

On-line Algorithms for 2-Coloring Hypergraphs via Chip Games

Javed A. Aslam*

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

Aditi Dhagat†

Department of Electrical Engineering and Computer Science
University of Wisconsin at Milwaukee
Milwaukee, WI 53201

Abstract

Erdős has shown that for all k -hypergraphs with fewer than 2^{k-1} edges, there exists a 2-coloring of the nodes so that no edge is monochromatic. Erdős has also shown that when the number of edges is greater than $k^2 2^{k+1}$, there exist k -hypergraphs with no such 2-coloring. These bounds are not constructive, however. In this paper, we take an “on-line” look at this problem, showing constructive upper and lower bounds on the number of edges of a hypergraph which allow it to be 2-colored *on-line*. These bounds become particularly interesting for degree- k k -hypergraphs which *always* have a good 2-coloring for all $k \geq 10$ by the Lovász Local Lemma. In this case, our upper bound demonstrates an inherent weakness of on-line strategies by constructing an adversary which defeats any on-line 2-coloring algorithm using degree- k k -hypergraph with $(3 + 2\sqrt{2})^k$ edges.

Keywords

Hypergraphs, Hypergraph Coloring, Graph Coloring, Property B, On-line Algorithms, On-line Computation, Lovász Local Lemma, Games

*Author was supported by DARPA Contract N00014-87-K-825 and National Science Foundation Grant CCR-8914428. Author’s net address: jaa@theory.lcs.mit.edu

†Portions of this work were done while the author was at the Laboratory for Computer Science at MIT. Author was supported by DARPA Contract N00014-87-K-825, National Science Foundation Grant CCR-8912586, and Air Force Contract AFOSR-89-0271. Author’s net address: aditi@cs.uwm.edu

1 Introduction

Often we can prove the existence of objects with certain properties without being able to find such objects efficiently. For example, it can be shown that a random graph on n nodes (where each edge is present with probability $1/2$) has a clique of size $2 \log_2 n$, but no efficient algorithm for finding a clique this large (or even one of size $(1 + \epsilon) \log_2 n$, for some $\epsilon > 0$) is known. In this paper, we consider another such problem: that of 2-coloring hypergraphs.

A hypergraph \mathcal{H} is a collection of subsets (edges) E_1, E_2, \dots, E_s of a set of nodes $V = \{1, \dots, n\}$. A hypergraph is said to be *2-colorable* (or to have “Property B”) if there exists a coloring of the set of nodes V using two colors (say, red and blue) so that no edge is monochromatic. A k -hypergraph is a hypergraph where each edge E_i contains exactly k nodes. A 2-hypergraph is therefore an ordinary graph. In this paper we consider the problem of finding 2-colorings for k -hypergraphs on-line.

Many of the early results on this problem are due to Erdős. Erdős and Hajnal [6] first posed the question, “What is $m(k)$, where $m(k)$ is the largest s such that each k -hypergraph with s edges has Property B?” Erdős [4, 5] has shown that:

$$2^{k-1} < m(k) < k^2 2^{k+1}$$

These bounds show that all k -hypergraphs with fewer than 2^{k-1} edges are 2-colorable, but if the number of edges is greater than $k^2 2^{k+1}$, then there exists a k -hypergraph which has no proper 2-coloring. These bounds on $m(k)$ are not constructive, however. The upper bound, for instance, does not yield an algorithm that finds a hypergraph which is not 2-colorable. The only known constructive upper bound is due to Abbott and Moser [1] who can find a “bad” hypergraph if the number of allowable edges is at least $(\sqrt{7} + \epsilon)^k$ (for arbitrarily small, positive ϵ). The lower bound on $m(k)$ was improved to $2^k(1 + 4k^{-1})^{-1}$ by Schmidt [11] and then to $\Omega(k^{\frac{1}{3}} 2^k)$ by Beck [2]. None of these bounds are algorithmic, however, in that they do not provide an efficient method for 2-coloring hypergraphs which have an appropriately small number of edges.

This paper focuses on the algorithmic question of finding good 2-colorings for hypergraphs. Unfortunately, the general problem of 2-coloring hypergraphs is equivalent to set splitting and thus known to be NP-complete [10]. We instead find 2-colorings of hypergraphs restricted by size and degree. Let $f(k)$ be the largest s such that all k -hypergraphs with s edges can be 2-colored *on-line*. Our first result matches the original lower bound of Erdős *constructively*, showing that $f(k) > 2^{k-1}$ by giving an on-line algorithm for 2-coloring any k -hypergraph with fewer than 2^{k-1} edges which runs in $O(ns)$ time and is $O(1)$ -competitive against any adaptive on-line adversary. For definitions of competitiveness of on-line algorithms and adversaries, see papers of Karlin et al. [9] and Ben-David et al. [3]. For this result we use a weighting technique similar to one that Erdős and Selfridge [8] used to solve another problem. Our second result constructively shows that $f(k) < k \cdot \phi^{2k} = k \cdot (\frac{3+\sqrt{5}}{2})^k$ by describing an adversary which produces an appropriately large k -hypergraph which cannot be 2-colored on-line (here, ϕ is the Golden Ratio $= \frac{1+\sqrt{5}}{2}$). Clearly the Erdős upper bound on $m(k)$ tells us that there exist even smaller k -hypergraphs which cannot be 2-colored on-line or off-line. But our result gives a way to *find* these k -hypergraphs which cannot be 2-colored on-line, asymptotically improving on the bound of $(\sqrt{7} + \epsilon)^k$ from the off-line adversary of Abbott and Moser.

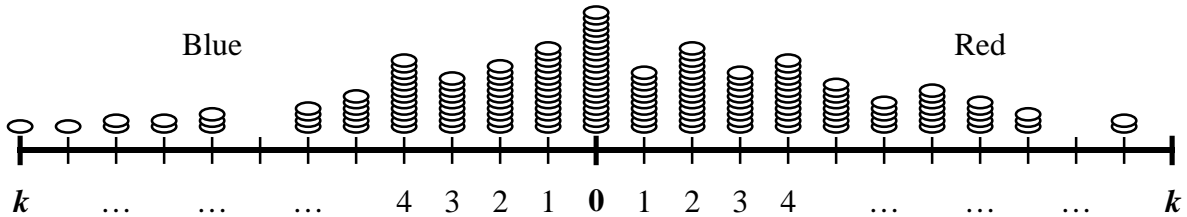


Figure 1: Chip Game

Our next result is motivated by the Lovász Local Lemma which first appeared in a paper by Erdős and Lovász [7]. Using this lemma, it can be shown that degree- k k -hypergraphs (k -hypergraphs where each node appears in at most k edges) are *always* 2-colorable for all $k \geq 10$. Our result gives evidence of the inherent weakness of on-line 2-coloring algorithms by showing that they fail against degree- k k -hypergraphs with more than $(3 + 2\sqrt{2})^k$ edges. Again, the adversary to the on-line algorithm is constructive, providing a hypergraph which the algorithm fails to 2-color.

As an application of these results, consider a scheduling problem which we will call the Committee Meeting Problem. Suppose that there are n committee members and s committees drawn from these members. At a conference there will be two general meetings held in parallel. Is there a way to assign each member to one of these two meetings so that each committee has at least one member at each meeting? Further, is it possible to make these assignments *on-line* (as each member registers or his registration material is received)? On-line algorithms for 2-coloring hypergraphs solve this problem.

We model our problem using Chip Games, first defined by Spencer [12]. We begin in the next section by defining the Chip Game. In sections 3 and 4 we prove certain properties of these Chip Games which we then generalize in section 5. These properties allow us to conclude the aforementioned results in section 6. Section 7 ends the paper with mention of some open problems.

2 Chip Games

At each step, an on-line algorithm for 2-coloring a k -hypergraph \mathcal{H} is given a node $j \in V$ and the hyperedges E_i which contain j . It must then choose a color for this node, say red or blue. If the algorithm is successful, then after $|V|$ such steps, none of the hyperedges E_i will be monochromatic. This procedure can easily be viewed as a game between two players, a Pusher and a Chooser. In every round, the Pusher selects which edges the current node is in, and the Chooser then selects a color for that node. The Pusher wins if and only if at least one of the edges is monochromatic at the end of round $|V|$. Clearly, a winning strategy for the Chooser translates into an on-line algorithm for 2-coloring \mathcal{H} . Such an algorithm will simply color each node according to the Chooser's winning strategy.

Consider the following game equivalent to the Pusher-Chooser game above. Here, each of the edges is represented by a chip which is located on a unidimensional board with levels marked "0" in the middle, "1" through " k " on the right, and "1" through " k " on the left, as shown in Figure 1. Each chip belongs to one of four categories:

1. Blank chips: These are chips at level 0, corresponding to empty hyperedges so far.
2. Red chips: These chips correspond to monochromatic red hyperedges and are placed at levels to the right of 0. A red chip at level j corresponds to a monochromatic red edge with j nodes.
3. Blue chips: These chips correspond to monochromatic blue hyperedges and are placed at levels to the left of 0. Again, a blue chip at level j corresponds to a monochromatic blue edge with j nodes.
4. Garbage chips: These correspond to nonempty hyperedges which are no longer monochromatic. Garbage chips are kept off the board.

At the beginning of the game, all s chips are blank. In each round of the game, the Pusher selects some number of blank, red, blue and garbage chips (corresponding to the hyperedges containing the next node), and the Chooser selects a color. If the chosen color is red, then the blank chips move one level to the right (since the corresponding edges are now monochromatic red with one node), the red chips also move one level to the right (since the corresponding monochromatic red edges have one extra red node) and the blue chips become garbage and move off the board. Analogously, if the chosen color is blue, then the blank and blue chips move one level to the left, and the red chips become garbage. The selected garbage chips remain garbage and stay off the board.

There are $|V|$ such rounds. The Pusher wins the game if he has forced at least one chip to either level k by the end of the game, since such a chip corresponds to a monochromatic edge in the k -hypergraph. For s chips and a board with $2k + 1$ levels (as in Figure 1), we call the game defined thus far the **General Chip Game** $G(s, k)$. This corresponds to our k -hypergraph 2-coloring problem. We will also be interested in a restricted Chip Game, where the Pusher is allowed to pick at most c chips in any round, for some fixed c . We call this the **c -Restricted Chip Game** $G(s, k)$. This restricted game corresponds to the degree- c k -hypergraph 2-coloring problem.

Both of these Chip Games are well-defined in the game-theoretic sense. So for any s and k , exactly one of the Pusher or Chooser has a winning strategy. Now our function f from the introduction can be equivalently defined as follows: $f(k)$ is the largest s for which the Chooser has a winning strategy in the General Chip Game $G(s, k)$, and $f_c(k)$ is the largest s for which the Chooser has a winning strategy in the c -Restricted Chip Game $G(s, k)$. In the following sections, we give lower and upper bounds for f and f_c .

3 Lower Bound for f and f_c

In this section we give a lower bound on the number of chips needed for the Pusher to win a Chip Game. This lower bound holds regardless of how many chips the Pusher is allowed to choose in each round. Thus, it applies to both the General and c -Restricted chip games.

The Pusher wins a Chip Game if he can force a chip to level k by the end of the game. A winning strategy for the Chooser must prevent this from happening. The intuition behind the Chooser's strategy is to give more importance to chips which are closer to level k .

The Chooser assigns a *weight* to each of the red and blue chips on the board. Each chip x at level $i > 0$ is given the weight $w(x) = 2^{i-1}$. We define the weight of a set of chips \mathcal{C} by $W(\mathcal{C}) = \sum_{x \in \mathcal{C}} w(x)$. The Chooser will color according to the following strategy. In each round j , the Pusher selects some number of blank, red, blue and garbage chips. The Chooser will calculate the weight of the selected blue chips and the weight of the selected red chips. If the selected blue chips are “heavier” than the selected red chips, then the Chooser will color the new node red. If the selected red chips are heavier, then the Chooser will color the new node blue. If the weights are equal (or no red or blue chips were selected), then he will color arbitrarily. Consider a Chip Game in which the Chooser follows the aforementioned strategy. Let $b(j)$ be the number of blank chips used in round j , and let $B(j)$ be the total number of blank chips used in rounds $1, 2, \dots, j$. Thus $B(j) = \sum_{i=1}^j b(i)$ and recursively $B(j) = B(j-1) + b(j)$. Let \mathcal{R}_j be the set of red chips after round j , and let \mathcal{B}_j be the set of blue chips after round j . Further, let \mathcal{R}_j^* be the set of red chips selected in round $j+1$ and \mathcal{B}_j^* be the set of blue chips selected in round $j+1$. We now show the following:

Lemma 1 $\forall j, B(j) \geq W(\mathcal{R}_j) + W(\mathcal{B}_j)$.

Proof: The proof is by induction on the number of rounds.

Base case: In round 1 the Pusher selects some m blank chips, and the Chooser arbitrarily colors them red or blue (placing them at the appropriate level 1). Assume without loss of generality that the Chooser colors blue. Then $B(1) = m$, and $W(\mathcal{R}_1) + W(\mathcal{B}_1) = 0 + W(\mathcal{B}_1) = m \cdot 2^{1-1} = m$. Thus, $B(1) = W(\mathcal{R}_1) + W(\mathcal{B}_1)$.

Inductive step: By induction, assume that after round j , $B(j) \geq W(\mathcal{R}_j) + W(\mathcal{B}_j)$. Assume without loss of generality that $W(\mathcal{R}_j^*) \geq W(\mathcal{B}_j^*)$ (and thus the Chooser colors blue). We then have the following:

$$\begin{aligned}
B(j+1) &= B(j) + b(j+1) \\
&\geq W(\mathcal{R}_j) + W(\mathcal{B}_j) + b(j+1) \\
&= W(\mathcal{R}_j^*) + W(\mathcal{R}_j - \mathcal{R}_j^*) + W(\mathcal{B}_j^*) + W(\mathcal{B}_j - \mathcal{B}_j^*) + b(j+1) \\
&\geq 2 \cdot W(\mathcal{B}_j^*) + W(\mathcal{R}_j - \mathcal{R}_j^*) + W(\mathcal{B}_j - \mathcal{B}_j^*) + b(j+1) \\
&= W(\mathcal{R}_{j+1}) + W(\mathcal{B}_{j+1})
\end{aligned}$$

The last equality can easily be seen by considering which chips will be in the sets \mathcal{R}_{j+1} and \mathcal{B}_{j+1} , and by considering what new weights these chips will have. ■

We now have the following theorem:

Theorem 1 *The Pusher requires at least 2^{k-1} blank chips to win, i.e. $f(k), f_c(k) \geq 2^{k-1}$.*

Proof: In order to win, the Pusher must have at least one chip at level k by the end of the game. This chip alone has weight 2^{k-1} , and by Lemma 1 at least 2^{k-1} blank chips will have been used by this time. Since the Chooser’s strategy does not depend on how many chips the Pusher picks at a time, the lower bound applies to both f and f_c . ■

4 Upper Bounds for f and f_c

We now show that $f(k) < k(\frac{3+\sqrt{5}}{2})^k \approx k(2.618)^k$ and $f_c(k) < (3 + 2\sqrt{2})^k$. It is interesting to note that $\frac{3+\sqrt{5}}{2}$ is in fact ϕ^2 , where ϕ is the Golden Ratio. Since our bound for $f(k)$ is worse than the original upper bound due to Erdős, we know that there exist even smaller hypergraphs which are not 2-colorable, on-line or off-line. But our bound is *constructive* in that an adversary prevents the on-line algorithm from 2-coloring large hypergraphs by actually giving an example of a hypergraph which proves to be “bad” for on-line 2-coloring. In addition, the bound is asymptotically better for on-line algorithms than the previously known constructive off-line bound of Abbott and Moser.

The upper bound for $f_c(k)$ is of greater interest, since it demonstrates the weakness of on-line strategies where a solution always exists and can be found off-line. In this case, the Lovász Local Lemma tells us that degree- k k -hypergraphs can always be 2-colored (for all $k \geq 10$). But our upper bound on $f_c(k)$ shows that no on-line algorithm will always succeed on hypergraphs with more than $(3 + 2\sqrt{2})^k$ edges. Again, the adversary produces an example of a bad hypergraph.

4.1 Upper Bounds for f

We begin by giving a simple Pusher strategy which requires only 4^k chips (thus implying that $f(k) < 4^k$). Starting with a stack of s chips at level 0, the Pusher initially chooses half of these chips. The Chooser must move them all the same way, thus creating a stack of $s/2$ chips at either blue or red level 1 and leaving a stack of $s/2$ chips at level 0. In each subsequent round, the Pusher picks half of the chips in each of the two outermost stacks, thus creating a new half-stack to one side and leaving a half stack on the other side. This process can be repeated at most $2k$ times (the width of the board) until the Pusher wins by creating a half-stack at level k . Thus, at most $2^{2k} = 4^k$ chips are required initially.

In the above strategy, the Pusher is “passive” in the following sense— he has the power to move *some* number of chips closer to the edge (by picking both red and blue chips), but he does not have the power to decide *which* chips (red or blue) will be moved. The following Pusher strategy is “aggressive” in the sense that the Pusher now *forces* the Chooser to move a set of chips in a particular direction.

4.1.1 The Aggressive Strategy

Suppose the Pusher has two stacks of chips, one each at blue and red levels i . Suppose further that the Pusher wants to move some number of chips towards the blue edge of the board. He starts by pitting one chip at blue level i against one chip at red level i . If the Chooser colors blue, then the Pusher repeats the process. If the Chooser colors red, then the Pusher pits the chip moved to red level $i + 1$ against γ chips at blue level i (where γ is some fixed constant > 1). If the Chooser now colors blue, the Pusher starts over (pitting one chip at blue level i against one chip at red level i). If the Chooser colors red, the Pusher then pits the chip moved to red level $i + 2$ against γ^2 chips at blue level i . In general, as long as the Chooser colors red, the Pusher will pit the chip moved to red level $i + j$ against γ^j chips at blue level i . When the Chooser finally *relents* (i.e. colors blue), the Pusher will

start over (pitting one chip at blue level i against one chip at red level i). Notice that the Chooser must relent after at most $k - i$ rounds, or a chip will be moved to red level k (and thus the Pusher will win).

Lemma 2 *If there are at least $\frac{\gamma^{k-i}-1}{\gamma-1}$ chips at blue level i , then the Pusher is guaranteed to be able to move some number of chips to blue level $i + 1$.*

Proof: In the above strategy, chips are moved to blue level $i + 1$ whenever the Chooser relents and colors blue. As mentioned above, the Chooser must relent after at most $k - i$ rounds. Now consider the number of chips needed at blue level i to insure that the Pusher does not run out before the Chooser relents. If the Chooser does not relent until round $k - i$, then

$$1 + \gamma + \gamma^2 + \gamma^3 + \dots + \gamma^{k-i-1} = \frac{\gamma^{k-i} - 1}{\gamma - 1}$$

chips will have been used from blue level i . Thus, if there are at least $\frac{\gamma^{k-i}-1}{\gamma-1}$ chips at blue level i , then the Chooser will be forced to relent (and move some chips to blue level $i + 1$) before the stack at blue level i runs out. ■

We define a *stage* to be the sequence of rounds starting when the Pusher pits one chip at blue level i against one chip at red level i and ending when the Chooser finally relents. In each stage, some number of chips will be moved to blue level $i + 1$, and some number of chips will have been used from blue level i . We define the *fraction of chips moved* to blue level $i + 1$ as the quotient of the number of chips moved to blue level $i + 1$ and the number of chips used from blue level i .

Lemma 3 *In each stage, the fraction of chips moved is $> \frac{\gamma-1}{\gamma}$.*

Proof: Suppose the Chooser relents after r rounds. We then have the following:

$$\begin{aligned} \text{Chips moved to blue level } i + 1 &= \gamma^{r-1} \\ \text{Chips used from blue level } i &= 1 + \gamma + \gamma^2 + \dots + \gamma^{r-1} \\ &= \frac{\gamma^r - 1}{\gamma - 1} \end{aligned}$$

Therefore,

$$\begin{aligned} \text{Fraction of chips moved} &= \frac{\text{Chips moved to blue level } i + 1}{\text{Chips used from blue level } i} \\ &= (\gamma^{r-1}) / \left(\frac{\gamma^r - 1}{\gamma - 1} \right) \\ &= (\gamma - 1) \cdot \frac{\gamma^{r-1}}{\gamma^r - 1} \\ &> (\gamma - 1) \cdot \frac{\gamma^{r-1}}{\gamma^r} \\ &= \frac{\gamma - 1}{\gamma} \end{aligned}$$

Notice that this is independent of r . ■

We define the *total fraction of chips moved* after some number of stages as the quotient of the *total* number of chips moved to blue level $i + 1$ and the *total* number of chips used from blue level i .

Lemma 4 *After any number of stages, the total fraction of chips moved is $> \frac{\gamma-1}{\gamma}$.*

Proof: Let s be the number of stages. Let a_j be the number of chips *moved* to blue level $i + 1$ in the j -th stage, and let b_j be the number of chips *used* from blue level i in the j -th stage. From Lemma 3, we have that $\forall j, \frac{a_j}{b_j} > \frac{\gamma-1}{\gamma}$. The total fraction of chips moved after s stages is $(\sum_{j=1}^s a_j) / (\sum_{j=1}^s b_j)$.

$$\begin{aligned} \frac{\sum_{j=1}^s a_j}{\sum_{j=1}^s b_j} &> \frac{\sum_{j=1}^s b_j \frac{\gamma-1}{\gamma}}{\sum_{j=1}^s b_j} \\ &= \frac{\gamma-1}{\gamma} \cdot \frac{\sum_{j=1}^s b_j}{\sum_{j=1}^s b_j} \\ &= \frac{\gamma-1}{\gamma} \end{aligned}$$

■

Lemma 5 *Given $m + \frac{\gamma^{k-i}-1}{\gamma-1}$ chips at blue level i and $m \cdot \frac{\gamma-1}{\gamma}$ chips at red level i , the Pusher can move at least $m \cdot \frac{\gamma-1}{\gamma}$ chips to blue level $i + 1$.*

Proof: The Pusher will follow the strategy described above until there are fewer than $\frac{\gamma^{k-i}-1}{\gamma-1}$ chips left at blue level i . Lemma 2 guarantees that the Pusher can keep moving chips to blue level $i + 1$ as long as there are at least $\frac{\gamma^{k-i}-1}{\gamma-1}$ chips at blue level i . Clearly, when there are fewer than $\frac{\gamma^{k-i}-1}{\gamma-1}$ chips at blue level i , at least m chips will have been used. Therefore, by Lemma 4, we know that at least $m \cdot \frac{\gamma-1}{\gamma}$ chips will have been moved to blue level $i + 1$.

No more than $m \cdot \frac{\gamma-1}{\gamma}$ chips are needed at red level i to move $m \cdot \frac{\gamma-1}{\gamma}$ chips to blue level $i + 1$, since in the worst case the Chooser can destroy at most one red chip for every blue chip moved. ■

For simplicity of analysis, we will use a strictly weaker version of this lemma. Note that the following lemma is implied by the above lemma when $\gamma \geq 2$.

Lemma 6 *Given $m + \gamma^{k-i}$ chips at both blue and red levels i , the Pusher can move at least $m \cdot \frac{\gamma-1}{\gamma}$ chips to blue level $i + 1$.*

4.1.2 The Iterative Pusher Strategy

We now show an iterative Pusher strategy which produces the desired bound. In the i -th iteration, the Pusher starts with two equal height stacks, one each at blue and red levels i . In the course of this i -th iteration, the Pusher will create two new equal height stacks, one

each at blue and red levels $i + 1$. The Pusher wins if he starts with enough chips so that he does not run out before the k -th iteration. The number of chips needed in the i -th iteration (at blue and red levels i) will be a function of $k - i$; let $g(k - i)$ be this function. To win, the Pusher needs $g(k - k) = g(0) = 1$. We can now build a recurrence for the number of chips required in any iteration.

In the i -th iteration, the Pusher has stacks of height $g(k - i)$ at blue and red levels i . His strategy is to first pick $g(k - [i + 1])$ chips from each of these stacks. The Chooser must then create a stack of height $g(k - [i + 1])$ at either blue or red level $i + 1$. Assume, without loss of generality, that the Chooser colors red (and thus creates a stack of height $g(k - [i + 1])$ at *red* level $i + 1$). The Pusher must now create a stack of height $g(k - [i + 1])$ at *blue* level $i + 1$. To do this, he needs only $\frac{\gamma}{\gamma - 1} \cdot g(k - [i + 1]) + \gamma^{k-i}$ remaining chips at blue and red levels i (by Lemma 6). We thus obtain the following recurrence:

$$\begin{aligned} g(k - i) &= g(k - [i + 1]) + \frac{\gamma}{\gamma - 1} \cdot g(k - [i + 1]) + \gamma^{k-i} \\ &= \frac{2\gamma - 1}{\gamma - 1} \cdot g(k - [i + 1]) + \gamma^{k-i} \end{aligned}$$

Letting $m = k - i$, we obtain the following:

$$g(m) = \frac{2\gamma - 1}{\gamma - 1} \cdot g(m - 1) + \gamma^m$$

This recurrence is minimized when $\gamma = \frac{2\gamma - 1}{\gamma - 1}$. At this point, $\gamma = \frac{3 + \sqrt{5}}{2} \approx 2.618$, and the recurrence solves to $g(m) = m\gamma^m$. Note that $\frac{3 + \sqrt{5}}{2} = (\frac{1 + \sqrt{5}}{2})^2 = \phi^2$ where ϕ is the Golden Ratio.

Theorem 2 *Given $s \geq k\phi^{2k}$ chips, the Pusher can win the General Chip Game $G(s, k)$.*

Proof: We are interested in the number of chips needed at level 0. This value is obtained from the recurrence when $m = k$. Thus, only $k\phi^{2k}$ chips are required. ■

It should also be noted that $\gamma = \phi^2$ is equivalent to $\phi + 1$, and the fraction of chips moved in Lemma 3 and Lemma 4 (given by $\frac{\gamma - 1}{\gamma}$) is in fact $1/\phi = \phi - 1$.

4.2 Upper Bound for f_c

We now present an upper bound for $f_c(k)$ by exhibiting a winning strategy for the Pusher in the c -Restricted Chip Game $G((3 + 2\sqrt{2})^k, k)$. We will consider the case $c = 2$ (i.e. where two chips are picked in each round). The upper bound for the general case of $c > 2$ follows in a similar way, where the Pusher picks $\lceil c/2 \rceil$ and $\lfloor c/2 \rfloor$ from the two stacks it is working on in the strategy given below.

We define the *diameter* of two stacks on opposite halves of the board to be number of levels between them. A stack at blue level $i \geq 0$ and a stack at red level $j \geq 0$ will have diameter $i + j$. A single stack at level 0 will have diameter 0. The Pusher will start with a stack of chips at level 0 and will initially create two stacks with diameter greater than 0. In the following phases, the Pusher will start with two stacks of diameter d and will create

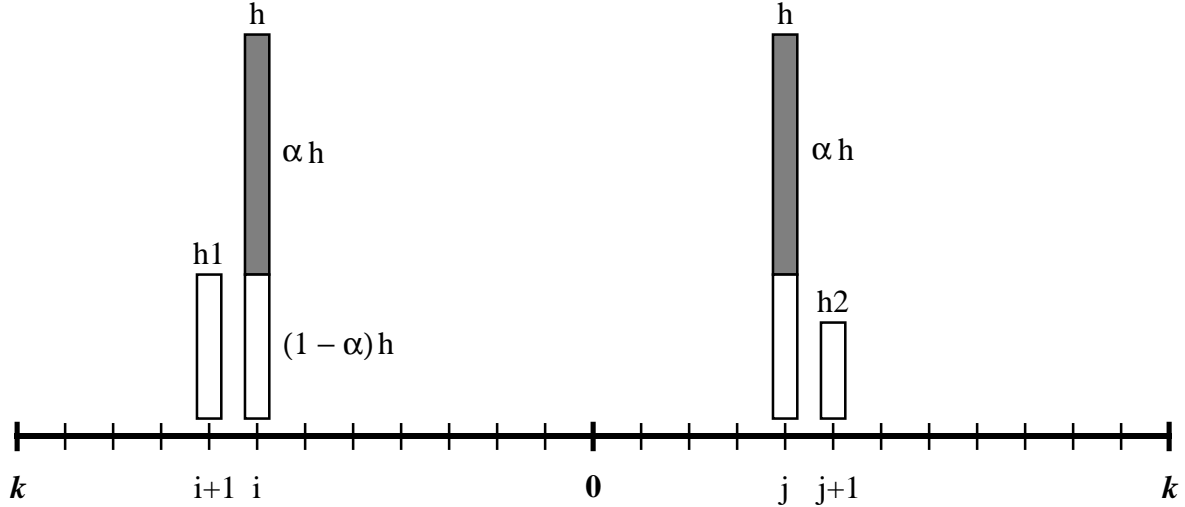


Figure 2: General Phase

new stacks of diameter greater than d . If after every phase, each of the newly created stacks is nonempty, then the Pusher will win in at most $2k$ phases (the “width” of the board). In the following paragraphs, we will show that for a particular Pusher strategy, the heights of the newly created stacks after each phase will be at least a constant fraction of the heights of the stacks used to create them. This will provide us with a bound on the height of the stack initially required at level 0.

Consider the following Pusher strategy. In the first phase, the Pusher will repeatedly select 2 chips from the stack at level 0 until the first point at which the height of one of the “outer” stacks (at levels 1) is equal to the height of the remaining stack at level 0. In each of the following phases, the Pusher will start with two equal height stacks, one at blue level i and one at red level j . The Pusher will repeatedly select one chip from blue level i and one chip from red level j in each round until the first point at which the height of one of the outer stacks is equal to the height of the remaining “inner” stacks (see Figure 2). The analysis that follows is for the latter general phase but also applies to the initial phase.

Let h be the height of the inner stacks initially, and let α be the fraction of the inner stacks used to create an outer stack of height equal to the remaining inner stacks. Clearly, $\frac{1}{2} \leq \alpha \leq \frac{2}{3}$. Let h_1 be the height of the outer stack which has height equal to the remaining inner stacks, and let h_2 be the height the other outer stack. Without loss of generality, assume that the stack of height h_1 is at level $i + 1$, and the stack of height h_2 is at level $j + 1$ as in Figure 2.

The Pusher now chooses which two stacks to use in the next phase according to the larger of $(\frac{h_1}{h})^2$ and $(\frac{h_2}{h})^2$. If $(\frac{h_1}{h})^2 > (\frac{h_2}{h})^2$, then the Pusher will use the stacks at levels $i + 1$ and j in the next phase. This corresponds to a decrease in height by the fraction $(\frac{h_1}{h})$ and an increase in diameter of 1. If $(\frac{h_2}{h})^2 \geq (\frac{h_1}{h})^2$, then the Pusher will use the stacks at levels $i + 1$ and $j + 1$ in the next iteration. This corresponds to a decrease in height by the fraction $(\frac{h_2}{h})$ and an increase in diameter of 2.

Lemma 7 *If $\max\left(\left(\frac{h_1}{h}\right)^2, \left(\frac{h_2}{h}\right)^2\right) \geq \beta$ for all phases, then at most $\left(\frac{1}{\beta}\right)^k$ chips are required for the Pusher to win.*

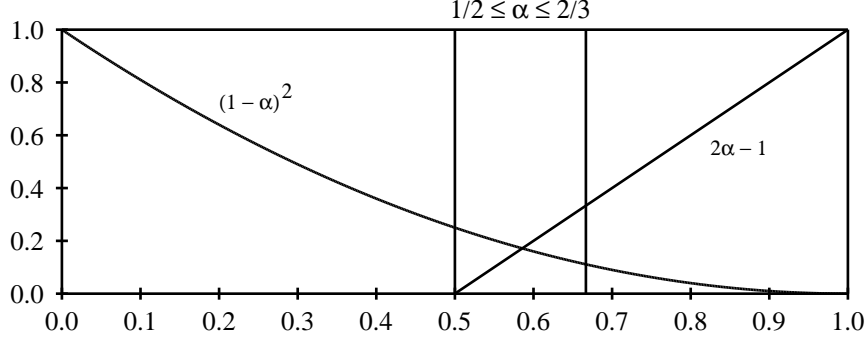


Figure 3: $z_1(\alpha)$ and $z_2(\alpha)$

Proof: Let N_1 be the number of phases where the change in diameter is 1, and let N_2 be the number of phases where the change in diameter is 2. Clearly, $N_1 + 2N_2 \leq 2k$. Further, in each of the N_1 phases of the former type, the height of the stacks will be decreased by at most $\beta^{\frac{1}{2}}$. In each of the N_2 phases of the latter type, the height of the stacks will be decreased by at most β . Let H_i be the initial height of the stack at level 0, and let H_f be the final heights of the outer stacks after $N_1 + N_2$ phases. We now have the following:

$$\begin{aligned} H_f &\geq H_i \beta^{\frac{N_1}{2}} \beta^{N_2} \\ &= H_i \beta^{\frac{N_1}{2} + N_2} \\ &\geq H_i \beta^k \end{aligned}$$

Thus, if $H_i = \left(\frac{1}{\beta}\right)^k$, then $H_f \geq 1$. ■

Lemma 8 *For the aforementioned Pusher strategy, $\max\left(\left(\frac{h_1}{h}\right)^2, \left(\frac{h_2}{h}\right)\right) \geq (3 - 2\sqrt{2})$ for all phases.*

Proof: Recall that α is the fraction of the inner stacks used to create an outer stack of height equal to that of the remaining inner stacks. As mentioned before, we have that $\frac{1}{2} \leq \alpha \leq \frac{2}{3}$. We also have that $h_1 = (1 - \alpha)h$ and $h_1 + h_2 = \alpha h$. This implies that

$$\begin{aligned} \left(\frac{h_1}{h}\right)^2 &= 1 - \alpha \\ \left(\frac{h_2}{h}\right) &= 2\alpha - 1 \end{aligned}$$

Consider the functions $z_1 = \left(\frac{h_1}{h}\right)^2 = (1 - \alpha)^2$ and $z_2 = \left(\frac{h_2}{h}\right) = 2\alpha - 1$ (see Figure 3). In the range $\frac{1}{2} \leq \alpha \leq \frac{2}{3}$, $\max(z_1, z_2)$ is clearly minimized at the intersection of z_1 and z_2 . Equating z_1 and z_2 , we obtain $\alpha = 2 - \sqrt{2}$. Therefore, we have $z_1(2 - \sqrt{2}) = z_2(2 - \sqrt{2}) = 3 - 2\sqrt{2}$. Thus, $\max(z_1, z_2) \geq 3 - 2\sqrt{2}$. ■

Theorem 3 *The Pusher requires at most $(3 + 2\sqrt{2})^k$ chips to win, i.e. $f_c(k) \leq (3 + 2\sqrt{2})^k$.*

Proof: This is an immediate consequence of Lemmas 7 and 8, and the fact that $\frac{1}{3 - 2\sqrt{2}} = 3 + 2\sqrt{2}$. ■

5 Generalizations to d -coloring

The 2^{k-1} lower bound and the 4^k upper bound shown in the previous sections generalize to bounds for chip games where more than two colors are used. Consider the d -color Chip Game $G_d(s, k)$, where d different colors are employed. In this case, the board forms a d -pointed star, each ray of the star being divided into levels from 1 through k . Now the Pusher wins if he can force a chip to level k on any of the d rays. Note that a winning strategy for the Chooser corresponds to an algorithm for d -coloring k -hypergraphs. The following results fall straightforwardly out of previous bounds:

Corollary 1 *In the d -color Chip Game (General or Restricted), the Pusher requires at least d^{k-1} chips to win.*

Corollary 2 *If $s \geq 2^{dk}$, then the Pusher has a winning strategy in the d -color General Chip Game $G_d(s, k)$.*

6 Conclusion

From the Chooser's winning strategy, we can conclude the following about the k -hypergraph 2-coloring problem:

Corollary 3 *Any k -hypergraph on n nodes with $s < 2^{k-1}$ edges can be 2-colored on-line in $O(ns)$ time.*

Proof: At every step, the algorithm is given a node from V and the edges that it belongs to. The algorithm colors the node according to the Chooser's winning strategy in the General Chip Game $G(s, k)$, as given in section 3.

To follow the Chooser's winning strategy, the algorithm computes the weight of $\leq s$ chips, assigns a color to the current node, and then adjusts the weights of these chips. So each round takes $O(s)$ time. There are at most $|V| = n$ rounds, so the entire algorithm runs on-line in $O(ns)$ time. ■

Similarly, for degree- k k -hypergraphs:

Corollary 4 *Any degree- k k -hypergraph on n nodes with $s < 2^{k-1}$ edges can be 2-colored on-line $O(kn)$ time.*

Proof: Again, the algorithm will follow the Chooser's winning strategy in the k -Restricted Chip Game $G(s, k)$. It needs to compute (and update) weights of at most k chips in each round, since the Pusher may only pick k chips in any round. Each round then involves $O(k)$ work. With at most $|V| = n$ rounds, it runs in $O(kn)$ time on-line. ■

Similar corollaries can be deduced for the d -coloring problem for k -hypergraphs from Corollary 1. These algorithms derive from Pusher-Chooser games. Since the Pusher acts like an adaptive on-line adversary, and since the algorithms take constant amount of time per input, they are $O(1)$ -competitive against adaptive on-line adversaries.

The upper bound on f_c says that for $s \geq (3 + 2\sqrt{2})^k$, the Pusher has a winning strategy in c -Restricted Chip Game $G(s, k)$, for any fixed c . In particular, the Pusher has a winning strategy in the k -Restricted Chip Game $G(s, k)$. Since any on-line algorithm to 2-color \mathcal{H} can be thought of as a strategy for the Chooser, we can conclude that:

Corollary 5 *There exists an on-line adversarial algorithm, which for any on-line 2-coloring algorithm A and every $s \geq (3 + 2\sqrt{2})^k$, produces a degree- k k -hypergraph with s edges which A fails to 2-color.*

Corollary 6 *There exists an on-line adversarial algorithm, which for any on-line 2-coloring algorithm A and every $s \geq k\phi^{2k} \approx k(2.618)^k$, produces a k -hypergraph with s edges which A fails to 2-color.*

Again, similar corollaries for the d -coloring problem for k -hypergraphs can be deduced from Corollary 2.

7 Open Problems

We conclude by mentioning some problems this work leaves open. Most importantly, what is the exact value of $f(k)$ and $f_c(k)$ with respect to the two Chip Games? Our lower bound of 2^{k-1} for $f(k)$ and $f_c(k)$ falls short of the existential lower bound of $k^{\frac{1}{3}}2^k$ for $m(k)$, and can probably be improved. Our strategy for the Chooser is aggressive only when the Pusher chooses chips of unequal weights; when the Pusher chooses chips of equal weight, the Chooser colors arbitrarily. A better strategy may be found by finding a way to make it unprofitable for the Pusher to pick equal weights. Also, there may be a better lower bound for the restricted chip game which takes advantage of the fact that the Pusher is limited to picking a constant number of chips each time.

If the exact values of $f(k)$ and $f_c(k)$ are difficult to determine, what is $\lim_{k \rightarrow \infty} f(k)^{\frac{1}{k}}$? (Similarly for f_c ?)

The General and c -Restricted Chip Games are interesting in themselves. Are there other problems which they model?

8 Acknowledgments

We would like to thank Joel Spencer for suggesting this problem to us.

References

- [1] A. L. Abbot and L. Moser. On a combinatorial problem of Erdős and Hajnal. *Canadian Mathematical Bulletin*, 7, 1964.
- [2] J. Beck. On 3-chromatic hypergraphs. *Discrete Mathematics*, 24:127–137, 1978.

- [3] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson. On the power of randomization in online algorithms. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 379–386. ACM Press, 1990.
- [4] P. Erdős. On a combinatorial problem, I. In J. Spencer, editor, *Paul Erdős: The Art of Counting*, pages 440–444. MIT Press, 1973.
- [5] P. Erdős. On a combinatorial problem, II. In J. Spencer, editor, *Paul Erdős: The Art of Counting*, pages 445–447. MIT Press, 1973.
- [6] P. Erdős and A. Hajnal. On a property of families of sets. *Acta Mathematica*, 15:87–123, 1961.
- [7] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and Finite Sets*, 2:609–627, 1975.
- [8] P. Erdős and J. L. Selfridge. On a combinatorial game. *Journal of Combinatorial Theory*, 14:298–301, 1973.
- [9] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.
- [10] L. Lovász. Coverings and colorings of hypergraphs. In *Proceedings of the Fourth South-eastern Conference on Combinatorics, Graph Theory, and Computing*, pages 3–11. Utilitas Mathematica Publishing, 1973.
- [11] W. M. Schmidt. Ein kombinatorisches problem von P. Erdős und A. Hajnal. *Acta Mathematica*, 15:373–374, 1964.
- [12] J. Spencer. *Ten Lectures on the Probabilistic Method*, page 59. Society for Industrial and Applied Mathematics, 1987.