List Update Posets

Draft, February 6, 1996
(Appendix added December, 2004)

Susanne Albers [1], Bernhard von Stengel [2], Ralph Werchner [3]

1. Introduction

This note discusses partially ordered sets (posets) as a structural device for proving or disproving 3/2 as the optimal competitive factor of a randomized on-line algorithm for the *list update problem* against an oblivious adversary. It represents incomplete work meant for communication to others interested in this problem. This work started in the fall of 1994 when the authors were at ICSI Berkeley. In discussions with Bernhard von Stengel in July 1995 in Berlin, Stefan Felsner raised many of the questions on posets mentioned below.

We use the conventions about the list update problem in [2]: The list is static and has n items. We try to deal only with free exchanges where a requested item may be moved free of charge further to the front after it has been requested. (Any other transposition of two items incurs cost one and is called a paid exchange.) We consider the $i-1$ cost model (also called partial cost model) where the cost of accessing the ith item in the list is $i-1$. With this cost model, the best known lower bound [5] for the competitive factor is exactly 3/2. In the i (or full) cost model, the cost of accessing the ith item in the list is i. There, 3/2 is only an asymptotic lower bound for long lists, since we have to subtract a term proportional to $1/n$ for a list of n items. We focus on the partial cost model since it is easier to analyze.

Finally, we try to base our analysis on pairs of items. Deleting all but the requests to two given items x and y from a request sequence σ yields the request sequence σ_{xy}. Considered for all such pairs, there is a lower bound $\overline{OPT}(\sigma)$ for the cost of the optimal

[1] Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany. Email: albers@mpi-sb.mpg.de

[2] Institut für Theoretische Informatik, ETH Zürich, CH-8092 Zürich, Switzerland. Email: stengel@inf.ethz.ch

[3] FB Mathematik, Universität Frankfurt, Postfach 11 19 32, 60054 Frankfurt/Main, Germany. Email: werchner@informatik.uni-frankfurt.de

offline algorithm *OPT* when serving σ. The cost $\overline{OPT}(\sigma_{xy})$ is simple to compute: When only two items are requested, it is optimal to move an item to the front whenever it is requested twice or more in succession.

Consider a list of two items x and y, and a sequence of requests to these items. Suppose x precedes y in the list, and there is a request to y. Then an optimal offline algorithm knows the optimal treatment (moving y to the front or not), whereas the online algorithm makes a "mistake" with probability at least 1/2. If it is possible to keep this mistake probability at 1/2, then we stay 3/2-competitive, since $\overline{OPT}$ pays only 1 (for the first request to y), whereas the online algorithm, at the next request to x or y, pays an extra expected cost of 1/2 since it mistakenly exchanged y with x or not. This can be made precise by partial orders.

2. Posets generated by request sequences

We define a partial order by an initial list L and a request sequence σ as follows. This partial order is denoted $\langle\sigma]$, and is defined inductively, starting with the empty request sequence $\emptyset$, for all items y,z with $y \neq z$ (we always compare only distinct elements):

$$y\,\langle\emptyset]\,z :\Longleftrightarrow \;\; y \text{ precedes } z \text{ in } L.$$

The sequence obtained by extending σ with an additional request to the item x is denoted σx. Then

$$y\,\langle\sigma x]\,z :\Longleftrightarrow\; (y\,\langle\sigma]\,z \quad \text{and} \quad z \neq x) \quad \text{or} \quad (y = x \quad \text{and not} \quad z\,\langle\sigma]\,x)$$

In other words, a request to an item x changes only the relationship of x to the other elements of the poset. Then x is moved in front of all items z except where $z < x$ held before (in the partial order $\langle\sigma]$); in those cases, x becomes parallel (uncomparable) to z in $\langle\sigma x]$. In particular, x is always minimal in $\langle\sigma x]$. Since the relationship between unrequested items is unchanged, this poset represents a pairwise property, i.e.

$$x\,\langle\sigma]\,y \iff x\,\langle\sigma_{xy}]\,y.$$

The use of this poset for a 3/2-competitive online algorithm follows from the following result.

Lemma 1. *Consider an online algorithm that uses only free exchanges and after any request sequence σ maintains the list such that at for any two items x,y*

$$x \text{ precedes } y \text{ with probability } \begin{cases} 1 & \text{if } x\,\langle\sigma]\,y \\ 0 & \text{if } y\,\langle\sigma]\,x \\ 1/2 & \text{otherwise.} \end{cases}$$

Then this algorithm is 3/2-competitive.

Proof. By pairwise analysis. Suppose initially x is in front of y. Projected to this pair of items, any request sequence σ_{xy} can be split into subsequences after which the order of the two items is determined for both offline and online algorithm, namely after two successive requests to one item. There are only two kinds of such subsequences: $x^l(yx)^k x$ or $x^l(yx)^{k-1}yy$ for some $l \geq 0$ and $k \geq 1$. Then the projected expected cost of serving either sequence is k for $\overline{OPT}$ and $3k/2$ for an online algorithm with the described properties: For example, the request sequence $yxyxx$ incurs cost 2 for $\overline{OPT}$ (y is never moved). Before serving the requests y,x,y,x,x, the partial order relation between x and y is, respectively: $x < y$, $x||y$ (x and y are uncomparable), $x < y$, $x||y$, $x < y$. The corresponding expected costs are $1, 1/2, 1, 1/2, 0$, with sum 3. In general, the relationship $x < y$ determines an expected cost $3/2$ for each pair of requests yx in $(yx)^k$ and in the final requests yy for the second sequence, but only cost 1 for $\overline{OPT}$. □

Is there an algorithm with the properties in Lemma 1? The answer is yes for a list with four items, and possibly for a list with five items. Any four-element poset is two-dimensional, that is, it can be represented as an intersection of two linear orders. With a single random choice, the (barely) random algorithm selects one of these linear orders which it maintains as the list, but remembers the other order. Any request to an item changes the partial order, which can be implemented as a free exchange on both linear extensions in some way; one of these changes is usually a move to the front of the requested item, in the other dimension the item is often (but not always) left in place. For example, Fig. 1 shows a poset P where the four items are a,b,c,d, and its change after a request to item d. The poset P is represented as the intersection of two linear orders $b < a < d < c$ and $a < c < d < b$. By requesting d, the new linear orders are $d < b < a < c$ and $a < d < c < b$. The first of these is obtained by a move to the front of the requested item d, the second by a partial forward move of d in front of c but behind a. This partial move is not necessary if item c, which is symmetric to d in P, is requested (then c is merely moved to the front in the first list), but for one of these items it is necessary.

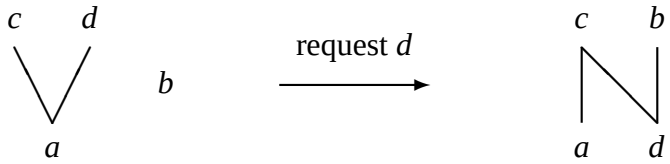

Figure 1.

Some care has to be taken to represent the poset: For example, if the only comparable items are a and b with $a < b$, then this poset should not be represented as the intersection of the two linear orders with $c < a < b < d$ and $d < a < b < c$. Namely, in this case a request to b has to move b in front of c in the first list and in front of d in the second list, and thus in front of a in both. However, a and b should be parallel. A

representation that works is $a < b < c < d$ intersected with $d < c < a < b$. Then a request to b is implemented by a move to the front in the second list.

By enumeration, one can examine the partial orders that occur and the possible requests to any item, and show the new representation (see the appendix). A systematic way of describing this approach would be desirable, also to extend it to $n = 5$ if possible.

A natural question is the following: Which partial orders can be generated in this way? This may be a restricted class of posets. For example, if all elements of the initial list are requested once in ascending order (by the request sequence α), they are all incomparable in the resulting poset $\langle\alpha]$. Starting from this symmetric situation, a request sequence σ requesting every element exactly twice generates an *interval* order, which is a poset where elements x, y are represented by intervals X, Y on the real line, with $x < y$ iff X is completely to the left of Y. The endpoints of the intervals representing $\langle\alpha\sigma]$ are the negative request times to the two requests to an item in σ. However, we can show the following.

Lemma 2. *Any poset P with n elements can be generated as $\langle\sigma]$ for a request sequence σ to the items in an n-element list.*

Proof. Consider a linear order L on the n items, where $x_1 < x_2 < \cdots < x_n$. Let $\Lambda(L)$ be the sequence of double requests to these items in reverse order:

$$\Lambda(L) = x_n x_n \ldots x_2 x_2 x_1 x_1 \,.$$

Clearly, $\langle\Lambda(L)] = L$ irrespective of the initial list. Furthermore, let $\lambda(L)$ be the sequence of requests requesting these items once in reverse order:

$$\lambda(L) = x_n x_{n-1} \ldots x_2 x_1 \,.$$

If P is the poset $\langle\sigma]$, then requesting all items once as in $\lambda(L)$ does the following: Clearly, if $x||y$ in P, then this relationship is unchanged by the requests $\lambda(L)$ since they contain x and y once. Similarly, if $x < y$ in both P and L, then $\lambda(L)$ contains first a request to y after which $x||y$, which is changed back to $x < y$ after the request to x. However, if $x < y$ in P but $x > y$ in L, then x is requested first in $\lambda(L)$ without effect, but the subsequent request to y produces $x||y$. In other words,

$$\langle\sigma\lambda(L)] = \langle\sigma] \cap L \,.$$

In particular, $\lambda(L)$ leaves P unchanged iff L is a linear extension of P (that is, $x < y$ in P implies $x < y$ in L).

Representing any given P as an intersection of d linear extensions of P (for example, all linear extensions),

$$P = L_1 \cap L_2 \cap \ldots \cap L_d \,,$$

we obtain

$$P = \langle\Lambda(L_1)\lambda(L_2)\ldots\lambda(L_d)]$$

which proves the claim. □

The minimum number d of linear orders whose intersection represents a partial order P is the dimension of P (see, for example, [4]). The preceding proof suggests that partial orders of high dimension need long request sequences to represent them. However, for the standard example of a d-dimensional order a much shorter request sequence suffices. This standard example is the order P whose graph is the complement of a perfect matching on $2d$ elements $x_1, x_2, \ldots, x_{2d}$. That is, the comparable elements in P are given by $x_i < x_{d+j}$ for $1 \leq i, j \leq n$ with $i \neq j$. Assume that the order of items in the initial list is $x_{d+1} < x_1 < x_{d+2} < x_2 < \cdots < x_{2d} < x_d$. Then $P = \langle \sigma]$ for the request sequence

$$\sigma = x_2 x_{d+2} x_3 x_{d+3} \ldots x_d x_{2d} x_1 x_2 \ldots x_d \,.$$

As an example, let $d = 3$ and write a, b, c, d, e, f for $x_1, \ldots, x_6$. Starting from the initial list $d < a < e < b < f < c$, the request sequence $becf$ generates the poset in Fig. 2, and the subsequent requests abc the three-dimensional poset in Fig. 3.

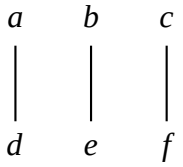

Figure 2.

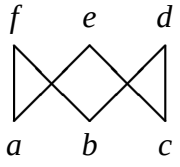

Figure 3.

The poset in Fig. 2 is two-dimensional, and its representation as an intersection of two linear orders is unique up to isomorphism: Namely, one of the minimal elements d, e, f, say d, has to be the smallest element of one of these linear orders. Since d is incomparable with all elements except a, these have to precede d in the other linear order. So the first linear order starts as $d < a < \cdots$ and the other ends as $\cdots < d < a$. Continuing this argument, the two orders are (up to symmetry) $d < a < e < b < f < c$ and $f < c < e < b < d < a$. This causes the approach that worked for four-element lists to fail, where the two dimensions have probability 1/2 and one of them represents the actual list maintained by the algorithm. Namely, requesting the item b means it has to be moved in front of a, c, d, f in both lists, and thereby will be moved in front of e although b should precede e in only one of these lists. This happens although the request to b (or similarly a or c) in the poset in Fig. 2 still results in a poset that is two-dimensional (although not for long, since Fig. 3 appears shortly). That is, even when we consider request sequences that generate only two-dimensional partial orders, it seems that these cannot be used to achieve a 3/2-competitive online algorithm according to Lemma 1.

3. Expected position of items

At first sight, Lemma 1 is stronger than it has to be. In proving a competitive factor of 3/2, it suffices that the *expected* position of an item is the same as prescribed by the partial order, as follows: Let $P = \langle\sigma]$. Then the expected position $p(x)$ according to P of an item x is defined as ($<$ referring to the partial order P and $||$ to incomparability in P as above)

$$p(x) = |\{y \mid y < x\}| + |\{y \mid y || x\}|/2\,.$$

This expected position (in terms of accessing cost, that is, as a number between 0 and $n-1$) results from a list where the relative position of items is distributed as in Lemma 1. It is this expected position that enters the cost comparison between $\overline{OPT}$ and the online algorithm (using pairwise analysis, see [2]).

Thus, it suffices to generate a distribution on lists that achieves the expected position $p(x)$ determined by the partial order P. Is this generally possible if P is arbitrary? The answer is yes if arbitrary linear orders can be used. The set of expected positions $(v_1,\ldots,v_n)$ resulting from a distribution on linear orders is a polytope known as the *permutahedron* (see, for example, [3]). It is characterized by the linear constraints

$$\sum_{i=1}^{n} v_i = \binom{n}{2}$$

and

$$\sum_{i\in A} v_i \geq \binom{|A|}{2} \qquad \text{for all } A \subseteq \{1,\ldots,n\}\,.$$

With $v_i := p(x_i)$ for the items $x_1,\ldots,x_n$, these constraints hold, because

$$\begin{aligned}\sum_{i\in A} p(x_i) &\geq \sum_{i\in A}\Big(|\{j\in A \mid x_j < x_i\}| + |\{j\in A \mid x_j || x_i\}|/2\Big)\\ &= 1/2\cdot\Big(\sum_{i,j\in A\,:\,x_j<x_i} 1 + \sum_{i,j\in A\,:\,x_j>x_i} 1 + \sum_{i,j\in A\,:\,x_j||x_i} 1\Big) = \binom{|A|}{2}\end{aligned}$$

where equality holds for $A = \{1,\ldots,n\}$.

However, this is no longer true if only linear extensions of P are allowed. Consider the poset P in Fig. 3. Each minimal element is parallel to three other elements, so $p(a) = p(b) = p(c) = 3/2$ and $p(d) = p(e) = p(f) = 7/2$. In any linear extension of P or a subset of P, minimal elements have to come first, so two of the minimal elements a,b,c have to be on the first two positions (counting 0 and 1) of the linear extension, similarly two elements of d,e,f occupy the last two positions (counting 4 and 5), and only positions 2 and 3 can be chosen as either a minimal or a maximal element of P. So the average position of a,b,c among linear extensions of P is at most $(0+1+3)/3 = 4/3$, and a,b,c cannot all have expected position $3/2$.

Why do we consider only linear extensions of P? This is justified by the use of free exchanges. For example, it is easy to show the following: Assume an initial list defining a linear order is given, and a single item x is requested. Then the lists obtainable by free exchanges after this request are exactly the linear extensions of $\langle x]$. This statement does no longer hold for sequences σ consisting of more than a single request, because an algorithm that does nothing uses also only free exchanges, but no longer keeps a list extending $\langle\sigma]$. However, the expected position $p(x)$ of an item x given by a partial order P can only be maintained all the time if the algorithm behaves as in Lemma 1:

Lemma 3. *Suppose a randomized online algorithm uses only free exchanges and maintains an expected position $p(x)$ for each item x as prescribed by the partial order $\langle\sigma]$ for all request sequences σ. Then after serving σ,*

$$x \text{ precedes } y \text{ with probability } \begin{cases} 1 & \text{if } x\langle\sigma]y \\ 0 & \text{if } y\langle\sigma]x \\ 1/2 & \text{otherwise.} \end{cases}$$

In particular, the algorithm gives positive probability only to lists that are linear extensions of $\langle\sigma]$.

Proof. First observe that an algorithm with the described property indeed maintains only lists that are linear extensions of $\langle\sigma]$ with positive probability. The initial list defines a linear order $\langle\emptyset]$. The expected position $p(x) = 0$ of the first item in that list can only be represented as a convex combination of positions in lists where x is at the first position. The same holds for the second up to the nth item, so the linear order representing these expected positions must have probability one (in other words, every linear order represents a vertex of the permutahedron). This shows the claim for $\sigma = \emptyset$.

As inductive hypothesis, assume the claim is true for the request sequence σ, and item x is requested afterwards. We want to show

$$y \text{ precedes } z \text{ with probability } \begin{cases} 1 & \text{if } y\langle\sigma x]z \\ 0 & \text{if } z\langle\sigma x]y \\ 1/2 & \text{otherwise.} \end{cases}$$

The probability that y precedes z in the list for items y and z other than x is not affected, since their transposition would require a paid exchange. So let $z = x$. If $x\langle\sigma]y$, then $x\langle\sigma x]y$ and x precedes y with probability one as before since y cannot be moved. If $y\langle\sigma]x$, then x and y are incomparable in $\langle\sigma x]$, so $p(y)$ is increased by 1/2. Since only x (and no other item) can be moved in front of y by a free exchange, this has to happen with probability 1/2, and afterwards y is indeed in front of x with probability 1/2. If x and y are incomparable in $\langle\sigma]$, then $x\langle\sigma x]y$, so $p(y)$ is again increased by 1/2. In those lists (with combined probability 1/2) where x is in front of y after serving σ, the position of y is not increased by moving x with a free exchange. This requires to move x in front of y in all other lists where x is behind y, thus placing x in front of y everywhere. This shows the claim for σx. □

The above argument and Lemma 3 show that the partial order in Fig. 3 defines expected positions $p(x)$ that cannot be achieved for all items x when only free exchanges are used.

4. Musings

Clearly, the above represents only a collection of partial results. The goal is, of course, to use this for getting a better upper or lower bound for the competitive factor. We have several conjectures in this direction.

One possibility would to be to improve the lower bound 3/2 by constructing a distribution on request sequences where the expected position of at least one item x is larger than $p(x)$, as indicated at end of Section 3, and then requesting that item. This has to be done in a repetitive fashion such as to actually generate the higher cost. These repetitions are maybe possible by requesting each item once as in the sequences $\lambda(L)$ used in the proof of Lemma 2 with L being linear extensions of the used partial orders. Assuming this works, which is speculative, we have only constructed a lower bound to $\overline{OPT}$ and have to show that OPT can actually serve these requests with this low offline cost. Furthermore, this lower bound is established for the partial cost model only. The approach would then have to be extended to lists of arbitrary length n for getting this result (3/2 is not the optimal competitive factor) for the full cost model. Finally, one would have to consider paid exchanges as well.

Alternatively, partial orders may have something to do — this is even more speculative — with the relationship of OPT and $\overline{OPT}$. That is, whenever the partial order is such that the expected costs of the online algorithm are necessarily larger than 3/2 times the cost of $\overline{OPT}$ for some items, then OPT cannot request these items like $\overline{OPT}$ but has to pay extra. In that way, the online algorithm can again catch up with OPT, and we can stay 3/2-competitive. There is no reason why this should be the case, but it may be worth trying.

It may also be useful to study the current algorithms with respect to the partial orders presented here. For example, the behavior of the BIT algorithm is not far from that described in Lemma 1: A requested item x passes with probability 1/2 items in front of it since it is moved to the front at every other request. However, this represents the partial order incompletely. The TIMESTAMP algorithm [1] is more sophisticated by not moving items completely to the front of the list. It may also be interesting to study its behavior with respect to the partial order.

References

[1] S. Albers, Improved randomized on-line algorithms for the list update problem, *Proc. 6th Annual ACM-SIAM Symposium on Discrete Algorithms* (1995) 412–419.

[2] S. Albers, B. von Stengel, and R. Werchner, A combined BIT and TIMESTAMP algorithm for the list update problem. *Information Processing Letters* **56** (1995) 135–139.

[3] A. von Arnim, U. Faigle, and R. Schrader, The permutahedron of series-parallel posets. *Discrete Applied Mathematics* **28** (1990) 3–9.

[4] I. Rival, Linear extensions of finite ordered sets. In: Orders: Description and Roles, eds. M. Pouzet and D. Richard, North-Holland, *Annals of Discrete Mathematics* **23** (1984) 355–370.

[5] B. Teia, A lower bound for randomized list update algorithms, *Information Processing Letters* **47** (1993) 5–9.

Appendix

In this appendix, we list the 3- and 4-element posets, all of which are two-dimensional, and show how to represent any access to a list element as on operation on one or both of the lists representing them. The first table shows the possible 3-element orders, also as intersections of two linear orders, on the elements a, b, c, and the effect of a request to either a, b, or c.

order	as dim 2	a	b	c
$a < b < c$	$abc \cap abc$	$abc \cap abc$	$abc \cap bac$	$abc \cap cab$
$(a \,\|\|\, b) < c$	$abc \cap bac$	$abc \cap abc$	$bac \cap bac$	$abc \cap cba$
$a < (b \,\|\|\, c)$	$abc \cap acb$	$abc \cap acb$	$bac \cap abc$	$cab \cap acb$
$(a < b) \,\|\|\, c$	$abc \cap cab$	$abc \cap acb$	$abc \cap bca$	$cab \cap cab$
$a \,\|\|\, b \,\|\|\, c$	$abc \cap cba$	$abc \cap acb$	$bac \cap bca$	$cab \cap cba$

The next table shows this for posets with 4 elements a, b, c, d. Not all entries are given. The posets are listed in breadth-first order of appearance after requests to list items, starting from the linear order. Some requests require that the item is moved partially forward in one list, not all the way to the front. This occurs for the requests to c for the two partial orders $(a < (b \,||\, c)) \,||\, d$ and $a < (b \,||\, c \,||\, d)$. The "N" shaped poset $(a \,||\, b) < c, b < d$ causes no difficulty.

order	as dim 2	a	b	c	d
$a<b<c<d$	$abcd \cap abcd$	$abcd \cap abcd$	$abcd \cap bacd$	$abcd \cap cabd$	$abcd \cap dabc$
$(a\|\|b)<c<d$	$abcd \cap bacd$	$abcd \cap abcd$	$bacd \cap bacd$	$abcd \cap cbad$	$abcd \cap dbac$
$((a<b)\|\|c)<d$	$abcd \cap cabd$	$abcd \cap acbd$	$abcd \cap bacd$	$cabd \cap cabd$	$abcd \cap dcab$
$(a<b<c)\|\|d$					
$(a\|\|b\|\|c)<d$					
$((a\|\|b)<c)\|\|d$					
$a<(b\|\|c)<d$					
$(a<b)\|\|c\|\|d$					
$a<((b<c)\|\|d)$					
$(a\|\|b)<c, b<d$	$abcd \cap bdac$				
$(a<b)\|\|(c<d)$					
$a\|\|b\|\|c\|\|d$					
$(a<(b\|\|c))\|\|d$	$abcd \cap dacb$	$abcd \cap adcb$	$abcd \cap bdac$	$acbd \cap cdab$	$dabc \cap dacb$
$a<(b\|\|c\|\|d)$	$abcd \cap adcb$	$abcd \cap adcb$	$abcd \cap badc$	$acbd \cap cadb$	$dabc \cap adcb$
$(a\|\|b)<(c\|\|d)$					
$a<b<(c\|\|d)$					