

In Response to Peg Jumping for Fun and Profit

Matthew Yancey
mpyancey@vt.edu

Department of Mathematics, Virginia Tech

May 1, 2006

Abstract

In this paper we begin by considering the optimal solution to a certain one dimensional puzzle game. In the complexity of the game and an optimal algorithm were given in a previous paper. Here, we will show that there are only two optimal algorithms, all optimal solutions will form to certain criteria, and how these results extend into alternative versions of the game.

1 One Dimensional Symmetric Boards

This paper addresses finding the optimal solution to a peg jumping game and similar games generated from simple modifications. We begin by describing the puzzle. Consider a one dimensional board with $2n + 1$ holes to hold pegs. The board begins with a configuration where blue pegs fill the n leftmost holes and red pegs fill the n rightmost holes. The middle hole is left empty. The objective is to use a series of actions to get the blue pegs on the left to fill the n rightmost holes and the red pegs on the right to fill the n leftmost holes. An optimal solution would obtain this objective in as few actions, or moves, as possible.

During each action, a single peg may move out of its current hole and into the open space. However, a peg may not move more than two spaces per action. Depending on where the only open space in the board is relative to the ends of the board, there will be two to four different actions possible. A 'step' is an action when a peg moves one space. A 'jump' is an action when a peg moves two spaces.

The weight of the board is a concept considered in depth in Section 2.3, but suffice it to say at this point that the weight of the board, as defined in this paper, is the summation of the distances each peg must travel in order to reach a board configuration that satisfies the final objective. The weight of the board before the first action has been performed is $2n^2 + 2n$. The weight of the board when the game is concluded is 0. Given that no move can reduce the weight of the board by more than two, our first lower bound on an optimal algorithm is $n^2 + n$ actions.

This game is a generalization of a game marketed by International Games of Canada, Ltd., Mississauga, Ontario. Their game was produced with $n = 5$ and marketed under the name of 'Brainbuster' in the 1970s. An optimal algorithm and a lower bound proving the complexity of the problem have been produced in *Peg Jumping for Fun and Profit*, by David Bradley and Hugh Thomas. The algorithm from that paper is presented in Section 1.1, and the lower bound is presented in Section 2.2. Using their work as a starting ground, we will extend the proofs into stronger results.

1.1 The Bradley-Thomas Algorithm

In order to show the sequence of moves that formulate a solution to this problem, we will need symbols for each type of move used. A jump is indicated by the letter 'J.' A step is indicated by the letter 'S.' A lower case letter means a red peg moving to the left. An upper case letter indicates a blue peg moving to the right. These letters will provide us with notation for four types of moves, which will be the only four moves that we use.

It is useful to describe the configurations of the board in a manner similar to the moves above. In these cases, a hole filled with a red peg is represented by the letter 'R,' a hole filled with a blue peg is represented by the letter 'B,' and the empty space without a peg is represented by the letter 'S.'

Theorem 1 *The Bradley-Thomas Algorithm* Let m be a positive integer. If $n = 2m$, then

$$\left(\prod_{k=1}^{m-1} S j^{2k-1} s J^{2k} \right) S j^{2m-1} s J^{2m} s j^{2m-1} S \left(\prod_{k=m-1}^1 J^{2k} s j^{2k-1} S \right)$$

is a sequence of $n^2 + 2n$ moves that solves the problem. If $n = 2m - 1$, then

$$\left(\prod_{k=1}^{m-1} S j^{2k-1} s J^{2k} \right) S j^{2m-1} S \left(\prod_{k=m-1}^1 J^{2k} s j^{2k-1} S \right)$$

is a sequence of $n^2 + 2n$ moves that solves the problem. The solution to game Brainbuster, ($n = 5, m = 3$), using the above algorithm is

$$S j s J J S j j j s J J J J S j j j j j S J J J J s j j j S J J s j S$$

Proof: We will show the even case, where $n = 2m$ in this paper. The case where $n = 2m - 1$ was proven in *Peg Jumping for Fun and Profit*. The two cases are similar enough that the reader should be able to deduce one from the other. We begin our study with the board in the configuration of $B^n S R^n$.

The first term in the first product is $S j$, followed by $s J J$. This creates a board configuration of $B^{n-1} R B S R^{n-1}$, followed by $B^{n-2} S (R B)^2 R^{n-2}$. Given a board of the form $B^{n-2a} S (R B)^{2a} R^{n-2a}$, performing the sequence of moves $S j^{2a+1}$, followed by $s J^{2a+2}$ will result in a board of the form $B^{n-2a-1} (R B)^{2a+1} S R^{n-2a-1}$, followed by $B^{n-2a-2} S (R B)^{2a+2} R^{n-2a-2}$.

Thus, after performing the first $m - 1$ products, we are left with a board of the form $B^2S(RB)^{n-2}R^2$.

The middle section of the algorithm is Sj^{2m-1} , sJ^{2m} , and $sj^{2m-1}S$, in order. The first set of moves will transform the board into $B(RB)^{n-1}SR$. The second set of moves will take that board into a configuration of $S(RB)^n$. The final set of moves will result in a board configuration of $R^2(BR)^{n-2}SB^2$.

Note that the board at the beginning and the board at the end of the middle section of the algorithm are symmetric opposites. This board configuration can then be solved by going backwards through the steps in the first product. The second product is just that: the first product backwards. The final resulting board is R^nSB^n . The discussion in the following sections will continue to consider these three sections of the algorithm independently.

Now that we have shown that the algorithm is a valid solution to the game, we must show that it runs in $n^2 + 2n$ actions, as claimed. In the algorithm, there are

$$(m - 1) \times 2 + 4 + (m - 1) \times 2 = 2n$$

steps. Also, we see that there are

$$\sum_{k=1}^{n-2} (k) + 2m - 1 + 2m + 2m - 1 + \sum_{k=1}^{n-2} (k) = \frac{(n - 1)(n - 2)}{2} + 3n - 2 + \frac{(n - 1)(n - 2)}{2} = n^2$$

jumps. All totaled, the algorithm where $n = 2m$ results in $n^2 + 2n$ actions.

1.2 The State Diagram

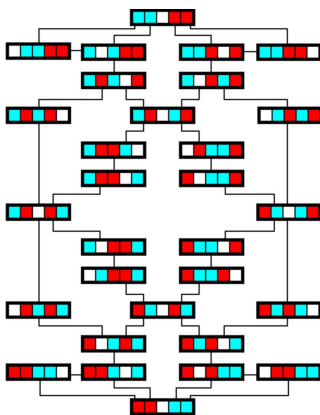


Figure 1: The complete game, where $n = 2$.

Occasionally, we will find it more useful to consider the configurations of the pegs placed on the board instead of the moves being performed on the board. Since there is a

finite number of board configurations and each board configuration has only a finite set of available actions, we can exhaustively describe the possible flows in and out of each board. By examining this, we can see the effects of certain moves and how they determine future possibilities. We show this in Figure 1, where the rectangles represent the possible board configurations, and the lines represent possible actions. Each square in the rectangle is a space for a peg. The white square represents the space with no peg in it, and the two colors represent a peg of that color in the square.

The number of board configurations possible is $\frac{(2n+1)!}{(n!)^2}$, and it becomes difficult rather quickly to lay out the entire game as a whole. However, we may use an abstract form of notation to represent the board configurations passed through during the Bradley-Thomas algorithm for games of any size.

In the state diagram shown in Figure 2, each board is represented as a member of a certain state. The rectangles represent the various boards, and the arrows show the sequence of moves used in the Bradley-Thomas algorithm. The diagram continues up to the middle section, where the final solution becomes symmetrical. Each rectangle is written out in letters. If the letters are in bold, then the letters grouped by parenthesis represent a repeating pattern. For example, **(BR)** is the same as saying $(BR)^i$, for some arbitrary $i \geq 1$. In this example, each rectangle represents multiple board configurations of similar structure.

It will be useful to note how a path is chosen when a possible branch is presented in the state diagram. In all cases where multiple arrows leave a single rectangle, the moves used for each arrow are the same. The choice of arrow to follow is not made by a player's choice, but is instead decided by the size of the repeating patterns in the particular board at that state in the algorithm. The choice of which branch that leads out of the loop in the middle of the algorithm to the other symmetrical side is controlled by whether n is even or n is odd. Hence, the path through this state diagram is strictly defined and is determined by the value of n .

2 Analyzing the Algorithm

There is more complexity behind the Bradley-Thomas algorithm than a simple ordering of moves. The algorithm creates and manipulates configurations of pegs according to a set of guidelines beyond the set of rules to the game. Using these guidelines, we can create lower bounds proving that the Bradley-Thomas algorithm is optimal. Through this deeper understanding of the game, we will also show that the algorithm is one of only two optimal solutions to the game.

2.1 Improper Moves

Definition 2 *The weight of the board is the minimum summation of the distances between each peg and that peg's position in a configuration that ends the game. Another method of describing the weight of the board is the summation of the minimum number of spaces*

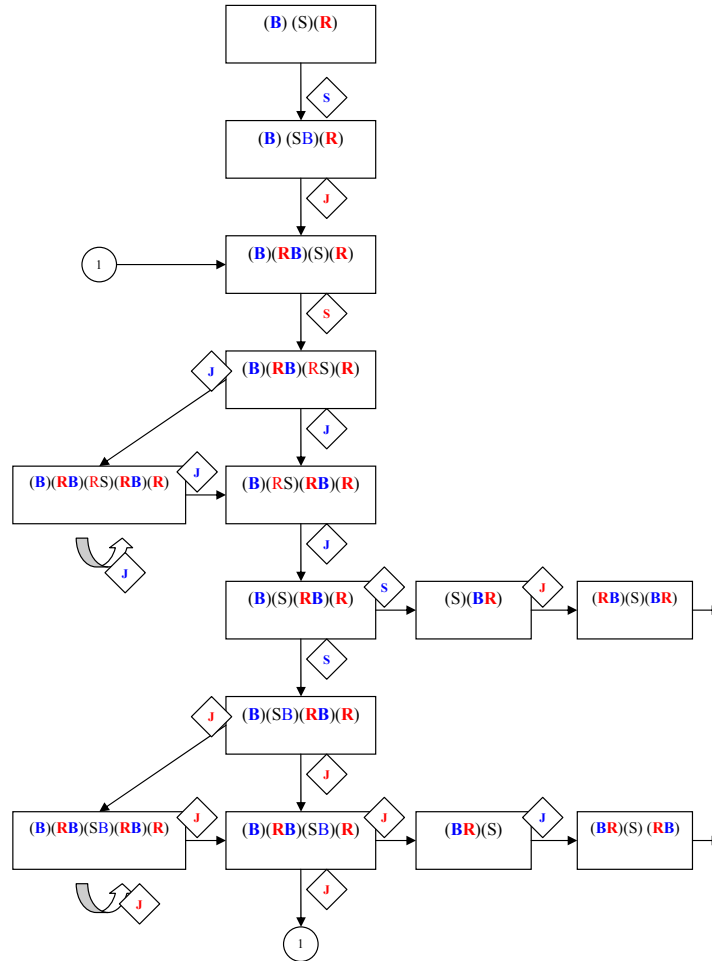


Figure 2: The first half of the Bradley-Thomas algorithm.

each peg must travel for the board as a whole to be in an ending state, without regard for any backwards moves that may be necessary given the rules of the game. The weight of the board is an integer that is strictly defined.

Certain moves are illegal. Just because a move is legal, however, does not mean that it is ideal. Some moves will remove more weight from the board than others. Some moves will

create preferable board configurations while others will create issues that ruin an optimal algorithm. Finding and avoiding non-optimal moves is a more complicated issue than attempting to always jump forward. In order to show that an algorithm is optimal, we need to move beyond the effects of a single action on the weight of the board to the effects of an action on the weight of the board and the resulting configuration of the board.

In order to identify optimal and non-optimal moves, we should categorize the types of moves possible. This paper will consider four types: hetero-colored jumps, homo-colored jumps, forward steps, and backward actions. A hetero-colored jump is when a red peg jumps across a blue peg or a blue peg jumps across a red peg. A homo-colored jump is when a blue peg jumps across a blue peg or a red peg jumps across a red peg.

It was originally shown by Bradley and Thomas that any optimal algorithm will only use hetero-colored jumps or forward steps. Here, we will re-word the proof that homo-colored jumps and backwards actions are excluded from any optimal algorithm, in a manner that leads into further conclusions.

Theorem 3 *Any algorithm, optimal or not, requires at least n^2 hetero-colored jumps to solve the game.*

Proof: In order to finish the game, every blue peg must cross every red peg. Since each peg may only cross one other peg per jump, each blue peg must be involved in n hetero-colored jumps, by either jumping or being jumped by each red peg. This comes to n^2 jumps in total.

Definition 4 *Consider an algorithm that solves a board with $2n + 1$ spaces. Let S be the series of moves used in the algorithm with n^2 arbitrarily chosen hetero-colored jumps removed. Those jumps are to be considered proper moves. Any other proper moves are actions that can be included in S such that the average weight removed per action in S is at least one. An improper move is an action where if it is included in the series S , then the average weight removed per action in S must be strictly less than one.*

Theorem 5 *The only actions that can be proper moves on a board of size $2n + 1$ spaces are hetero-colored jumps to a maximum of n^2 times and forward steps.*

Two lemmas are required to prove this theorem. We will show this theorem as a result from the following two statements.

Lemma 6 *Let $x, y \in \mathbb{Z}^+$. For an algorithm to have $n^2 + x$ forward hetero-colored jumps, it is necessary for that algorithm to also have x backwards hetero-colored jumps. For an algorithm to have y forward homo-colored jumps, it must also have y backwards moves of arbitrary type.*

Proof: It was proven in theorem 3 that there must be n^2 forward hetero-colored jumps. This is because the n blue pegs must cross the n red pegs. When these jumps are completed,

all blue pegs will be across the red pegs. If there is to be another forward hetero-colored jump, a blue peg must cross back behind a red peg. Hence, in order to have $n^2 + x$ forward hetero-colored jumps, there must be x backwards hetero-colored jumps.

Without loss of generality, we will only consider the homo-colored jumps that use blue pegs. There are two situations where this can happen. Either there is a red peg to the right of the position of the blue homo-colored jump in question, or there is one to the left.

In the first situation, where there is a red peg to the right of the jump, the board after the jump is in a configuration that looks like

$$*SBB * R*$$

where the $*$ symbol represents any permutation of pegs that is legal in respect to the rest of the board.

Note that the open space is to the left of the two blue pegs, and remember that the space moves around the board in a manner equal and opposite to the movement of pegs. Since the red peg to the right still needs to move left of the blue pegs in order to reach a finishing position, the open space must cross the two adjacent blue pegs at some point. But since the pegs can not move more than two spaces per action, the space can not jump across the two blue pegs without one of them moving first. Since the space can not cross the blue pegs, we know that the space will still be to the left of the two adjacent blue pegs when one of the two blue pegs does move, and therefore the blue peg will be backing up to the left when it happens.

Every time a homo-colored jump happens like this, a new pair of adjacent blue pegs will form. And for each pair, at least one move backwards will be forced.

In the next situation, where there is a red peg to the left of the jump, the board before the jump occurs is in a configuration that looks like

$$*R * BBS*$$

Once again, there are two adjacent blue pegs; however, this time the space is to the right of them. Once again, we know that the space can not cross the two adjacent blue pegs without one of them moving first. Because there is a red peg to the left of the blue pegs, we know that this is not the original configuration of the two blue pegs. From that, we know that at least one of the two blue pegs must have moved into its current position, and we know that it came from the right, and hence it was backing up as it did so.

Every time a homo-colored jump occurs like this, a pair of adjacent blue pegs must form. And since they must form with a space to the right, each must form with a backwards move.

Since each homo-colored jump requires a backwards move to either finish the game or to be set up, then we know that there are at least as many backwards moves as there are homo-colored jumps.

Lemma 7 *When calculating the average weight removed per action on a set of moves, any set that includes a backwards move is not proper.*

Proof: Consider an algorithm that uses $z \neq 0$ backwards moves. Since backwards moves remove negative weight from the board, the calculated average weight begins below one weight per move, and other moves must bring the average up sufficiently for the entire set of moves to become proper. Since the algorithm must be finite in order to finish, no number of forward steps will bring the average up to the level of proper moves. Therefore, the only way that a set of backwards moves can become proper is by averaging them with homo-colored jumps and additional hetero-colored jumps beyond the n^2 removed from the definition of a proper move. For the rest of this proof, forward steps will be ignored.

We know from lemma 6 that $z \geq x$ and $z \geq y$. We also know that at least x of the z backwards moves are jumps. Suppose $x \geq y$. Then there must be at least x backwards jumps, no more than x forward hetero-colored jumps, and no more than x homo-colored jumps. The maximum efficiency rating is $\frac{2}{3}$ weight per move, which is insufficient for a proper move. Now, suppose $x < y$. There are at least x backwards jumps, $y - x$ generic backwards moves, no more than x forward hetero-colored jumps, and no more than y forward homo-colored jumps. The maximum efficiency rating is $\frac{y+x}{2y+x}$ weight per move, which is also insufficient for a proper move.

Proof of Theorem 5: The Bradley-Thomas Algorithm completes the game using only forward steps in addition to the n^2 jumps necessary, proving that the forward step can be proper. This is not to imply that forward steps and hetero-colored jumps up to n^2 are always proper; they are only proper if they are not used in a situation that would then force an improper move. Section 2.3 further discusses the types of situations where these moves would not be proper.

Lemma 7 proved that any backwards move is an improper move. It also showed that anything forcing a backwards move is also improper. We may conclude from Lemma 6 that homo-colored jumps and hetero-colored jumps in excess of n^2 are improper. We have thus concluded showing what is and is not a proper move.

2.2 Lower Bound One: Limited Jumps

Theorem 8 *An algorithm is optimal if and only if it uses only proper moves. The Bradley-Thomas Algorithm is optimal.*

Proof: As proven in Theorem 3, any algorithm will use at least n^2 hetero-colored jumps. It was proven in Theorem 5 that no other hetero-colored jumps would be proper. Then any algorithm that uses only proper moves could only be comprised of those n^2 jumps and forward steps. The jumps would remove $2n^2$ weight from the board, so $2n$ steps would be necessary to complete the game. As proper moves, by definition, are strictly more efficient than improper moves, we may conclude that the $n^2 + 2n$ moves in a proper-move-only algorithm is a lower bound for the moves necessary to solve the game. The Bradley-Thomas Algorithm runs in $n^2 + 2n$ moves, and thus it is proven that this is the complexity of the game. The rest follows from the above.

2.3 Improper Boards

Improper boards are equally important to our discussion. These boards are configured such that they will force the use of an improper move, and thus they are configured such that they will never be seen in an optimal algorithm. Referring to the proof given in Lemma 6, where why homo-colored jumping is improper is discussed, we also know that the following boards are improper:

$$*S * BB * R*$$

$$*R * BB * S*$$

$$*S * RR * B*$$

$$*B * RR * S*$$

As a direct result of these boards being improper, we can determine another board as being improper:

$$*BRSBR*$$

This board is improper because the only two proper moves that can be made, either jumping the right red peg to the left or jumping the left blue peg to the right, will result in the fourth and first improper board configurations respectively, as listed above. We can now expand the set of improper moves from hetero-colored jumps and backwards actions to include moves that generate an improper board configuration.

2.4 Proof Two: Necessary Steps

Definition 9 *The conjugate of an algorithm is the algorithm when all red moves are replaced with identical blue moves and all blue moves are replaced with identical red moves from the original algorithm; in other words, it is the mirror solution reflected about the middle.*

Theorem 10 *The Bradley-Thomas Algorithm and its conjugate algorithm are the only optimal solutions.*

Proof: This is a proof by inspection. Return to the state diagram in Figure 2. Upon close examination, it becomes clear that, excluding one exception, each state in the diagram has only one available move into another state if only proper moves are used. In order to arrive at this, one must consider the results of Section 2.3 as well as of Section 2.1.

The only exception to this rule is that in the original state, before any pegs have been moved, there is a decision that can be made by the player on whether a red or blue peg will be moved first. Should a red peg be stepped forward first instead of a blue peg, then a perfectly symmetric algorithm, or the conjugate algorithm, is created.

Given this observation, we can see that this algorithm is the best possible because there is no available location to modify it while only using proper moves.

3 Unbalanced Boards: a Generalized Scenario

The work in the previous sections has been performed in a robust manner allowing us to consider generalizations on the problem by making slight changes to the results. The outcomes of these generalizations help clarify and test the mathematics used previously.

The first generalization we will consider is an asymmetric board. Let N_R represent the number of red pegs on the board. Similarly, let N_B represent the number of blue pegs on the board. Previously, we have been letting $N_R = N_B$. For this section, we will consider how the algorithm and lower bound are affected when $N_R \geq N_B$. Note that in the starting configuration, the red pegs will still be in the N_R rightmost holes, the blue pegs will still be in the N_B leftmost holes, and the open space will be between them.

Theorem 11 *The unbalanced algorithm* Let m, k be positive integers. Set the value $x = N_R - N_B$. If $N_B = 2m - 1$ and $x = 2k$, then

$$\left(\prod_{i=1}^{m-1} S j^{2i-1} s J^{2i} \right) S \left(\prod_{i=1}^k j^{2m-1} s J^{2m-1} s \right) j^{2m-1} S \left(\prod_{i=m-1}^1 J^{2i} s j^{2i-1} S \right)$$

is a sequence of $N_R N_B + N_R + N_B$ moves that solves the problem. If $N_B = 2m$ and $x = 2k$, then

$$\left(\prod_{i=1}^{m-1} S j^{2i-1} s J^{2i} \right) S j^{2m-1} s \left(\prod_{i=1}^k J^{2m} s j^{2m} s \right) J^{2m} s j^{2m-1} S \left(\prod_{i=m-1}^1 J^{2i} s j^{2i-1} S \right)$$

is a sequence of $N_R N_B + N_R + N_B$ moves that solves the problem. If $N_B = 2m - 1$ and $x = 2k - 1$, then

$$\left(\prod_{i=1}^{m-1} S j^{2i-1} s J^{2i} \right) S \left(\prod_{i=1}^k j^{2m-1} s J^{2m-1} s \right) \left(\prod_{i=m-1}^1 j^{2i} S J^{2i-1} s \right)$$

is a sequence of $N_R N_B + N_R + N_B$ moves that solves the problem. If $N_B = 2m$ and $x = 2k - 1$, then

$$\left(\prod_{i=1}^{m-1} S j^{2i-1} s J^{2i} \right) S j^{2m-1} \left(\prod_{i=1}^k s J^{2m} s j^{2m} \right) S J^{2m-1} s \left(\prod_{i=m-1}^1 j^{2i} S J^{2i-1} s \right)$$

is a sequence of $N_R N_B + N_R + N_B$ moves that solves the problem.

Proof: We will only be showing the proof of when $N_B = 2m - 1$ and $x = 2k$. Because this theorem is the author's own work, the other cases are not shown anywhere else. The proofs are very similar and are left to the reader to complete.

Consider a small portion of the board where the x rightmost red pegs are ignored. We know by the proof of Theorem 1 that we can perform the actions $\left(\prod_{i=1}^{m-1} S j^{2i-1} s J^{2i} \right) S$ to

create a board that looks like $S(BR)^{2m-1}(R)^x$ where the x rightmost pegs have been left untouched. At this point, we begin the middle product term. The first half of the term in the product is $j^{2m-1}s$, which turns the board into the configuration $(RB)^{2m-1}RS(R)^{x-1}$. The second half of the term in the product is $J^{2m-1}s$, which turns the board into the configuration $R^2S(BR)^{2m-1}(R)^{x-2}$. If this process is repeated k times, then the resulting board will be of configuration $R^xS(BR)^{2m-1}$. Performing a final sequence of moves j^{2m-1} , the board changes into the form of $R^x(RB)^{2m-1}S$. If we choose to ignore the leftmost x red pegs at this point, we once again know from the proof of Theorem 1 that we can perform the actions $S\left(\prod_{i=m-1}^1 J^{2i}s^{2i-1}S\right)$ to arrive at the final solution.

The number of steps in this algorithm is

$$2(m-1) + 1 + 2k + 1 + 2(m-1) = 4m - 2 + N_R - N_B = N_B + N_R$$

The number of jumps in this algorithm is

$$\sum_{i=1}^{N_B-1} i + 2k(N_B) + N_B + \sum_{i=1}^{N_B-1} i = \frac{1}{2}(N_B)(N_B-1) + N_R N_B - N_B^2 + N_B + \frac{1}{2}(N_B)(N_B-1) = N_R N_B$$

The total number of moves in the algorithm amounts to $N_R N_B + N_R + N_B$.

3.1 A New Algorithm with a New Flowchart

Note that the above formulas work even for the case where $N_B = N_R$. The algorithms for when $x = 2k$ are the exact same as in Theorem 1 should $k = 0$. The state diagram in Figure 2 is largely applicable to this modified game. Some small additions to the diagram are necessary first.

Shown in Figure 3 are additional states to the state diagram allowing a transition from the board configuration remaining when the first part of the Bradley-Thomas Algorithm concludes to the board configuration suitable for use in the last part of the Bradley-Thomas Algorithm. These are the additional steps necessary when $k \neq 0$. One change from before is that a boldface element representing a repeating element may now also be a null element. For example, **(BR)** is the same as saying $(BR)^i$ for some arbitrary $i \geq 0$, where it used to be that $i \geq 1$.

There are several observations to make about this state diagram. First, there are no steps used as moves among the blue pegs. Second, all paths shown are the only paths possible when using only proper boards. Third, all branches are decided by the size of the repeated segments, not by the choice of a player. Hence, there is only one path possible, which is determined by N_R and N_B .

3.2 Proving the Complexity, Repeated

Theorem 12 *Even if the board is unbalanced, an algorithm is optimal if and only if it only uses proper moves. The only optimal algorithms are the unbalanced algorithm and its conjugate.*

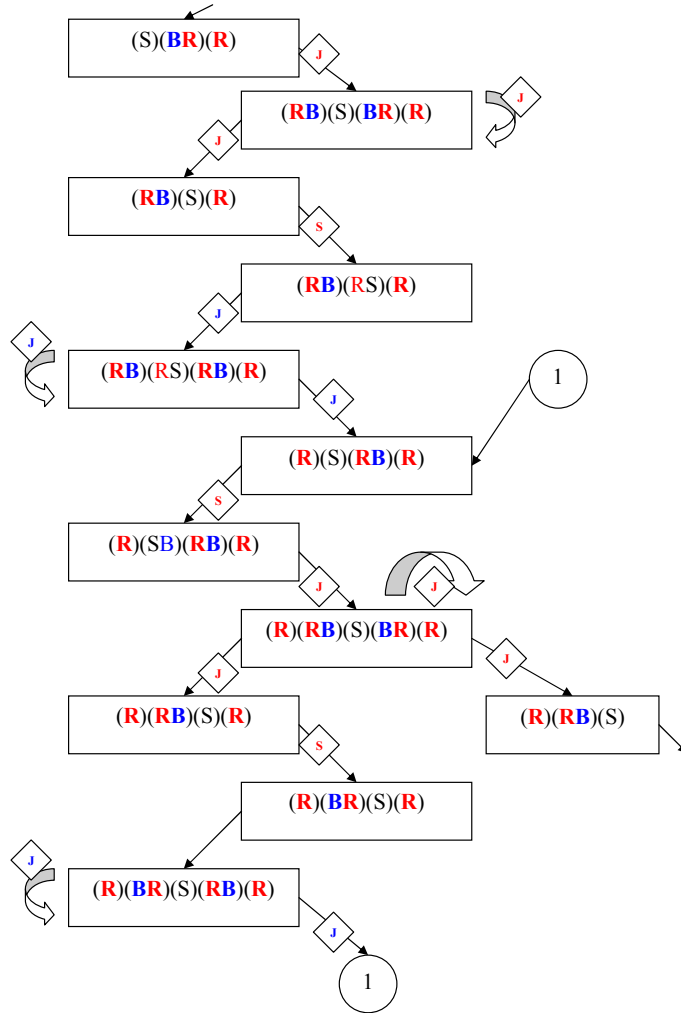


Figure 3: Alterations to the state diagram when $N_B = 2m - 1$ and $X = 2k$.

Proof: Reconsider the proofs addressing the number of hetero-colored jumps that are allowed and that are necessary. Once again, every red peg must cross every blue peg. Once again, if an additional hetero-colored jump is attempted, then it must be going backwards. So, using new numbers applied to the old reasoning, the number of hetero-colored jumps is now $N_B \times N_R$.

If an algorithm chooses to only use proper moves, then the rest of the weight must be removed by steps. Given that each jump reduces the weight by two and the board begins with $(N_R + 1) \times N_B + (N_B + 1) \times N_R$ weight, we can calculate that there is still $N_R + N_B$ weight to be removed. With $N_R + N_B$ steps in a proper-move-only algorithm, the total number of moves would be $N_B N_R + N_R + N_B$, which is the number of steps in the unbalanced algorithm. The argument that proper moves are more efficient still holds, so this serves as a lower bound. We have once again found the complexity of the problem. The bijection between proper moves and an optimal algorithm still holds.

By inspection of Figure 3, we can once again see that no alternative path is available. When we consider the algorithm as a whole, including the portion from the Bradley-Thomas Algorithm, the only place where there was a player choice in the moves made was at the very beginning, when the conjugate algorithm could be selected. The result still holds that there are only two optimal algorithms for this generalization of the game.

4 Forks in the Road

Another generalization that can be applied to this game is to think about it in multiple dimensions. In this section we will consider the effects of placing an additional path on the board by adding additional lengths of board, each attached to a single space on the board called a branching point. The extent of this paper will not cover such topics as cycles, paths with a thickness greater than one, or diagonal moves across holes. Those are open problems left to be solved. We will consider two generalizations: where the branching point is the middle hole where the space originally starts and where the branching points are to the side of the board.

4.1 Three Colors

The first generalization of this type is where the board extends in three directions from the center. In each direction there will be a different color: blue, red, and now yellow. To complete the game, the blue pegs must move into the yellow branch; the red pegs must move into the blue branch; and the yellow pegs must move into the red branch. In the case where each branch is only one deep, the obvious solution involves four moves.

Theorem 13 *There is no algorithm for the general three colored board that only uses proper moves.*

Proof: Our categorization of a proper move becomes shaky here, as homo-colored jumps may be permissible near the center of the board. The proofs that these moves were to be avoided depended on the linearity of the board. However, the general ideas proposed about proper and improper boards in Section 2.3 may still be used under certain conditions. Hence, we still know that the first move of any algorithm should be a step, not a jump. From these limited beginnings, we can show that an improper move will be forced.

When each branch reaches a length of at least two, complications arise. In order to show the issue, let us walk through the game. Without loss of generality, let blue move first. If blue jumps first, a deadlock is created, so the first move will be a blue step. Now the space is in the blue branch. Red is the color that is interested in the blue branch, so the second move will be a red jump. Now the space is in the red branch, so the third move will be a yellow jump. At the end of these moves, a blue peg is in the center, a red peg is in the blue branch, a yellow peg is in the red branch, and the three outside pegs have not moved yet.

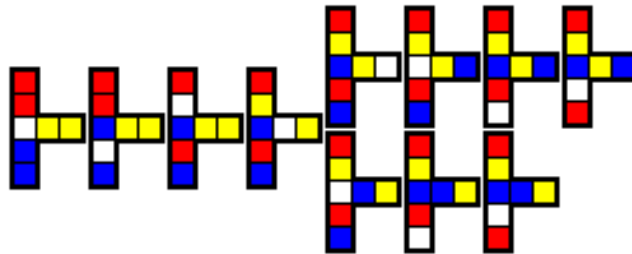


Figure 4: The failure of three colors.

At this point, there are two choices: either move the blue peg in the center into the yellow branch, or move the outside yellow peg towards the center. In the first scenario, the next move is to jump the outside blue peg as the only other forward move would be jumping the red peg, which would create a deadlock. However, a deadlock has been created anyway, as neither the outside yellow peg nor the outside red peg will be able to get out without a blue or yellow peg backing up.

In the other option, the next move would be a blue jump, which would put the space back in the center. If the yellow peg moves next, the red outside peg will become blocked until a yellow peg backs up. If the red peg moves next, the outside blue peg will be become blocked until a red peg backs up. If a blue peg moves next, then the outside red peg becomes blocked until either a blue or yellow peg backs up.

It thus becomes impossible to solve this game without incorporating improper moves into the algorithm. Although this only shown in example form, it should be apparent that there will be no solution with only proper moves for any board where the branches have length of at least two.

4.2 Branching Out

The next generalization is to instead have the board extend in three dimensions from a non-center position. This may happen in multiple locations on the board and would most likely (although not necessarily) mean that the board is no longer symmetrical. All positions 'to the right' of the center will still be filled with red pegs and all positions 'to the left' of center will still be filled with blue pegs. In other words, one side will still be a contiguous block of red pegs and the other side will still be a contiguous block of blue pegs. For the sake of simplicity, we will assume that the two sides are balanced sufficiently that a final board configuration will involve the space ending in between the two branches it started in so that no ambiguity about placement of the space in a particular branch may arise.

Theorem 14 *There is an algorithm for the two colored branching problem that uses only proper moves. It is based on the flowchart of the Bradley-Thomas Algorithm.*

Proof: The ideas behind the Bradley-Thomas Algorithm and the use proper moves do operate here, even if the expressed formula does not. The general state diagram can be followed for all instances leading up to a branching point. When a branching point is reached, the player arbitrarily chooses one of the branches that has not reached a finished configuration and momentarily disregards all other paths. Then the algorithm is continued as described for a linear board. When the algorithm returns to this branching point the next time, another branch is once again arbitrarily chosen.

Because the algorithm confines itself to boards of specific configurations, the different branches will see the same configuration board every time the algorithm reaches the branching point regardless of the number of iterations that have passed through each branch. Hence, when using this algorithm, the order of branches chosen is irrelevant. This also means that a new choice is presented every time a branching point is reached, and therefore there are now many solutions presented by this algorithm beyond the one conjugate alternative.

Theorem 15 *Let the problem of a two colored board with a single branch on the right side of the board be described as follows: there are N_B blue pegs, N_R red pegs, N_1 spaces in the first branch, N_2 spaces in the second branch, and $N_0 = N_R - N_1 - N_2$. Using proper moves and the general state diagram, this problem can be solved in $N_B + N_R$ steps and $N_B N_R - N_1 N_2$ jumps.*

The complexity of this algorithm is more difficult to determine than before. When a blue peg travels down a branch of red pegs, it is missing all of the pegs in the other branches of red pegs. We can no longer assume that there will be $N_R \times N_B$ hetero-colored jumps. The long strings of jumps have been shortened, as now the end of a branch of red pegs can be reached well before the summation of all the red pegs. The lower bounds of previous sections only apply if it can be proven that a backwards move will never provide three additional jumps. Although it is not yet proven, it is believed to be true.

The weight of the board has also changed. If all N_2 pegs in the second branch of the starting configuration of the board were to be picked up and placed behind the first branch, $2N_1 \times N_2$ weight would be added (consider the weight for both the blue pegs and the red pegs). The weight of this new board would also be $2N_B N_R + N_B + N_R$, which means the weight of the board with a branch in it is $2N_B N_R + N_B + N_R - 2N_1 N_2$.

Remember by our first assumption that in the final board, the space would stay inside of the branch points it from the original configuration, which gives us $N_B > N_1 + N_2$. Consider what happens when one of the first blue pegs moves to the right. It will cross N_0 red pegs and then enter a branch. When it enters a branch, it pushes a red peg into the N_0 area. The next peg will then have to cross $N_0 + 1$ red pegs before it reaches the branching point. This will continue until all of the red pegs have been flushed out of the branches. The remaining blue pegs will cross all of the red pegs. So there will be $\left(\sum_{i=N_0}^{N_R-1} i\right) + N_R(N_B - N_1 - N_2)$ peg jumps in the N_0 area. Every time a blue peg enters a branch, it flushes one red peg out of that branch. Thus, each successive blue peg entering a branch will jump one fewer time in that branch until the branch is full. So the number of jumps in the branches is $\left(\sum_{i=N_1}^0 i\right) + \left(\sum_{j=N_2}^0 j\right)$. The total number of jumps then is

$$\begin{aligned}
& N_R N_B - N_R N_1 - N_R N_2 + \frac{1}{2} N_R^2 - \frac{1}{2} N_R - \frac{1}{2} N_0^2 + \frac{1}{2} N_0 + \frac{1}{2} N_1^2 + \frac{1}{2} N_1 + \frac{1}{2} N_2^2 + \frac{1}{2} N_2 \\
& \quad N_R N_B - (N_0 + N_1 + N_2) N_1 - (N_0 + N_1 + N_2) N_2 \\
& \quad + \frac{1}{2} (N_0^2 + N_1^2 + N_2^2 + 2N_0 N_1 + 2N_0 N_2 + 2N_1 N_2) - \frac{1}{2} N_0^2 + \frac{1}{2} N_1^2 + \frac{1}{2} N_2^2 \\
& \quad N_R N_B - N_1 N_2
\end{aligned}$$

The algorithm only uses two moves: steps and hetero-colored jumps. The number of steps follows from the above.

If we assume that no optimal algorithm involves backward moves, then we may assume that following the general transitions of the Bradley-Thomas Algorithm will result in an optimal solution to this particular branching problem. Since there is player choice about which branch to take every time a branching point is reached, there are many such existing solutions. It is conjectured that the Bradley-Thomas Algorithm is optimal under all conditions of a branching problem.

References

- [1] David M. Bradley and Hugh Thomas, Peg Jumping For Fun and Profit, arXiv:math.CO/0411275 v2, April 1, 2005