Optimality of an Algorithm Solving the Bottleneck Tower of Hanoi Problem *

Yefim Dinitz † Shay Solomon †

Abstract

We study the Bottleneck Tower of Hanoi puzzle posed by D. Wood in 1981. There, a relaxed placement rule allows a larger disk to be placed *higher* than a smaller one if their size difference is less than a pregiven value k. A shortest sequence of moves (optimal algorithm) transferring all the disks placed on some peg in decreasing order of size, to another peg in the same order is in question. In 1992, D. Poole suggested a natural disk-moving strategy for this problem, and computed the length of the shortest move sequence under its framework. However, other strategies were overlooked, so the lower bound/optimality question remained open. In 1998, Benditkis, Berend, and Safro proved the optimality of Poole's algorithm for the first non-trivial case $k = 2$. We prove Poole's algorithm to be optimal in the general case.

1 Introduction

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

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

†Department of Computer Science, Ben-Gurion University of the Negev, POB 653, Beer-Sheva 84105, Israel. E-mail: {dinitz,shayso}@cs.bgu.ac.il

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

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

Over the last two decades, there has been an increasing interest in Tower of Hanoi problems (see bibliography in [7]). As usual, the most difficult task is to show that a suggested algorithm is optimal. A distinguished example is the Frame-Stewart algorithm (1941), solving the generalization of the ToH problem to four or more pegs. It is simple, and extensive research was conducted on its behavior. However, its optimality remains an open question; in 1999, the algorithm was proved to be "approximately optimal" [8], which was considered a breakthrough.

In 1981, Wood [9] suggested a generalization of TH, 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 to it as the *Bottleneck Tower of Hanoi problem* (following Poole [5]) and denote it $BTH_n = BTH_{n,k}$. If k is 1, we arrive at the classical ToH problem, so we restrict our attention to the case $k \geq 2$. Note that this rule offers more than one legal way to place a given set of disks on the same peg. Refer to Figure 1 for an illustration.

In 1992, Poole [5] suggested a natural algorithm for solving BTH_n and declared its optimality. However, his (straightforward) optimality proof is made under the naive assumption that *disk n moves just once*, thus leaving a substantial gap. To be more precise, he implicitly assumed that before the last move of disk n to the (empty) target peg, *all* other $n - 1$ disks are gathered on the spare peg. His proof is therefore incomplete, since during

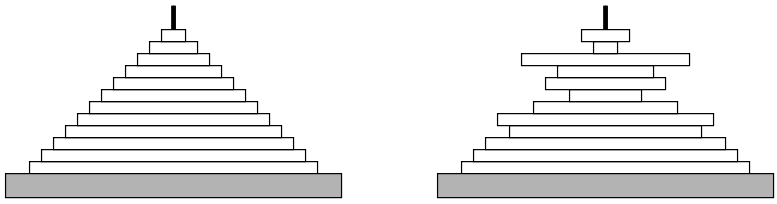

Figure 1: On the left, the decreasing arrangement of the disk set [1..12] is shown. On the right, the arrangement (12, 11, 10, 8, 9, 6, 3, 5, 4, 7, 1, 2) is presented; it is legal for $BTH_{12,k}$ in the case $k \geq 5$ only, as disk 7 is placed higher than disk 3.

that move the $k-1$ disks $n-k+1, \ldots, n-2, n-1$ may be dispersed arbitrarily on *source* and *auxiliary*.

Benditkis, Berend, and Safro have reinvented Poole's algorithm for BTH_n and gave a (rather involved) proof of its optimality for the first non-trivial case $k = 2$ [1].

We prove optimality of Poole's algorithm in the general case.

Remark: Chen et al. [2] considered a few ToH problems independently, including BTH_n. They also reinvented Poole's algorithm and suggested a proof of its optimality, based on another technical approach.

A preliminary version of this paper is presented in [3].

2 Definitions and Notation

A *configuration* is a specification of the arrangement of all the disks in consideration on the three pegs; in this paper we restrict our attention to configurations *legal* w.r.t. the k-relaxed placement rule. A configuration is called *gathered* if all disks are gathered on a single peg; if they are also in decreasing order of size, we call it *perfect*.

Let D be a disk set and $D' \subseteq D$. For any configuration C of D, the *restriction of* C *to* D', denoted $C|_{D'}$, is the configuration of D' obtained by removing all disks in $D \setminus D'$ from C. Similarly, for any sequence of moves S of D, its *restriction to* D', denoted $S|_{D'}$, is the result of omitting all moves of disks in $D \setminus D'$ from S. Note that restrictions of legal configurations and move sequences are legal as well. If C_1 is the initial configuration and C_2 is the final one with respect to S, then $S|_{D'}$ is a move sequence transforming $C_1|_{D'}$ to $C_2|_{D'}$. The *length* $|S|$ of a move sequence S is the number of moves in

it; to avoid ambiguity in formulae, the length of $S|_{D'}$ is denoted $|(S|_{D'})|$. We say that a move sequence S of D *contains* a move sequence S' of D' if $D' = D$ and S' is an interval of S or if $D' \subset D$ and S' is the restriction of an interval of S to D'. Move sequences contained in S are called *disjoint* if the respective intervals are disjoint. We say that a move sequence $S = S_1||S_2||\ldots||S_r$ is *composed* of the move sequences $S_1, S_2, \ldots, S_r$.

A move sequence P of a disk set D is called a *packet-move of* D if it transfers the entire set D from an initial gathered configuration on one peg to a final gathered configuration on another peg. With respect to P, the former peg is called $source = source(P)$, the latter $target = target(P)$, and the third $auxiliary = auxiliary(P)$. Clearly, for any $D' \subseteq D$, the restriction of a packet-move of D to D' is a packet-move of D'. Note that if D is partitioned into D' and D'', then $|P| = |(P|_{D'})| + |(P|_{D''})|$.

We say that P is a *perfect-to-perfect* (*p.t.p.*) packet-move if both its initial and final configurations are perfect. We define $D_n = [1..n]$, and if $n > k$, we partition it into $Big(n) = [(n-k+1)..n]$ and $Small(n) = D_{n-k}$. Thus, BTH_n concerns finding the shortest p.t.p. packet-move of D_n.

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

Consider a sequence of moves S containing two *consecutive* 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 of S; it is easy to see that such a result is independent on the order of particular prunings and is legal, so the sequence of moves $Prune(S)$ is well-defined.

3 Shortest Packet-Moves

In both [5] and [1], a shortest possible packet-move of D_n (among general, not necessarily p.t.p. packet-moves) is found and used for solving BTH_n. We denote it $\beta_n = \beta_n(source, target)$ (for illustration see Figure 2):

- If $n \leq k$, move all disks from *source* to *target* one by one.
- Otherwise:

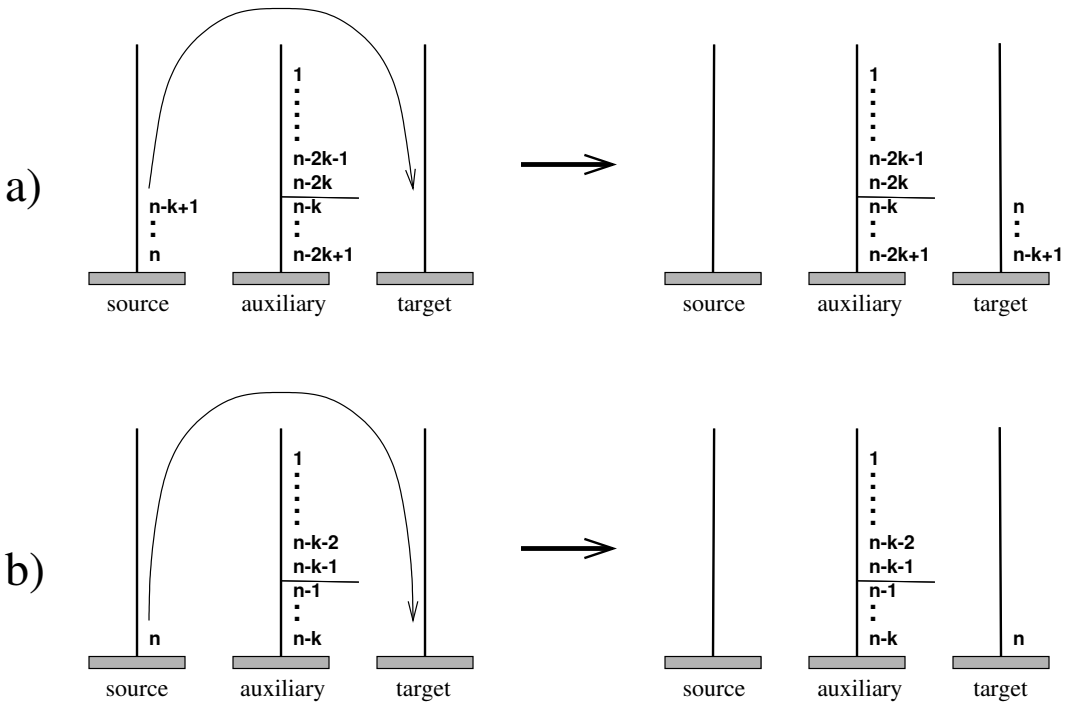

Figure 2: Illustration of (a) β_n and of (b) α_n.

- recursively perform $\beta_{n-k}(source, auxiliary)$;
- move disks in $Big(n)$ from $source$ to $target$ one by one;
- recursively perform $\beta_{n-k}(auxiliary, target)$.

It is easy to prove, by induction on the depth of recursion, that β_n is legal if D_n is placed initially in decreasing order or in the order differing from it by *the inverse arrangement of the k lowest (largest) disks* (*or of all disks if* $n \leq k$). To see this, note that β_n applied to each of these orders transforms it to the other one.

Let b_n denote the length of β_n. By definition of β_n:

$$b_n = \begin{cases} n & \text{if } n \leq k \\ 2b_{n-k} + k & \text{if } n > k \ . \end{cases} \tag{1}$$

In [5], the recurrence relation (1) is shown to imply the explicit formula:

$$b_n = k \cdot (2^{\lfloor \frac{n}{k} \rfloor} - 1) + r \cdot 2^{\lfloor \frac{n}{k} \rfloor} = (k + r) \cdot 2^{\lfloor \frac{n}{k} \rfloor} - k, \text{ where } r = n \bmod k \ . \tag{2}$$

Note that $b_{n+1} - b_n = 2^{\lfloor \frac{n}{k} \rfloor}$; therefore, the sequence $\{b_i\}$ is *strictly* monotonous.

It is essentially proved in [5] (with minor details left out) that β_n is a shortest possible packet-move of D_n; we provide such a proof for completeness. Notice that by the k-relaxed placement rule, during any move (i, X, Y) all disks in $Small(i)$ are gathered on the spare peg $Z \neq X, Y$. We will use the following observation in the sequel.

Fact 3.1 *For $n > k$, if some sequence of moves begins from a configuration, where disk n and $Small(n)$ are gathered on peg X, and finishes at a configuration, where disk n and $Small(n)$ are gathered on another peg Y, then it contains two disjoint packet-moves of $Small(n)$: one (from X) before the first move of disk n and another (to Y) after its last move.*

Theorem 3.2 *Under the k-relaxed placement rule, the length of any packet-move of D_n is at least b_n.*

Proof: By a complete induction on n. *Basis*: The case $n \leq k$ is trivial. *Induction step*: For any $n > k$, we consider an arbitrary packet-move P of D_n, assuming the statement holds for all lesser values of n. By Fact 3.1, $P|_{Small(n)}$ contains two disjoint packet-moves of $Small(n)$; by the induction hypothesis, their total length is at least $2 \cdot b_{n-k}$. Every disk in $Big(n)$ must move at least once, which sums to at least k moves. Hence, $|P| = |(P|_{Small(n)})| + |(P|_{Big(n)})| \geq 2 \cdot b_{n-k} + k = b_n$. □

Poole's algorithm for solving BTH_n, henceforth denoted α_n, is as follows:

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

Recall that the gap in Poole's proof of optimality was in overlooking algorithms containing more than one move of disk n. Let us show that there are reasonable strategies for solving BTH_n while moving disk n twice. Notice first that $\beta_n(source, auxiliary)$ followed by $\beta_n(auxilary, target)$ is a feasible solution to BTH_n. Another (generic) strategy is as follows:

- move D_{n-k} from *source* to *target* somehow;

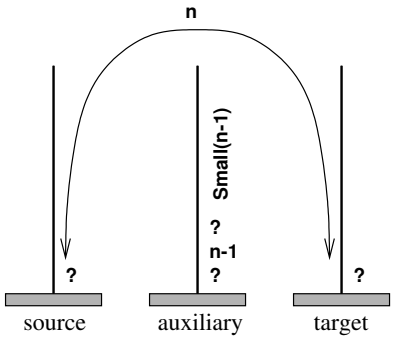

Figure 3: A distinguished move of disk n. Disks in $[(n-k+1)..(n-2)]$ can be dispersed on all pegs; on *auxiliary* they, as well as disk $n-k$, may be below or above disk $n-1$ and may be interleaved with disks in $Small(n-1)$.

- move disks $n-k+1, \ldots, n$ from *source* to *auxilary* one by one;
- move D_{n-k} from *target* to *source* somehow;
- move disks $n, \ldots, n-k+1$ from *auxilary* to *target* one by one;
- move D_{n-k} from *source* to *target* somehow.

Clearly, the disks $n-k+1, \ldots, n$ arrive on *target* in decreasing order. So, aiming to overperform α_n, it would be enough to find a sufficiently short *triplet* of consecutive packet-moves of D_{n-k}, which results in the *perfect* configuration of D_{n-k} on *target*. Notice that α_n contains *four* shortest packet-moves of D_{n-k-1}, so this task does not seem hopeless.

4 Optimality of Poole's algorithm for BTH_n

In this section, we prove that α_n is optimal for BTH_n.

For any packet-move, we call a move of the largest disk *distinguished* if it is from *source* to *target* or from *target* to *source* and if the second largest disk resides on *auxiliary* during that move (for illustration see Figure 3).

Lemma 4.1 *Any packet-move P of D_n, which preserves the initial order between disks n and $n-1$ and such that disk n never moves to auxiliary, contains a distinguished move of disk n.*

Proof: Define $P' := Prune(P|_{\{n-1,n\}})$. Note that P' preserves the initial order between disks n and $n-1$, since P preserves it. Furthermore, a move of n is distinguished w.r.t. P' if and only if it is distinguished w.r.t. P. By the definition of $Prune$, the moves of disks n and $n-1$ must *alternate* in P'.

We claim that the first move of disk $n-1$ in P' should be from *source* to *auxiliary*. Indeed, otherwise it is from *source* to *target*. Then, by the second condition of the lemma, the first two moves in P' transfer both disks to *target*. This should be the end of P', since the existence of a third move would contradict the alternation of moves of disks n and $n-1$ in P'. It follows that the final order of n and $n-1$ is inverse w.r.t. their initial order, yielding a contradiction.

Now, if at the initial configuration of P' disk $n-1$ is placed above disk n, then the first two moves should be $(n-1, source, auxiliary)$ and $(n, source, target)$; the latter is distinguished. Otherwise, disk $n-1$ is initially placed below disk n. Then, the first three moves should be $(n, source, target)$, $(n-1, source, auxiliary)$, and $(n, target, source)$; the latter is distinguished. □

Corollary 4.2 *For any $n > k+1$ and any packet-move P of D_n, which preserves the initial order between disks n and $n-1$ and such that disk n never moves to auxiliary, P contains four disjoint packet-moves of $Small(n-1)$.*

Proof: By Lemma 4.1, there exists at least one distinguished move of disk n in P. During the first such move, all disks in $Small(n-1)$ are on *auxiliary* together with disk $n-1$; by the k-relaxed placement rule, they are placed above disk $n-1$. By Fact 3.1, the parts of P before and after that move contain two disjoint packet-moves of $Small(n-1)$ each, which totals to four disjoint packet-moves. □

Lemma 4.3 *For any $n > k$, if a packet-move P of D_n contains a move of disk n to auxiliary, then P contains three disjoint packet-moves of $Small(n)$.*

Proof: Suppose first that P contains a move $(n, source, auxiliary)$. By the k-relaxed placement rule, $Small(n)$ is gathered on *target* during that move, and it is gathered on another peg $X \neq target$ during the last move of disk n. Hence, P contains the following disjoint packet-moves of $Small(n)$: $source \to target$, $target \to X$, and $X \to target$.

If disk n never moves from *source* to *auxiliary* during P, then the first move of disk n is $(n, source, target)$, and at some later point disk n moves from *target* to *auxiliary*. During the former move, $Small(n)$ is gathered on *auxiliary*, and it is gathered on *source* during the latter. Therefore, P contains the following disjoint packet-moves of $Small(n)$: $source \rightarrow auxiliary$, $auxiliary \rightarrow source$, and $source \rightarrow target$. □

Lemma 4.4 *The length of any p.t.p. packet-move of D composed of $2l+1$ packet-moves of D is at least $(2l+2)|D|-1$.*

Proof: We call a disk *cheap* w.r.t. some packet-move P', if it does not move to $auxiliary(P')$ during P'. We claim that at most one disk is cheap w.r.t. each one of the $2l+1$ contained packet-moves. Assume to the contrary that there are two such disks, i and j. Then, after each one of the contained packet-moves, their order is reversed. Since there is an odd number of such packet-moves, the order of i and j at the final configuration is inverse w.r.t. their order at the initial configuration—a contradiction.

It follows that there are $|D|-1$ disks which make at least $2l+2$ moves each, and a single disk which makes at least $2l+1$ moves. Altogether, at least $(2l+2)|D|-1$ moves are made. □

Following is the central statement. Note that the length of α_n, denoted by a_n, equals $2 \cdot b_{n-1}+1$ by definition.

Proposition 4.5 *For any $l \geq 0$ and $n \geq 1$, let P be a p.t.p. packet-move of D_n containing $2l+1$ disjoint packet-moves of D_n. Then, $|P| \geq 2l \cdot b_n + 2 \cdot b_{n-1} + 1 = 2l \cdot b_n + a_n$, and this bound is tight.*

Proof: The lower bound proof is by a complete induction on n, for all l.

We first claim that P is *composed* of at least $2l+1$ packet-moves of D_n. Consider the non-empty intervals of P (if any) before the first, between the ith and the $(i+1)$st, $1 \leq i \leq 2l$, and after the last one out of the $2l+1$ contained packet-moves. For any such interval I, if the gathered configurations of D_n at its beginning and end are on the same peg, we extend one of the neighboring packet-moves with the moves in I; otherwise, we define I as an additional packet-move of D_n. As a result, P is composed of at least $2l+1$ packet-moves of D_n, as required.

If there are at least $2l+2$ such packet-moves, then by Theorem 3.2 and the strict monotonicity of (b_i), $|P| \geq (2l+2) \cdot b_n \geq 2l \cdot b_n + 2 \cdot b_{n-1} + 2$. We henceforth suppose that P *is composed of exactly* $2l+1$ *packet-moves of* D_n.

Basis: $n \leq k$.

Since $b_r = r$ for any $r \leq k$, we conclude by Lemma 4.4 that $|P| \geq (2l+2)n - 1 = 2ln + 2(n-1) + 1 = 2l \cdot b_n + 2 \cdot b_{n-1} + 1$.

Induction step: We suppose that the claim holds for all lesser values of n and for all l, and prove it for n and all l.

Note that disk n is placed below disk $n-1$ at both the initial and the final configurations of P. Since P is composed of an odd number of packet-moves of D_n, at least one of them, henceforth denoted by $\tilde{P}$, *preserves the order between disks* n *and* $n-1$. Note that $|P| = |(P|_{Small(n)})| + |(P|_{Big(n)})|$ and that by Lemma 4.4 applied to $Big(n)$, we have $|(P|_{Big(n)})| \geq (2l+2)k - 1$.

Case 1: *During* $\tilde{P}$, *disk* n *never moves to auxiliary*$(\tilde{P})$.

By Corollary 4.2, $\tilde{P}$ contains four disjoint packet-moves of $Small(n-1)$ (this holds vacuously in the case $n = k+1$, as $Small(n-1) = \emptyset$ in that situation). Disk $n-k$ makes at least two moves during $\tilde{P}$: at least one before the first move of disk n and at least one after its last move. By Theorem 3.2, $|(\tilde{P}|_{Small(n)})| \geq 4 \cdot b_{n-k-1} + 2 = 2 \cdot b_{n-1} - 2k + 2$. By Fact 3.1, the other $2l$ packet-moves of D_n contain two disjoint packet-moves of $Small(n)$ each; their total length is at least $4l \cdot b_{n-k} = 2l(b_n - k)$. Therefore, $|(P|_{Small(n)})| \geq 2l \cdot b_n - 2lk + 2 \cdot b_{n-1} - 2k + 2 = 2l \cdot b_n + 2 \cdot b_{n-1} - (2l+2)k + 2$. By adding $|(P|_{Big(n)})| \geq (2l+2)k - 1$, we obtain the required bound for $|P|$.

Case 2: $\tilde{P}$ *contains a move of disk* n *to auxiliary*$(\tilde{P})$.

By Lemma 4.3, $\tilde{P}|_{Small(n)}$ contains three disjoint packet-moves of $Small(n)$. By Fact 3.1, the other $2l$ packet-moves of D_n contain two disjoint packet-moves of $Small(n)$ each. Thus, $P|_{Small(n)}$ contains $4l+3$ disjoint packet-moves of $Small(n)$. By the induction hypothesis and the strong monotonicity of (b_i), $|(P|_{Small(n)})| \geq (4l+2) \cdot b_{n-k} + 2 \cdot b_{n-k-1} + 1 \geq 4l \cdot b_{n-k} + 4 \cdot b_{n-k-1} + 3 = 2l \cdot b_n + 2 \cdot b_{n-1} - (2l+2)k + 3$. By adding $|(P|_{Big(n)})| \geq (2l+2)k - 1$, we establish that $|P|$ strictly exceeds the required bound.

The bound for $|P|$ is tight, since the sequence composed of $2l$ β_n and one α_n is a p.t.p. packet-move of D_n of length as in the bound. □

Proposition 4.5 in the particular case $l = 0$ implies:

Theorem 4.6 *Algorithm* α_n *is optimal for* BTH_n.

The case analysis in the proof of Proposition 4.5 for $l = 0$ (when $\tilde{P} = P$), shows that it is impossible to reach the bound a_n in Case 2. Besides, in order to attain this bound in Case 1, disk n should move exactly once. Indeed, otherwise it should make at least three moves, which prevents attaining the equality $|(P|_{Big(n)})| = (2l+2)k-1$. We deduce the following characterization:

Corollary 4.7 Any *optimal algorithm for BTH_n contains a single move of disk n, from source to target, preceded by a packet-move of D_{n-1} of length b_{n-1} from source to auxiliary, and followed by such a packet-move from auxiliary to target.*

5 Further Research

In [4, 6], we show that β_n is *not* a unique shortest possible packet-move, for any $k \geq 2$, thus disproving Corollary 3 of [5]. As a consequence, α_n is *not* a unique optimal algorithm for BTH_n. Further, we describe there the *family of all optimal solutions* to BTH_n, and present a closed formula for their number.

In the same paper, we present a pair of initial and final configurations of BTH_n, such that the shortest sequence of moves between them is of length $\Omega(k \cdot a_n)$. Moreover, we establish there the tight $\Theta(k \cdot a_n)$ bound, asymptotic w.r.t. n, for both the *diameter* of and the *average distance* between nodes in the configuration graph of BTH_n.

In [3, 6], we generalize $BTH_{n,k}$ by allowing the disk sizes to form any set of n distinct integers. For this "subset" setting, we describe a family of at most n algorithms, so that the best one among them is optimal. Poole's algorithm is the simplest one in this family; no other member obeys the *single move of disk n* assumption made by Poole [5].

Following a suggestion of Berend, we define in [3, 6] the "ultimate" relaxed placement rule, where the sets of larger disks allowed to be placed above each disk may be arbitrary, obeying monotonicity only. We extend the results we established for the "subset" setting to the "ultimate" setting.

Acknowledgment

The authors thank Daniel Berend for his suggestion to consider the question on the optimal algorithm for the Bottleneck Tower of Hanoi problem.

References

[1] S. Benditkis and I. Safro. Generalizations of the Tower of Hanoi Problem. Final Project Report, supervised by D. Berend, Dept. of Math. and Comp. Sci., Ben-Gurion University, 1998.

[2] X. Chen, B. Tian, and L. Wang. Santa Claus' Towers of Hanoi. *Graphs and Combinatorics* **23**[Supplement] (2007), 153-167.

[3] Y. Dinitz and S. Solomon. Optimal Algorithms for Tower of Hanoi Problems with Relaxed Placement Rules. In: *Proc. of the 17th International Symposium on Algorithms and Computation (ISAAC'2006)*, LNCS **4288**, Springer-Verlag, 2006, 36–47.

[4] Y. Dinitz and S. Solomon. On Optimal Solutions for the Bottleneck Tower of Hanoi Problem. In: *Proc. of the 33rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'2007)*, LNCS **4362**, Springer-Verlag, 2007, 248–259.

[5] D. Poole. The Bottleneck Towers of Hanoi Problem. *J. of Recreational Math.* **24** (1992), no. 3, 203–207.

[6] S. Solomon. *Algorithms and Lower Bounds on Hanoi Tower Problems.* M.Sc. thesis in Computer Science. Dept. of Comp. Sci., Ben-Gurion University, Beer-Sheva, Israel, 2006.

[7] P.K. Stockmeyer. The Tower of Hanoi: A Bibliography. (1998). Available via http://www.cs.wm.edu/∼pkstoc/biblio2.pdf .

[8] M. Szegedy, In How Many Steps the k Peg Version of the Towers of Hanoi Game Can Be Solved? In: *Proc. of the 16th Symposium on Theoretical Aspects of Computer Science (STACS'1999)*, LNCS **1563**, Springer-Verlag, 1999, 356–361.

[9] D. Wood. The Towers of Brahma and Hanoi Revisited. *J. of Recreational Math.* **14** (1981–1982), no. 1, 17–24.