

# Optimal Algorithms for Tower of Hanoi Problems with Relaxed Placement Rules\*

Yefim Dinitz and Shay Solomon

Dept. of Computer Science  
Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel  
{dinitz, shayso}@cs.bgu.ac.il

**Abstract.** We study generalizations of the Tower of Hanoi (ToH) puzzle with relaxed placement rules. In 1981, D. Wood suggested a variant, where a bigger disk may be placed *higher than* a smaller one if their size difference is less than  $k$ . In 1992, D. Poole suggested a natural disk-moving strategy, and computed the length of the shortest move sequence (algorithm) under its framework. However, other strategies were not considered, so the lower bound/optimality question remained open. In 1998, Beneditkis, Berend, and Safro were able to prove the optimality of Poole’s algorithm for the first non-trivial case  $k = 2$  only. We prove it be optimal in the general case. Besides, we prove a tight bound for the diameter of the configuration graph of the problem suggested by Wood. Further, we consider a generalized setting, where the disk sizes should not form a continuous interval of integers. To this end, we describe a finite family of potentially optimal algorithms and prove that for any set of disk sizes, the best one among those algorithms is optimal. Finally, a setting with the *ultimate* relaxed placement rule (suggested by D. Berend) is defined. We show that it is not more general, by finding a reduction to the second setting.

## 1 Introduction

The classic Tower of Hanoi (ToH) puzzle is well known. It consists of three pegs and disks of sizes  $1, 2, \dots, n$  arranged on one of the pegs as a “tower”, in decreasing, bottom-to-top size. The goal of the puzzle is to transfer all disks to another peg, placed in the same order. The rules are to move a single disk from (the top of) one peg to (the top of) another one, at each step, subject to the divine rule: to never have a larger disk above a smaller one.

The goal of the corresponding mathematical problem, which we denote by  $HT = HT_n$ , is to find a sequence of moves (“algorithm”) of a minimal length (“optimal”), solving the puzzle. We denote the pegs naturally as *source*, *target*, and *auxiliary*, while the size of a disk is referred as its name. The following algorithm  $\gamma_n$  is taught in introductory CS courses as a nice example of a recursive algorithm. It is known and easy to prove that it solves  $HT_n$  in  $2^n - 1$  disk moves, and is the unique optimal algorithm for it.

---

\* Partially supported by the Lynn and William Frankel Center for Computer Science.

- If  $n$  is 1, move disk  $n$  from *source* to *target*.
- Otherwise:
  - recursively perform  $\gamma_{n-1}(\textit{source}, \textit{auxiliary})$ ;
  - move disk  $n$  from *source* to *target*;
  - recursively perform  $\gamma_{n-1}(\textit{auxiliary}, \textit{target})$ .

In the recent two decades, various ToH type problems were considered in the mathematical literature. Many algorithms were suggested, and extensive related analysis was performed. As usual, the most difficult, far not always achievable task is showing that a certain algorithm is optimal, by providing the matching *lower bound*. A distinguished example is the Frame-Stewart algorithm (of 1941), solving the generalization of the ToH problem to four or more pegs. It is simple, and an extensive research was conducted on its behavior, since then. However, its optimality still remains an open problem; the proof of its *approximate* optimality [6] was considered a breakthrough, in 1999. This paper contributes to the difficult sub-area of the ToH research—optimality proofs.

In 1981, D. Wood [7] suggested a generalization of *HT*, characterized by the *k-relaxed placement rule*,  $k \geq 1$ : *If disk  $j$  is placed higher than disk  $i$  on the same peg (not necessarily neighboring it), then their size difference  $j - i$  is less than  $k$ .* In this paper, we refer it as the *bottleneck Tower of Hanoi problem* (following D. Poole [3]), and denote it  $BTH_n = BTH_{n,k}$ . Now, there are more than one legal way to place a given set of disks on the same peg, in general; we refer the decreasing bottom-to-top placement of all disks on the same peg as the *perfect* disk configuration. If  $k$  is 1, we arrive at the classic ToH problem.

In 1992, D. Poole [3] suggested a natural algorithm for  $BTH_n$  and declared its optimality. However, his (straightforward) proof is done under the fundamental assumption that before the last move of disk  $n$  to the (empty) target peg, *all* other  $n - 1$  disks are gathered on the spare peg. This situation is far not general, since before the last move of disk  $n$ , from some peg  $X$  to the target peg, any set of the disks  $n - 1, n - 2, \dots, n - k + 1$  may be placed below disk  $n$  on peg  $X$ .

In 1998, S. Beneditkis, D. Berend, and I. Safro [1] gave a (far not trivial) proof of optimality of Poole’s algorithm for the first non-trivial case  $k = 2$  only.

We prove it for the general case, by different techniques. Besides, we prove a tight bound for the diameter of the configuration graph of  $BTH_n$ . In other words, we find the length of the *longest* one among all shortest sequence of moves, over all pairs of initial and final configurations, up to a constant factor. We also prove that the *average* length of shortest sequence of moves, over all pairs of initial and final configurations, is the same as the above diameter for all values of  $n \leq k$  and  $n > 3k$ , up to a constant factor.

X. Chen et al. [2] considered independently a few ToH problems, including the bottleneck ToH problem. They also suggested a proof of optimality of Poole’s algorithm; it is based on another technical approach, and is not less difficult than our proof.

Further, we consider the “subset” setting, generalizing  $BTH_{n,k}$ , where the disk sizes should not form a continuous interval of integers. Here, for different disks, there are different number of bigger disks allowed to be placed above them.

For this setting, we describe the finite family of potentially optimal algorithms, and prove that for any set of disk sizes, the best one among those algorithms is optimal. Poole’s algorithm is the simplest one in this family; all of its other members do not obey the fundamental assumption made by Poole.

Finally, following a suggestion of D. Berend, we define the most general setting of relaxed placement rule, where the sets of bigger disks allowed to be placed above each disk may be arbitrary, obeying monotonicity only. We show that this “ultimate” setting is not more difficult than the “subset” setting. This is done in two ways: by generalizing the relevant proofs and by finding a reduction from the “ultimate” to the “subset” setting.

The preliminary version of this paper (with results on  $BTH_n$  only) was presented at the Workshop on the Tower of Hanoi and Related Problems (2005).

## 2 Definitions and Notation

A configuration of a disk set  $D$  is called *gathered*, if all disks in  $D$  are on the same peg. Such a configuration is called *perfect*, if  $D$  is an initial interval of naturals, and the order of disks (on a single peg) is decreasing. For any configuration  $C$  of  $D$  and any  $D' \subseteq D$ , the *restriction*  $C|_{D'}$  is  $C$  with all disks not in  $D'$  removed.

A move of disk  $m$  from peg  $X$  to peg  $Y$  is denoted by the triplet  $(m, X, Y)$ . For a disk set  $D$ , the configuration of  $D \setminus \{m\}$  is the same before and after such a move; we refer it as the *configuration of  $D \setminus \{m\}$  during  $(m, X, Y)$* .

A *packet-move*,  $P$ , of  $D$  is a sequence of moves transferring  $D$  from one peg to another. W.r.t.  $P$ , the former peg is called *source*, the latter *target*, and the third peg *auxiliary*. The *length*  $|P|$  of  $P$  is the number of moves in it. If both initial and final configurations of  $P$  are perfect, we call  $P$  a *perfect-to-perfect* (or *p.t.p.*, for short) packet-move of  $D$ .

For better mnemonics (following [2]), the entire set of disks  $[1..m]$  is divided into  $\lceil \frac{m}{k} \rceil$  blocks  $B_i = B_i(m)$ :  $B_1 = [(m-k+1)..m]$ ,  $B_2 = [(m-2k+1)..(m-k)]$ ,  $\dots$ ,  $B_{\lceil \frac{m}{k} \rceil} = [1..(m - (\lceil \frac{m}{k} \rceil - 1) \cdot k)]$ . Note that the set of disks in any block is allowed to be placed on the same peg in an arbitrary order. For any  $m \geq 1$ , let  $D_m$  denote  $[1..m]$ , and  $Small(m)$  denote  $D_m \setminus B_1(D_m)$ . In the above notion,  $BTH_n$  concerns finding the shortest perfect-to-perfect packet-move of  $D_n$ .

We say that a packet-move  $P$  *contains* a packet-move  $P'$  if  $P'$  is a subsequence of  $P$ . Several packet-moves  $P_i$ ,  $1 \leq i \leq r$ , contained in  $P$ , are called *disjoint* if the last move in  $P_i$  precedes the first move in  $P_{i+1}$ , for each  $1 \leq i \leq r-1$ .

For any sequence of moves  $S$  of  $D$  and any  $D' \subseteq D$ , the *restriction of  $S$  to  $D'$* , denoted by  $S|_{D'}$ , is the result of omission from  $S$  all moves of disks not in  $D'$ . Note that any restriction of a legal sequence of moves is legal as well, and a restriction of a packet-move to  $D'$  is a packet-move of  $D'$ . Clearly, if  $D$  is partitioned into  $D'$  and  $D''$ , then  $|P| = |P|_{D'} + |P|_{D''}$ .

Consider a sequence of moves,  $S$ , containing two *consequent* moves of the same disk:  $(i, X, Y)$  and  $(i, Y, Z)$ . Their replacement by the single move  $(i, X, Z)$ , if  $X \neq Z$ , or the deletion of both, if  $X = Z$ , is called a *pruning* of  $S$ . We denote by  $Prune(S)$  the result of all possible prunings, at  $S$ ; it is easy to see that

such a result is independent on the order of particular prunings and is legal, so  $Prune(S)$  is well defined.

### 3 The Shortest “Somehow” Packet-Move

In this section, we consider general, not p.t.p. packet-moves.

In the attempts of S. Beneditkis, D. Berend, and I. Safro [1] to solve  $BTH_n$ , much like in [3], another related problem of “moving somehow”, under the  $k$ -relaxed placement rule arose: *To move  $m$  disks  $[1..m]$ , placed entirely on one peg, to another peg, in any order.* For solving it, the following algorithm  $\beta_m = \beta_m(source, target)$  was presented:

- If  $m$  is at most  $k$ , move all disks from  $source$  to  $target$  one by one.
- Otherwise:
  - recursively perform  $\beta_{m-k}(source, auxiliary)$ ;
  - move disks  $[(m - k + 1)..m]$  from  $source$  to  $target$  one by one;
  - recursively perform  $\beta_{m-k}(auxiliary, target)$ .

Notice that the sequence  $\beta_n$  is similar to  $\gamma_n$ , but deals with blocks, instead of single disks. When  $\beta_m$  is applied to the perfect disk configuration, it is legal, and results in the configuration, different from the perfect one by the *increasing* order of disks in  $B_1(m)$ . When  $\beta_m$  is applied to the latter configuration, it is legal and results in the perfect configuration.

For  $k = 2$ , it is proved in [1] that no sequence of moves shorter than  $\beta_m$  can transfer  $m$  disks from one peg to another. We generalize this result for general  $k$ . Let  $b_m$  denote the length of  $\beta_m$ . By definition of  $\beta_m$ ,  $b_m = m$  if  $m \leq k$ , and  $b_m = 2b_{m-k} + k$ , otherwise. In [3], this recurrence relation is shown to imply the explicit formula  $b_n = k \cdot (2^{\lfloor \frac{n}{k} \rfloor} - 1) + r \cdot 2^{\lfloor \frac{n}{k} \rfloor} = (k + r) \cdot 2^{\lfloor \frac{n}{k} \rfloor} - k$ , where  $r = n \bmod k$ . As a consequence,  $b_n - b_{n-1} = 2^{\lceil n/k \rceil - 1}$ ; in particular, the sequence  $\{b_i\}$  is *strictly* monotonous.

Note that during a move  $(m, X, Y)$ , all disks in  $Small(m)$  are on the spare peg  $Z \neq X, Y$ . As a corollary, holds:

**Fact 1.** *If a sequence of moves  $S$  begins from a configuration, where disk  $m$  and  $Small(m)$  are on  $X$ , and finishes at a configuration, where they are on  $Y$ ,  $X \neq Y$ , then it contains two disjoint packet-moves of  $Small(m)$ : one (from  $X$ ) before the first move of disk  $m$  in  $S$  and another (to  $Y$ ) after its last move.*

**Theorem 2.** *Under the rules of  $BTH_m$ , the length of any packet-move of  $D_m$  is at least  $b_m$ . (proof is omitted)*

### 4 Optimal Solution to $BTH_n$

Let us describe algorithm  $\alpha_n$ , presented first in [3] and independently afterwards in [1], solving  $BTH_n$ :

- perform  $\beta_{n-1}(\text{source}, \text{auxiliary})$ ;
- move disk  $n$  from *source* to *target*;
- perform  $\beta_{n-1}(\text{auxiliary}, \text{target})$ .

For  $k = 2$ , it is proved in [1] that  $\alpha_n$  is an optimal solution to  $BTH_n$ . In this section, we generalize this result to the case of general  $k$ .

Let  $a_n$  denote the length of  $\alpha_n$ . The explicit formula for  $a_n$  (established in [3]) is implied straight-forwardly by that for  $b_{n-1}$ :  $a_n = 2(r+k) \cdot 2^{\lfloor \frac{n-1}{k} \rfloor} - 2k + 1$ , where  $r = (n-1) \bmod k$ . The following theorem implies that  $\alpha_n$  is an optimal algorithm that solves the puzzle.

**Theorem 3.** *Any p.t.p. packet-move of  $D_n$  solving  $BTH_n$  is of length at least  $a_n$ .*

The rest of this section is devoted to the proof of this Theorem. Recall that  $Small(m) = [1..(m-k)]$ ,  $B_1(m) = [(m-k+1)..m]$ . In our study of a packet-move  $P$  of  $D_m$ ,  $m > k$ , we usually consider separately  $P|_{Small(m)}$  and  $P|_{B_1(m)}$ . The reason is that any move  $(m, X, Y)$  defines completely the placement of  $Small(m)$ —on the spare peg  $Z$ ,—while the placement of disks in  $B_1(m) \setminus \{m\}$  may be arbitrary, during such a move. For the analysis of  $P|_{B_1(m)}$ , we use the following statement:

**Fact 4.** *Any p.t.p. packet-move  $P$  of  $D_m$  contains at least two moves of any disk  $i$ ,  $i \neq m$ : at least one before the first move of disk  $m$  and at least one after its last move. Hence, for any  $D \subseteq D_{m-1}$ , holds  $|P|_D \geq 2|D|$ .*

#### 4.1 Case 1: Disk $n$ Never Moves to the Auxiliary Peg

We call a move of disk  $m$ , in a packet-move of  $D_m$ , *distinguished* if it is between *source* and *target* and disk  $m-1$  is at *auxiliary*, during that move.

**Lemma 1.** *Any packet-move  $P$  of  $D_m$ , which preserves the initial order between disks  $m$  and  $m-1$ , and such that disk  $m$  moves only between the source and target pegs, contains a distinguished move of disk  $m$ .*

*Proof.* Define  $P' := Prune(P|_{\{m-1, m\}})$ . Note that  $P'$  preserves the initial order between disks  $m$  and  $m-1$ , since  $P$  preserves it. We will study  $P'$ , since a move of  $m$  is distinguished in  $P'$  if and only if it is distinguished in  $P$ . By the definition of  $Prune$ , moves of disks  $m$  and  $m-1$  must interchange, in  $P'$ .

*Claim.* Under the conditions of the Lemma, the first move of disk  $m-1$  should be  $(m-1, \text{source}, \text{auxiliary})$ . (the proof is omitted)

Based on this Claim, we consider the possible scenarios of  $P'$ . Assume that disk  $m-1$  is initially above disk  $m$ . Then, necessarily, the first two moves are:  $(m-1, \text{source}, \text{auxiliary})$ ,  $(m, \text{source}, \text{target})$ ; the latter one is distinguished. Assume that disk  $m-1$  is initially below disk  $m$ . Then, necessarily, the first three moves are:  $(m, \text{source}, \text{target})$ ,  $(m-1, \text{source}, \text{auxiliary})$ ,  $(m, \text{target}, \text{source})$ ; the latter one is distinguished. Hence, also  $P$  contains a distinguished move.  $\square$

**Proposition 1.** *For any  $m > k + 1$  and any packet-move  $P$  of  $D_m$  preserving the initial order between disks  $m$  and  $m - 1$ , s.t. disk  $m$  moves only between the source and target pegs,  $P$  contains four disjoint packet-moves of  $Small(m - 1)$ .*

*Proof.* Let us consider the first distinguished move of disk  $m$  during  $P$ ; it exists by Lemma 1. Clearly, during such a move, all disks in  $Small(m - 1)$  are on *auxiliary*, together with disk  $m - 1$ ; by the  $k$ -relaxed placement rule, they are placed above disk  $m - 1$ . Hence, by Fact 1, the parts of  $P$  before and after that move contain two disjoint packet-moves of  $Small(m - 1)$  each. Altogether, there are four disjoint packet-moves of  $Small(m - 1)$ , in  $P$ .  $\square$

**Corollary 1.** *The length of any p.t.p. packet-move of  $D_n$ , such that disk  $n$  moves only between the source and target pegs, is at least  $a_n$ .*

*Proof.* Let  $P$  be a packet-move as in the Corollary. By Fact 4,  $|P|_{B_1(n-1)}| \geq 2 \cdot |B_1(n-1)|$ . If  $n \leq k + 1$ ,  $|P| = |P|_{B_1(n-1)}| + |P|_{\{n\}}| \geq 2(n-1) + 1 = 2 \cdot b_{n-1} + 1 = a_n$ . Otherwise, by Proposition 1 and Theorem 2,  $|P| = |P|_{Small(n-1)}| + |P|_{B_1(n-1)}| + |P|_{\{n\}}| \geq 4 \cdot b_{n-k-1} + 2k + 1 = 2 \cdot b_{n-1} + 1 = a_n$ .  $\square$

## 4.2 Case 2: Disk $n$ Moves to the Auxiliary Peg

**Lemma 2.** *If  $m > k$  and a packet-move  $P$  of  $D_m$  contains a move of disk  $m$  to auxiliary, then  $P|_{Small(m)}$  contains three disjoint packet-moves of  $Small(m)$ . (the proof is omitted.)*

Following is the central statement of this section.

**Proposition 2.** *For any  $m, l \geq 0$ , let  $P$  be a p.t.p. packet-move of  $D_m$  containing  $2l + 1$  disjoint packet-moves of  $D_m$ . Then,  $|P| \geq 2l \cdot b_m + 2 \cdot b_{m-1} + 1 = 2l \cdot b_m + a_m$ , and this bound is tight.*

*Proof.* It can be easily proved that any packet-move as in the Proposition may be divided into  $t$ ,  $t \geq 2l + 1$ , packet-moves of  $D_m$ . If  $t > 2l + 1$ , then, by Theorem 2 and the strict monotonicity of  $(b_i)$ ,  $|P| \geq (2l + 2) \cdot b_m \geq 2l \cdot b_m + 2 \cdot b_{m-1} + 2$ . We henceforth assume that  $P$  is divided into  $2l + 1$  packet-moves of  $D_m$ .

We prove by a complete induction on  $m$ , for all  $l$ . *Basis:*  $m \leq k$ . (Note that in this basic case, the disks may be placed at pegs in an arbitrary order).

**Lemma 3.** *In the case  $m \leq k$ , the length of any p.t.p. packet-move  $P$  of  $D_m$ , divided into  $2l + 1$  packet-moves of  $D_m$ , is at least  $(2l + 2)m - 1$ .*

*Proof.* We call a disk in  $D_m$  *expensive* w.r.t. some packet-move  $P'$ , if it moves to the auxiliary peg during  $P'$ . Let us prove that there could be at most one disk, which is not expensive w.r.t. any one out of the  $2l + 1$  packet-moves as in the Lemma. Assume to the contrary that there are two such disks,  $i$  and  $j$ . Then, after each packet-move, their order is reversed. Since there is an odd number of packet-moves, the order of  $i$  and  $j$  at the final configuration is inverse to that in the initial configuration,—a contradiction.

It follows that either all  $m$  disks or some  $m - 1$  disks make at least  $2l + 2$  moves each, while the remained disk, if any, makes at least  $2l + 1$  moves. Altogether, at least  $(2l + 2)m - 1$  moves are made.

As a corollary, since  $b_r = r$ , for any  $r$ ,  $1 \leq r \leq k$ , holds  $|P| \geq (2l + 2)m - 1 = 2lm + 2(m - 1) + 1 = 2l \cdot b_m + 2 \cdot b_{m-1} + 1$ .

*Induction step:  $m > k$ .* We suppose that the claim holds for all lesser values of  $m$  and for all  $l$ , and prove it for  $m$  and all  $l$ .

Note that at the initial configuration of  $P$ , as well as at its final configuration, disk  $m$  is placed below disk  $m - 1$ . Since  $P$  is a composition of an odd number of disjoint packet-moves of  $D_m$ , there exists a packet-move among them, henceforth denoted by  $\tilde{P}$ , which preserves the order between disks  $m$  and  $m - 1$ . We bound  $|P|_{Small(m)}$  separately, for two complementary types of  $\tilde{P}$ .

*Case 1: During  $\tilde{P}$ , disk  $m$  never moves to the auxiliary peg.* By Proposition 1,  $\tilde{P}$  contains four disjoint packet-moves of  $Small(m - 1)$ . We notice that  $\tilde{P}$  contains at least two moves of disk  $m - k$ : at least one before the first move of disk  $m$ , and at least one after its last move. By Theorem 2,  $|\tilde{P}|_{Small(m)} \geq 4 \cdot b_{m-k-1} + 2 = 2 \cdot b_{m-1} - 2k + 2$ . By Corollary 3, the other  $2l$  packet-moves of  $D_m$  contain two disjoint packet-moves of  $Small(m)$  each. Hence, their total length is at least  $4l \cdot b_{m-k} = 2l(b_m - k)$ . Therefore,  $|P|_{Small(m)} \geq 2l \cdot b_m - 2lk + 2 \cdot b_{m-1} - 2k + 2 = 2l \cdot b_m + 2 \cdot b_{m-1} - (2l + 2)k + 2$ . We denote this value by  $N$ .

*Case 2:  $\tilde{P}$  contains a move of disk  $m$  to the auxiliary peg.* By Lemma 2,  $|\tilde{P}|_{Small(m)}$  contains three disjoint packet-moves of  $Small(m)$ . By Corollary 3, the other  $2l$  packet-moves of  $D_m$  contain two disjoint packet-moves of  $Small(m)$  each. Thus,  $P|_{Small(m)}$  contains  $4l + 3$  disjoint packet-moves of  $Small(m)$ . By the induction hypothesis,  $|P|_{Small(m)} \geq (4l + 2) \cdot b_{m-k} + 2 \cdot b_{m-k-1} + 1 \geq 4l \cdot b_{m-k} + 4 \cdot b_{m-k-1} + 3 = 2l \cdot b_m + 2 \cdot b_{m-1} - (2l + 2)k + 3 = N + 1$  (the second inequality holds since the sequence  $(b_i)$  is strictly monotonous).

By Lemma 3,  $|P|_{B_1(m)}$  is at least  $(2l + 2)k - 1$ . So,  $|P| = |P|_{Small(m)} + |P|_{B_1(m)} \geq N + |P|_{B_1(m)} \geq 2l \cdot b_m + 2 \cdot b_{m-1} + 1$ , as required.

The bound is tight, since the sequence composed from  $2l \beta_m$  and one  $\alpha_m$ , in this order, is a p.t.p. packet-move of length equal to this bound.  $\square$

Now, Theorem 3 follows from Proposition 2 with  $l = 0$  and  $m = n$ .

The difference of 1 between bounds at Cases 1 and 2, together with the tightness of the bound of Case 1, implies:

- Corollary 2.**
1. No optimal algorithm for  $BTH_n$  contains a move of disk  $n$  to the auxiliary peg.
  2. Any optimal algorithm for  $BTH_n$  contains just a single move of disk  $n$ , from the source peg to the target peg.
  3. The only difference of an arbitrary optimal algorithm for  $BTH_n$  from  $\alpha_n$  could be in choosing another optimal algorithms, for the two included optimal “somehow” packet-moves of  $D_{n-1}$ , instead of  $\beta_{n-1}$ .

## 5 Diameter of the Configuration Graph of $BTH_n$

While the shortest perfect-to-perfect sequence of moves had been already found, we do not know what is such a sequence for transforming an arbitrary (legal)

configuration to another given (legal) one, and what is its length. We study, what is the length of the *longest* one among all shortest sequences of moves, over all pairs of initial and final configurations. In other words, what is the diameter, denoted by  $D(n, k)$ , of the directed graph of all the configurations of the disks in  $[1..n]$ , under the  $k$ -relaxed placement rule?

The proof of the following theorem is omitted.

**Theorem 5.**

$$Diam(n, k) = \begin{cases} \Theta(n \cdot \log n) & \text{if } n \leq k \\ \Theta(k \cdot \log k + (n - k)^2) & \text{if } k < n \leq 2k \\ \Theta(k^2 \cdot 2^{\frac{n}{k}}) & \text{if } n > 2k . \end{cases}$$

Another question is what is the *average* length of shortest sequences of moves, over all pairs of initial and final configurations, denoted by  $Avg(n, k)$ . The following theorem states that it is the same as  $D(n, k)$  for all values of  $n \leq k$  and  $n > 3k$ , up to a constant factor (its proof is omitted).

**Theorem 6.**

$$Avg(n, k) = \begin{cases} \Theta(n \cdot \log n) & \text{if } n \leq k \\ \Theta(k^2 \cdot 2^{\frac{n}{k}}) & \text{if } n > 3k . \end{cases}$$

## 6 “Subset” Setting

In this section, the previous study is generalized to  $BTH_D = BTH_{D,k}$ , where the disk placement is still subject to the  $k$ -relaxed rule, but the disk set  $D$  is an *arbitrary* set, not necessarily a contiguous interval of naturals.

### 6.1 Preliminaries

Let us generalize some of the definitions given for  $BTH$ . For a non-empty disk set  $D$ ,  $|D| = n$ , let us denote the disks of maximal and minimal size by  $max(D)$  and  $min(D)$ , respectively, and by  $s(D)$  the “stretch”  $max(D) - min(D) + 1$ . The  $i$ th biggest disk in  $D$  is denoted by  $D(i)$ . We define  $D^- = D \setminus \{max(D)\}$ . For a disk set  $D$ , its division into blocks is as follows:  $B_1 = B_1(D) = D \cap [(max(D) - k + 1)..max(D)]$ , and for any  $i > 1$ , s.t.  $\bigcup_{j < i} B_j(D) \neq D$ ,  $B_i = B_i(D) = B_1(D \setminus \bigcup_{j < i} B_j(D))$ . The size of each block is between 1 and  $k$ , and the number of blocks  $\#(D) = \#_k(D)$  is between  $\lceil n/k \rceil$  and  $\min\{n, s(D)/k\}$ . We define  $Small(D) = \bigcup_{j \geq 2} B_j(D)$ . We generalize the optimal “moving somehow” algorithm  $\beta_m$  to the optimal algorithm  $\beta_D = \beta_D(source, target)$ :

- If  $s(D) \leq k$ , move all disks from *source* to *target* one by one.
- Otherwise:
  - recursively perform  $\beta_{Small(D)}(source, auxiliary)$ ;
  - move disks in  $B_1(D)$  from *source* to *target* one by one;
  - recursively perform  $\beta_{Small(D)}(auxiliary, target)$ .

Denote by  $b_D$  the length of  $\beta_D$ . By definition of  $\beta_D$ , holds  $b_D = n$ , if  $s(D) \leq k$ , and  $b_D = 2 \cdot b_{Small(D)} + |B_1(D)|$ , otherwise. It can be proved that the value  $b_D$  is *strictly* inclusion-wise monotonous in  $D$ .

The proofs of the following statements are similar to those for the corresponding ones for  $BTH_n$ .

**Theorem 7.** *Any packet-move of  $D$  is of length at least  $b_D$ .*

**Corollary 3.** *For any packet-move  $P$  of  $D$ ,  $P|_{Small(D)}$  contains two disjoint packet-moves of  $Small(D)$ .*

**Fact 8.** *Any p.t.p. packet-move  $P$  of  $D$  contains at least two moves of any disk  $i$ ,  $i \neq \max(D)$ . Thus, for any  $D' \subseteq D^-$ , holds  $|P|_{D'} \geq 2|D'|$ .*

## 6.2 The Set of Potentially Optimal Algorithms

Let us generalize algorithm  $\alpha_n$  to  $\alpha_D$ :

- perform  $\beta_{D^-}(source, auxiliary)$ ;
- move disk  $\max(D)$  from *source* to *target*;
- perform  $\beta_{D^-}(auxiliary, target)$ .

We denote  $\alpha_D^0 = \alpha_D$ , and define  $\alpha_D^i$ ,  $1 \leq i \leq \#(D) - 1 \leq n - 1$ , as follows:

- perform  $\beta_{Small(D)}(source, target)$ .
- move disks in  $B_1(D)$  from *source* to *auxiliary* one by one;
- perform  $\beta_{Small(D)}(target, source)$ ;
- move disks in  $B_1(D)$  from *auxiliary* to *target* one by one;
- recursively perform  $\alpha_{Small(D)}^{i-1}(source, target)$ .

We define  $a_D^i = |\alpha_D^i|$ . Clearly,  $a_D^i = a_{Small(D)}^{i-1} + 2 \cdot b_{Small(D)} + 2 \cdot |B_1(D)|$ . Let us denote  $\bar{a}_D = \min_{0 \leq i \leq \#(D)-1} a_D^i$ .

It is easy to show, by induction on  $i$ , that each one of the algorithms  $\alpha_D^i$ ,  $0 \leq i \leq \#(D) - 1$ , solves  $BTH_D$ : the first and third items move out and return  $Small(D)$  to *source*, the second and fourth items move  $B_1(D)$  from *source* to *target*, in the right order, and the fifth item does the same for  $Small(D)$ . The following Theorem shows that the length of the shortest sequence of moves solving  $BTH_D$  is  $\bar{a}_D$ .

**Theorem 9.** *The best one out of the  $\#(D)$  algorithms  $\alpha_D^i$  is optimal for  $BTH_D$ .*

The proof of this Theorem has the same structure as that of Theorem 3. It is based on the following statements, whose proofs are similar to those of the corresponding ones for  $BTH_n$ . The difficulty of proofs remains approximately the same, except for that of Proposition 4, which becomes more involved.

**Proposition 3.** *Let  $P$  be a packet-move of disks  $D'$ , preserving the initial order between disks  $\max(D')$  and  $\max(D'^-)$ , and such that disk  $\max(D')$  moves only between the source and target pegs. Then,  $P|_{Small(D'^-)}$  contains four disjoint packet-moves of  $Small(D'^-)$ .*

**Corollary 4.** *If  $P$  is a p.t.p. packet-move of  $D$ , such that disk  $\max(D)$  moves only between the source and target pegs, then  $|P| \geq a_D^0$ .*

**Lemma 4.** *If a packet-move  $P$  of  $D'$  contains a move of disk  $\max(D')$  to the auxiliary peg, then  $P|_{\text{Small}(D')}$  contains three disjoint packet-moves of  $\text{Small}(D')$ .*

**Proposition 4.** *Let  $P$  be a p.t.p. packet-move of  $D'$ , containing  $2l + 1$  disjoint packet-moves of  $D'$ . Then,  $|P| \geq 2l \cdot b_{D'} + \bar{a}_{D'}$ .*

*Proof.* The proof is made by induction on  $\#(D')$ , for all  $l$ . The basis of the induction is similar to that of Proposition 2. At the induction step, the case analysis is more involved. As in the proof of Proposition 2, we point our finger at a "guilty" packet-move, denoted by  $\tilde{P}$ , among the  $2l + 1$  disjoint packet-moves contained in  $P$ . Due to space constraints, we show only the case satisfying the following conditions: 1.  $\tilde{P}$  contains a move of disk  $\max(D')$  to the auxiliary peg; 2.  $|B_1(D')| \geq 2$ ; 3.  $|P|_{B_1(D')} = (2l + 2)|B_1(D')| - 1$ . When considering packet-moves forming  $P$ , we will use the names *source*, *target*, and *auxiliary* w.r.t. the currently considered packet-move.

It is easy to prove that there exists a disk  $d \neq \max(D')$  in  $B_1(D')$  that does not move to *auxiliary* during  $P$ , and all disks in  $B_1(D') - \{d\}$  move to *auxiliary* exactly once during  $P$ . In any packet-move other than  $\tilde{P}$ ,  $d$  and  $\max(D')$  reverse their order. Since  $P$  is a composition of an odd number of disjoint packet-moves of  $D'$ ,  $d$  and  $\max(D')$  preserve their order in  $\tilde{P}$ . In the initial configuration of  $\tilde{P}$ ,  $\max(D')$  must be placed higher than  $d$ , for otherwise they would reverse their order. The first move of  $\max(D')$  is to *auxiliary*. Before that move, all disks in  $\text{Small}(D')$  have been moved to *target*. Before the move of disk  $d$  to *target*, all disks in  $\text{Small}(D')$  have been moved to *auxiliary* above disk  $\max(D')$ , otherwise disk  $\max(D')$  would not be able to move above disk  $d$ , on *target*. Before disk  $\max(D')$  moves from *auxiliary* to *target*, all disks in  $\text{Small}(D')$  have been moved to *source*, and are due to be moved to *target* at some point. In total,  $\tilde{P}|_{\text{Small}(D')} \geq 4 \cdot b_{\text{Small}(D')}$ . Thus,  $P|_{\text{Small}(D')} \geq (4l + 4) \cdot b_{\text{Small}(D')}$ . We recall that  $|P|_{B_1(D')} = (2l + 2)|B_1(D')| - 1$ . Altogether,  $|P| = |P|_{\text{Small}(D')} + |P|_{B_1(D')} \geq (4l + 4) \cdot b_{\text{Small}(D')} + (2l + 2)|B_1(D')| - 1 \geq$  (details are omitted)  $2lb_{D'} + a_{D'}$ .

### 6.3 Tightness and Other Related Issues

Recall that for the case  $k = 1$ , for any disk set  $D$ ,  $\alpha_D = \alpha_D^0$  is the shortest p.t.p. packet-move of  $D$ . It can be proven that it is the unique optimal algorithm.

**Proposition 5.** *For any  $k \geq 2$ ,  $n$ , and  $\lceil \frac{n}{k} \rceil \leq p \leq \lceil \frac{n}{2} \rceil$ , there exists a set  $D$ , with  $|D| = n$  and  $\#(D) = p$ , s.t. for any  $0 \leq i, j \leq p - 1$ ,  $|\alpha_D^i| = |\alpha_D^j|$ .  
(the proof is omitted)*

**Theorem 10.** *For any  $k \geq 2$ ,  $n$ ,  $\lceil \frac{n-1}{k} \rceil + 1 \leq p \leq \lceil \frac{n}{2} \rceil$ , and  $0 \leq j \leq p - 2$ , there exists a set  $D$ , with  $|D| = n$  and  $\#(D) = p$ , s.t. for each  $l \neq j$ ,  $0 \leq l \leq p - 1$ , holds  $|\alpha_D^j| < |\alpha_D^l|$ .  
(the proof is omitted)*

**Proposition 6.** *For any  $k \geq 2$ ,  $n$ ,  $\lceil (n-4)/k \rceil + 3 \leq p \leq n-1$ , and  $0 \leq j \leq p-1$ , there exists a set  $D$ , with  $|D| = n$  and  $\#(D) = p$ , s.t. for each  $l \neq j$ ,  $0 \leq l \leq p-2$ , holds  $|\alpha_D^j| > |\alpha_D^l|$ . (the proof is omitted)*

We define the partial order “denser or equal”  $\preceq_d$  between two integer sets,  $D_1$  and  $D_2$ , of an equal cardinality,  $n$ , by  $D_1 \preceq_d D_2 \Leftrightarrow \forall 1 \leq i \leq n-1 : D_1(i+1) - D_1(i) \leq D_2(i+1) - D_2(i)$ .

**Proposition 7.** *1. For any  $D_1 \preceq_d D_2$ , holds  $\bar{a}_{D_1} \leq \bar{a}_{D_2}$ .  
2. For any  $D$ , holds  $a_{|D|} \leq \bar{a}_D \leq 2^{|D|} - 1$ .  
(the proof is omitted)*

In this paper, we considered a fixed value of  $k$ . The following statement shows that when  $k$  grows,  $BTH_D$  becomes more general.

**Proposition 8.** *For any two values  $k_1 < k_2$  and any set  $D_1$ , there exists a set  $D_2$ ,  $|D_2| = |D_1|$ , such that the instances  $BTH_{D_1, k_1}$  and  $BTH_{D_2, k_2}$  are equivalent in the following sense. The natural bijection  $D_1(i) \leftrightarrow D_2(i)$  induces bijections between the families of legal configurations of  $BTH_{D_1, k_1}$  and  $BTH_{D_2, k_2}$  and thus between their families of legal sequences of moves. (the proof is omitted)*

## 7 Setting with the Ultimate Placement Rule

In this paper, we study shortest p.t.p. packet-moves under different types of relaxed placement rules. In this section, we consider the problem  $BTH_f$ , defined by the *ultimate* in a sense, relaxed placement rule, suggested by D. Berend. There are  $n$  disks  $1, 2, \dots, n$ , and an (arbitrary) monotonous non-decreasing function  $f : [1..n] \rightarrow [1..n]$ , s.t.  $f(i) \geq i$ , for all  $1 \leq i \leq n$ . For each disk  $i$ , only disks of size at most  $f(i)$  may be placed higher than  $i$ .

We consider the following restrictions natural: if some disk may be placed above  $i$ , then any smaller disk may also be placed above  $i$ , and if  $i$  may be placed above some disk, then  $i$  may also be placed above any larger disk. Besides, any disk smaller than  $i$  may be placed above  $i$ , since this is so at the initial, perfect configuration. Therefore, the free choice of  $f$  makes the above rule ultimate.

We state that  $BTH_f$  is equivalent to  $BTH_D$ . For this, let us first show that  $BTH_f$  is at least as general as  $BTH_D$ , by a reduction. For any instance of  $BTH_D$ , we set  $n = |D|$  and consider the bijection  $g_D : [1..n] \rightarrow D$ , s.t.  $g_D(i) = D(i)$ . It induces naturally a function  $f$  as above, and thus an instance of  $BTH_f$ . It is easy to see that  $g_D$  and  $g_D^{-1}$  induce a bijection between the families of configurations at the two problem instances, which keeps the legality; thus, the two instances are equivalent.

Proving that  $BTH_D$  is as general as  $BTH_f$  becomes more complicated. One way is as follows. We define algorithms  $\alpha_f^i$  and the value  $\bar{a}_f$ , similarly to  $\alpha_D^i$  and  $\bar{a}_D$ . First, we observe that Theorem 11 below, analogous to Theorem 9, and certain analogues of Theorem 7, Proposition 5, Theorem 10, and Proposition 6 are valid, since the proofs can be generalized almost straightforwardly.

**Theorem 11.** *The best one out of the algorithms  $\alpha_f^i$  is optimal for  $BTH_f$ .*

Besides, we have a reduction, which for any instance of  $BTH_f$ , constructs an instance of  $BTH_D$ , which is equivalent to it via the bijection  $g_D$ . The size of the instance constructed in this reduction is polynomial in  $n$ .

Hence, the theory of  $BTH_D$  can be extended to  $BTH_f$ .

**Acknowledgment.** Authors thank Daniel Berend for his suggestion to consider the question on the optimal algorithm for the bottleneck Tower of Hanoi problem, and for his constant willingness to help.

## References

1. S. Beditkis and I. Safro. Generalizations of the Tower of Hanoi Problem. Final Project Report, supervised by D. Berend, Dept. of Mathematics and Computer Science, Ben-Gurion University, 1998.
2. X. Chen, B. Tian, and L. Wang. Santa Claus' Towers of Hanoi. Manuscript, 2005.
3. D. Poole. The Bottleneck Towers of Hanoi Problem. *J. of Recreational Math.* **24** (1992), no. 3, 203-207.
4. P.K. Stockmayer. Variations on the Four-Post Tower of Hanoi Puzzle. *CONGRESSUS NUMERANTIUM* **102** (1994), 3-12.
5. P.K. Stockmayer. The Tower of Hanoi: A Bibliography. (1998). Available via [http://www.cs.wm.edu/~pkstoc/h\\_Papers.html](http://www.cs.wm.edu/~pkstoc/h_Papers.html).
6. M. Szegedy, In How Many Steps the  $k$  Peg Version of the Towers of Hanoi Game Can Be Solved?, *Symposium on Theoretical Aspects of Computer Science* **1563** (1999), 356.
7. D. Wood. The Towers of Brahma and Hanoi revisited. *J. of Recreational Math.* **14** (1981-1982), no. 1, 17-24.