

A randomized approximation algorithm for computing bucket orders

Antti Ukkonen^{a,b,*}, Kai Puolamäki^{a,b}, Aristides Gionis^d, Heikki Mannila^{a,b,c}

^a Helsinki Institute for Information Technology HIIT, Finland

^b Helsinki University of Technology, Department of Information and Computer Science, P.O. Box 5400, Helsinki, Finland

^c University of Helsinki, Finland

^d Yahoo! Research Barcelona, Spain

ARTICLE INFO

Article history:

Received 2 September 2008

Received in revised form 2 December 2008

Available online 7 December 2008

Communicated by C. Scheideleer

Keywords:

Approximation algorithms

Randomized algorithms

Ranking

ABSTRACT

We show that a simple randomized algorithm has an expected constant factor approximation guarantee for fitting bucket orders to a set of pairwise preferences.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction and problem definition

Let $M = \{1, \dots, m\}$ be a set of m items. A *bucket order* \mathcal{B} is a partial order defined by an ordered partition of M , i.e., a sequence M_1, \dots, M_k of k disjoint subsets (buckets) of M , with $\bigcup_i M_i = M$. The item $u \in M$ precedes the item $v \in M$ according to \mathcal{B} , denoted $u <_{\mathcal{B}} v$, if and only if $u \in M_i$ and $v \in M_j$ with $i < j$. If two items belong to the same bucket they are unordered by \mathcal{B} , denoted $u \sim_{\mathcal{B}} v$. We can represent \mathcal{B} as the $m \times m$ matrix B , with $B(u, v) = 1$ if $u <_{\mathcal{B}} v$, $B(u, v) = \frac{1}{2}$ if $u \sim_{\mathcal{B}} v$, and $B(u, v) = 0$ if $v <_{\mathcal{B}} u$. We call B a *bucket matrix*. Note that we always have $B(u, v) + B(v, u) = 1$.

Let C be an $m \times m$ matrix with entries in the interval $[0, 1]$, so that $C(u, v) + C(v, u) = 1$. ($C(u, u) = \frac{1}{2}$ for all u .) We call C the *pair order matrix*, and interpret $C(u, v)$ as the probability that the item u precedes the item v . Our task is to find a bucket order \mathcal{B} that fits these probabilities as well as possible. We do this by finding a bucket matrix B

that is a good approximation of C in terms of the L_1 norm. More formally, we consider the following problem:

Problem OPTIMAL-BUCKET-ORDER. Given a pair order matrix C , find the bucket matrix B that minimizes the cost $c(B, C) = \sum_{u,v} |B(u, v) - C(u, v)|$.

The problem is closely related to the feedback-arc-set problem (GT8 in [3]) that has been studied recently from the point of view of approximation [1,2,5]. The OPTIMAL-BUCKET-ORDER problem has applications, e.g., in ranking and in seriation for paleontological data. It is NP-hard, as it can be shown to include the feedback arc set problem as a special case [4]. In this paper we consider a simple quicksort-like algorithm for the problem, and show that it has an expected constant factor approximation guarantee.¹

2. A simple algorithm and its analysis

The Bucket Pivot Algorithm (Algorithm 1) receives as input a set of items M , the pair order matrix C , and a pa-

* Corresponding author at: Helsinki University of Technology, Department of Information and Computer Science, P.O. Box 5400, Helsinki, Finland.

E-mail address: antti.ukkonen@tkk.fi (A. Ukkonen).

¹ The algorithm was originally given by us in [4], where no proof of the approximation ratio was given. The current paper is based on the PhD thesis of the first author [6].

```

1: BP( $M, C, \beta$ )
   {Input:  $M$ , set of items;  $C$ , pair order matrix;  $\beta \geq 0$ , parameter.
   Output: Bucket order.}
2: if  $M = \emptyset$  then
3:   RETURN  $\emptyset$ 
4: end if
5: Pick a pivot  $p \in M$  uniformly at random.
6:  $L \leftarrow \emptyset, S \leftarrow \{p\}, R \leftarrow \emptyset$ 
7: for all items  $u \in M \setminus \{p\}$  do
8:   if  $C(p, u) < \frac{1}{2} - \beta$  then
9:     Add  $u$  to  $L$ .
10:  else if  $\frac{1}{2} - \beta \leq C(p, u) \leq \frac{1}{2} + \beta$  then
11:    Add  $u$  to  $S$ .
12:  else if  $\frac{1}{2} + \beta < C(p, u)$  then
13:    Add  $u$  to  $R$ .
14:  end if
15: end for
16: RETURN order (BP( $L, C, \beta$ ),  $S$ , BP( $R, C, \beta$ ))

```

Algorithm 1. The Bucket Pivot Algorithm. By default, we use $\beta = \frac{1}{4}$.

parameter $\beta \in [0, \frac{1}{2})$. It selects a random element $p \in M$ as a pivot, and then places the remaining items in M to the sets L, S , or R , depending how they are related to p according to C . If the probability that an item precedes (follows) p is large enough, it is put to the set L (R), otherwise it is put to the set S . For details, please see lines 8–14 of Algorithm 1. Finally the algorithm is applied recursively to the sets L and R . This algorithm is a generalization of a similar algorithm presented in [1], and with $\beta = 0$ it is equivalent to the one in [1].

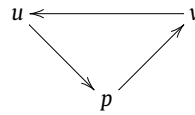
We show that Algorithm 1 is an expected constant factor approximation algorithm when $\beta = \frac{1}{4}$. The proof follows the technique used in [1] with some modifications. Let B^A be the bucket matrix found by Algorithm 1, and denote by B^* the optimal solution to OPTIMAL-BUCKET-ORDER. Since Algorithm 1 is randomized, B^A is a random variable. We show that the expected cost $E[c(B^A, C)]$ is at most $\alpha c(B^*, C)$, where α is a constant. (Either 9 or 5, depending on our assumptions about the pair order matrix.)

For the proof it is useful to think that C contains weights of the edges of a complete, directed graph with the items in M as vertices. By linearity of expectation the expected cost $E[c(B^A, C)]$ can be written as the sum of the costs of individual edges (u, v) , that is, we can write $E[c(B^A, C)] = \sum_{u,v} E[c(B^A, (u, v))]$, where $c(B^A, (u, v)) = |B^A(u, v) - C(u, v)|$. In the following we denote by e any pair (u, v) . The key of the approximation result is to find a proper expression for the expected cost $E[c(B^A, e)]$ of an edge e . To do this we consider the possible cases that can happen to e during the execution of the algorithm. Crucial here is how e is related to the pivot vertex p .

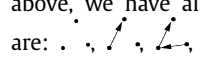
First of all, when p is one of the endpoints of e , the cost $c(B^A, e)$ will always be optimal. For example, let $e = (p, u)$ and suppose that $C(p, u) \in [0, \frac{1}{4})$. The algorithm will put u in the set L , and hence u will appear before p in the final output. We will thus have $B^A(p, u) = 0$, which is optimal given the value of $C(p, u)$. It is easy to see that this holds also for other cases where $C(p, u)$ belongs either to $[\frac{1}{4}, \frac{3}{4}]$ or $(\frac{3}{4}, 1]$.

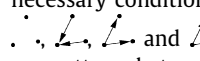
Therefore, $c(B^A, e)$ can be nonoptimal only when the pair e is assigned a cost without the pivot p being a part of e . This can only happen when e is opposite to the pivot


in a triangle of three vertices. See the figure below for an example:

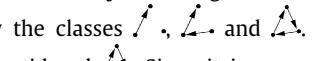

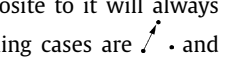




The arrows indicate which direction has a large weight in C , i.e., when the arrow points from u to p , we have $C(u, p) > \frac{3}{4}$. In this case if p is chosen as the pivot and $e = (u, v)$, the cost $c(B^A, e)$ will be nonoptimal. The vertex u will be put to L and v will be put to R . Hence u will precede v in the final output, and we will have $B^A(u, v) = 1$, although $C(u, v) < \frac{1}{4}$, and the optimal placement of u and v could place v before u .

The weights in C can be used to define other kinds of triangles as well. In addition to the directed cycle shown above, we have also six other classes of triangles. These are: . We define them as follows: two vertices u and v are connected by an arrow whenever $C(u, v)$ is either larger than $\frac{3}{4}$ or less than $\frac{1}{4}$. The arrow is directed from u to v if $C(u, v) > \frac{3}{4}$. If $C(u, v) < \frac{1}{4}$, the arrow is directed from v to u . If $C(u, v)$ is in the range $[\frac{1}{4}, \frac{3}{4}]$ we do not draw an arrow between the points.

We already argued that the cost $c(B^A, e)$ will be smallest when either endpoint of e is chosen as the pivot, and claimed that the cost of an edge e can only be nonoptimal when it is opposite to the pivot vertex. This is only a necessary condition, however. Consider the triangle classes . It is easy to see that in every case, no matter what vertex is chosen as the pivot, the edge opposite to it will always either a) have the smallest possible cost, i.e., its orientation in the final output will be concordant with C , or b) both of its endpoints will be put to the set L (or R), and hence its cost will be defined in a later step of the algorithm.

For example, in case of , if we choose the vertex with two outgoing (or incoming) edges as the pivot, the opposite edge will be put to the set R (or L) and its cost is not yet defined. If we choose the vertex with one incoming and one outgoing edge as the pivot, the edge opposite to it will be oriented in the correct way according to C .

This leaves us with only the classes . In the example above we considered . Since it is completely symmetric, it is obvious that no matter what vertex is chosen as the pivot, the edge opposite to it will always incur a nonoptimal cost. The remaining cases are  and . In case of , if we choose the vertex that is adjacent to the undirected edges, both endpoints of the directed edge (u, v) will be put to the set S and hence we will have $B^A(u, v) = \frac{1}{2}$ even though $C(u, v) \in (\frac{3}{4}, 1]$. If either of the vertices adjacent to the directed edge is chosen as the pivot, one of the vertices of the undirected edge (u, v) that is opposite to it will be put to the set S , while the other one is put to the set R (or L). This means $B^A(u, v) = 1$ or $B^A(u, v) = 0$, even though $C(u, v) \in [\frac{1}{4}, \frac{3}{4}]$.

In case of \triangleleft the following occurs: If the vertex with one incoming and one outgoing edge is chosen as the pivot, the edge on the opposite side will be assigned a direction even though both of its endpoints should belong to the same bucket. If the vertex with one outgoing (or incoming) edge is chosen as the pivot, the other vertex of the opposite edge will be put to the set R (or L) while the other one is put to the set S . In both cases the arrow will end up pointing the wrong way.

Consider an edge e , and the randomly chosen pivot $p \notin e$. Denote the triangle formed by these with t_e^p . Let $\Omega = \{\triangleleft, \triangle, \triangleright\}$ be the set of those triangle classes that may lead to suboptimal behavior as argued above. I.e., if t_e^p belongs to a triangle class $T \in \Omega$, the cost $c(B^A, e)$ will be nonoptimal. Denote by $c_T(e)$ the cost that e incurs given a fixed class $T \in \Omega$. Note that $c_T(e)$ is the same for all triangles that belong to $T \in \Omega$. Denote the probability that t_e^p belongs to a fixed $T \in \Omega$ with $p_T(e)$. Furthermore, let $c_{\text{opt}}(e)$ denote the cost incurred to e when it either is adjacent to the pivot or appears opposite to the pivot in one of the triangle classes *not* belonging to Ω . We have $c_{\text{opt}}(e) = \min_{x \in [0, 0.5, 1]} |x - C(u, v)|$, where $e = (u, v)$. Finally, let $c^*(e)$ denote the cost of the edge e in the globally optimal solution B^* . Obviously we have $c_{\text{opt}}(e) \leq c^*(e)$. For instance, in case of the directed cycle (triangles belonging to class \triangleleft) one of the edges *always* has to pay a nonoptimal cost.

Using the above, we can write the expected cost of e as follows:

$$E[c(B^A, e)] = \sum_{T \in \Omega} p_T(e) c_T(e) + \left(1 - \sum_{T \in \Omega} p_T(e)\right) c_{\text{opt}}(e). \quad (1)$$

Either e pays the nonoptimal cost $c_T(e)$ when it ends up opposite to the pivot in a triangle t_e^p belonging to $T \in \Omega$, or it pays $c_{\text{opt}}(e)$ in the remaining cases. By summing Eq. (1) over all possible $e = (u, v)$ pairs, we get

$$E[c(B^A, C)] = \underbrace{\sum_e \sum_{T \in \Omega} p_T(e) c_T(e)}_{H^A} + \underbrace{\sum_e \left(1 - \sum_{T \in \Omega} p_T(e)\right) c_{\text{opt}}(e)}_{L^A}, \quad (2)$$

where H^A is the part with the nonoptimal (“high”) costs, and L^A the part with the locally optimal (“low”) costs.

Next we derive $p_T(e)$. Define the events $X_T(t, e)$ and $B(e)$. The event $X_T(t, e)$ means that all vertices of a triangle $t \in T$, one side of which is e , appear in a recursive call of the algorithm, and one of t 's vertices is chosen as the pivot. The event $B(e)$ happens when the pivot chosen is the vertex opposite to edge e . Given that $X_T(t, e)$ happens, the probability of $B(e)$ is just $\frac{1}{3}$ as each of the three vertices has equal probability of becoming the pivot. If p_t is the probability of $X_T(t, e)$, we can write

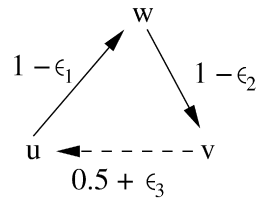


Fig. 1. A triangle belonging to one of the classes in Ω . To each edge is associated its possible value in C . For the cases where the weight is $1 - \epsilon_i$, we have $\epsilon_i \in [0, \frac{1}{4}]$. When the weight is $0.5 + \epsilon_3$, we have $\epsilon_3 \in [-\frac{1}{4}, \frac{1}{4}]$.

$$\Pr(X_T(t, e) \wedge B(e)) = \Pr(B(e) | X_T(t, e)) \Pr(X_T(t, e)) = \frac{1}{3} p_t.$$

This is the probability of one $t \in T$ causing e to pay a nonoptimal cost. For the entire class T we have $p_T(e) = \sum_{t: e \in t \in T} \frac{1}{3} p_t$.

This must be a probability (≤ 1) because the events $X_T(t, e) \wedge B(e)$ and $X_T(t', e) \wedge B(e)$ are disjoint for all $t, t' \in T$, since e is charged to triangle t , it cannot be charged to triangle t' . In fact, the same holds for all triangles in the input graph. Any edge e can only cause a nonoptimal cost with *one* triangle, no matter what class this triangle belongs to. This means that for all e we have $\sum_{T \in \Omega} \sum_{t: e \in t \in T} \frac{1}{3} p_t \leq 1$.

When $p_T(e)$ is substituted into H^A we obtain $H^A = \sum_{T \in \Omega} \sum_{t \in T} \frac{1}{3} p_t c_T(t)$, where $c_T(t) = \sum_{e \in t} c_T(e)$. To proceed we present the following lemma:

Lemma 2.1. *Let $\alpha > 1$ be a constant. We have $E[c(B^A, C)] \leq \alpha c(B^*, C)$, if for all triangle classes $T \in \Omega$ it holds that $c_T(t) \leq \alpha c^*(t)$, where $c^*(t) = \sum_{e \in t} c^*(e)$.*

Proof. We decompose the optimal cost as follows: Let $c(B^*, C) = H^* + L^*$, where $H^* = \sum_{T \in \Omega} \sum_{t \in T} \frac{1}{3} p_t c^*(t)$ and $L^* = \sum_{e \in A} (1 - \sum_{T \in \Omega} p_T(e)) c^*(e)$. If we assume there exists an α so that $c_T(t) \leq \alpha c^*(t)$ for all $T \in \Omega$, then we obtain that $H^A \leq \alpha H^*$, which implies the claim of the lemma, as it must be the case that $L^A \leq L^*$, since $c_{\text{opt}}(e) \leq c^*(e)$ for all e . \square

The exact value(s) of α are determined by case analysis, where we consider all triangle classes in Ω separately. As an example we look at the class $T = \triangleleft$. We tabulate the costs of each edge for different choices of the pivot (see also Fig. 1):

Pivot	$\{v, w\}$	$\{u, w\}$	$\{u, v\}$
u	$1 - \epsilon_2$	ϵ_1	ϵ_3
v	ϵ_2	$1 - \epsilon_1$	ϵ_3
w	ϵ_2	ϵ_1	$\frac{1}{2} + \epsilon_3$

This gives us $c_T(t) = \frac{5}{2} - \epsilon_1 - \epsilon_2 + \epsilon_3$, and $c^*(t) = \frac{1}{2} + \epsilon_1 + \epsilon_2 + \epsilon_3$, where $\epsilon_1, \epsilon_2 \in [0, \frac{1}{4}]$ and $\epsilon_3 \in [-\frac{1}{4}, \frac{1}{4}]$. For the worst case we set $\epsilon_1 = \epsilon_2 = 0$. If $\epsilon_3 \geq 0$, we have $c_T(t)/c^*(t) \leq 5$, and when $\epsilon_3 = -\frac{1}{4}$, we have $c_T(t) = \frac{9}{4}$ and $c^*(t) = \frac{1}{4}$ which gives $\alpha \geq 9$.

Using the same technique we obtain $\alpha \geq 5$ for \curvearrowright and $\alpha \geq \frac{11}{3}$ for \triangleleft . Thus, Lemma 2.1 holds for $\alpha = 9$ and we obtain the following theorem.

Theorem 2.2. *Algorithm 1 is a randomized 9-approximation algorithm.*

If we require that the probabilities in the pair order matrix C satisfy the triangle inequality, we note that for instance the assignment of values to ϵ_i used above to obtain $\alpha \geq 9$ is not valid. Also, triangles from the class \triangleleft do not occur at all. By repeating the same case analysis so that the triangle inequality is satisfied, we obtain $\alpha \geq 5$ for \curvearrowright and $\alpha \geq 4$ for \triangleleft .

Theorem 2.3. *Algorithm 1 is a randomized 5-approximation algorithm if the input C satisfies the triangle inequality.*

It is not known whether the bounds 9 and 5 are tight.

References

- [1] N. Ailon, M. Charikar, and A. Newman, Aggregating inconsistent information: ranking and clustering, in: Proceedings of the 37th ACM Symposium on Theory of Computing, 2005, pp. 684–693.
- [2] D. Coppersmith, L. Fleischer, A. Rudra, Ordering by weighted number of wins gives a good ranking for weighted tournaments, in: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2006, pp. 776–782.
- [3] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, 1979.
- [4] A. Gionis, H. Mannila, K. Puolamäki, A. Ukkonen, Algorithms for discovering bucket orders from data, in: Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, pp. 561–566.
- [5] C. Kenyon-Mathieu and W. Schudy, How to rank with few errors, in: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing, 2007, pp. 95–103.
- [6] A. Ukkonen, Algorithms for finding orders and analyzing sets of chains, PhD thesis, Helsinki University of Technology, 2008.