A Hierarchical Model for Auto-adjusting of Information Issuance

Chenguang Luo[†], Rong Pan[††] and Shan Wang[†††],

[†] Peking University, Beijing, 100871, China
[††] Wuhan University of Science and Technology, Wuhan, 430081, China
[†††] Hunan University of Science and Technology, Xiangtan, 411201, China

Summary
This paper brings out a mathematical model of information for its auto-adjusting when issued in different hardware environment from a hierarchical perspective, discusses possible solutions to the construction of the model, and gives two implementations of algorithms for solutions together with their complexity analysis.

Key words:
information issuance; auto-adjusting; hierarchical model; algorithms

Introduction

As the world is entering an information era, the scale of sorts of information is expanding at an exponential speed. This brings about a significant and practical problem with the issuance of information. It mainly lies in that different hardware devices have different capability of information exhibition, which leads to misrepresentation of information on some terminals and inconvenience for people to recognize it. For example, an ordinary web page would fit almost any computer display, but on the screen of a PDA or even a mobile phone it may turn out a mess: the limited screen cannot hold so many contents and the user will have to make every effort to scroll the page for something she is interested in. With the development of various interconnected devices, the situation is still deteriorating.

Such problem calls for the auto-adjusting in the issuance of information. That is, the device should automatically choose a best way to display it. Take the former example, a web page with the ability of auto-adjusting will exhibit itself in a multimedia style when the terminal is a computer; however, it will reduce the quality of pictures on a PDA, and only display some word-based descriptions if shown on a mobile phone. In a word, the auto-adjusting of information will allow the terminal to display contents according to the competence of hardware device, to change the information into an acceptable form, and to reduce the misrepresentation to a lowest level.

This paper will discuss the auto-adjusting of information with the following arrangements. The next section will create the hierarchical model of information, then the two sections followed will propose and compare two algorithms about the model, and the last section summarizes.

Manuscript received December 14, 2006.
Manuscript revised December 18, 2006.

1. Hierarchical Model for Auto-adjusting of Information Issuance

1.1 Preliminaries

The introduction has given a basic description to the problem of information issuance, and the objective of this section is to formalize the method of information auto-adjusting by modeling the information to be issued, and to simplify the model to a solvable level.

Before any formal discussion, the final goal of auto-adjusting of information issuance should be explained. As is depicted, the same information should be exhibited in certain forms on different devices respectively, according to the capability of the device, with the aim that it should make full use of the limited resource of the terminal to ensure the correctness and wholeness of the information.

Based on this goal, we focus on the abstract structure of the information to find a suitable model for it. The structure should have the feature to enable the information to be condensed at a proper degree when necessary, such as being browsed on a mobile phone; meanwhile, the structure must be feasible in computational complexity, to guarantee a tolerable time cost for readers.

For these two features, one possible solution is the hierarchical model of information, providing different degrees of information abstraction on different levels of the model, and the needed information can be computed from the model efficiently.

Prior to the definition of the model, an informal (and ideal) way to compress the information will be described. First divide the information into 2^n equal parts, and then combine each adjacent two parts into one part with the same size of any original part, as the abstraction of the former two parts. Now the 2^{n-1} parts of information are a compressed edition of the original information. Repeat the process until the result turns out only one part.

In fact, the former compression method generates a hierarchical structure of information. Denote the result of 2^i parts of each dividing and compression process as one level of information, and we gain $(n + 1)$ levels of information. Then, on competent display devices (such as a

computer monitor) we may issue the information on lower levels to get a better view, and on devices with limited resources (such as mobile phones) the information may be shown in an abstracted style for the reader's convenience. In this way, the information will have a correct and whole exhibition on any device, and our aim can be achieved.

The informal discussion above has shown a hierarchical model for information issuance, of which the search algorithm will set a base for the auto-adjusting.

1.2 The Hierarchical Model

Suppose in the process of information division above, any two parts at a lower level will take no less resource for issuance than their counterpart at a high level does. Then the problem can be described as the following form.

Definition 1. Let $m, n \in N$, denote m as n 's *parent* if $n=2m+1$ or $n=2m+2$.

Definition 2. Let $m, n \in N$, denote m as n 's *ancestor* if a sequence $m, m_1, m_2, \ldots, m_k, n$ exists, such that in each adjacent pair, the left one is the parent of the right one.

With the two definitions above, the problem can be changed to a new form: given a sequence of positive real values $w_0, w_1, \ldots, w_{2^k-2}$, where $w_m \le w_{2m+2} + w_{2m+3}$, construct a set $S \subseteq \{0,1,\ldots, 2^k-2\}$, satisfying that for any a, $\forall 2^{k-1}-1 \le a \le 2^k-2, \exists b \in S$, $b=a$ or b is a 's parent, and $\sum_{i \in S} w_i < w$, where w is a given value.

According to the former informal discussion about information division and its levels, natural numbers from $2^{k-1}-1$ to 2^k-2 in the conditions above represent the 2^{k-1} parts divided from the original information. Then each two parts will be combined to a new part on a higher level of information, and the parts on the new level are still combined to construct a higher level … until only one part left on the top level. Each level consists of 2^{i-1} parts ($1 \le i \le k$), standing for different degrees of information abstraction. Among all these parts on all the levels, some will be selected for issuance, taking no more resource than w (the upper limit of terminal device).

In data structures, a natural way to model different levels of data is trees. So the division of information and the resulted levels can be formalized as the following definition.

Definition 3. Denote a full binary tree T as a *hierarchical model*, if each node of T has been assigned a natural number as a subscription (the assignment is in breadth-first order and begins at zero), and each node n_i has unique weight w_i, where parent node n_i and its children n_{2i+1}, n_{2i+2} have an inequality $w_i \le w_{2i+1} + w_{2i+2}$ for their weights.

Now the problem of information division and issuance can be transformed to a problem of the hierarchical model. To describe the solution to the problem we still need another definition.

Definition 4. Set S is defined to be a *cover set* of a hierarchical model T, if $\forall n \in S, n \in T$ holds for any node n, and for any leaf node $n_a (2^{k-1}-1 \le a \le 2^k-1)$ in T, on the path from it to the root node n_0, there exists one and only one node n_s, such that $n_s \in S$. (For empty hierarchical models, their cover sets are defined as empty sets.)

Intuitively, the nodes in a cover set "cover" all the original information to be issued; so in the process of issuance, the nodes in a cover set are both necessary and sufficient. Now the problem is changed to a new form: for certain hierarchical model T and constant w, find a cover set S of T, such that provided the sum of weights of the nodes in S does not exceed w, $|S|$ should be maximized.

2. The Best-solution Algorithm for the Model

2.1 Analysis of Cover Sets

In the last section we have defined the hierarchical model for information issuance, and turned the problem to a more formal one. For any given hierarchical model T and constant w, a natural way to find a best (or largest) cover set S is to try every cover set of T, and choose the one with the most nodes and not exceeding the weight limit w. This section will go far in this way to find a best solution.

First we do not take weights into consideration, but just focus on cover set itself. The set containing only the root, namely, $\{n_0\}$, is a cover set. Besides this one, there are still many cover sets, which all follow the rule that any of them are built up by two smaller cover sets, one of which is a cover set of the left child tree of the model's root, while the other is a cover set of the right child tree. It is not difficult to prove that only these two types of cover sets exist. So an equivalent definition of cover set may be stated as follows.

Definition 5. Set S is defined to be a *cover set* of a hierarchical model T, if $S = \{n_0\}$, or $S = S_l \cup S_r$, where S_l and S_r are cover sets of n_0's left and right child trees, respectively. (For any empty hierarchical model, its cover set is defined as empty set.)

This recursive definition reveals the essence of cover sets; meanwhile, it shows a direct way to find all the cover sets of a hierarchical model. First find all the cover sets of the left child tree, then all of them of the right child tree, and at last make a combination with each cover set from each side, plus the cover set containing only the root. In the following section, this idea will be implemented as an algorithm.

2.2 The Algorithm Based on Direct Search

In the last section a recursive definition of cover set was given, which inspires a natural way to generate all the cover sets of a hierarchical model. This section first introduces an algorithm to search for all the cover sets, and then the best one could be chosen from them.

Suppose the model has on every node a *flag* (initially 0) and two variables to store cover sets (*leftcoverset* and *rightcoverset*, initially NULL), then the algorithm to find next cover set of a tree can be designed as follows.

```
function NEXT_COVER_SET(btree) returns next
  cover set of btree
  inputs: btree, the binary tree
  if btree.root.flag = 0 then
    btree.root.flag ← 1
    return {btree.root}
  elseif btree.root.leftchild = NULL then
    return NULL
  else
    if btree.root.leftcoverset = NULL then
      btree.root.leftcoverset ← NEXT_COVER_
        SET (btree.root.leftchildtree)
    end if
    btree.root.rightcoverset ← NEXT_COVER_
      SET (btree.root.rightchildtree)
    if btree.root.rightcoverset = NULL then
      btree.root.leftcoverset ←
      NEXT_COVER_SET (btree.root.leftchildtree)
      if btree.root.leftcoverset = NULL then
        btree.root.flag ← 0
      else
        btree.root.rightcoverset ← NEXT_
        COVER_SET (btree.root.rightchildtree)
      end if
    end if
  end if
  return btree.root.rightcoverset ∪
    btree.root.rightcoverset
end NEXT_COVER_SET
```

Figure 1: Algorithm for Best Solution

It is easy to see the algorithm is based on the direct search strategy. It just utilizes the recursive definition of cover sets, and returns the next cover set by each call.

With this algorithm, the task to find the best cover set turns out simple. The procedure NEXT_COVER_SET should be invoked once and again, with the root as the parameter, until NULL is returned. Then the largest cover set gained before is the best solution to the problem. Below is an example of such solution, with weights marked beside the nodes, and the weight limit w = 6. The black nodes are the components of the cover set.

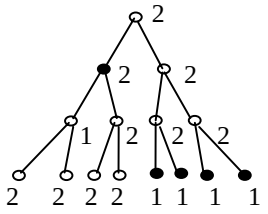

Figure 2: Best solution of a problem

2.3 Analysis of the Algorithm

The algorithm above provides the best solution to any hierarchical model; however, the complexity of the algorithm is far from the best. The algorithm has to traverse all the possible cover sets and finds the best at last. So it is essential to estimate the scale of solution space according to the size of the model.

When the model has only one node, it has only one cover set, say, $T_1 = 1$. Then, on the condition of a model with three nodes, considering the recursive definition of cover set, it may be found that each child tree of the root has exactly T_1 cover set (s), so the total cover sets add up to $T_3 = 1 + T_1^2 = 2$. In this way, consequent results are $T_7 = 5$ and $T_{15} = 26$. To conclude, the general formula is

$$T_{2^k-1} = 1 + T_{2^{k-1}-1}^2. \qquad (k > 1) \quad (1)$$

Take another sequence S_i, where $S_i = T_{i-1}$. Then (1) can be written in a new form

$$S_{2^k} - 1 = S_{2^{k-1}}^2. \qquad (k > 1) \quad (2)$$

From the discussion above, it can be seen that the scale of cover sets is exponential to that of nodes, since for sequence $R_4 = 2, R_{2^k} = R_{2^{k-1}}^2 (k > 2)$, or $R_{2^k} = 2^{2^{k-2}} (k \geq 2)$, the value of each item, compared with its subscription, is the level of power of 2, and for each meaningful i we even have $R_i < S_i$. In a word, the scale of cover sets is, unfortunately, an exponential blow-up of that of nodes.

To this algorithm, some improvement can be made to better its performance. For example, it may use a depth-first search with certain strategy to generate the cover set with the least weight first, then increase the weight of cover sets gradually and cut out any cover sets exceeding the weight limit and their possible descendants. This pruning method may reduce the cost for best solution to a minimal degree, but it cannot get off the trap of the $O(2^n)$ complexity.

Based on the result above, the best solution algorithm is not practical for applications of information issuance. So we need a trade-off for a better balance between the quality of the solution and the time cost.

3. An Algorithm for Second-best Solutions

3.1 The Algorithm Based on Greedy Principle

Last section presents the algorithm for the best solution to the model, and its impracticality. For more feasible complexity we cannot search the whole space of cover sets, but should restrict the range of searching. The aim of this section is to propose an algorithm for second-best solutions to the problem, which will result in cover sets that are "not bad" and acceptable time cost.

There is no formal definition for what is a second-best solution; hence they can be achieved in many different ways. Here the greedy principle is adopted to conduct a local search for a possible solution to the model. It restricts the search within two adjacent levels of the model, namely, for given weight limit w and hierarchical model T, if $\sum_{i=2^k-1}^{2^{k+1}-2} w_i \le w \le \sum_{i=2^{k+1}-1}^{2^{k+2}-2} w_i$ holds, say, the k-th level of T has a sum of weight no more than w and the (k+1)-th level has a weight sum no less than w, then the search will base on all the nodes on the k-th level, and some of them will be replaced with nodes on the (k+1)-th level for a better result.

The design of the algorithm is directly from the idea above, and is exhibited as follows.

```
function SEEK(btree, w) returns the "max" cover set
  of btree
  inputs: btree, the binary tree
          w, the weight limit
  i ← 0
  while i ≤ n and weight ≤ w
    weight ← the total weight of the nodes on floor_i
    if i < 0 then result ← NULL
    elseif i=N+1 then result ← set of leaf nodes
    elseif 0 ≤ i ≤ N then
      SORT(the sum of node couples on floor_(i-1))
      j ← 0
      while j ≤ 2^i − 2
        if weight ≤ w after replacing the node_j on the
          floor_(i-1) with its children then
          Replace(j-th node, its children)
        end if
        j ← j + 1
      end while
    end if
  end while
  return result
end SEEK
```

Figure 3: Algorithm for Second-best Solution

The procedure of this algorithm can be divided into two steps. The first is to find a proper level as the search base, and the second is to replace some nodes on the base level with their two children. In the second step another greedy strategy is used, that the weight sums of children pairs on the level to be searched are calculated and sorted in an ascendant order, and pairs with less weight sums are earlier replaced. It also can be seen that the solutions are restricted in the two adjacent levels.

3.2 Analysis of the Algorithm

The greedy idea restricts the search within the nodes, not the cover sets, of the model, and each node is considered once at most. So the complexity of the algorithm is highly improved, that is, $O(n\log n)$ (where n is the number of nodes). This is feasible enough for most applications.

However, the solution got in this way is generally a possible solution. If take the number of nodes in the cover set as the only standard to judge the quality of a solution, then sometimes this solution can have a considerable gap from the best one, which is because of the imbalance of the weights on the left and the right trees.

Below is a somewhat extreme example.

Example 1. Suppose the nodes of a hierarchical model have their weights listed as w_0=2;w_1=2;w_2=2;w_3=2;w_4=2;w_5=1;w_6=2;w_7=2;w_8=2;w_9=2;w_{10}=2;w_{11}=1;w_{12}=1;w_{13}=1;w_{14}=1, and the weight limit is 6. Then the best cover set is $S=\{n_1, n_{11}, n_{12}, n_{13}, n_{14}\}$, shown in Figure 2. However, the second-best solution algorithm will find a "worse" result $S'=\{n_2, n_3, n_4\}$, shown in the figure below.

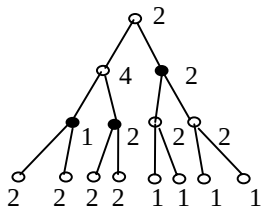

Figure 4: Second-best solution of the same problem

Now that this algorithm sometimes generates solutions that are not that good, then why should this algorithm be introduced? The most significant reason is that its complexity perfectly fits the needs of information issuance, especially in the network environment — this algorithm will not take the place of transmission speed as the bottleneck. Meanwhile, the nodes that this algorithm returns are placed on the two adjacent levels, so they are similar in abstraction degree as to the information issuance, and are more convenient for design and display of information format. At last, the solutions given by this algorithm are

mostly acceptable when the weight limit is not too small and the model is quite balanced, and below is a positive example, where the second-best solution is equal to the best one (the weight limit of which is 7).

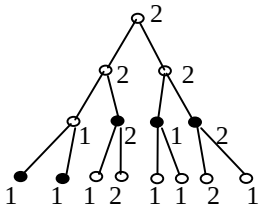

Figure 5: A best and second-best solution to another problem

Therefore, the algorithm for the second-best solution is still valuable for us to research on, and its implementation can be more suitable for practical use.

3.3 A Comparison between the Two Algorithms

We have made experiments over many different sorts of information and done statistics with them. Of the entire hierarchical model constructed, the total nodes count up to 3187, where best cover sets take up 1029, and second-best cover sets contain 954 of them; the ratio of the two types of solutions is about 1.08:1. It is found that, on a very large hierarchical model (and this is the only one), the best solution has ten more nodes than the second-best one; in other situations the difference of nodes gained by the two algorithms has not exceeded five, and many times there are only one node in difference.

With a little gap in nodes quantity, the two algorithms have distinct time performance. For all the 3187 nodes, the best solution algorithm takes 172.774 seconds in total and it is just 0.516 seconds for the second-best one. The former is over 300 times more than the latter one.

Now it is quite clear that the second-best solution algorithm is much better than the best solution one, taking both the service quality and the service time into consideration. As a matter of fact, in network environment, time is a precious resource. An HTTP server usually has a millisecond level time limit for the response of a webpage. So it would be better to utilize the algorithm second-best solutions in practical use.

4. Conclusion

In this paper the auto-adjusting of information issuance is briefly discussed, and a corresponding hierarchical model has been constructed for it. Upon this model, the problem is turned formal, and a best solution algorithm is found and analyzed. Meanwhile, another algorithm for second-best solutions is also designed, and compared with the former one, from perspectives of solution quality and time complexity. In the end, a conclusion is drawn that the second-best solution algorithm is more suitable in practical use.

References

[1] Aho, A., Hopcroft, J. and Ullman, J.: *Data Structures and Algorithms*, 1st ed. Pearson Education, 2003.
[2] Cormen, T., Leiserson, C. and et al: *Introduction to Algorithms*, 2nd ed. Beijing: Higher Education Press, May 2002.
[3] Kamimura, R.: Mediated and multi-level information processing, *IJCNN 99*, Jul 1999, vol. 2: 1397 – 1402.
[4] Knuth, D. E.: *The Art of Computer Programming, Vol 3, Sorting and Searching*. Addison-Wesley Co. Ltd. 1998.
[5] Martin, P. and Bateman, A.: Multi-level data transmission over mobile radio channels, *IEE Colloquium on Multi-Level Modulation Techniques and Point-to-Point and Mobile Radio*, Mar 1990.

Chenguang Luo received his both degrees of Bachelor of Science and Bachelor of Management in Peking University in 2005, and is now working in the PKU-TCL Laboratory for his Master of Engineering. His interests include computation and complexity theory, formal models and their respective algorithms.

Rong Pan received his degree of Bachelor of Engineering in Wuhan University of Science and Technology in 2006. His focus is mainly on data structures and algorithms, and their applications in practice. During his undergraduate career he had spent much time on the independence of information issuance.

Shan Wang received her degree of Bachelor of Arts in Hunan University of Science and Technology in 2006. As a student major in arts, her main focus is on the morphology of languages. Meanwhile she has a special interest in information processing and interchange, its formal models and related topics.