10.** *There exists a polynomial-time reduction from 3SAT to EPMX.*

Proof. Let X be the player whose goal is to make the formula true in EPMX, Y be the player whose goal is to make it false. (In our reduction, Y will not make meaningful decisions in the course of the game.) Let n be the number of variables and m will be the number of clauses from the 3SAT instance.

Note that 3SAT is hard even if every variable appears only at most 3 times, at least once negated and at least once unnegated. It is also hard adding on a further restriction that there are an odd number of variables in the formula. We will assume both of these are true in this reduction.

For each clause in 3SAT, we will create a clause in EPMX. We will also have two separate clauses c_x and c_y. For each variable (x_i) in 3SAT, we will create an variable (x_i) in EPMX with five states: $\{x_{i_a}, x_{i_b}, x_{i_c}, x_{i_d}, x_{i_e}\}$. WLOG, we assume that x_i appears once unnegated and twice negated, in clauses r, s, and t in 3SAT respectively. Every state appears in c_x. In the EPMX formula, the first state (x_{i_a}) appears in no other clauses. The second state, x_{i_b}, corresponds to the unnegated appearance, and appears in clause r. The third state, x_{i_c}, corresponds to both negated 3SAT literals, and appears in clauses s and t. The fourth state, x_{i_d}, corresponds to only the first negated 3SAT literal, and appears only in clause s. Finally, the fifth state, x_{i_e}, corresponds to only the second negated literal, and only appears in clause t.

In order to be an EPMX position, we need to include the same number of y variables as x. We can do this by creating n dummy variables with two states each: a and b. We can include all of the states of each of the variables into c_y. Note that since there are an odd number of y variables, c_y will always be true (the same is true of c_x).

If there is a solution to the 3SAT formula, then a solution to EPMX can be constructed by iterating over the variables. For each x_i, if the 3SAT assignment is true, then we choose x_{i_b}, unless the EPMX clause r has already been satisfied, in which case we choose x_{i_a}. If x_i is assigned to false, then we select the correct choice of a, c, d, and e, depending on which of clauses r and t have already been satisfied.

The inverse direction is simpler: an assignment of x_{i_a} means it doesn't matter what we pick. x_{i_b} means x_i must be true to satisfy the 3SAT formula, and any of the others means it must be assigned to false. ◀

As explained in Section 2.2, we introduce the term comet to refer to the objects called superstars in *On Numbers And Games*[5]. Each superstar has an associated comet. We do not provide a full explanation of all cases of comets here. However, if the superstar, S, is a left-0, then the comet will be $\downarrow + * + S$, where $\downarrow = \{ * \mid 0 \}$. If S is a right-0, then it will be $\uparrow + * + S$ where $\uparrow = \{ 0 \mid * \}$. Finally if S is a nimber, then S is its own comet [5, 13]. Note that $\downarrow + \uparrow = 0$.

▶ **Corollary 11.** *A sum of comets is* NP*-hard.*

Proof. We can express our sums of superstars and a nimber resulting from the reduction from EPMX as a sum of comets. If we replace each x_i with the comet $\uparrow + * + x_i$ and each y_i with the comet $\downarrow + * + y_i$, then sum all of those comets, all the $\uparrow$ and $\downarrow$ components will cancel out and all of the individual $*$ will cancel each other out. Thus the game is the same sum of superstars. ◀

A minor note is that comets have something known as atomic weight 0, which also gives the result that sums of atomic weight 0 games can be NP-hard.

4 From Logic to Board Game: Blackout

In this section, we use a simplified version of the logical game that appeared in our earlier analysis to design a two-player board game, which we call Blackout. The board of this game contains an array of light bulbs and two sets of switches, one above the light bulbs and one below. Two players, denoted by AllOff (Left) and OneOn (Right), each control one set of switches.

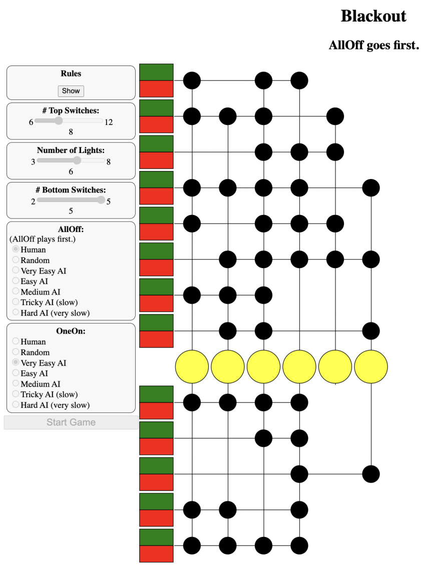

Figure 4 An Example Game of Blackout.

The AllOff player wins if all lights are off at the end of the game. They control the switches at the top of the board. The OneOn player wins if at least one light is on in the end. They use the switches at the bottom of the board. On their turn, a player turns one of their unchosen switches off (red) or on (green). If they turn the switch on, then all of the lights that switch is connected to get toggled (off becomes on and on becomes off). The OneOn player has an easier objective, so they may have fewer switches. Once all bottom switches are played, they can pass as long as all the lights are not out. During these final turns, the AllOff player is searching for configuration of their remaining switches to turn all lights off in order to win.

Blackout is in the Maker-Breaker style, but it differs from traditional Maker-Breaker games in one crucial aspect: Even if AllOff turns all lights off at some point during the game, if OneOn still has some unflicked switches, then the game will continue until all switches for both players have been selected. I.e., Blackout could potentially continue even after all light bulbs have been turned off, while Maker-Breaker games end as soon as the desired structure has been formed (or continue until there are no moves).

Figure 4 is an illustration from our two-dimensional board layout in our Web implementation, which can be found at the following link: `http://kyleburke.info/DB/combGames/blackout.html`.

4.1 Blackout Ruleset Formalities

In this subsection, we discuss the mathematical representation of positions and rules in Blackout to set up our computational analysis. For Blackout games with p lights, a position has three components $P = (L, S^{\text{AllOff}}, S^{\text{OneOn}})$:

(a) L is a Boolean vector in $\{0,1\}^p$, indicating whether each of the lights is off or on. That is, $L(i) = 1$ denotes that the i^{th} light is on.

(b) S^{AllOff} is a Boolean matrix with p columns in which every row has at least one 1. Each row represents a switch that AllOff can use. If the switch controls the i^{th} bulb, then its i^{th} entry equals to 1.

(c) S^{OneOn} is a Boolean matrix with p columns, in which every row has at least one 1. Each row represents a switch that player OneOn can use. If the switch controls the i^{th} bulb, then its i^{th} entry equals to 1.

In this position, we define the options as follows:

- AllOff has two options for each row in S^{AllOff}. (If the matrix is empty, then they have no options remaining.) For their turn, AllOff selects a row r and a binary action $\alpha \in \{0,1\}$. If the action is $\alpha = 0$, then the position is moved to $(L, S^{\text{AllOff}}_{-r}, S^{\text{OneOn}})$, where S^{AllOff}_{-r} denotes the Boolean matrix obtained from S^{AllOff} by removing its r^{th} row. If the action is $\alpha = 1$, then let L' be the entry-wise exclusive-or of L and the r^{th} row of S^{AllOff}, and the position is moved to $(L', S^{\text{AllOff}}_{-r}, S^{\text{OneOn}})$. Note L' represents the result when player AllOff activates its r^{th} switch.
- OneOn's options depend on two cases: (1) If S^{OneOn} is not an empty matrix, then OneOn can select a row s and a binary action $\beta \in \{0,1\}$. If the action is $\beta = 0$, then the position is moved to $(L, S^{\text{AllOff}}, S^{\text{OneOn}}_{-s})$, where S^{OneOn}_{-s} denotes the Boolean matrix obtained from S^{OneOn} by removing its s^{th} row. If the action is $\beta = 1$, then let L' be the entry-wise exclusive-or of L and the s^{th} row of S^{OneOn}, and position is moved to $(L', S^{\text{AllOff}}, S^{\text{OneOn}}_{-s})$. Note L' represents the result when player OneOn flicks on its s^{th} switch. (2) If S^{OneOn} is an empty matrix (i.e., player OneOn has no more switches to flick), then OneOn can make a pass move so long as there is at least one light on in L. If the lights are all out, then they have no options available.[8])

4.2 The Intractability of Blackout

We now analyze the complexity of Blackout and prove the following intractability result.

▶ **Theorem 12** (Intractability of Blackout). *Deciding whether or not player ALLOFF has a winning strategy at a given Blackout position is NP-hard.*

Proof. We begin by defining two decision problems, Set Cover and Exact Cover.

In Set Cover, there is a collection V containing n sets $S_1, S_2, \ldots, S_n$ which each contain some subset of a ground set $E = \{e_1, \ldots, e_m\}$ of m elements. There is also a given integer k, indicating a target number of sets to choose.

A Set Cover instance is feasible if there exists a selection of k sets in V such that every element in E is in at least one of the selected k sets. We call such selection a *cover*. A cover is *exact* if for each element $e \in E$, e appears exactly once in the selected sets. Exact Cover determines whether the input has an exact cover.

[8] In order to prevent OneOn from making unbounded passes in the context of a game sum, they should have a maximum number of passes at the beginning equal to the difference in heights of the matrices, $\text{height}(S^{\text{AllOff}}) - \text{height}(S^{\text{OneOn}})$.

In our desired problem of PURE SET COVER, we want to have a promise that if there is a SET COVER of size k, then there is also a EXACT COVER of size k.[9].

Next, we note that SET COVER is NP-complete even if there are only three elements in each set in V [7].

Now we can reduce from SET COVER with three elements in each set. For our reduction, we will enrich the input by adding new sets for each subset of those sets. E.g., if $S_1 = \{e_1, e_2, e_3\}$, then we include the six sets $\{e_1\}$, $\{e_2\}$, $\{e_3\}$, $\{e_1, e_2\}$, $\{e_1, e_3\}$, and $\{e_2, e_3\}$ in our new collection V as well. This enforces the promise, because if there is a set cover of size k, for each overlap, for one of the overlapping sets, one can instead choose to select a subset that doesn't overlap. We can repeat this for all overlapping sets without changing k.

In other words, with this enrichment, we have proved that PURE SET COVER is also NP-complete.

To reduce from PURE SET COVER to BLACKOUT, we create a light switch for the ALLOff player for each set in the PURE SET COVER and $n - k$ light switches for the OneOn-player. We create a light for each element in E.

The switches are set as the following:

- **(AllOff)**: For the n light switches controlled by the AllOff player, we connect them to the lights corresponding to the elements contained in the sets, one for each set.
- **(OneOn)**: We connect $n - k - 1$ of the light switches controlled by the OneOn-player only to the lights corresponding to the elements in S_1, which the AllOff player also has a switch for. Finally, the OneOn-player's last light switch is connected to all of the lights.

All lights begin in the on state and OneOn goes first.

We claim that the AllOff-player has a winning strategy if and only if there exists a Set Cover in the PURE SET COVER.

The OneOn player has $n - k - 1$ switches for S_1, $n - k - 2$ of which are redundant, so they should start by playing all of these, without the setting changing the outcome of the game.

If there is a working k-sized cover, AllOff will spend their first $n - k - 2$ turns choosing to turn off switches that are not in their pure cover and not S_1. OneOn, seeing that AllOff has a switch to negate theirs for S_1, should play that again. If S_1 is part of the cover, then AllOff can choose the setting that shuts them off. Otherwise, they should choose to turn them on. OneOn will then choose to leave all lights on (otherwise AllOff can win immediately). Now AllOff has k turns to flip all switches in their cover of size k to turn all lights off. If OneOn decides to activate the switch connected to all lights earlier, they should choose to leave them all on, in which case AllOff just has extra turns to put their cover to work. No matter what, AllOff will win.

Then, if there exists no Exact Cover of size k, then there exists no set cover of size k. OneOn can win by saving their all-lights-switch for their last move. Since there is no set cover of size k, there must be at least one light not covered by the AllOff player's remaining switches. The OneOn player can either flip or not flip the final switch to make sure that light is on and win the game. ◀

[9] We thank Neal Young for this idea.

5 Conclusion

Going beyond winnability, Sprague-Grundy Theory [14, 8] introduced the first notion of *game values* and *game algebra* for combinatorial games, which was later expanded to partizan games [2, 5]. It demonstrated that every impartial game can be mathematically reduced to a single-pile of Nim in the algebra of disjunctive sums. Because the nim sum can be computed in linear-time in the size of the binary representations of the summands [3], Sprague-Grundy Theory further captures the computational benefit of knowing game values of impartial games, rather than just their game rules or winnability [4].

By showing that the winnability of the sum of superstars and comets are intractable, our result highlights the fundamental subtlety of tepid partizan game values. It takes a significant step beyond Morris' [10] classical intractability result by demonstrating that intractability happens just one step above nimbers (stars). This hardness result implies that the Bouton-like result for Nim [3] is unlikely for superstars and comets.

Part of our proof has also inspired the design of a board game, Blackout, which enjoys intriguing complexity based on the shape of game boards: On the one hand, if both players have the same number of switches, which we refer to as the balanced case, then the game can be solved in polynomial time [10]. On the other hand, if players have a different number of switches, then the game is intractable in general.

Whereas the sum of superstars has been shown to be NP-hard to solve, its precise complexity remains open. It can be shown that the outcome class of a sum of superstars can be computed in polynomial space in the number of bits representing the sum.[11] Therefore, as part of the next step in our future research, we would like to settle the following conjecture.

▶ **Conjecture 13.** *A sum of superstars, and hence Paint Can, is* PSPACE*-complete.*

It is worth noting that it seems challenging to naturally extend the current setup of our intractability proof for this conjecture. Reducing directly from QSAT to EPMX seems fruitless since there is no clear way to "punish" player Y from covering a variable multiple times. For any reduction, the fundamental difficulty lies in the strong asymmetry between players X and Y in that Y is just too powerful with options. In other words, as soon as EPMX allows Y to be a "real" decision maker – as opposed to the construction in Theorem 10 – the game shifts dramatically in Y's favor, and indeed, it is difficult to even find complicated positions where X wins with a non-trival Y player. As it stands, either a different approach is needed, or a very clever reduction to EPMX is required.

References

1 M. H. Albert, R. J. Nowakowski, and D. Wolfe. *Lessons in Play: An Introduction to Combinatorial Game Theory.* A. K. Peters, Wellesley, Massachusetts, 2007.

2 Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for your Mathematical Plays*, volume 1. A K Peters, Wellesley, Massachsetts, 2001.

3 Charles L. Bouton. Nim, a game with a complete mathematical theory. *Annals of Mathematics*, 3(1/4):pp. 35–39, 1901. URL: `http://www.jstor.org/stable/1967631`.

4 Kyle Burke, Matthew Ferland, and Shang-Hua Teng. Winning the war by (strategically) losing battles: Settling the complexity of Grundy-values in undirected geography. In *Proceedings of the 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2021.

[10] This statement follows from the basic idea in [11].
[11] One of the proofs was given by Aaron Siegel.

5 J.H. Conway. *On Numbers and Games*. A.K. Peters. A.K. Peters, 2000. URL: https://books.google.com/books?id=tXiVo8qA5PQC.

6 Aviezri S. Fraenkel, Edward R. Scheinerman, and Daniel Ullman. Undirected edge geography. *Theor. Comput. Sci.*, 112(2):371–381, 1993.

7 M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified np-complete problems. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC '74, pages 47–63, New York, NY, USA, 1974. Association for Computing Machinery. doi:10.1145/800119.803884.

8 P. M. Grundy. Mathematics and games. *Eureka*, 2:198—-211, 1939.

9 David Moews. *On some combinatorial games connected with Go*. PhD thesis, Citeseer, 1993.

10 FL Morris. Playing disjunctive sums is polynomial space complete. *International Journal of Game Theory*, 10(3-4):195–205, 1981.

11 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 216–226, New York, NY, USA, 1978. Association for Computing Machinery. doi:10.1145/800133.804350.

12 A.N. Siegel. *Combinatorial Game Theory*. Graduate Studies in Mathematics. American Mathematical Society, 2013. URL: https://books.google.com/books?id=VUVrAAAAQBAJ.

13 Alexandre Silva, Carlos Pereira dos Santos, João Pedro Neto, and Richard J. Nowakowski. Disjunctive sums of quasi-nimbers. *Theoretical Computer Science*, 945:113665, 2023. doi:10.1016/j.tcs.2022.12.015.

14 R. P. Sprague. Über mathematische Kampfspiele. *Tôhoku Mathematical Journal*, 41:438—-444, 1935-36.

15 David Wolfe. Go endgames are PSPACE-hard. In Richard J. Nowakowski, editor, *More Games of No Chance*, volume 42 of *Mathematical Sciences Research Institute Publications*, pages 125–136. Cambridge University Press, 2002.

16 Laura Jo Yedwab. *On playing well in a sum of games*. PhD thesis, Massachusetts Institute of Technology, 1985.