ON THE COMPLEXITY OF THE GAME OF SET

KAMALIKA CHAUDHURI, BRIGHTEN GODFREY, DAVID RATAJCZAK, AND HOETECK WEE

{kamalika,pbg,dratajcz,hoeteck}@cs.berkeley.edu

ABSTRACT. Set® is a card game played with a deck in which each card has four properties. Each property takes on one of three values. Players compete to quickly find sets of three cards such that for each property, the cards have all the same or all different values.

We show that a natural generalization of the game, where the number of properties and values vary, is **NP**-complete. We also study the case where the number of values is fixed at 3, as in the classic game, and the number of properties p varies. In a restricted model involving checking pairs of cards, we give an exactly (not just asymptotically) optimum $\Theta(n^2p)$-time algorithm.

1. INTRODUCTION

1.1. **The Game of Set.** The card game Set® [6] was invented in 1974 by Marsha Jean Falco, and was set loose upon the world in 1991. The game is played with a deck of $3^4 = 81$ cards. Each card is unique and is printed with figures that have four *properties*: color, shape, shading, and number of figures. Each of these properties takes on one of three *values*:

- *color* is red, green, or purple;
- *shape* is oval, diamond, or squiggle;
- *shading* is solid, shaded, or open; and
- *number* of figures is one, two, or three.

Define a *valid set* as three cards such that for each property, the three cards either have the same value or no two of them have a common value.[1] Examples of a valid set and an invalid set are shown in Figure 1.1.

[1]What we refer to as a valid set is simply called a "set" in traditional game play. We adopt the moniker "valid set" to avoid overloading the mathematical meaning of "set".

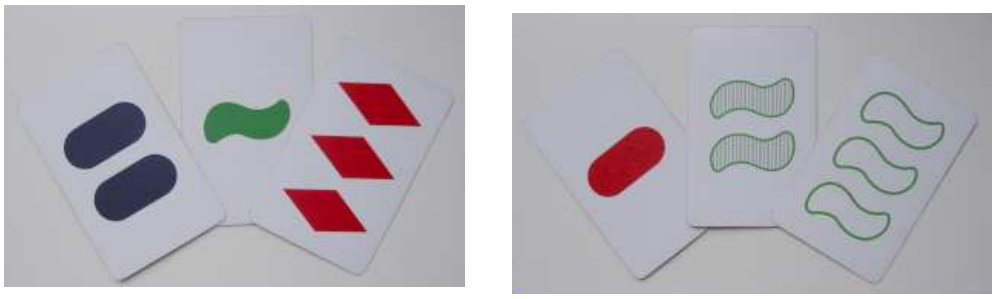

FIGURE 1.1. The three cards on the left (which are purple, green, and red, respectively) form a valid set; the three on the right (which are red, green and green) do not.

The game is played as follows.[2] Twelve cards are dealt face-up. Any number of players search for valid sets among the dealt cards, and asynchronously claim them as soon as they find them. The three cards that form the valid set are removed and three new cards are dealt in their place. Play continues in this manner until all cards have been dealt and all valid sets among the dealt cards have been claimed. The winner is the player who has claimed the most valid sets.

1.2. **The model.** Because the goal is to claim valid sets faster than other players, studying the computational complexity of finding valid sets is natural. To put asymptotic analysis to use, we generalize the game to variable-size decks in a natural way: we let the number of properties p, the number of values v, and the number of dealt cards vary.

Our deck will be $D = \{1, \ldots, v\}^p$. When $p = 4$ and $v = 3$, this is just the classic Set deck. For a card $c \in D$ and $i \in \{1, \ldots, p\}$ we denote by $c[i]$ the value of the ith property (dimension) of c; that is, $c = (c[1], \ldots, c[p])$.

Definition 1.1. *Given v distinct cards $c_1, \ldots, c_v \in D$, we say that $S = \{c_1, \ldots, c_v\}$ is* valid *if for all properties $i \in \{1, \ldots, p\}$, either $c_1[i] = \cdots = c_v[i]$ or the values $c_1[i], \ldots, c_v[i]$ are distinct.*

Definition 1.2. *The decision problem* SET *is as follows. Given a set of cards $C \subseteq \{1, \ldots, v\}^p$ for any $v > 0$ and $p > 0$, is there a valid set $S \subseteq C$? The problems k-*VALUE SET *and k-*PROPERTY SET *differ from* SET *only through added restrictions on their input: in the former, we require that $v = k$; in the latter, we require that $p = k$.*

Another common characterization of Set is to define a valid set as a line in $\mathbb{Z}_3^4$, which implies an alternate generalization of finding lines in $\mathbb{Z}_v^p$. This is equivalent to our model when $v = 3$, but not in general. The authors believe that the above model is the most natural way to generalize the original game; the reader should be aware, however, that the alternate generalization can be solved in polynomial time.

1.3. **Our results.** In Section 2 we prove that SET is **NP**-complete, as is k-PROPERTY SET for $k \geq 3$. The proof uses a simple reduction from the problem 3-DIMENSIONAL MATCHING.

In section Section 3, we study the difficulty of 3-VALUE SET. In a restricted model in which algorithms are only allowed to check pairs of cards to see if they are part of a valid set in the input, we give a worst-case lower bound of $\lceil n^2/4 - n/2 \rceil$ checks and an algorithm which performs at most this many checks.

1.4. **Related work.** The authors are unaware of any previous work on the computational complexity of Set. However, problems involving combinatorial properties of the game have been considered by many. The most popular is to determine the smallest x for which any set of x cards must contain a valid set. The answer turns out to be 21 [2].

2. SET IS **NP**-COMPLETE

We first give a reduction from k-DIMENSIONAL MATCHING to k-PROPERTY SET. Since 3-DIMENSIONAL MATCHING is **NP**-complete [4, 5], the main result follows easily. Note that we use the terms *dimension* and *property* interchangeably.

Definition 2.1. *Let $v > 0$, $k > 0$, and $M \subseteq \{1, \ldots, v\}^k$. We say that M is a* perfect matching *if $|M| = v$ so that $M = \{m_1, \ldots, m_v\}$ and for all dimensions $i \in \{1, \ldots, k\}$, the values $m_1[k], \ldots, m_v[k]$ are distinct.*

Definition 2.2. *The decision problem k-*DIMENSIONAL MATCHING *is as follows. Given a set $C \subseteq \{1, \ldots, v\}^k$ for some $v > 0$, is there a perfect matching $M \subseteq C$?*

Lemma 2.3. *There is a polynomial-time mapping reduction from k-*DIMENSIONAL MATCHING *to k-*PROPERTY SET.

[2] We leave some out details which do not concern our analysis, such as dealing more than 12 cards when no valid sets are present, and scoring when players claim valid sets incorrectly.

Proof. We are given $C \subseteq \{1, \ldots, v\}^k$ and will produce $C' \subseteq \{1, \ldots, v+1\}^k$ such that C' contains a valid set if and only if C contains a perfect matching. Let $c \in \{1, \ldots, v+1\}^k$ be the element $(v+1, \ldots, v+1)$, and pick $C' = C \cup c$. This reduction is illustrated in Figure 2.1. Clearly we can construct C' in polynomial time. Why does it work?

Suppose there is a perfect matching $M \subseteq C$. Let $M' = M \cup c$. Since for all $i \in \{1, \ldots, k\}$ and all $m \in M$, $m[i] \neq v+1 = c[i]$, and $|M'| = v+1$, M' is a perfect matching and is also valid, since by definition every perfect matching is a valid set. Finally, $M' \subseteq C'$, so C' contains a valid set.

Now suppose C' contains some valid set $M' = \{c_1, \ldots, c_{v+1}\}$. There must be some property $i \in \{1, \ldots, k\}$ for which $c_1[i], \ldots, c_{v+1}[i]$ are distinct (because otherwise the cards $c_1, \ldots, c_{v+1}$ would all be the same card, and we need v distinct cards to form a valid set). By the pigeonhole principle, for some $c_j \in M'$ we have $c_j[i] = v+1$. c is the only element of C' which takes on the value $v+1$ in any dimension, so $c \in M'$. Furthermore, since no other elements of M' agree with c in any dimension, the elements of M' must be distinct in all dimensions, which means M' is a perfect matching. Thus $M' - \{c\}$ is a perfect matching in C. □

Corollary 2.4. *k-PROPERTY SET for $k \geq 3$ and SET are **NP**-complete.*

Proof. SET and k-PROPERTY SET are clearly in **NP**: a non-deterministic algorithm can simply guess a subset of the cards and check to see whether it is valid. By Lemma 2.3 and the fact that 3-DIMENSIONAL MATCHING is **NP**-hard, 3-PROPERTY SET is **NP**-hard, which implies that k-PROPERTY SET for $k \geq 3$ and SET are also **NP**-hard, since they are generalizations of 3-PROPERTY SET. □

Note that 2-PROPERTY SET can be solved in polynomial time, as follows: View the deck as a $v \times v$ matrix. A valid set is either a row or column of this matrix, or a perfect matching. We can clearly find sets of the former type in polynomial time. For sets of the latter type, we can use a polynomial-time algorithm for 2-DIMENSIONAL MATCHING [1].

3. COMPLEXITY OF k-VALUE SET

Having shown that Set is hard when the number of values varies, we now fix the number of values at k and allow the number of properties p to vary. This is the problem k-VALUE SET, defined in Section 1.2. In the discussion here, we focus on the case $k = 3$, which is of particular interest since the classic Set deck uses three values, and because three is the smallest k for which the problem is nontrivial.

There is an obvious brute force $\Theta(n^2 p)$-time algorithm for 3-VALUE SET[3]:

[3]The generalization of this algorithm to arbitrary k runs in time $\Theta(n^{k-1}p)$.

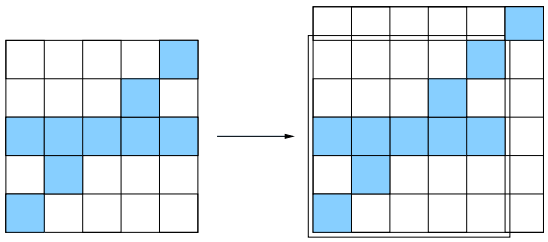

FIGURE 2.1. The reduction from k-DIMENSIONAL MATCHING to k-PROPERTY SET with $k = 2$ and $v = 5$. In the figure on the left, each cell represents an element of the set $\{1, \ldots, 5\}^2$, and the shaded cells are the elements of the input C. The reduction translates this into the figure on the right, where the cells are elements of $\{1, \ldots, 6\}^2$, and the shaded cells are the elements of C'. In C, both the row and the diagonal are valid sets, but only the diagonal is a perfect matching. In C', the row is no longer a valid set, but the diagonal remains both a valid set and a perfect matching, because we have added the element c in the upper right corner.

Algorithm BruteForce: "On input $C = \{c_1, \ldots, c_n\}$:
(1) For each pair of cards $c_h, c_i \in C_1$:
  (a) Compute the unique card c_j such that $\{c_h, c_i, c_j\}$ is valid.
  (b) If $c_j \in C$, then return **valid set**.
(2) Return **no valid set**."

There are $\binom{n}{2} = \Theta(n^2)$ pairs to check in the loop in Step 1. The computation of c_j takes time $\Theta(p)$ (i.e. linear in the number of bits used to represent a card), and the check to see whether $c_j \in C$ can be done in expected time $\Theta(p)$ through use of a hash table. Thus the algorithm runs in expected time $\Theta(n^2 p)$. Note that the number of bits used to represent the input is $m = \Theta(pn)$, so the expected runtime can alternately be written as $\Theta(m^2/p)$.

Naturally one would like to do better than BruteForce. In particular, can we choose the pairs of cards to check smartly, rather than just checking every pair? The following theorem states that we can make only a constant factor fewer checks than BruteForce.

Definition 3.1. *A* pair-checking algorithm *is one that solves* 3-VALUE SET *given only n and indirect access to its input $C = \{c_1, \ldots, c_n\}$ through queries to an oracle which, when given two card identifiers $h, i \in \{1, \ldots, n\}$, returns **true** if there is a card $c_j \in C$ such that $\{c_h, c_i, c_j\}$ is valid, and returns **false** otherwise.*

Theorem 3.2. *Any pair-checking algorithm must perform at least $\lceil n^2/4 - n/2 \rceil$ checks in the worst case.*

Proof. Let A be any pair-checking algorithm. For a particular input C, let G_i be the graph which has a node for each card in C and an edge (c_1, c_2) when after the ith check, A has not yet checked (c_1, c_2). Thus G_0 is the complete graph on $n = |C|$ nodes, and a check corresponds to the removal of an edge.

The infinite family of inputs $X_p = \{1, 2\}^p$ contains no valid sets. (In the classic deck, this corresponds to, for example, no reds, no ovals, no solids, and no ones.) On these inputs, A must check one pair of cards from every triple of cards. To see this, suppose A didn't check some triple. Then A must behave incorrectly either on X_p or on inputs of the same size as X_p in which the unchecked triple was the one and only valid set. Such an input is some ordering of the cards $Y_p = X_p \cup \{(1, \ldots, 1, 3)\} \setminus \{(2, \ldots, 2)\}$, whose only valid set is $\{(1, \ldots, 1, 1), (1, \ldots, 1, 2), (1, \ldots, 1, 3)\}$.

This implies that if A terminates after t checks on input X_p, G_t must not contain a triangle. By Turán's Theorem [7, 8] this implies that G_t can have at most $n^2/4$ edges. Since the initial complete graph G_0 has $n^2/2 - n/2$ edges, A must have made at least $\lceil n^2/4 - n/2 \rceil$ checks. □

A triangle-free graph which has the maximum number of edges, as given by Turán's Theorem, is the complete bipartite graph $K_{\lfloor n/2 \rfloor, \lceil n/2 \rceil}$. Taking the complement of this graph to find the edges that we should check, we arrive at the following optimum pair-checking algorithm.

Algorithm OptimumPairCheck: "On input $C = \{c_1, \ldots, c_n\}$:
(1) Partition C into two sets $C_1 = \{c_1, \ldots, c_{\lfloor n/2 \rfloor}\}$ and $C_2 = \{c_{\lfloor n/2 \rfloor + 1}, \ldots, c_n\}$.
(2) For each pair of cards $c_h, c_i \in C_1$ and each pair of cards $c_h, c_i \in C_2$:
  (a) Compute the unique card c_j such that $\{c_h, c_i, c_j\}$ is valid.
  (b) If $c_j \in C$, then return **valid set**.
(3) Return **no valid set**."

In the classic Set game with 12 cards on the table, OptimumPairCheck uses 30 checks, as opposed to BruteForce's 66 checks. Asymptotically, OptimumPairCheck uses half as many checks as BruteForce.

4. CONCLUSION

We generalized the game of Set to variable-sized inputs, letting the number of properties p and the number of values v vary. We proved that when $p \geq 3$, the game is **NP**-complete. When $p \leq 2$ or v is fixed, the problem is in **P**.

We then considered the case when v is fixed at 3, as in the classic game, and p varies. In a restricted model in which algorithms are only allowed to check pairs of cards to see if they are part of a valid set in

the input, we gave a worst-case lower bound of $\lceil n^2/4 - n/2 \rceil$ checks and an algorithm which performs at most this many checks.

The main question we leave open is the following: is there a $o(n^2 d)$-time non-pair-checking algorithm for 3-VALUE SET? Noting that in this case the problem is equivalent to finding three collinear points from a subset of $\mathbb{Z}_v^p$, it may be interesting to explore possible connections with computational geometry problems such as that of finding three collinear points from a set of n points in the plane. That problem is 3SUM-hard [3]; it is an open problem to find a $o(n^2)$-time[4] algorithm for any 3SUM-hard problem.

REFERENCES

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, McGraw-Hill, New York, NY, 1990, pps. 600 - 604.

[2] B. Davis and D. Maclagan. The Game of SET! *The Mathematical Intelligencer.* To appear.

[3] A. Gajentaan and M. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, 5:165-185, 1995.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979, p. 50.

[5] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, ed. by J. W. Thatcher and R. E. Miller, Plenum Press, New York, 1972, pps. 85-103.

[6] Set Enterprises. `http://setgame.com`.

[7] P. Turán. Eine Extremalaufgabe aus der Graphentheorie. *Mat. Fiz Lapook* 48:436-452, 1941.

[8] D. West. *Introduction to Graph Theory.* Prentice Hall, Upper Saddle River, NJ, 1996, pps. 32-35.

[4]Note that the $\Theta(n^2)$ runtime of the best known 3SUM algorithm assumes that arithmetic operations can be performed on integers in constant time. If we made the corresponding assumption – that arithmetic operations can be performed on a card's vector in constant time, regardless of the number of properties – then the runtime of BruteForce and OptimumPairCheck would also be $\Theta(n^2)$.