Hyperbolic Dovetailing

David Kirkpatrick

University of British Columbia, Vancouver, Canada
kirk@cs.ubc.ca

Abstract. A familiar quandary arises when there are several *possible* alternatives for the solution of a problem, but no way of knowing which, if any, are viable for a particular problem instance. Faced with this uncertainty, one is forced to simulate the parallel exploration of alternatives through some kind of co-ordinated interleaving (*dovetailing*) process. As usual, the goal is to find a solution with low total cost. Much of the existing work on such problems has assumed, implicitly or explicitly, that at most one of the alternatives is viable, providing support for a competitive analysis of algorithms (using the cost of the unique viable alternative as a benchmark). In this paper, we relax this worst-case assumption in revisiting several familiar dovetailing problems.

Our main contribution is the introduction of a novel process interleaving technique, called *hyperbolic dovetailing* that achieves a competitive ratio that is within a logarithmic factor of optimal on *all* inputs in the worst, average and expected cases, over all possible deterministic (and randomized) dovetailing schemes. We also show that no other dovetailing strategy can guarantee an asymptotically smaller competitive ratio for all inputs.

An interesting application of hyperbolic dovetailing arises in the design of what we call *input-thrifty* algorithms, algorithms that are designed to minimize the total precision of the input requested in order to evaluate some given predicate. We show that for some very basic predicates involving real numbers we can use hyperbolic dovetailing to provide input-thrifty algorithms that are competitive, in this novel cost measure, with the best algorithms that solve these problems.

1 Introduction

You are trapped underground in a mine following a massive earthquake. While there are many potential escape routes, you have no way of knowing how much effort will be required to clear any one of them. You quickly realize the first exploration strategies that come to mind might be poor choices for your particular situation. If it happens that all of the escape routes have about the same amount of debris, then you may as well simply choose one and start digging. On the other hand, if only a few of routes are viable, it makes sense to keep trying all of the routes, more or less equitably, until one of these few is discovered.

Upon further reflection, you recognize that being trapped in this way is not such an uncommon occurrence. Being more adept at thinking than digging, you wonder if there are strategies that are arguably good, or even best, to adopt in situations like this...

1.1 Dovetailed Execution of Multiply-Viable Process Sets

Finding an escape route in a mine has much in common with problems that arise in many computational settings where several possible avenues are available for the solution of a problem but there is no way of knowing which, if any, are viable (or adequately efficient) for a particular problem instance. Faced with this uncertainty, we elect to simulate their parallel exploration through some kind of co-ordinated interleaving (*dovetailing*) of their associated processes. The goal, of course, is to find a solution with low total cost. (Examples include geometric or graph search, the synthesis of hybrid algorithms based on a suite of heuristics, and adaptive raising strategies.) Existing work on such problems invariably makes the assumption that at most one of the processes is viable. This provides support for a competitive analysis of algorithms (using the cost of running the unique viable process alone as a benchmark). Algorithms with optimal competitive ratios, based on variants of round-robin doubling search, have been formulated for a large variety of such problems [2,6,8,9,10,13,16].

Competitive analysis was introduced to provide a more realistic alternative to worst-case analysis, which in our setting would make all strategies equivalent since each could, with sufficiently bad input, be forced to run an arbitrarily long time. However, it is similarly unrealistic in many scenarios, including those mentioned above, to adopt the worst-case assumption that at most one of the underlying process is viable.

In this paper, we relax this assumption in revisiting some of these dovetailing problems. Our contributions are of four types: (i) we formulate a natural notion of intrinsic cost that provides a basis for a modified form of competitive analysis that can be applied in this more general setting; (ii) we introduce a non-uniform process interleaving technique, called *hyperbolic dovetailing*, that can be viewed as a hybrid (or mixture) of a family of bounded depth-first strategies covering the full spectrum between a pure breadth-first (i.e. round robin) and a pure depth-first strategy, (iii) we prove that hyperbolic dovetailing achieves a competitive ratio that is within a logarithmic factor of optimal on *all* inputs in the worst, average and expected cases, over all possible deterministic (and randomized) dovetailing schemes, and (iv) we prove that no other dovetailing strategy can guarantee an asymptotically smaller competitive ratio for all inputs.

Our work was motivated, in part, by the need to develop efficient algorithms in computational settings in which knowledge about the input is incomplete but extendable. For instance, input numbers may initially be known only up to some precision, and additional precision may be obtainable, but only at a high cost. Here are two examples for such a scenario: 1) the input numbers are initially the result of cheap measurements, and with further, more elaborate measurements additional precision can be obtained; 2) the input numbers are produced bit by bit by a computational process such as root-finding via bisection. In such scenarios it is of interest to design algorithms that solve the problem using as little total input precision as possible.

We show that the hyperbolic dovetailing technique can be applied to the design of algorithms for certifying certain simple properties of a set of input numbers, with the goal of minimizing the number of input bits that need to be examined by the algorithm, for each set of possible inputs. We refer to this as the *leading-input-bits-cost*, or LIB-cost, of the algorithm. Algorithms whose LIB-cost is "small", in some quantifiable sense, relative to the intrinsic LIB-cost for every input, are said to be *input-thrifty* algorithms.

1.2 Multi-list and Cow-Path Traversal Problems

The following *multi-list traversal* problem captures the essence of the mine-escape-route search problem; it is a simplified version of other multi-process dovetailing problems that we wish to study and, as such, it allows us to introduce our strategies and analyses in their most basic setting. Let $\mathcal{L}$ be a given multi-list, that is a sequence of m not-necessarily-finite-length lists. Let λ_i denote the length of the i-th longest list in $\mathcal{L}$, so $\infty \geq \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m$. The goal is to traverse at least one list to its end, while minimizing the total *cost* (the number of list positions examined). In general, we assume that the (multi)set of list lengths $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ is *not* known to the strategy, and we do worst-case or average-case analysis over the set of all multi-lists with associated lengths set Λ (what we refer to as *presentations* of Λ). However, for the sake of comparison, we also consider the the behaviour of strategies that know Λ (so, they are not constrained to run efficiently, or even terminate correctly, for multi-lists outside of this restricted class).

Our multi-list traversal problem bears a strong resemblance to what has come to be known as the *m-lane cow-path problem*. An instance of the 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 to formulate a provably good strategy for an agent (*cow*) to reach the goal, starting from the origin.

The cow-path problem was introduced by Baeza-Yates *et al.* [2] as a simple graph-theoretic abstraction of search problems in the plane. It has been studied in several variations including directionally dependent traversal costs, turnaround penalties, shortcuts and dead-ends [6,8,12,13,15]. It has also been analysed in terms of worst-case and average-case competitive ratio (using the distance d as a benchmark), as well as in a game-theoretic framework [2,9,10,16,17,18].

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 optimal (or near optimal) competitive ratios [1,9]. The setting is one in which there are a number of *basic* algorithms 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. In this context, memory limitations may impose restrictions on the number of processes that can be suspended at any given time (the alternative being a complete restart with successively larger computation bounds).

As we have already suggested, the multi-list traversal problem provides a natural generalization of the cow-path problem in which *every ray* leads to a goal (perhaps arbitrarily far from the origin) Of course, the list traversal cost is not the same as that employed in the cow-path problem; in particular, it does not take into account the cost of re-traversal of paths. However, as we shall see later, our multi-list traversal algorithm can be implemented in such a way that the two traversal cost functions agree to within a small constant factor, without increasing the total list traversal cost by more than a small constant factor.

1.3 Competitive Analysis

In order to formulate a compelling notion of competitiveness (in the spirit of [19]) for our multi-list traversal problems, we need to specify some measure of *inherent complexity* that permits us to distinguish easy from more difficult problem instances. The objective, of course, is to formulate *adaptive* strategies that run relatively more efficiently on instances of relatively low inherent complexity.

One natural candidate is simply the length of the shortest list. Since this corresponds to the length of the shortest proof that some specified list has some specified length, we refer to this as the *minimum certification cost* of the instance. Clearly this provides a lower bound on the cost of any multi-list traversal strategy, and is realizable by a strategy that knows (or correctly guesses) the identity of the shortest list. It also coincides with the notion of inherent complexity that underlies the competitive analysis used for the conventional m-lane cow-path problem.

In general, however, traversal strategies must deal with uncertainty in both the *set* Λ of numbers that correspond to the path lengths (and completely determine the minimum certification cost) and the *presentation* of these numbers as a sequence (that is their assignment to individual paths). For a given set Λ, we specify the maximum-traversal-cost of an algorithm $\mathcal{A}$ *for* Λ, and then define the intrinsic maximum-traversal-cost of Λ to be the minimum, over all algorithms $\mathcal{A}$ that are only guaranteed to solve the problem for multi-lists that are presentations of Λ, of the maximum-traversal-cost of $\mathcal{A}$ for Λ. These definitions resemble refinements of competitive analysis (in particular, the so-called relative worst order ratio) introduced to provide a more meaningful/sensitive analysis for certain on-line algorithms [3,4,5,11]. (See [7] for a comprehensive overview of these alternative measures.)

It turns out to be reasonably straightforward to specify the intrinsic maximum-traversal-cost of an arbitrary input set Λ in this way. With this in hand, we do competitive analysis of two types: (i) with respect to the family of algorithms that are constrained to answer correctly only when the input multi-list is a presentation of Λ (i.e. relative to the intrinsic maximum traversal-cost); and (ii) with respect to the family of algorithms that are constrained to answer correctly only when its input multi-list is a presentation of some input set Λ' whose intrinsic maximum-traversal-cost is the same as that of Λ.

In the former case, we show that our algorithms are competitive to within a logarithmic factor. More precisely, for every set Λ, if $\xi(\Lambda)$ denotes the intrinsic maximum-traversal-cost of input set Λ, then our algorithm achieves maximum-traversal-cost $O(\xi(\Lambda) \log \min\{\xi(\Lambda), |\Lambda|\})$. In the latter case, we show that our algorithms are competitive in maximum-traversal-cost to within a constant factor. We also develop similar results for average-traversal-cost.

Results that have striking similarities to those in this paper were presented by Luby *et al.* [14] for the problem of minimizing the expected time to complete the execution of Las Vegas algorithms (which can be viewed as an infinite sequence of deterministic algorithms with unknown completion times).

2 Multi-list Traversal Strategies

Let Λ be a (multi)set of m (positive) list lengths, and let λ_i denote the ith largest element of Λ, for $1 \leq i \leq m$. We denote by $\mathcal{L}$ a generic presentation of Λ (that is, a multi-list whose associated set of list lengths coincides with Λ).

2.1 Intrinsic Maximum-Traversal-Cost and Average-Traversal-Cost

We define the maximum-traversal-cost of some multi-list traversal strategy $\mathcal{A}$ on Λ to be the maximum cost of $\mathcal{A}$ over all multi-list presentations of Λ. The intrinsic maximum-traversal-cost of Λ, denoted $\xi(\Lambda)$, is the minimum, over all multi-list traversal strategies $\mathcal{A}$ of the maximum-traversal-cost of $\mathcal{A}$ on Λ.

Theorem 1. $\xi(\Lambda) = \min_{1 \leq i \leq m} i\lambda_i$.

Proof. Consider any multi-list traversal strategy $\mathcal{A}$ that works correctly on all presentations of Λ. Such a strategy specifies a sequence of traversal steps that culminates in the discovery of the end of some list. At any point in time, up to termination, the strategy has traversed the i-th list to some depth d_i. Provided that $\hat{d}_i$, the i-th largest element of $\{d_1, \ldots, d_m\}$, satisfies $\hat{d}_i < \lambda_i$, for $1 \leq i \leq m$, the strategy will have not terminated for at least one multi-list presentation of Λ. Thus, at termination, we must have $\hat{d}_i = \lambda_i$, for some i, and $\hat{d}_j \geq \lambda_i$, for all j, $1 \leq j < i$. It follows that $\mathcal{A}$ must explore at least $\min_{1 \leq i \leq m} i\lambda_i$ list positions.

On the other hand, if $i_\Lambda = \text{argmin}_{1 \leq i \leq m} i\lambda_i$, then the strategy that explores lists to fixed depth λ_{i_Λ}, in any sequence, will never explore more than $\min_{1 \leq i \leq m} i\lambda_i$ list positions. □

In the average case, where the average is taken over all presentations of the given length set Λ, we can hope to get away with something considerably less than the intrinsic maximum-traversal cost. We note that the average case behaviour of any deterministic multi-list traversal strategy is realized as the expected behaviour of a two-phased randomized algorithm that first randomly permutes the indices of the lists in the input multi-list and then continues in an entirely deterministic fashion. Thus, a lower bound on the expected cost of any randomized list-traversal strategy is also a lower bound on the average-case cost of (deterministic) list traversal.

Consider first the fixed-depth traversal strategy (denoted FDT(d)), already encountered, that explores the lists, one after another, to some fixed depth d, stopping if and when some list is completely traversed.

Lemma 1. *Strategy* FDT*(λ_k) solves the multi-list traversal problem, with average cost (over all presentations of Λ) at most $\lambda_k(m+1)/(m-k+2)$.*

Proof. By definition, at least $m-k+1$ of the lists have length at most λ_k. Since lists are explored to depth λ_k, it suffices to argue that, among all possible permutations of the input lists (i.e. all possible presentations of Λ) the average position of the first list with length at most λ_k is at most $(m+1)/(m-k+2)$. (Equivalently, this is the expected position of the first 1 in a random permutation of a binary string of length m containing $n-k+1$ 1's.) □

Let $\bar{i}_\Lambda = \arg\min_{1\le i\le n}\{\lambda_i/(m-i+2)\}$. Then, assuming that Λ is known, a strategy, namely $\mathrm{FDT}(\lambda_{\bar{i}_\Lambda})$, exists that solves the list traversal problem with average cost at most $m\lambda_{\bar{i}_\Lambda}/(m-\bar{i}_\Lambda+2)$. Thus the *intrinsic average-traversal-cost* of the list length set Λ, which we denote by $\bar{\xi}(\Lambda)$, is at most $m\lambda_{\bar{i}_\Lambda}/(m-\bar{i}_\Lambda+2)$.

As it turns out, this bound is essentially tight:

Lemma 2. *Any randomized multi-list traversal strategy $\mathcal{B}$ that terminates after at most $m\lambda_{\bar{i}_\Lambda}/3(m-\bar{i}_\Lambda+2)$ steps on all presentations of Λ, fails with probability at least one half on a random presentation of Λ.*

Proof. Since an adversary is free to choose the least favourable list indexing, the expected cost of strategy $\mathcal{B}$, forced by an adversary, is at least that which is required for a random permutation of the input lists. Hence, we can assume that $\mathcal{B}$ behaves the same on all list orderings (without loss of generality, it begins by randomly permuting the lists) and thus the i-th list has length $\lambda_{\pi(i)}$, for some random permutation π.

Denote the expression $m\lambda_{\bar{i}_\Lambda}/(m-\bar{i}_\Lambda+2)$ by c_Λ. To prove the desired result, it suffices to argue that any randomized list-traversal strategy that has been modified to terminate after exploring at most $c_\Lambda/3$ list positions, must fail (i.e. not complete the traversal of any list) with probability at least 1/2. We note that any such truncated list-traversal strategy can be interpreted as a probability distribution over the set of all (deterministic) traversals where list i is traversed to depth d_i and $\sum_{1\le i\le n} d_i \le c_\Lambda/3$.

Because the list indices are assigned randomly, we can assume, without loss of generality, that $d_1 \ge d_2 \ge \ldots \ge d_m$. We will argue that every such $(d_1, d_2, \ldots, d_m)$-traversal fails with probability at least 1/2. Let $\hat{\Lambda} = \{\hat{\lambda}_1, \ldots, \hat{\lambda}_m\}$, where $\hat{\lambda}_i = (m-i+2)c_\Lambda/m$, for $1 \le i \le m$. Since $\hat{\lambda}_i \le \lambda_i$, for $1 \le i \le m$, and $\bar{\xi}(\hat{\Lambda}) = \bar{\xi}(\Lambda)$, it suffices to prove the result under the assumption that $\Lambda = \hat{\Lambda}$. Furthermore, since the strategy need only work for presentations of Λ, there is no loss of generality in assuming that the strategy exploits the knowledge that all of the λ_i values are integral multiples of c_Λ/m and thus the exploration depth values $d_1, d_2, \ldots, d_m$ satisfy $d_i = k_i c_\Lambda/m$, for some integers $k_1 \ge k_2 \ldots \ge k_m \ge 0$.

Since $\sum_i k_i = \sum_i d_i m/c_\Lambda \le m/3$, it follows that $k_i = 0$, for $m/3 < i \le m$. Furthermore, since $\lambda_j > d_i$ just when $j < m - k_i + 2$, at least $m - k_i$ of the lists have length greater than d_i. Thus, $\Pr(\text{failure}) = \prod_{j=1}^{m} \Pr(\lambda_{\pi(i)} > d_j) \ge \prod_{j=1}^{m}\left(\frac{m-j+1-k_j}{m-j+1}\right) = \prod_{j=1}^{m/3}\left(\frac{m-j+1-k_j}{m-j+1}\right)$. If we relax the constraint that $k_i \ge k_j$, for $i \leq j$, the expression $\prod_{j=1}^{m/3} \left(\frac{m-j+1-k_j}{m-j+1}\right)$ is minimized when $k_{m/3} = m/3$ and $k_j = 0$, for $j < m/3$. Hence, Pr(failure) $\geq 1/2$. □

Corollary 1. *The average cost of any deterministic multi-list traversal strategy, over all presentations of Λ, is at least $m\lambda_{\bar{i}_\Lambda}/6(m - \bar{i}_\Lambda + 2)$, even if Λ is known.*

Theorem 2. $\bar{\xi}(\Lambda) = \Theta(m\lambda_{\bar{i}_\Lambda}/(m - \bar{i}_\Lambda + 2))$.

2.2 Competitive Ratio of Conventional Dovetailing Strategies

It is instructive to analyse the competitive ratios achieved by two familiar dovetailing strategies applied to multi-list traversal. These correspond to the extremes of uniformity in traversal of lists; breadth-first (usually called round-robin) traversal explores all lists in a completely equitable fashion and depth-first traversal chooses one list and explores it to its end.

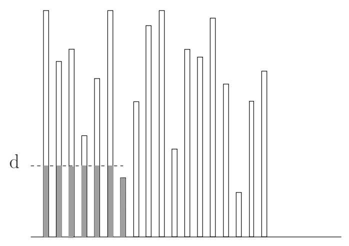
Theorem 3. *For both the maximum and average-traversal-costs, the competitive ratio of breadth-first traversal is $\Omega(m)$ and the competitive ratio of depth-first traversal is unbounded.*

Proof. Suppose first that we have a multi-list whose associated length set Λ consists of one finite value d and $m-1$ infinite (or arbitrarily large) values. In this case, it is easy to confirm that the intrinsic maximum-traversal-cost of Λ, $\xi(\Lambda)$, satisfies $\xi(\Lambda) = md$ and the intrinsic average-traversal-cost of Λ, $\bar{\xi}(\Lambda)$, satisfies $\bar{\xi}(\Lambda) = \Theta(d)$. This coincides with the familiar observation that an adversary can "hide" the identity of the sole finite list until all other lists have been explored to depth at least d. Breadth-first traversal achieves maximum-traversal-cost and average-traversal-cost $\Theta(md)$, but depth-first traversal has an unbounded cost in this case, for both maximum-traversal-cost and average-traversal-cost.

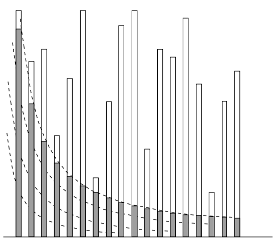
On the other hand, if we have a multi-list whose associated length set Λ consists of m lists all of which have the same length d, then $\xi(\Lambda)$ and $\bar{\xi}(\Lambda)$ are both $\Theta(d)$ (despite the fact that the minimum certification cost d is the same as in the previous example). In this case the intrinsic maximum-traversal-cost is achieved by any depth-first traversal but breadth-first traversal has maximum-traversal-cost and average-traversal-cost $\Theta(md)$. □

2.3 Hyperbolic Dovetailing

We now introduce a novel multi-list traversal strategy that makes use of a technique that we call *hyperbolic dovetailing*. The strategy maintains an indexing of all the list positions in the input multi-list and, in terms of this indexing, defines a *rank* function on the next unexplored position of each list. The traversal explores positions in order of increasing rank until some list is exhausted. In general, the rank of the t-th position of the i-th list (in the input order) is just the product ti. Thus if we view the list positions as points in the plane (where the t-th position of the i-th list has coordinates (i, t)), the positions are examined in the order encountered by a hyperbola $t = c/i$, for increasing values of c (see Fig. 1(b)). This interpretation explains the term "hyperbolic dovetailing".



(a) Bounded depth-first traversal



(b) Hyperbolic traversal

Fig. 1.

```
HYPERBOLIC-TRAVERSAL
    $c \leftarrow 1$;
    **repeat** until some list is fully explored
           **for** $i = 1$ to $m$
               **do** continue exploration of list $i$ up to depth $\lfloor c/i \rfloor$;
           increment $c$;
```

2.4 Worst-Case Competitive Case Analysis of Hyperbolic Traversal

Here we argue that hyperbolic traversal is competitive, up to a logarithmic factor, even against strategies that know the set of list lengths.

Theorem 4. *The hyperbolic-traversal strategy solves the multi-list traversal problem for a multi-list with associated length set Λ, with maximum-traversal-cost $O(\xi(\Lambda) \lg \min\{|\Lambda|, \xi(\Lambda)\})$.*

Proof. Suppose that the last list position explored in the hyperbolic traversal has rank c. Then (i) $c \leq \xi(\Lambda)$, and (ii) the total number of list positions traversed is $O(c \lg \min\{m, c\})$. Point (i) follows from Theorem 1. Point (ii) follows from the fact that, according to our hyperbolic traversal, all of the explored positions in each list have rank at most c and, since there are at most $\lfloor c/t \rfloor$ positions of rank at most c in the t-th list, the number of list positions explored is bounded above by $\sum_{t=1}^{m} \lfloor c/t \rfloor \leq \sum_{t=1}^{\min\{m,c\}} c/t \leq c + \int_{1}^{\min\{m,c\}} \frac{c}{t} dt = c(1 + \ln \min\{m, c\})$. □

Furthermore, the logarithmic competitiveness bound is the best that one could hope for in a general strategy.

Theorem 5. *Any deterministic list-traversal strategy that behaves correctly on all multi-list presentations of length sets Λ satisfying $\xi(\Lambda) = \xi_0$, must have maximum-traversal-cost at least $\xi_0 \ln \min\{|\Lambda|, \xi_0\}$, on at least one such presentation.*

Proof (Sketch). We define a family of canonical input length sets Λ, with $\xi(\Lambda) = \xi_0$ and argue that any deterministic traversal strategy that explores fewer than $\xi_0 \ln \xi_0$ list positions must, in the worst case, fail to complete the traversal of at least one presentation of some Λ in this family. □

2.5 Average and Expected Case Competitive Analysis of Hyperbolic Traversal

We now turn to the average case behaviour of our hyperbolic multi-list traversal strategy. As in the worst case, hyperbolic traversal is competitive, up to a logarithmic factor, even against strategies that know the set of list lengths.

Theorem 6. *The hyperbolic-traversal strategy solves the multi-list traversal problem for a multi-list with associated length set Λ with average-traversal-cost $O(\bar{\xi}(\Lambda) \lg \min\{|\Lambda|, \bar{\xi}(\Lambda)\})$.*

Proof. Let $T^{(j)} = \{\delta_1^{(j)}, \ldots, \delta_m^{(j)}\}$, where $\delta_i^{(j)} = \lambda_j$, for $j \le i \le m$ and $\delta_i^{(j)} = \infty$, for $i < j$. By the monotonicity of average traversal cost, avg-trav-cost$(\Lambda) \le \min_{1 \le j \le m}$ avg-trav-cost$(T^{(j)})$. Given this is suffices to prove that for the hyperbolic-traversal strategy avg-trav-cost$(T^{(j)})$ is $O(\hat{c}_j \ln \hat{c}_j)$, where $\hat{c}_j = \lambda_j m/(m-j+1)$.

By the nature of $T^{(j)}$, any strategy discovers the end of a list at depth exactly λ_j. But, since exactly $j-1$ lists have length greater than λ_j, the probability that the number of lists that need to be explored exceeds $m/(m-j+1)$ (which happens just when the first $m/(m-j+1)$ lists all have length exceeding λ_j) is just $\binom{j-1}{m/(m-j+1)} / \binom{m}{m/(m-j+1)} < \left(\frac{j-1}{m}\right)^{m/(m-j+1)} < \left(\frac{1}{e}\right)$.

Since the hyperbolic-traversal strategy explores $\lfloor c/\lambda_j \rfloor$ lists to depth λ_j when its traversal parameter has the value c, it follows that the event, denoted τ_t, that it fails to terminate by the time its traversal parameter reaches $t\hat{c}_j$ has probability at most $\left(\frac{1}{e}\right)^t$. As we have already seen, if τ_t holds, the total exploration cost is $O(t\hat{c}_j \lg(t\hat{c}_j))$. Thus, the total expected cost is $\sum_{r=-\infty}^{\infty} \sum_{t=2^r}^{2^{r+1}} \Pr(\tau_t) t\hat{c}_j \lg(t\hat{c}_j)) = O(\hat{c}_j \lg(t\hat{c}_j))$. $\square$

If we precede the hyperbolic traversal strategy with a step that randomly permutes the indices of the input lists, we produce a randomized algorithm whose expected cost coincides with the average cost of deterministic hyperbolic traversal. This randomized hyperbolic traversal strategy, which works correctly on *all* input list sets, is essentially optimal even among randomized list traversal strategies whose only constraint is that they succeed with probability at least 1/2 on multi-lists whose associated length sets Λ satisfy $\bar{\xi}(\Lambda) = \bar{\xi}_0$, for some fixed $\bar{\xi}_0$.

Theorem 7. *Any randomized list-traversal strategy that succeeds with probability at least 1/2 on all multi-list presentations of length sets Λ satisfying $\bar{\xi}(\Lambda) = \bar{\xi}_0$, must have expected cost at least $c\bar{\xi}_0 \lg \bar{\xi}_0$, for some fixed constant $c > 0$, on at least one such presentation.*

Proof (Sketch). We view a randomized algorithm as being a distribution over deterministic strategies. Suppose that $\sum_{1 \le i \le n} d_i = \bar{\xi}_0 \lg \bar{\xi}_0/3$. We argue that any $(d_1, d_2, \ldots, d_m)$-traversal, with $d_1 \le d_2 \le \ldots \le d_m$, will fail on at least half of the presentations of at least half of the members of a family of canonical input length sets Λ, with $\bar{\xi}(\Lambda) = \bar{\xi}_0$. It follows that at least one of these sets will, for at least half of its presentations, be incompletely traversed by deterministic strategies whose total assigned probability is at least 1/2.

Corollary 2. *Any deterministic list-traversal strategy that behaves correctly on all multi-list presentations of length sets Λ satisfying $\bar{\xi}(\Lambda) = \bar{\xi}_0$, must have average cost at least $c\bar{\xi}_0 \lg \bar{\xi}_0$, for some fixed constant $c > 0$, on at least one such presentation.*

3 Applications of Hyperbolic Dovetailing

3.1 Generalized Cow-Path Search and Hybrid Algorithm Synthesis

As discussed in the introduction, the multi-list traversal problem provides a well-motivated generalization of cow-path search and hybrid algorithm synthesis. The variations in search cost functions, while significant for the exact competitive analysis of interest in the single goal versions of these problems, are essentially negligible when we consider asymptotic competitiveness bounds. For example, our lower bounds for multi-list traversal obviously still hold in the setting of cow-path search (where search cost is counted for revisiting path locations) but our hyperbolic traversal algorithm can be modified (by re-exploring a list only when the hyperbolic rank function doubles its value from the preceding exploration) so that the total search cost (counting revisits) can be assigned in such a way that every explored location has $O(1)$ charges. Thus all of our multi-list traversal results carry over (with modified asymptotic constants) to these other problems.

Our results also extend to a slightly more general form of multi-list search in which individual lists may contain zero or more *goal* locations and the objective is to traverse at least one list to the location of its first goal. By truncating lists at their first goal, this reduces to what we call the *signed* multi-list traversal problem in which some lists are *positive* (i.e. they have a goal location at their end) and some are *negative* (i.e. they terminate, if at all, in a dead-end). Signed multi-list traversal provides a natural model of dovetailing processes that may terminate without success.

3.2 Input-Thrifty Algorithms

In many natural problem settings the knowledge about the input is incomplete. For instance, input numbers may initially be known only up to some precision, and additional precision may be obtainable, but only at a high cost. In such situations it is of interest to design algorithms that solve the problem using as little total input precision as possible.

Our results on list searching provide a modest but non-trivial contribution to the study of such algorithms, taking the extreme point of view that the computation within such an algorithm is free, and the only cost incurred is the number of input bits that need to be examined by the algorithm. Specifically, for a given algorithm and input sequence, we define the *leading-input-bits-cost*, or LIB-cost, for short, to be the number of input bits that the algorithm examines. We are interested in *input-thrifty algorithms*, i.e. algorithms whose LIB-cost is "small" in some quantifiable sense.

We consider problems whose input is a sequence $p_1, \dots, p_n$ of real numbers in the half-open interval $[0, 1)$. An algorithm can access each such number $p = \sum_{j>0} p^{(j)} 2^{-j}$ via its binary representation $p^{(1)}, p^{(2)}, \cdots$, and this happens by examining the bits $p^{(j)}$ individually *in order of decreasing significance*, i.e. an algorithm can examine bit $p^{(j)}$ only after it has examined bits $p^{(1)}$ through $p^{(j-1)}$ already. In practice, we will assume that each p_i has a finite, although arbitrarily long, representation (thereby guaranteeing finite cost for all inputs).

As a concrete application, suppose we are given a set of s numbers $\{p_1, \dots, p_s\}$. Our goal is to determine if there exists a pair (p_i, p_j), such that $p_i \neq p_j$. We can map this to an instance of the multi-list traversal problem as follows: each number p_i is interpreted as a list of length λ_i, where $\lambda_i = \arg\min\{j \mid p_i^{(j)} \neq p_1^{(j)}\}$, (that is the position of the most significant bit on which p_i and p_1 differ). It should be clear that (i) if the input set contains at least two distinct numbers then the associated multi-list contains at least one finite length list, and (ii) any multi-list traversal scheme corresponds to an not-all-equal certification algorithm in the LIB-cost model.

Note that any not-all-equal certification algorithm can be reformulated in such a way that at least as many bits of input p_1 are explored as any other input, with at most a constant factor increase in the LIB-cost. (In effect this says that in the LIB-cost model the cost of certifying not-all-equal is essentially the same as certifying that some input number differs from one specific input, p_1.) It follows that competitive algorithms for multi-list traversal translate directly to input-thrifty not-all-equal certification algorithms.

Acknowledgements

The author gratefully acknowledges helpful discussions concerning the material of this paper with Rolf Klein, Robert Tseng, and especially Raimund Seidel.

References

1. 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)
2. Baeza-Yates, R.A., Culberson, J.C., Rawlins, G.J.E.: Searching in the plane. Information and Computation 106(2), 234–252 (1993)
3. Ben-David, S., Borodin, A.: A new measure for the study of on-line algorithms. Algorithmica 11, 73–91 (1994)
4. Boyar, J., Favrholdt, L.M.: The relative worst order ratio for on-line algorithms. ACM Trans. on Algorithms 3(2) (2007)
5. Boyar, J., Favrholdt, L.M., Larsen, K.S.: The relative worst order ratio applied to paging. In: Proc. 16th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 718–727 (2005)
6. Demaine, E., Fekete, S., Gal, S.: Online searching with turn cost. Theoretical Computer Science 361, 342–355 (2006)

7. Dorrigiv, R., Lopez-Ortiz, A.: A survey of performance measures for on-line algorithms. ACM SIGACT News 36(3), 67–81 (2005)
8. Kao, M.-Y., Littman, M.L.: Algorithms for informed cows. In: AAAI 1997 Workshop on On-Line Search (1997)
9. Kao, M.-Y., Ma, Y., Sipser, M., Yin, Y.: Optimal constructions of hybrid algorithms. J. Algorithms 29(1), 142–164 (1998)
10. Kao, M.-Y., Reif, J.H., Tate, S.R.: Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. Information and Computation 131(1), 63–79 (1996)
11. Kenyon, C.: Best-fit bin-packing with random order. In: Proc. 7th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 359–364 (1996)
12. Koutsoupias, E., Papadimitriou, C., Yannakakis, M.: Searching a fixed graph. In: Proc. 23rd International Colloquium on Automata, Languages and Programming, pp. 280–289 (1996)
13. Lopez-Ortiz, A., Schuierer, S.: The ultimate strategy to search on m rays. Theoretical Computer Science 2(28), 267–295 (2001)
14. Luby, M., Sinclair, A., Zuckerman, D.: Optimal speedup of Las Vegas algorithms. In: Proc. Second Israel Symposium on Theory of Computing and Systems, June 1993, pp. 128–133 (1993)
15. Papadimitriou, C.H., Yannakakis, M.: Shortest path without a map. In: Proc. 16th International Colloquium on Automata, Languages and Programming, pp. 610–620 (1989)
16. Schonhage, 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. Computational Geometry: Theory and Applications 18(1), 37–53 (2001)
18. Schuierer, S.: A lower bound for randomized searching on m rays. In: Klein, R., Six, H.-W., Wegner, L. (eds.) Computer Science in Perspective. LNCS, vol. 2598, pp. 264–277. Springer, Heidelberg (2003)
19. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Comm. ACM, 202–208 (February 1985)