Competitive Search in Symmetric Trees

David Kirkpatrick[1] and Sandra Zilles[2]

[1] Department of Computer Science, University of British Columbia, Canada
kirk@cs.ubc.ca
[2] Department of Computer Science, University of Regina, Canada
zilles@cs.uregina.ca

Abstract. We consider the problem of searching for one of possibly many goals situated at unknown nodes in an unknown tree $\mathcal{T}$. We formulate a universal search strategy and analyse the competitiveness of its average (over all presentations of $\mathcal{T}$) total search cost with respect to strategies that are informed concerning the number and location of goals in $\mathcal{T}$. Our results generalize earlier work on the multi-list traversal problem, which itself generalizes the well-studied m-lane cow-path problem. Like these earlier works our results have applications in areas beyond geometric search problems, including the design of hybrid algorithms and the minimization of expected completion time for Las Vegas algorithms.

1 Introduction

The *m-lane cow-path problem* specifies a sequence of m rays (*lanes*) of unbounded length incident on a common origin (*crossroad*). A goal (*pasture*) lies at some unknown distance d from the origin along some (unknown) ray. The objective is to formulate a provably good strategy (minimizing the total search cost) for an agent (*cow*) to reach the goal, starting from the origin.

The cow-path problem is a special instance of a family of problems called *search games*, in which a searcher tries to minimize the time needed to find a hidden goal. For a detailed study of search games, the reader is referred to [1]. The cow-path problem itself has been studied in several variations, including directionally dependent traversal costs, turnaround penalties, shortcuts and dead-ends [4,6,11,12,15]. It has also been analysed in terms of worst-case and average-case competitive ratio (using d as a benchmark), as well as in a game-theoretic framework [3,8,16,17,18].

Essentially the same ideas as those used in solving the cow-path problem have been used in the synthesis of deterministic and randomized hybrid algorithms with (near) optimal competitive ratios [2,7]. Given are a number of *basic* algorithms each of which might (or might not) be useful in solving some problem. The goal is to synthesize a hybrid algorithm from these basic components by some kind of dovetailing process. Memory limitations may restrict the number of processes that can be suspended at any given time (the alternative being a complete restart with successively larger computation bounds).

More recently, the cow-path problem has been generalized in a new and fundamentally different direction. The *multi-list traversal problem* [10] assumes that *every ray* leads to a goal, and the objective is to minimize the total search cost in finding a goal on at least one path. (Conventional one-goal cow-path problems correspond to the special case in which all goals but one are located arbitrarily far from the origin). Essentially the same problem has been studied by McGregor *et al.* [14] as an "oil searching problem", where the objective is to maximize the number of goals (wells) discovered for a specified budget. Even earlier, similar results were presented by Luby *et al.* [13] for the problem of minimizing the expected execution time of Las Vegas algorithms (viewed as an infinite sequence of deterministic algorithms with unknown completion times.)

The m-lane cow paths problem, the multi-list traversal problem, and its variants can all be thought of as search-with-backtracking problems, in which backtracking always brings the algorithm back to the origin of search, from where a new path can be chosen or search in a previously visited path can be resumed. In many real-world search problems, it is often the case that part of the search effort invested into one search path eases the search along another path. Backtracking would then allow the search algorithm to return to a fork part-way along the current path and to search along a new path branching from the current one (without repeating the search effort to reach the fork from the origin). The simplest search domain allowing this kind of backtracking is a tree.

Motivated by the desire to understand the limitations of oblivious backtracking algorithms, we consider a generalization of the multi-list traversal problem in which the search domain is an unknown unbounded fully-symmetric tree $\mathcal{T}$ with goals at one or more nodes. Fleischer *et al.* [5] considered search problems on trees as part of a more general study of algorithms that minimize the optimal search ratio: essentially the worst case ratio of the cost of finding a goal in a search domain to the length of the shortest path to that goal. For our competitive analysis we compare uninformed algorithms to those that know $\mathcal{T}$, including the locations of all goals, but not the specific presentation of $\mathcal{T}$ (i.e. the ordering of the children at each of its internal nodes). In fact, McGregor *et al.* [14] already introduced a generalization of their oil-searching problem to symmetric trees in an appendix to their paper. Unfortunately, their algorithm exhibits a rather poor competitive ratio for general symmetric trees, motivating a more in-depth treatment of the symmetric tree search problem. Note that while it is possible to study backtracking in asymmetric trees (or more general graphs), it is natural to restrict attention to search domains in which all search paths are equivalent up to relabeling: as McGregor *et al.* [14] point out, asymmetries serve to amplify the knowledge (concerning goal locations) of informed algorithms, making competitive analysis simultaneously more difficult and less meaningful.

1.1 Symmetric Tree Traversal

In many respects our treatment of search in symmetric trees parallels and generalizes earlier work on the multi-list search problem. Where previously an algorithm would be evaluated with respect to possible input presentations ranging over *all possible permutations of a multi-set of list lengths*, we are now interested in inputs that correspond to presentations of some fixed symmetric tree. Thus an *instance* of our *symmetric tree traversal problem* is an unbounded rooted unordered fully-symmetric[1] tree $\mathcal{T}$, one or more nodes of which are distinguished as *goal* nodes, called *goals* for short. We assume,

[1] All nodes at the same level ℓ have the same number of children d_ℓ.

without loss of generality, that the path from the root to any goal does not contain any other goal. We denote by $\Pi(\mathcal{T})$ the set of all *presentations* of the problem instance $\mathcal{T}$. Each such presentation is an ordering of $\mathcal{T}$, i.e. for each internal node x of $\mathcal{T}$, a bijection from the set $\{1, \ldots, d_x\}$ to the edges joining x to the d_x children of x. In this way, every presentation of $\mathcal{T}$ assigns to every node x, and in particular every goal, in $\mathcal{T}$ a labeled path from the root to x. We interpret the concatenation of labels on this path as the *index* of x in the given presentation.

We assume that in general algorithms take an arbitrary presentation π of $\mathcal{T}$ as input, and know nothing about the number or location of goals in $\mathcal{T}$. Algorithms proceed in a stepwise fashion. In the first step the root node is explored, and in every subsequent step a specified child of some previously explored node is explored, continuing until some goal node is reached. We denote by $\mathrm{search\_cost}(\mathcal{A}, \pi)$ the total search cost (number of explored nodes) of algorithm $\mathcal{A}$ on input presentation π.[2] We analyse this search cost of algorithms (both deterministic and randomized) for specific problem instances $\mathcal{T}$ in both the worst and average cases (over all presentations of $\mathcal{T}$). For worst-case behaviour we can think of an adversary choosing the least favorable presentation of $\mathcal{T}$, knowing the search strategy of the algorithm. We view randomized (Las Vegas) algorithms as probability distributions over deterministic algorithms; in this case we are interested in *expected* search cost.

For the purpose of competitive analysis we contrast general *uninformed* algorithms with several *informed* variants that are only required to behave correctly on problem instances that satisfy certain constraints on the number or location of the goals. A *instance-informed* algorithm knows the problem instance, i.e. the location of goals in $\mathcal{T}$, but not their index in the given input presentation. A *level-count-informed* algorithm knows the number of goals at each level of $\mathcal{T}$, but not their location. A *cost-informed* algorithm knows an upper bound on the worst-case search cost that is realizable by the best instance-informed algorithm for the given instance.

We start by restricting our attention to the case where $\mathcal{T}$ is a full binary tree (i.e. $d_\ell = 2$, at every level). Section 2 considers the situation where all goals are known to lie on one fixed level of $\mathcal{T}$, and results are developed for both the full search cost as well as the search cost restricted to the goal level. These results are extended, in Section 3, to the general situation where goals may appear on multiple levels. Finally, the restriction to binary trees is relaxed in Section 4. (Most of the proofs in this last section are most easily understood as elaborations of the proofs of corresponding results for binary trees; full details of these proofs are presented in [19].)

In general, our oblivious search algorithms not only significantly improve the search bounds provided by the tree-searching algorithm of McGregor *et al.* [14], but they also are arguably close to optimal in the competitive ratio of their search cost with that of non-oblivious counterparts. For example, for binary trees with k goals on one fixed level h, our algorithm guarantees an average search cost that is within a factor h of that achievable by any algorithm that is only required to perform efficiently on presentations of one fixed tree. In the same situation, the strategy proposed in [14] is only claimed to

[2] Our results apply equally well when the cost of backtracking is taken into account, *i.e.*, when the search cost includes the cost of re-visiting nodes.

have a corresponding competitive ratio which is bounded by the square of the number of nodes in the tree!

2 The Case Where All Goals Are Known to Lie at the Same Level

In the multi-list traversal problem the best uninformed strategy employs a non-uniform interleaving (dubbed "hyperbolic dovetailing" in [10]) of strategies each of which searches all lists uniformly to some fixed depth. Motivated by that, we first consider the case where all goals are known to lie at some fixed level h. In this case, it does not make any sense for an algorithm to explore paths in $\mathcal{T}$ to a level more or less than h. Therefore we initially consider $\mathcal{T}$ to be truncated at level h and count just the number of *probes* an algorithm makes of nodes at the leaf level h, ignoring the cost associated with reaching those nodes. In this restricted setting, a level-count-informed algorithm knows the number k of goals at level h in $\mathcal{T}$, but not their location. We denote by $\mathrm{probe\_cost}(\mathcal{A}, \pi)$ the total number of nodes on level h explored by algorithm $\mathcal{A}$ on input presentation π.

Since every presentation of the full binary tree $\mathcal{T}$ of height h fixes, for each of its $2^h - 1$ internal nodes x, one of two possible labelings on the pair of edges leading to the children of x , we have the following:

Observation 1. *If $\mathcal{T}$ is a full binary tree of height h then $|\Pi(\mathcal{T})| = 2^{2^h-1}$.*

2.1 Worst-Case Probe Cost

It is clear that an arbitrary uninformed probing algorithm will never need to make more than $2^h - k + 1$ probes at level h, when faced with a problem instance $\mathcal{T}$ with exactly k goals at level h. On the other hand, an adversary can force this many probes by any fixed (even count-informed) algorithm by choosing a suitable problem instance $\mathcal{T}$ with exactly k goals at level h and a suitable presentation $\pi \in \Pi(\mathcal{T})$. Thus,

Observation 2. *For every deterministic level-count-informed algorithm $\mathcal{A}$, there exists a problem instance $\mathcal{T}$ with exactly k goals at level h such that* $\max_{\pi \in \Pi(\mathcal{T})} \mathrm{probe\_cost}(\mathcal{A}, \pi) = 2^h - k + 1$.

As we observe next, fully informed probing algorithms can, at least for some problem instances, have significantly lower worst-case probe cost. In the next section, we show that similar reductions are always achievable if we measure instead the average or expected probe cost.

Observation 3. *There exists a deterministic instance-informed algorithm $\mathcal{A}$ and, for every $j \geq 0$, a problem instance $\mathcal{T}_j$ with exactly 2^j goals at level h, such that* $\max_{\pi \in \Pi(\mathcal{T}_j)} \mathrm{probe\_cost}(\mathcal{A}, \pi) \leq 2^{h-j}$.

Proof. If tree $\mathcal{T}_j$ has goals at all 2^j leaves of some subtree rooted at an internal node at level $h - j$, then it suffices to probe one leaf in each of the 2^{h-j} subtrees rooted at level $h - j$, in any presentation of $\mathcal{T}_j$. As a second example, if $\mathcal{T}_j$ has one goal in each of its 2^j subtrees rooted at internal nodes at level j, it suffices to explore all 2^{h-j} leaves in any one of these subtrees, in any presentation of $\mathcal{T}_j$. □

It follows from Theorem 4 below that instances like $\mathcal{T}_j$ above are the *least* complex, in terms of their worst-case probe cost, for fully informed algorithms. As Theorem 5 and its corollary demonstrate, the *most* complex such instances have a significantly higher worst-case probe cost.

Theorem 4. *For every deterministic instance-informed algorithm $\mathcal{A}$, and every problem instance $\mathcal{T}$ with exactly k goals at level h,*
$\max_{\pi \in \Pi(\mathcal{T})} \text{probe\_cost}(\mathcal{A}, \pi) \geq 2^h/k$.

Proof. For any node x in $\mathcal{T}$ and any index i of a fixed probe location at level h, x is assigned index i in exactly $\frac{2^{2^h-1}}{2^h}$ presentations of $\mathcal{T}$, since any presentation that maps x to a fixed probe location fixes the labels associated with the h edges on the path to that goal, and only those edges. Thus, for any i, there are exactly $k\frac{2^{2^h-1}}{2^h}$ presentations that assign one of k goals to the probe with location index i. It follows that any deterministic algorithm that uses fewer than $\frac{2^h}{k}$ probes at level h fails to detect a goal for at least one presentation of $\mathcal{T}$. □

Theorem 5. *For any r, $0 \leq r \leq h$, there exists a problem instance $\mathcal{T}_{r,h}$ with $k = \sum_{j=r}^{h} \binom{h}{j}$ goals at level h, such that for every deterministic instance-informed algorithm $\mathcal{A}$, $\max_{\pi \in \Pi(\mathcal{T}_{r,h})} \text{probe\_cost}(\mathcal{A}, \pi) \geq 2^r$.*

Proof. (Sketch) The tree $\mathcal{T}_{r,h}$ is defined recursively for $0 \leq r \leq h$: (i) $\mathcal{T}_{0,h}$ is the complete tree with 2^h leaves, all of which are goals; (ii) $\mathcal{T}_{h,h}$ is the complete tree with 2^h leaves, exactly one of which is a goal; and (iii) $\mathcal{T}_{r,h}$ is the complete tree whose root has subtrees $\mathcal{T}_{r,h-1}$ and $\mathcal{T}_{r-1,h-1}$, when $0 < r < h$.

One can show, by induction on r and h, that (i) $\mathcal{T}_{r,h}$ has $k = \sum_{j=r}^{h} \binom{h}{j}$ goals at level h and (ii) for any set of fewer than 2^r probes in $\mathcal{T}_{r,h}$ there is a presentation of $\mathcal{T}_{r,h}$ for which no probe detects a goal. (See [19] for details.) □

2.2 Average and Expected-Case Probe Cost

Theorem 4 extends to average case behaviour of fully informed algorithms:

Theorem 6. *For every deterministic instance-informed algorithm $\mathcal{A}$, and every problem instance $\mathcal{T}$ with exactly k goals at level h,*
$\text{avg}_{\pi \in \Pi(\mathcal{T})} \text{probe\_cost}(\mathcal{A}, \pi) \geq 2^{h-2}/k$.

Proof. As shown in the proof of Theorem 4, for any i, there are exactly $k\frac{2^{2^h-1}}{2^h}$ presentations that assign one of k goals to the location index i. Thus, any deterministic algorithm using fewer than $\frac{2^{h-1}}{k}$ probes at level h fails to detect a goal in at least half of the presentations of $\mathcal{T}$. Hence every deterministic algorithm uses at least $\frac{2^{h-1}}{k}$ probes at level h on at least half of its input presentations. □

Theorem 6 can be strengthened to apply to the expected case behaviour of randomized instance-informed algorithms $\mathbf{A}$, by viewing $\mathbf{A}$ as a probability distribution over deterministic algorithms in the standard way (see [19] for details).

Theorem 7. *For every randomized instance-informed algorithm* $\mathbf{A}$*, and every problem instance* $\mathcal{T}$ *with exactly* k *goals at level* h*,*
$\mathrm{avg}_{\pi \in \Pi(\mathcal{T})} E[\mathrm{probe\_cost}(\mathbf{A}, \pi)] \geq 2^{h-2}/k$.

The following theorem, whose proof embodies the central idea of our general oblivious tree-searching strategy, shows that the lower bound of Theorem 6 is realizable to within a constant factor, even by an uninformed algorithm.

Theorem 8. *There is a deterministic uninformed algorithm* $\mathcal{A}_0$ *such that, for every problem instance* $\mathcal{T}$ *with exactly* k *goals at level* h*,*
$\mathrm{avg}_{\pi \in \Pi(\mathcal{T})} \mathrm{probe\_cost}(\mathcal{A}_0, \pi) \leq 2^{h+2}/k$.

Proof. For any r, $0 \leq r \leq h$, we can interpret an arbitrary presentation of $\mathcal{T}$ as a *bottom* tree $\mathcal{T}'$, consisting of all nodes of $\mathcal{T}$ at level at most r, together with 2^r *top* trees, each with 2^{h-r} leaves.

The algorithm $\mathcal{A}_0$ proceeds in rounds: at the completion of round $r \geq 0$, exactly one leaf in each of the 2^r trees rooted at nodes on level r has been probed. The algorithm terminates if a goal is discovered in at least one of its probe locations. The total number of probes in round r is just $2^r - 2^{r-1} = 2^{r-1}$.

We count the fraction Φ_r of presentations of $\mathcal{T}$ for which algorithm $\mathcal{A}_0$ terminates by the end of round r. Each goal resides in one of the 2^r top trees with 2^{h-r} leaves, and coincides with the probed leaf in that tree in exactly $\frac{1}{2^{h-r}}$ of the presentations of that top tree. Thus each individual goal is probed in $\frac{1}{2^{h-r}}$ of the presentations of $\mathcal{T}$, by the end of round r.

Of course, some presentations map two or more goals to probe positions. So to count Φ_r we number the goals arbitrarily and, for $1 \leq i \leq k$, we count, among the presentations of $\mathcal{T}$ that map none of the first $i-1$ goals to a probe position, the fraction f_i that map the i-th goal to a probe position. Clearly, $\Phi_r = \sum_{1 \leq i \leq k} f_i \cdot [\prod_{1 \leq j < i}(1 - f_j)]$. Furthermore, $f_i \geq \frac{1}{2^{h-r}}$, where equality holds just when none of the first $i-1$ goals occupy the same top tree as the i-th goal.

If we define $F_x = \sum_{x \leq i \leq k} f_i \cdot [\prod_{x \leq j < i}(1 - f_j)]$, for $1 \leq x \leq k$, then $F_k = f_k$ and, for $1 \leq x < k$, $F_x = f_x + (1 - f_x)F_{x+1}$. It is straightforward to confirm by induction that $F_x \geq 1 - (1 - \frac{1}{2^{h-r}})^{k-x+1}$. Thus $\Phi_r = F_1 \geq 1 - (1 - \frac{1}{2^{h-r}})^k > 1 - (\frac{1}{e})^{k/2^{h-r}}$.

Now if $2^{h-j} \leq k < 2^{h+1-j}$, then at most $(\frac{1}{e})^{k/2^{h-j-i}} \leq (\frac{1}{e})^{2^i}$ of the presentations of $\mathcal{T}$ have not terminated after $r = j + i$ rounds. Hence the average, over all presentations of $\mathcal{T}$, of the number of probes of algorithm $\mathcal{A}_0$ is at most
$2^j + \sum_{i \geq 1}(2^{j+i-1}(\frac{1}{e})^{2^{i-1}}) < 2^j(1 + \sum_{s \geq 1}(s(\frac{1}{e})^s)) < 2^j(1 + \frac{e}{(e-1)^2}) < 4\frac{2^h}{k}$. □

Remark 1. Choosing $k = 2^{h-1}$ in Theorem 8 and $r = h/2$ in Theorem 5 demonstrates a large gap between the average and worst-case behaviours of deterministic instance-informed algorithms. Specifically, the problem instance $\mathcal{T}_{h/2,h}$ with 2^{h-1} goals at level h has the property that algorithm $\mathcal{A}_0$ has average probe cost of at most 8, whereas every deterministic instance-informed algorithm requires at least $2^{h/2}$ probes in the worst case.

Remark 2. It is easy to see that the total additional search cost in round r of Algorithm $\mathcal{A}_0$ is $2^{r-1}(h - r + 1)$. Thus if $2^{h-j} \leq k < 2^{h+1-j}$ the proof above implies that the average total search cost is at most $2^j(h-j)+\sum_{i\geq 1}(2^{j+i-1}(h-j-i+1)(\frac{1}{e})^{2^{i-1}}) < 2^j(h-j)(1+\sum_{s\geq 1}(s(\frac{1}{e})^s)) = O((h-j)\frac{2^h}{k}) = O(\frac{2^h}{k}(1+\lg k))$.

By simply randomizing the given presentation before running algorithm $\mathcal{A}_0$ the average-case bound of Theorem 8 can be realized as the worst-case expected cost, providing a tight complement to the lower bound of Theorem 7:

Corollary 9. *There is a randomized uninformed algorithm* $\mathbf{A}_1$ *such that, for every problem instance* $\mathcal{T}$ *with exactly* k *goals at level* h*,*
$\max_{\pi\in\Pi(\mathcal{T})} E[\text{probe\_cost}(\mathbf{A}_1,\pi)] \leq 2^{h+2}/k$.

2.3 Taking Full Search Cost into Consideration

As noted above, the algorithm $\mathcal{A}_0$ outlined in Section 2.2 has probe cost $O(\frac{2^h}{k})$ but total search cost $O(\frac{2^h}{k}(1+\lg k))$. For some problem instances, e.g., the tree $\mathcal{T}_j$ (described in Theorem 3) with goals at its leftmost $k=2^j$ leaves, even fully informed algorithms require average total search cost $\Omega(\frac{2^h}{k}(1+\lg k))$, since at least one probe must be made in at least half of the top-level trees of size k, or the algorithm will fail on at least half of the permutations. Hence this additional $\lg k$ factor is unavoidable in some cases, even when $k=o(2^h)$).

Nevertheless, we have not been able to formulate a notion of intrinsic total search cost that would permit a tighter general competitive bound than that given by the following:

Theorem 10. *The uninformed algorithm* $\mathcal{A}_0$ *has the property that, for every problem instance* $\mathcal{T}$*,* $\text{avg}_{\pi\in\Pi(\mathcal{T})}\text{search\_cost}(\mathcal{A}_0,\pi) = O(c_{\text{inf}}(\mathcal{T})\cdot(h+1-\lg(c_{\text{inf}}(\mathcal{T}))))$*, where* $c_{\text{inf}}(\mathcal{T})$ *denotes the minimum, over all informed algorithms* $\mathcal{B}$*, of* $\text{avg}_{\pi\in\Pi(\mathcal{T})}$ $\text{probe\_cost}(\mathcal{B},\pi)$.

Proof. Suppose that input $\mathcal{T}$ has k goals. By Theorem 6, $c_{\text{inf}}(\mathcal{T})$ is $\Omega(2^h/k)$. Furthermore, it is easy to see from the proof of Theorem 8 that the average, over all presentations $\pi\in\Pi(\mathcal{T})$, of the total search cost of $\mathcal{A}_0$ on presentation π is
$O(\frac{2^h}{k}(1+h-\lg(\frac{2^h}{k}))) = O(c_{\text{inf}}(\mathcal{T})\cdot(h+1-\lg(c_{\text{inf}}(\mathcal{T}))))$. □

Following Corollary 9, it is easy to see that the competitive bound in Theorem 10 holds for the expected search cost of Algorithm $\mathbf{A}_1$ as well. This should be contrasted with the $O(c_{\text{inf}}(\mathcal{T})\cdot 4^h)$ bound, given by Theorem 23 of McGregor *et al.* [14], for the expected cost of their uninformed search strategy in this same situation.

3 The Case Where Goals May Appear on Many Different Levels

To this point we have assumed that all problem instances have the property that all goals lie on one fixed level h. In this section we develop a dovetailing strategy that allows us to relax this assumption.

We have already noted that the uninformed algorithm $\mathcal{A}_0$ described in Theorem 8 is competitive (in terms of expected total search cost), to within a factor of at most h, with the best fully informed algorithm, for input instances all of whose goals lie on level h. For more general instances, we first generalize Theorem 6, establishing a lower bound on the intrinsic total expected search cost, and then show how algorithm $\mathcal{A}_0$ can be modified to minimize its competitive ratio with this bound. We then argue that the competitive ratio achieved by this modified uninformed algorithm cannot be improved, by more than a logarithmic factor, even by an algorithm that is cost-informed (that is, is constrained only to work correctly for problem instances of a known bounded intrinsic cost).

Theorem 11. *For every deterministic instance-informed algorithm $\mathcal{A}$, and every problem instance $\mathcal{T}$ with exactly k_t goals at level t,*
$\operatorname{avg}_{\pi\in\Pi(\mathcal{T})}\text{search\_cost}(\mathcal{A},\pi) \geq \min_{t\geq 0}\{t+\frac{2^t}{2k_t}\}/2$.

Proof. Let $\mathcal{T}$ be any problem instance with exactly k_t goals at level t and let $\mathcal{A}$ be any informed goal-searching algorithm. Suppose $\mathcal{A}$ makes p_t probes at level t, and let $m=\max\{t \mid p_t>0\}$ and $p=\sum_{t\geq 0}p_t$. We consider the fraction of the presentations of $\mathcal{T}$ that take some goal to some probe location. We can restrict our attention to the 2^{2^m-1} presentations of $\mathcal{T}$ truncated at level m. By the argument in Theorem 4, at most $p_t k_t \frac{2^{2^m-1}}{2^t}$ presentations take a goal on level t to a probe on level t. Thus at most a fraction $\sum_{t=0}^{m}\frac{p_t k_t}{2^t} \leq p/\min_{t\leq m}\{\frac{2^t}{k_t}\}$ of the presentations of $\mathcal{T}$ take some goal to some probe location. It follows that if $p<\min_{t\leq m}\{\frac{2^t}{k_t}\}/2$ then $\mathcal{A}$ fails to detect a goal for at least half of the presentations of $\mathcal{T}$. Thus, any deterministic algorithm must make at least $\min_{m\geq 0}\max\{m,\min_{t\leq m}\{\frac{2^t}{k_t}\}/2\}=\min_{t\geq 0}\max\{t,\frac{2^t}{2k_t}\}\geq \min_{t\geq 0}\{t+\frac{2^t}{2k_t}\}/2$ probes on at least half of the presentations of $\mathcal{T}$. □

Algorithm $\mathcal{A}_0$, as described in the proof of Theorem 8, makes 2^r equally spaced probes, for increasing values of r, at one fixed level h, at a total cost of $2^r(h-r+1)$. To spread the cost equitably among levels we formulate a modification $\mathcal{A}_2$ of algorithm $\mathcal{A}_0$ that, for increasing values of r, probes all 2^r nodes at level r, and makes 2^{r-i} equally spaced probes at all 2^i levels in the interval $(r-2+2^i, r-2+2^{i+1}]$, for $1\leq i<r$.

Algorithm $\mathcal{A}_2$ effectively simulates algorithm $\mathcal{A}_0$, for all values of h. The total cost of algorithm $\mathcal{A}_2$, up to a fixed value r_0 of the parameter r, is $(r_0+1)2^{r_0}$. Let $t_0=\arg\min_{t\geq 0}\{(t+1)2^t/k_t\}$. Then, from the proof of Theorem 8, we know that the fraction of presentations for which algorithm $\mathcal{A}_2$ requires more than $2^j 2^{t_0}/k_{t_0}$ probes on level t_0 before hitting a goal is less than $(\frac{1}{e})^{2^j}$. It follows that the average number of probes made on level t_0 before hitting a goal on that level is $O(2^{t_0}/k_{t_0})$ and the average total search cost of algorithm $\mathcal{A}_2$ is $O((r_0+1)2^{r_0+1})$, provided $2^{r_0}\geq (t_0-r_0+1)2^{t_0}/k_{t_0}$.

We summarize this result in the following:

Theorem 12. *The uninformed algorithm $\mathcal{A}_2$ has the property that, for every problem instance $\mathcal{T}$ with exactly k_t goals at level t,*
$\operatorname{avg}_{\pi\in\Pi(\mathcal{T})}\text{search\_cost}(\mathcal{A}_2,\pi)\leq \min_{t\geq 0}\{(t+1)\frac{2^t}{k_t}\}\cdot \lg(\min_{t\geq 0}\{(t+1)\frac{2^t}{k_t}\})$.

When $k_{t_0}=2^{t_0}/t_0$ and $k_t=0$, when $t\neq t_0$, the ratio of the $O(t_0^2\lg t_0)$ average search cost of Algorithm $\mathcal{A}_2$ (given by Theorem 12) and the $\Omega(t_0)$ lower bound on the same cost for any *instance-informed* algorithm (given by Theorem 11), is maximized. It turns out that at least a quadratic cost inflation is unavoidable, even for cost-informed algorithms:

Theorem 13. *For every cost $c \geq 0$, there is a family $\mathcal{F}$ of problem instances, each member of which can be searched with worst-case total search cost at most c by some fully informed deterministic search algorithm, such that any cost-informed search algorithm $\mathcal{A}$ must have average, over all input presentations, total search cost at least $\Omega(c^2)$, on at least half of the instances in the family.*

Proof. (Sketch) $\mathcal{F}$ includes instances $\mathcal{T}_i$ with $2^{i+1}/(c-i)$ goals equally-spaced on level i. For each such instance $(c-i)/2$ probes at level i (and at most c total search cost) suffices in the worst case, by a instance-informed algorithm (cf. Theorem 3), and $(c-i)/8$ probes at level i are necessary on average (by Theorem 6). □

4 General Symmetric Trees

To this point we have restricted our attention to full binary trees. Not surprisingly, all of our results generalize to arbitrary symmetric trees. There are some subtleties, however, arising both from nodes with just one child, which can be used to form trees whose number of leaves is significantly smaller than the number of internal nodes, and nodes with a very large number of children, which complicate our round-based algorithms. In the remainder of this section, we outline our generalized results.

We denote by $D_{i,j}$ the expression $\prod_{\ell=i}^{j} d_\ell$, where d_ℓ, recall, denotes the number of children of all internal nodes at level ℓ. Clearly, the number of nodes at level h is now $D_{0,h-1}$, and Observation 1 generalizes to the following:

Observation 14. *If $\mathcal{T}$ is a general symmetric tree of height h then $|\Pi(\mathcal{T})| = \prod_{j=0}^{h-1} d_j^{D_{0,j-1}}$.*

Using this, Theorems 4, 6 and 7 generalize directly to arbitrary symmetric trees, with 2^h replaced by $D_{0,h-1}$. Theorem 8 generalizes in the same way, by a relatively straightforward modification of algorithm $\mathcal{A}_0$:

Theorem 15. *There is a deterministic uninformed algorithm $\mathcal{A}_0$ such that, for every problem instance $\mathcal{T}$ with exactly k goals at level h,*
$\text{avg}_{\pi \in \Pi(\mathcal{T})} \text{probe\_cost}(\mathcal{A}_0, \pi) = O(D_{0,h-1}/k)$.

The next theorem gives a generalization of Theorem 11. It should be noted that our analysis presented here sacrifices comprehensiveness for brevity; it is possible to tighten the analysis to better exploit the situation where the degrees on many successive levels are all one (giving rise to subtrees whose number of leaves is far exceeded by their number of internal nodes).

Theorem 16. *For every deterministic instance-informed algorithm $\mathcal{A}$, and every problem instance $\mathcal{T}$ with exactly k_t goals at level t,*
$\text{avg}_{\pi \in \Pi(\mathcal{T})} \text{search\_cost}(\mathcal{A}, \pi) = \Omega(\min_{t \geq 0}\{t + D_{0,t-1}/k_t\})$.

Next, we give a generalization of Theorem 12. We begin by describing algorithm $\mathcal{A}_3$, the general tree variant of binary tree search algorithm $\mathcal{A}_2$. We dovetail, as in Theorem 12, but in rounds that are partitioned into sub-rounds. Let $\sigma_r = \sum_{0 \le j \le r} D_{0,j-1}$, the total number of nodes of $\mathcal{T}$ on levels 0 through r. After round $r \ge 0$, the tree $\mathcal{T}$ has been completely searched up to level r, at a cost of σ_r. In addition, for $0 \le j < \lg D_{0,r-1}$, $D_{0,r-1}/2^j$ nodes on all levels in the interval $(r - 1 + \sigma_r/D_{0,r-1}2^j, r - 1 + \sigma_r/D_{0,r-1}2^{j+1}]$ have been searched, at an additional total cost of $\sigma_r \lg D_{0,r-1}$.

More generally, after sub-round s of round r, $s+1$ of the d_{r-1} children of each node on level $r-1$ have been probed, at a cost of $\sigma_{r-1} + (s+1)D_{0,r-2}$. In addition, for $0 \le j < \lg((s+1)D_{0,r-2})$, $(s+1)D_{0,r-2}/2^j$ nodes on all levels in the interval $(r - 2 + (\sigma_{r-1} + (s+1)D_{0,r-2})/((s+1)D_{0,r-2})2^j, r - 2 + (\sigma_{r-1} + (s+1)D_{0,r-2})/((s+1)D_{0,r-2})2^{j+1}]$ have been searched, at an additional total cost of $(\sigma_{r-1} + (s+1)D_{0,r-2}) \lg((s+1)D_{0,r-2})$.

Theorem 17. *The uninformed algorithm $\mathcal{A}_3$ has the property that, for every problem instance $\mathcal{T}$ with exactly k_t goals at level t,* $\mathrm{avg}_{\pi \in \Pi(\mathcal{T})} \mathrm{search\_cost}(\mathcal{A}_3, \pi)$ $= O(\min_{t \ge 0}\{(t+1)\frac{D_{0,t-1}}{k_t}\} \cdot \lg(\min_{t \ge 0}\{(t+1)\frac{D_{0,t-1}}{k_t}\}))$.

Contrasting Theorems 16 and 17, we obtain competitive bounds comparable to those achieved in the case of binary trees; of course, the competitive limit captured by Theorem 13 still applies.

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Alpern, S., Gal, S.: The Theory of Search Games and Rendezvous. Kluwer Academic Publishers, Dordrecht (2003)
2. Azar, Y., Broder, A.Z., Manasse, M.S.: On-line choice of on-line algorithms. In: Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 432–440 (1993)
3. Baeza-Yates, R.A., Culberson, J.C., Rawlins, G.J.E.: Searching in the plane. Inform. Comput. 106(2), 234–252 (1993)
4. Demaine, E., Fekete, S., Gal, S.: Online searching with turn cost. Theoret. Comput. Sci. 361, 342–355 (2006)
5. Fleischer, R., Kamphans, T., Klein, R., Langetepe, E., Trippen, G.: Competitive online approximation of the optimal search ratio. In: Proc. 12th Annual European Symposium on Algorithms, pp. 335–346 (2004)
6. Kao, M.-Y., Littman, M.L.: Algorithms for informed cows. In: AAAI 1997 Workshop on On-Line Search (1997)
7. Kao, M.-Y., Ma, Y., Sipser, M., Yin, Y.: Optimal constructions of hybrid algorithms. J. Algorithms 29(1), 142–164 (1998)
8. Kao, M.-Y., Reif, J.H., Tate, S.R.: Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. Inform. Comput. 131(1), 63–79 (1996)

9. Kenyon, C.: Best-fit bin-packing with random order. In: Proc. 7th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 359–364 (1996)
10. Kirkpatrick, D.: Hyperbolic dovetailing. In: Proc. 17th Annual European Symposium on Algorithms, pp. 516–527 (2009)
11. Koutsoupias, E., Papadimitriou, C., Yannakakis, M.: Searching a fixed graph. In: Proc. 23rd International Colloquium on Automata, Languages and Programming, pp. 280–289 (1996)
12. Lopez-Ortiz, A., Schuierer, S.: The ultimate strategy to search on ï rays. Theoret. Comput. Sci. 261, 267–295 (2001)
13. Luby, M., Sinclair, A., Zuckerman, D.: Optimal speedup of Las Vegas algorithms. In: Proc. Second Israel Symposium on Theory of Computing and Systems, Jerusalem, pp. 128–133 (June 1993)
14. McGregor, A., Onak, K., Panigrahy, R.: The oil searching problem. In: Proc. 17th Annual European Symposium on Algorithms, pp. 504–515 (2009)
15. Papadimitriou, C.H., Yannakakis, M.: Shortest paths without a map. In: Ronchi Della Rocca, S., Ausiello, G., Dezani-Ciancaglini, M. (eds.) ICALP 1989. LNCS, vol. 372, pp. 610–620. Springer, Heidelberg (1989)
16. Schönhage, A.: Adaptive raising strategies optimizing relative efficiency. In: Proc. 30th International Colloquium on Automata, Languages and Programming, pp. 611–623 (2003)
17. Schuierer, S.: Lower bounds in on-line geometric searching. Comp. Geom. 18, 37–53 (2001)
18. Schuierer, S.: A lower bound for randomized searching on ï rays. In: Klein, R., Six, H.-W., Wegner, L. (eds.) Computer Science in Perspective. LNCS, vol. 2598, pp. 264–277. Springer, Heidelberg (2003)
19. `http://www2.cs.uregina.ca/~zilles/kirkpatrickZ11b.pdf`