# On Approximation of Bookmark Assignments

Asahiro, Yuichi
Department of Social Information Systems, Kyushu Sangyo University

Miyano, Eiji
Department of Systems Innovation and Informatics, Kyushu Institute of Technology

Murata, Toshihide
Department of Systems Innovation and Informatics, Kyushu Institute of Technology

Ono, Hirotaka
Department of Computer Science and Communication Engineering, Kyushu University

https://hdl.handle.net/2324/14891

出版情報：Lecture Notes in Computer Science. 4708, pp.115-124, 2007-08. Springer
バージョン：
権利関係：

# On Approximation of Bookmark Assignments [⋆]

Yuichi Asahiro[1], Eiji Miyano[2], Toshihide Murata[2], and Hirotaka Ono[3]

[1] Department of Social Information Systems, Kyushu Sangyo University, Fukuoka 813-8503, Japan. asahiro@is.kyusan-u.ac.jp

[2] Department of Systems Innovation and Informatics, Kyushu Institute of Technology, Fukuoka 820-8502, Japan. miyano@ces.kyutech.ac.jp, hide_m@theory.ces.kyutech.ac.jp

[3] Department of Computer Science and Communication Engineering, Kyushu University, Fukuoka 819-0395, Japan. ono@csce.kyushu-u.ac.jp

**Abstract.** Consider a rooted directed acyclic graph $G = (V, E)$ with root $r$, representing a collection $V$ of web pages connected via a set $E$ of hyperlinks. Each node $v$ is associated with the probability that a user wants to access the node $v$. The access cost is defined as the expected number of steps required to reach a node from the root $r$. A bookmark is an additional shortcut from $r$ to a node of $G$, which may reduce the access cost. The bookmark assignment problem is to find a set of bookmarks that achieves the greatest improvement in the access cost. For the problem, the paper presents a polynomial time approximation algorithm with factor $(1-1/e)$, and shows that there exists no polynomial time approximation algorithm with a better constant factor than $(1-1/e)$ unless $\mathcal{NP} \subseteq \mathcal{DTIME}(N^{O(\log \log N)})$, where $N$ is the size of the inputs.

## 1 Introduction

### 1.1 Motivation and Formulation

The world wide web is one of the most famous and the hugest repository of information, which consists of web pages and hyperlinks. The hyperlink is a reference from one page to another, and hence it gives us a non-linear searchability; a user can travel to access desired pages by starting from a home page, clicking correct hyperlinks, and displaying pages one after another. In order to display the target (desired or favorite) pages by clicking as few hyperlinks as possible, one of the most popular approaches is to add a "shortcut" linkage, usually called a *bookmark*.

The above natural solution for improving web access can be formulated as the following optimization problem, called the $k$-Bookmark Assignment Problem ($k$-BAP), which has been introduced in [19]: Let $G = (V, E)$ be a rooted, connected, directed acyclic graph (DAG for short) with root $r$, representing a

[⋆] This work is partially supported by Grant-in-Aid for Scientific Research on Priority Areas No. 16092222 and 16092223, and by Grant-in-Aid for Young Scientists (B) No. 17700022, No. 18700014 and No. 18700015.

web hierarchical structure. Let $p_v$ be the *access probability* associated to a node $v \in V$, where $\sum_{v \in V} p_v = 1$ and each $p_v \geq 0$. For a node $v$, let $d(v)$ denote the *distance* from the root $r$ to the node $v$, i.e., the length of the shortest directed path. The *expected number of steps* to reach a node from the root $r$ is defined by $\sum_{v \in V} p_v \cdot d(v)$. A *bookmark to a node* $v$ is an additional directed edge $(r, v) \notin E$. Only for simplicity, throughout the paper, we identify the bookmark $(r, v)$ with the node $v$ itself. Let $B = \{b_1, b_2, \cdots, b_k\}$, $b_i \in V$, be a set of bookmarks and also let $G \oplus B$ be the graph resulting from the assignment of $B$. The distance from the root $r$ to a node $v$ in $G \oplus B$ is denoted by $d_B(v)$. Thus, the expected number of steps to reach a node of $G \oplus B$ from the root is also defined by $\sum_{v \in V} p_v \cdot d_B(v)$. For simplicity, let $p_v \cdot d(v)$ and $p_v \cdot d_B(v)$ be represented by $c(v)$ and $c_B(v)$, respectively, in the following. A *gain* $g(B)$ of bookmarks $B$ is defined as

$$g(B) = \sum_{v \in V} p_v \left(d(v) - d_B(v)\right) = \sum_{v \in V} \left(c(v) - c_B(v)\right),$$

and thus the problem is formulated as follows:

**$k$-Bookmark Assignment Problem ($k$-BAP):**
INSTANCE: A directed acyclic graph $G = (V, E)$, the access probability $p_v$ for each node $v \in V$, and a positive integer $k \leq |V|$.
GOAL: Find a bookmark set $B \subseteq V$ of size $k$ maximizing $g(B)$.

## 1.2 Previous and Our Results

The problem has been considered also in the context of locating web proxies in the Internet. For restricted network topologies, such as paths and trees, polynomial time exact algorithms are successfully developed: For paths, an $O(k|V|^2)$ time algorithm is proposed by Li, Deng, Golin, and Sohraby in [10] and for trees Li, Golin, Italiano, Deng and Sohraby proposed an $O(k^2|V|^3)$ time algorithm in [11]. In particular, Czyzowicz, Kranakis, Krizanc, Pelc, and Vergas Martin showed that if the input graph is a perfect binary tree and $k \leq \sqrt{|V| + 1}$, the optimal set of bookmarks satisfies a good property [4] and a faster $O(|V|)$ time algorithm can be developed based on that property [19]. On the other hand, Vergas Martin shows that $k$-BAP is $\mathcal{NP}$-hard for directed acyclic graphs in [19]. As far as the authors know, there is no result with respect to approximation for $k$-BAP.

In this paper, we consider (in)approximability of $k$-BAP, and obtain the following results.

**(Approximability)** There is a polynomial time approximation algorithm with factor $(1 - 1/e)$.
**(Inapproximability)** There is no polynomial time approximation algorithm with a better constant factor than
- $(1 - 1/e)$ unless $\mathcal{NP} \subseteq \mathcal{DTIME}(N^{O(\log \log N)})$, where $N$ is the size of the inputs, and

– $(1-\delta)$ under a weaker assumption, i.e., $\mathcal{P} \neq \mathcal{NP}$, where $\delta$ is a fixed small positive constant.

An algorithm is called an *approximation algorithm with factor* $\alpha$, or an $\alpha$ *factor approximation algorithm* for $k$-BAP, if $g(\mathsf{ALG})/g(\mathsf{OPT}) \geq \alpha$ holds for any instance $G$, where $\mathsf{ALG}$ and $\mathsf{OPT}$ are sets of bookmarks obtained by the algorithm and an optimal algorithm, respectively. $\alpha$ is called *approximation ratio*.

### 1.3 Related Work

There is a large amount of literature on the problem of improving the accessibility of the most popular pages by adding hyperlinks over the underlying structure (e.g., [3, 6, 13, 17, 18]). A variant of a bookmark is called a *hotlink*, which is defined as a hyperlink that links an *arbitrary* page to its descendant page. The problem variant is known as the Hotlink Assignment Problem (HAP), which has been introduced by Bose et al. in [3]. It is known that HAP is $\mathcal{NP}$-hard for arbitrary DAGs [3] but unknown for trees whether it is $\mathcal{NP}$-hard or not. Many researchers have studied on HAP in perfect binary tree structures [3, 6]. Matichin and Peleg present a polynomial time 1/2 factor approximation algorithm for DAGs in [13], and also a 1/2 factor approximation algorithm for trees under some realistic access model in [14]. It is important to note that no result on inapproximability has been obtained so far for HAP.

### 1.4 Organization

In Section 2, we present the polynomial time approximation algorithm with factor $(1-1/e)$. The inapproximability of $k$-BAP is discussed in Section 3. Then we conclude in Section 4.

## 2 Approximation Guarantee of a Greedy Algorithm

Consider the following greedy algorithm for a DAG $G = (V, E)$, which selects $k$ bookmarks in total by iteratively assigning a new bookmark that obtains the maximum gain:

```
Algorithm Greedy:
    $B = \emptyset$
    for $i = 1$ to $k$ do
        select a bookmark $b \in V \setminus B$ that maximizes $g(B \cup \{b\})$
        $B = B \cup \{b\}$
    endfor
    output $B$
```

**Theorem 1.** *Algorithm* Greedy *achieves an approximation ratio of* $(1 - \frac{1}{e})$ *for* $k$*-BAP, and its running time is* $O(k|V||E|)$.

*Proof.* First we show the approximation ratio. The basic framework of the proof is the same as in [20]. That is, we utilize a result of Nemhauser et al. [15], which analyzes the approximation ratio of a greedy algorithm for a certain maximization problem whose cost is defined as a monotone submodular function. The submodularity and its monotonicity are defined as follows:

**Definition 1.** *Let* $S$ *be a finite set, and* $f : 2^S \to \mathbb{R}$ *be a function with* $f(\emptyset) = 0$. $f$ *is called* submodular *if for any sets* $X, Y \subseteq S$,

$$f(X \cup Y) + f(X \cap Y) \leq f(X) + f(Y).$$

$f$ *is called* monotone *if for any set* $X \subset S$ *and* $s \in S \setminus X$

$$f(X \cup \{s\}) - f(X) \geq 0.$$

The result of Nemhauser et al. is that the problem of selecting $k$-element subsets maximizing a monotone submodular function can be approximated within a constant factor $(1 - \frac{1}{e})$ by a greedily picking algorithm [15]. By Proposition 2 (monotonicity) and Lemma 1 (submodularity) shown later, the gain function $g$ is monotone submodular, and hence the approximation ratio of the above algorithm Greedy is also $(1 - \frac{1}{e})$.

As for the running time, Greedy selects $k$ bookmarks by evaluating the values of $g(B \cup \{b\})$'s for each $b \in V$. Computing $g(B \cup \{b\})$ requires to solve the shortest path problem for the DAG $G \oplus (B \cup \{b\})$, which takes $O(|E|)$ time [5]. Since the number of possible bookmarks in each iteration is at most $|V|$ and the number of iterations is $k$, the total running time is $O(k|V||E|)$. □

Now we show that the gain function $g$ is monotone and submodular. First we briefly note the monotonicity of $g$. The following property obviously holds.

**Proposition 1.** *For* $S' \subseteq S \subseteq V$ *and* $u \in V$, $d_{S'}(u) \geq d_S(u)$ *holds.* □

Then, from the above proposition, we can see that $g$ is monotone in a straightforward way:

**Proposition 2 (monotonicity).** *For* $S' \subseteq S \subseteq V$, $g(S') \leq g(S)$ *holds.* □

We next prove the submodularity of the gain function $g$. Let $V(X)$ be a set of descendant nodes that are reachable from a set $X$ of nodes in $G$.

**Proposition 3.** *For two subsets of nodes* $X, Y \subseteq V$

(i) $V(X) \cup V(Y) = V(X \cup Y)$
(ii) $V(X) \cap V(Y) \supseteq V(X \cap Y)$

*Proof.* (i) It is easy to see that $V(S') \subseteq V(S)$ holds for $S' \subseteq S \subseteq V$. Thus, $V(X) \cup V(Y) \subseteq V(X \cup Y)$ holds. We show that $V(X \cup Y) \subseteq V(X) \cup V(Y)$ also holds in the following: First, note that for $\forall u \in V(X \cup Y)$, there exists a node $v \in X \cup Y$ such that $u$ is reachable from $v$, i.e., for $\forall u \in V(X \cup Y)$, $u$ belongs to $V(X) \cup V(Y)$. It follows that $V(X \cup Y) \subseteq V(X) \cup V(Y)$ holds. As a result, $V(X \cup Y) = V(X) \cup V(Y)$ holds.

(ii) By a similar discussion, we can show that $V(X) \supseteq V(X \cap Y)$ and $V(Y) \supseteq V(X \cap Y)$ hold, which means that $V(X) \cap V(Y) \supseteq V(X \cap Y)$ is satisfied. $\square$

Using Propositions 1, 2, and 3, we can show the submodularity of the gain function $g$:

**Lemma 1 (submodularity).** *The gain function $g : 2^V \to \mathbb{R}$ is submodular, i.e., for any subsets of nodes $X, Y \subseteq V$, $g(X \cup Y) + g(X \cap Y) \leq g(X) + g(Y)$ holds.*

*Proof.* Note that the following holds for a node set $S \subseteq V$ by definition:

$$g(S) = \sum_{v \in V(S)} (c(v) - c_S(v)).$$

Thus,

$$\begin{aligned}
& g(X) + g(Y) - (g(X \cup Y) + g(X \cap Y)) \\
= & \sum_{v \in V(X)} (c(v) - c_X(v)) + \sum_{v \in V(Y)} (c(v) - c_Y(v)) \\
& - \left( \sum_{v \in V(X \cup Y)} (c(v) - c_{X \cup Y}(v)) + \sum_{v \in V(X \cap Y)} (c(v) - c_{X \cap Y}(v)) \right) \\
= & \left( \sum_{v \in V(X) \cup V(Y)} c_{X \cup Y}(v) + \sum_{v \in V(X) \cap V(Y)} c(v) \right) \qquad (1) \\
& - \left( \sum_{v \in V(X)} c_X(v) + \sum_{v \in V(Y)} c_Y(v) \right) \qquad (2) \\
& - \sum_{v \in V(X \cap Y)} (c(v) - c_{X \cap Y}(v))
\end{aligned}$$

holds, where the last equality comes from Proposition 3 (i). By definition, the following two equations are satisfied:

$$\begin{aligned}
\sum_{v \in V(X)} c_X(v) &= \sum_{v \in V(X) \setminus V(Y)} c_X(v) + \sum_{v \in V(X) \cap V(Y)} c_X(v) \\
\sum_{v \in V(Y)} c_Y(v) &= \sum_{v \in V(Y) \setminus V(X)} c_Y(v) + \sum_{v \in V(X) \cap V(Y)} c_Y(v).
\end{aligned}$$

Also, as for the union $X \cup Y$ of bookmarks,

$$\begin{aligned}&\sum_{v \in V(X) \cup V(Y)} c_{X \cup Y}(v) \\ =& \sum_{v \in V(X) \setminus V(Y)} c_{X \cup Y}(v) + \sum_{v \in V(Y) \setminus V(X)} c_{X \cup Y}(v) + \sum_{v \in V(X) \cap V(Y)} c_{X \cup Y}(v)\end{aligned}$$

holds. Thus, the above two terms (1) and (2) can be replaced by the following terms:

$$\begin{aligned}\sum_{v \in V(X) \setminus V(Y)} (c_{X \cup Y}(v) - c_X(v)) + \sum_{v \in V(Y) \setminus V(X)} (c_{X \cup Y}(v) - c_Y(v)) \\ + \sum_{v \in V(X) \cap V(Y)} (c(v) + c_{X \cup Y}(v) - c_X(v) - c_Y(v)) \, .\end{aligned}$$

Note that even if we add bookmarks to nodes in $Y$, the length of the shortest path from $r$ to a node $v$ does not change if $v \in V(X) \setminus V(Y)$. Thus,

$$\sum_{v \in V(X) \setminus V(Y)} (c_{X \cup Y}(v) - c_X(v)) = 0.$$

From a similar reason,

$$\sum_{v \in V(Y) \setminus V(X)} (c_{X \cup Y}(v) - c_Y(v)) = 0.$$

As a result,

$$\begin{aligned}& g(X) + g(Y) - (g(X \cup Y) + g(X \cap Y)) \\ =& \sum_{v \in V(X) \cap V(Y)} (c(v) + c_{X \cup Y}(v) - c_X(v) - c_Y(v)) & (3) \\ & - \sum_{v \in V(X \cap Y)} (c(v) - c_{X \cap Y}(v)). & (4)\end{aligned}$$

Since $c_{X \cup Y}(v) = \min\{c_X(v), c_Y(v)\}$ for $v \in V(X) \cap V(Y)$ from Proposition 3-(i), the term (3) can be replaced to the following:

$$\sum_{v \in V(X) \cap V(Y)} (c(v) - \max\{c_X(v), c_Y(v)\}) \, . \tag{5}$$

Since $c_{X \cap Y}(u) \geq c_X(u)$ and $c_{X \cap Y}(u) \geq c_Y(u)$ hold for any vertex $u$ from $X \cap Y \subseteq X$ and $X \cap Y \subseteq Y$, respectively, the inequality $c_{X \cap Y}(v) \geq \max\{c_X(v), c_Y(v)\}$ holds for $v \in V(X \cap Y)$. Hence the following inequality is true for the term (4):

$$\sum_{v \in V(X \cap Y)} (c(v) - c_{X \cap Y}(v)) \leq \sum_{v \in V(X \cap Y)} (c(v) - \max\{c_X(v), c_Y(v)\}) \, . \tag{6}$$

From (5), (6), and Proposition 3-(ii), we can obtain the following:

$$\begin{aligned} & g(X) + g(Y) - (g(X \cup Y) + g(X \cap Y)) \\ \geq & \sum_{v \in (V(X) \cap V(Y)) \setminus V(X \cap Y)} (c(v) - \max\{c_X(v), c_Y(v)\}) \\ \geq & \ 0. \end{aligned}$$

Therefore, the gain function $g$ is submodular. □

Before concluding this section, we mention a possibility of speeding the algorithm up. In Greedy, we need to compute the shortest path problem $O(|V|)$ times for each iteration. Although the shortest path of a directed acyclic graph can be computed in $O(|E|)$ time, it still might be time-consuming and actually many parts of the computation may be redundant; since most of the graph structure is preserved even if we add a bookmark, the shortest path computation can reuse the previous computation. This idea can be implemented by using several results from dynamic topological sort algorithms (cf., [12, 1]), although we do not give the detail here because the main scope of the paper is the approximation factor.

Another issue of the running time is on the restriction of graph classes. One of the most common subclasses of directed acyclic graphs is a tree. In the case of trees, we actually can reduce the running time by the following: We prepare $q(u)$ and $w(u)$ for each node $u$ to store the distance of $u$ from the root $r$ and $w(u) = \sum_{v \in V(u)} p_v$, respectively. Then, in each iteration of the for-loop, the algorithm processes the following operations:

1. Pick a node $b$ which has the maximum gain $q(b) \cdot w(b)$ as a new bookmark,
2. Replace $q(u)$ with $q(u) - q(b) + 1$ for each node $u \in V(b)$, and
3. Replace $w(u)$ with $w(u) - w(b)$ for each node $u$ on the simple path from $r$ to $b$ (exclusive).

It is easy to see that the above procedure maintains $q$ and $w$ correctly, and each step can be done in $O(|V|)$ time. Hence the total running time of Greedy for trees is reduced to $O(k|V|)$. It is much faster than the running time of the known exact algorithm in [11], $O(k^2|V|^3)$, although Greedy obtains only approximate solutions.

## 3 Lower Bounds

We show that the $(1 - \frac{1}{e})$ approximation ratio proved in the previous section is the best possible for $k$-BAP, in the sense that no approximation algorithm with factor $(1 - \frac{1}{e} + \varepsilon)$ exists for any $\varepsilon > 0$, unless $\mathcal{NP} \subseteq \mathcal{DTIME}(N^{O(\log \log N)})$. The hardness of approximation is shown via a *gap-preserving reduction* [2] from the Unit Cost Maximum Coverage Problem (UCMCP):

**Unit Cost Maximum Coverage Problem (UCMCP):**

Instance: A collection of sets $S = \{S_1, S_2, \cdots, S_m\}$ with associated unit cost $c_i = 1$ for each $i = 1, 2, \cdots, m$ defined over a domain of elements $X = \{x_1, x_2, \cdots, x_n\}$ with associated unit weight $w_j = 1$ for each $j = 1, 2, \cdots, n$, and a positive integer $\ell \le |S|$.

Goal: Find a collection of sets $S' \subseteq S$ such that the total cost of sets in $S'$ does not exceed $\ell$, and the total weight of elements covered by $S'$ is maximized.

**Theorem 2 ([8]).** *No approximation algorithm with approximation ratio better than $(1 - \frac{1}{e})$ exists for UCMCP unless $\mathcal{NP} \subseteq \mathcal{DTIME}(N^{O(\log \log N)})$.*

Let $OPT_{mcp}(I)$ denote the weight of elements covered by a collection of sets output by an optimal algorithm for the instance $I$ of UCMCP. Also, let $OPT_{bap}(G)$ be the gain of bookmarks output by an optimal algorithm for the graph $G$ of $k$-BAP.

**Lemma 2.** *There is a gap-preserving reduction from UCMCP to $k$-BAP that transforms an instance $I$ of UCMCP to a graph $G = (V, E)$ of $k$-BAP such that*

**(i)** *if $OPT_{mcp}(I) = \texttt{max}$, then $OPT_{bap}(G) \ge \frac{h}{n}\texttt{max}$, and*

**(ii)** *if $OPT_{mcp}(I) \le (1 - \frac{1}{e})\texttt{max}$, then $OPT_{bap}(G) \le (1 - \frac{1}{e} + \frac{\ell}{h \cdot \texttt{max}})\frac{h}{n}\texttt{max}$, where $h$ is a sufficiently large integer such that $h = O(|I|^q)$ for some constant $q$.*

*Proof.* Consider an instance $I$ of UCMCP; a collection of sets $S = \{S_1, S_2, \cdots, S_m\}$ defined over a domain of elements $X = \{x_1, x_2, \cdots, x_n\}$, and a positive integer $\ell$. Then, we construct the following directed graph, illustrated in Figure 1. Let $V_R = \{r, r_{1,1}, r_{1,2}, \cdots, r_{1,h}, r_{2,1}, r_{2,2}, \cdots, r_{m,h-1}, r_{m,h}\}$ be a set of $1 + mh$ nodes associated with $|S| = m$. Also, let $V_S = \{\alpha_1, \alpha_2, \cdots, \alpha_m\}$ be a set of $m$ nodes corresponding to the $m$ sets, $S_1$ through $S_m$, and $V_X = \{\beta_1, \beta_2, \cdots, \beta_n\}$ be a set of $n$ nodes corresponding to the $n$ elements, $x_1$ through $x_n$. Here, $V_R$, $V_S$, and $V_X$ are pairwise disjoint, and let $V = V_R \cup V_S \cup V_X$. The set $E$ of directed edges is defined as follows: $E_1 = \{(r, r_{i,1}), (r_{i,1}, r_{i,2}), \cdots, (r_{i,h}, \alpha_i) \mid i = 1, 2, \cdots, m\}$, $E_2 = \{(\alpha_i, \beta_j) \mid x_j \in S_i$ for each $i$ and each $j\}$, and $E = E_1 \cup E_2$. As for the access probabilities of nodes, we define $p_{\beta_j} = \frac{1}{n}$ for $j = 1, 2, \cdots, n$, $p_r = 0$, and $p_{r_{i,1}} = p_{r_{i,2}} = \cdots = p_{r_{i,h}} = p_{\alpha_i} = 0$ for $i = 1, 2, \cdots, m$. Finally, we set $k = \ell$. Clearly this reduction can be done in polynomial time since $h = O(|I|^q)$.

(i) Suppose that $OPT_{mcp}(I) = \texttt{max}$ and the optimal collection of sets is $OPT = \{S_{i_1}, S_{i_2}, \cdots, S_{i_\ell}\}$ where $\{i_1, i_2, \cdots, i_\ell\} \subseteq \{1, 2, \cdots, m\}$. Then, if we select a set $B = \{\alpha_{i_1}, \alpha_{i_2}, \cdots, \alpha_{i_k}\}$ of $k$ $(= \ell)$ nodes as bookmarks, then we can obtain the gain of $\frac{h}{n}\texttt{max}$ since $d(\alpha_{i_j}) - d_B(\alpha_{i_j}) = h$ for each $j$ and the nodes $\alpha_{i_j}$'s are connected with exactly $\texttt{max}$ leaves $\beta$'s of probability $\frac{1}{n}$.

(ii) Next suppose that $OPT_{mcp}(I) \le (1 - \frac{1}{e})\texttt{max}$ and again the optimal collection of sets is $OPT = \{S_{i_1}, S_{i_2}, \cdots, S_{i_\ell}\}$ where $\{i_1, i_2, \cdots, i_\ell\} \subseteq \{1, 2, \cdots, m\}$. If a set $B = \{\alpha_{i_1}, \alpha_{i_2}, \cdots, \alpha_{i_k}\}$ of $k$ $(= \ell)$ nodes is selected as bookmarks, then the gain is at most $(1 - \frac{1}{e})\frac{h}{n}\texttt{max}$. For example, the replacement of $\alpha_{i_j}$ with, say, $r_{i_j,h}$ decreases the gain by $\frac{1}{n}$ because $d(r_{i_j,h}) - d_B(r_{i_j,h}) = h - 1$. On the other hand,

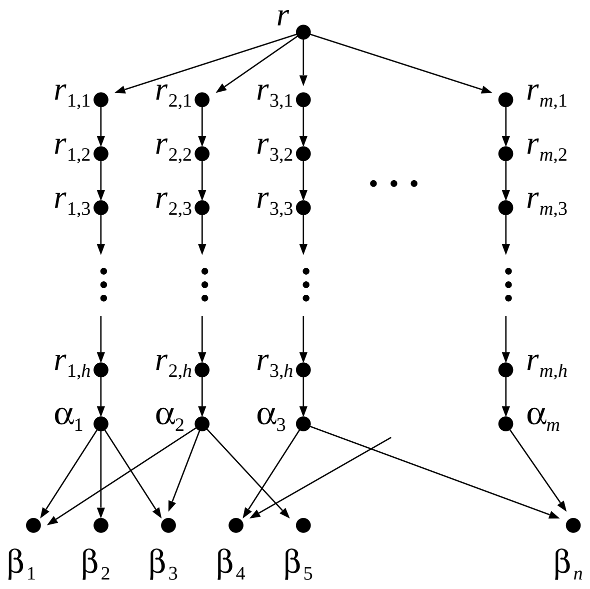

**Fig. 1.** Gap-preserving reduction

if we select, say, $\beta_1$ instead of $\alpha_{i_j}$ as a bookmark, then the gain possibly increases at most by $(h+1)\cdot\frac{1}{n} - h\cdot\frac{1}{n} = \frac{1}{n}$, i.e., at most $\frac{1}{n}$ gain per such replacement. As a result, the gain on $G$ is at most $(1-\frac{1}{e})\frac{h}{n}\mathtt{max} + \frac{k}{n} = (1-\frac{1}{e}+\frac{\ell}{h\cdot\mathtt{max}})\frac{h}{n}\mathtt{max}$. This completes the proof. □

The following theorem is obtained by the above lemma:

**Theorem 3.** *No $(1-\frac{1}{e}+\varepsilon)$ factor approximation algorithm exists for k-BAP unless $\mathcal{NP} \subseteq \mathcal{DTIME}(N^{O(\log\log N)})$, where $\varepsilon$ is an arbitrarily small positive constant.*

A similar gap-preserving reduction from the Maximum $k$-Vertex Cover Problem [7, 9, 16] gives us the following hardness of approximation under the different weak assumption:

**Theorem 4.** *No $(1-\delta)$ factor approximation algorithm exists for k-BAP unless $\mathcal{P} = \mathcal{NP}$, where $\delta$ is a fixed small positive constant.*

*Proof.* Let $\delta_0 < 1$ be the approximation hardness factor of the Maximum $k$-Vertex Cover Problem. By using similar ideas in [2, 9, 7], we can provide the gap-preserving reduction with $\delta = \frac{1305}{1349}\cdot(1-\delta_0)$. Details are omitted. □

## 4 Conclusion

In this paper we have considered the problem of assigning bookmarks from the root to the nodes of a DAG in order to maximize the gain in the expected cost.

Then we have shown that there is a polynomial time approximation algorithm with factor $(1 - 1/e)$ and the factor is the best possible under the assumption that $\mathcal{NP}$ is not in $\mathcal{DTIME}(N^{O(\log \log N)})$.

As for further researches, there is a gap between the inapproximability result we have shown under the assumption that $\mathcal{P} \neq \mathcal{NP}$, and the approximability result of $(1 - 1/e)$ ratio. To reduce the time complexity for trees is another interesting topic.

## References

1. B. Alpern, R. Hoover, B.K. Rosen, P.F. Sweeney and F.K. Zadeck, Incremental evaluation of computational circuits, in *Proc. SODA'90*, pp. 32–42, (1990)
2. S. Arora and C. Lund. Hardness of Approximation, in *Approximation Algorithms for NP-hard problems* (D.S. Hochbaum, ed), PWS publishing company, pp. 399–446 (1995)
3. P. Bose, E. Kranakis, D. Krizanc, M. Vergas Martin, J. Czyzowicz, A. Pelc, J. Gasieniec. Strategies for hotlink assignments. In *Proc. ISAAC'00*, pp.23–34 (2000)
4. J. Czyzowicz, E. Kranakis, D. Krizanc, A. Pelc, and M. Vergas Martin. Assigning bookmarks in perfect binary trees. *Ars Combinatoria*, **LXXXII** (2007)
5. T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms (Second Edition)*, MIT Press (2001)
6. S. Fuhrmann, S.O. Krumke, and H.-C. Wirth. Multiple hotlink assignment. In *Proc. WG'01*, pp.189–200 (2000)
7. U. Feige and M. Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *J. Algorithms* **41** (2), pp.174–211 (2001)
8. S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *IPL* **70** (1), pp.39–45 (1999)
9. M. Langberg. Approximation Algorithms for Maximization Problems arising in Graph Partitioning. *M. Sc. thesis, Weizmann Institute of Science* (1998)
10. B. Li, X. Deng, M.J. Golin, and K. Sohraby. On the optimal placement of web proxies in the Internet: Linear Topology, In *Proc. HPN'98*, pp.485–495 (1998)
11. B. Li, M.J. Golin, G.F. Italiano, X. Deng, and K. Sohraby. On the optimal placement of web proxies in the Internet. In *Proc. INFOCOMM'99*, pp.1282–1290 (1999)
12. A. Marchetti-Spaccamela U. Nanni and H. Rohnert, Maintaining a topological order under edge insertions, *IPL*, **59** (1), pp.53–58 (1996)
13. R. Matichin and D. Peleg. Approximation algorithm for hotlink assignments in web directories. In *Proc. WADS'03*, pp.271–280 (2003)
14. R. Matichin and D. Peleg. Approximation algorithm for hotlink assignment in the greedy model. In *Proc. SIROCCO'04*, pp.233–244 (2004)
15. G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming* **14**, pp.265–294 (1978)
16. E. Petrank. The hardness of approximation: gap location. *Computational Complexity* **4**, pp.133–157 (1994)
17. M. Perkowitz and O. Etzioni. Towards adaptive web sites: conceptual framework and case study. *Computer Networks* **31**, pp.1245–1258 (1999)
18. A.A. Pessoa, E.S. Laber, C. de Souza. Efficient algorithms for the hotlink assignment problem: the worst case search. In *Proc. ISAAC'04*, pp.778–792 (2004)

19. M. Vergas Martin. Enhancing Hyperlink Structure for Improving Web Performance. *PhD thesis, School of Computer Science, Carleton University* (2002)
20. R.V. Vohra and N.G. Hall. A probabilistic analysis of the maximal covering location problem. *Discrete Applied Mathmatics* **43** (2), pp.175–183 (1993)