Complexity of Benndorf's "The Game"

Jason S. Ku* Mikhail Rudoy*

"The Game" is the title of a neat cooperative card game designed by Steffen Benndorf and published by Nürnberger-Spielkarten-Verlag [1]. In this paper, we describe The Game, provide a number of generalizations, and discuss the computational complexity of winning them, provided a given input card distribution.

The Game requires players to cooperatively discard all cards in a deck of 100 uniquely numbered cards into a set of piles that follows certain rules. In the original, cards placed on a given pile must be either monotonically increasing or decreasing, with the exception that any card exactly 10 away from the top card in the pile may also be played. These rules define a valid placement graph for each pile which we call a *transition graph.* The players win if they can place all the cards in turn onto piles in a way that is consistent with each pile's transition graph.

We consider a one-player version of The Game that is *offline*, i.e., the order of the cards in the deck is known. We will generalize this game for variable deck size n, variable hand size h, variable play number p, with a variable number of discard piles k, each having a transition graph that may be any general directed graph. We show that if either the hand size or the number of piles is non-constant, then deciding whether one can win The Game is NP-Complete.

We begin with some definitions. A *Card* is a number in $[1..n]$; cards are unique. A *Deck* is a permutation on the cards from which we draw. The *Hand* is a temporary storage in which we place cards drawn from the Deck. *Hand Size* is the max number of cards in the Hand. The *Play Number* is the min number of cards that can leave the Hand per turn. A *Pile* is a container into which we can place cards from the Hand. A *Pile Transition Graph* is a graph on vertices $[0..n]$, with 0 referencing no card, and vertex i corresponding to card i. Edge (i, j) says that card j may be placed on top of card i.

Every turn, we remove at least or exactly the Play Number of cards from the Hand and place these cards one at a time onto piles on which they can legally be placed. Then we replenish the Hand to the Hand Size from the Deck. The Game ends if no more actions can be made. The player wins if every card is in a pile and the decision problem for The Game is then: can the player win for a given input? This decision problem is in NP, certified by a play order.

Theorem 1 *The Game can be decided in polynomial time if the hand size and the number of piles are constant.*

Proof. For a constant number of Piles and a constant Hand Size, the game state graph is polynomial in size $O(n^{h+k+1})$, so we can easily determine whether a path exists from the starting state to any win state. Here state encodes the hand contents, the top card of each pile, and the progress through the deck. □

When either hand size or number of piles is not constant, and Pile Transition Graphs are allowed to encode general graphs provided as input, the problem becomes hard.

Theorem 2 *The Game is NP-Complete when hand size is a constant fraction of deck size, allowing general transition graphs for one discard pile.*

Proof. For one pile with a general directed transition graph and hand size that is a constant fraction r of deck size (or part of the input), we can reduce from Hamiltonian Path. We construct a transition graph as follows. Let the Hand Size be the size n of the input graph, and fix the Deck size to n/r. Let the transition graph on vertices $[1..n/r]$ be our Hamiltonian Path instance with vertex 1 the start vertex and vertex n the end vertex. Add edge $(0, 1)$ and edge $(i, i + 1)$ for $n \leq i < n/r$. Set the Deck to be $[1..n]$ in that order. Analysis is straightforward and not detailed here. □

Theorem 3 *The Game is NP-Complete for constant hand size and play number, allowing general transition graphs for many piles.*

This can be proved via a chain of reductions starting from the following NP-Complete problem (from [2]):

[Swap or Not Reachability] Given permutations σ_1, σ_2, ..., σ_k, and T on n elements such that σ_i is a swap of some two elements and T is an arbitrary permutation, decide whether there exists a bitstring $b_1 b_2 \dots b_k$ such that $\sigma_k^{b_k} \circ \sigma_{k-1}^{b_{k-1}} \circ \dots \circ \sigma_2^{b_2} \circ \sigma_1^{b_1} = T$.

We reduce this problem to the following more general problem:

[DAG Path Cover] Given a DAG and a list of pairs of vertices (s_i, t_i), decide whether there exists disjoint paths from each s_i to its corresponding t_i such that each vertex of the DAG exists in one of the paths.

*MIT, {jasonku,mrudoy}@mit.edu

Lemma 4 DAG Path Cover *is NP-Complete.*

Proof. Given a Swap or Not Reachability instance S, we construct an instance of DAG Path Cover D with $m = n(k+1)$ vertices and n paths. Let (i, j) be a vertex of the DAG for $i \in \{1, \ldots, n\}$ and $j \in \{0, \ldots, k\}$. We say that vertex (i, j) is in column i on level j. For $i \in \{1, \ldots, n\}$, let s_i be vertex $(i, 0)$ and t_i be vertex $(T(i), k)$. For $i \in \{1, \ldots, n\}$ and $j \in \{0, \ldots, k-1\}$, a vertex (i, j) has an outgoing edge to $(i', j+1)$ if $i' = \sigma_{j+1}(i)$ or $i' = i$.

Given a solution to S consisting of bit-string $b_1 b_2 \ldots b_k$ where $\sigma_k^{b_k} \circ \sigma_{k-1}^{b_{k-1}} \circ \ldots \circ \sigma_2^{b_2} \circ \sigma_1^{b_1} = T$, we construct a solution to D. For $p \in \{1, \ldots, n\}$, consider the sequence of vertices $\mathcal{S}_p$ of the form (i_j, j) for $j \in \{1, \ldots, n\}$ where transformation $T_j = \sigma_j^{b_j} \circ \sigma_{j-1}^{b_{j-1}} \circ \ldots \circ \sigma_1^{b_1}$ maps p to i_j. The first vertex in $\mathcal{S}_p$ is $(p, 0) = s_p$ since T_0 is the identity transformation. Similarly, the last vertex of $\mathcal{S}_p$ is t_p since $T_k = T$. Since $T_{j+1} = \sigma_{j+1}^{b_{j+1}} \circ T_j$, there exists an edge from $(T_j(p), j)$ to $(T_{j+1}(p), j+1)$ by construction. Therefore $\mathcal{S}_p$ is a path from s_p to t_p. Two paths $\mathcal{S}_p$ and $\mathcal{S}_{p'}$ intersect only if $(T_j(p), j) = (T_j(p'), j)$. Since T_j is a permutation on the elements, $T_j(p) \neq T_j(p')$ for $p \neq p'$ so the paths are disjoint. Furthermore, since each path starts on level 0 and ends on level $k+1$, each path contains one vertex in every level. Because there are exactly n vertices per level, the disjoint paths $\mathcal{S}_p$ cover every vertex and comprise a solution to D.

Given a solution to D, a set of paths $\mathcal{S}_p$, we construct a solution to S. By the same argument above, each $\mathcal{S}_p$ contains one vertex from each level. Consider level j. Let b_j be zero if every path $\mathcal{S}_p$ has vertices in the same column at levels j and $j+1$; otherwise, $b_j = 1$. We claim that $b_1 b_2 \ldots b_k$ is a solution to S. Define T_j with respect to these b_j as above. We prove by induction that path $\mathcal{S}_p$ contains vertex $(T_j(p), j)$. The first vertex of $\mathcal{S}_p$ is $(p, 0)$ as desired since T_0 is the identity transformation. Then assume $(T_{j-1}(p), j-1)$ is on $\mathcal{S}_p$. Note that $T_j = \sigma_j^{b_j} \circ T_{j-1}$. There are three cases.

1. Case when $b_j = 0$. Then $\mathcal{S}_p$ passes through $(T_{j-1}(p), j)$ and $T_j = T_{j-1}$, so $\mathcal{S}_p$ contains vertex $(T_j(p), j)$ as desired.

2. Case when $b_j = 1$ and σ_j does not swap element $i = T_{j-1}(p)$. The possible outgoing edges from $(i, j-1)$ are to (i, j) and $(\sigma_j(i), j)$, but since $\sigma_j(i) = i$ these are the same edge. Since $(i, j-1)$ is on $\mathcal{S}_p$ with unique outgoing edge to $(\sigma_j(i), j)$, then $(\sigma_j(i), j)$ must also be on the path. But $\sigma_j(i) = T_j(p)$, so $\mathcal{S}_p$ contains vertex $(T_j(p), j)$ as desired.

3. Case when $b_j = 1$ and σ_j swaps element $i = T_{j-1}(p)$. The argument from the previous case shows that at most two vertices on level $j-1$ have two outgoing edges, so at most two edges change column from level $j-1$ to j. Since $b_j = 1$, some path must change columns between levels $j-1$ and j. In fact at least two must change columns or else two paths would be in the same column at level j and would not be disjoint. Because there are at most two edges that change column from level $j-1$ to j, every such edge must be traversed by some path.

   The edge from $(i, j-1)$ to $(\sigma_j(i), j)$ is in the graph by construction. Furthermore $\sigma_j(i) \neq i$ because σ_j swaps element i. Thus this edge must be contained in some path. This path must be $\mathcal{S}_p$ since it contains vertex $(i, j-1)$. Therefore $\mathcal{S}_p$ contains vertex $(\sigma_j(i), j)$. But $\sigma_j(i) = T_j(p)$, so $\mathcal{S}_p$ contains vertex $(T_j(p), j)$ as desired.

By induction, path $\mathcal{S}_p$ contains vertex $(T_j(p), j)$ for all $j \in \{0, \ldots, k\}$ and $p \in \{1, \ldots, n\}$. Then for $j = k$, $\mathcal{S}_p$ contains vertex $t_p = (T(p), k)$, so since only one vertex in each level is in each path, $(T_k(p), k) = (T(p), k)$. So $T_k(p) = T(p)$ for all p. Thus $T = T_k = \sigma_k^{b_k} \circ \sigma_{k-1}^{b_{k-1}} \circ \ldots \circ \sigma_2^{b_2} \circ \sigma_1^{b_1}$ as desired.

Specifying paths that satisfy the requirements can be described in polynomial time, so the problem is in NP. □

Proof of this Lemma is omitted for brevity. We then use this lemma to prove the preceding theorem:

Proof. Given a DAG Path Cover instance D, with DAG G on m vertices requiring k paths, we construct an instance I of The Game with hand size h, play number p, k piles, and $n = (h-p) + m + k$ cards.

The deck is given in order from 1 to n. We call cards $\{1, \ldots, h-p\}$ *trash*, cards $\{h-p+1, \ldots, m-k\}$ *cool*, and cards $\{m-k+1, \ldots, m\}$ *sentinel*. Our strategy will be to construct a transition graph for each pile, each containing G as a subgraph such that the play order on pile i contains a path in G from s_i to t_i.

We construct a correspondence between cool cards and vertices of G, such that if (u, v) is an edge, then corresponding cards c_u and c_v satisfy $c_u < c_v$. This is possible using any topological sort on G. In the transition graph for every pile, we add an edge from card c_u to c_v if (u, v) is an edge of G. For pile i, we add an edge from the empty state 0 to the card corresponding to vertex s_i, and an edge from the card corresponding to vertex t_i to sentinal card $m-k+i$. Lastly, we form a directed chain through the trash cards starting at sentinal card m in the transition graph of pile k. By this construction, the trash cards can only be played after the last card in the deck.

Given a solution to D consisting of paths $\mathcal{S}_p$, we construct a solution to I. We will play all non-trash cards in order and then all the trash cards. Assuming there are piles that can accept this order of cards, this play order is possible given the deck and play number. We now show that there always exists a pile on to which we can place these cards. We place cards on to piles as follows:

C1 For each cool card c corresponding to start vertex s_i, we will place that card onto pile i. Provided the pile is empty, we will be able to place this card.

C2 For each cool card c corresponding to non-start vertex v, we will place c on to the card corresponding to the previous vertex u along the path containing v. Provided the card corresponding to u is on the top of some pile, we will be able to play c.

C3 For every sentinel card $m - k + i$, we will place it onto the card corresponding to t_i. Provided the card corresponding to t_i is on the top of pile i.

C4 For every trash card c, we will place it on pile k on top of the parent of c in the transition graph of pile k.

Now we argue that the above placements are possible.

C1 No card in [C2], [C3], [C4] is placed on an empty pile and no two cards in [C1] are placed onto the same pile, so we can place cards in [C1].

C2 Because the cards corresponding to path $\mathcal{S}_j$ are played in path order based on the topological sort, no card in [C2] will be played before the card it should be played on. Further, no two cards are placed onto the same card, so we can place cards in [C2]. Note that all cards corresponding to vertices in path $\mathcal{S}_p$ are placed in pile p by induction.

C3 Because cool cards are placed before sentinel cards, no card in [C3] will be played before the card it is played on. Since the card corresponding to t_i will be placed in pile i, we will place the sentinel card $m - k + i$ in the correct pile, so we can place cards in [C3].

C4 Since card m is in [C3] and can be played, we can play the trash cards in the order in which they appear in the transition graph of pile k.

Thus we can play all cards, solving I. Given a solution to I, we construct a solution to D. Consider pile i. Card $m-k+i$ can only be played on this pile, and only immediately after the card corresponding to t_i. The card corresponding to s_i is the only card that can be played onto pile i when it is empty. The sequence of cards played on pile i starting with the card corresponding to s_i and ending with the card corresponding to t_i will be a path in the transition graph of pile i. The only transition in the transition graph of pile i between cool cards and non-cool cards is from t_i, so this path is in G. No card can be placed in two piles, so the paths are disjoint, and every card must be placed, so the paths cover the vertices of G. These paths comprise a solution to D.

Specifying play order and location can be described in polynomial time, so the problem is in NP. □

In our generalized models, we allow for transition graphs to be general graphs, but the original game uses only very particular transition graphs. If transition graphs are not part of the input and only depend on the size of the deck, is The Game still NP-Hard? We leave this question to future work.

References

[1] S. Benndorf. The Game. Nürnberger-Spielkarten-Verlag: `https://boardgamegeek.com/boardgame/173090/game`, 2015.

[2] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.