Knapsack Problem and Piece-Picking Algorithms for Layered Video Streaming

Tibor Szkaliczki*

Computer and Automation Research Institute
of the Hungarian Academy of Sciences
Kende u. 13-17, 1111 Budapest, Hungary
sztibor@sztaki.hu

Michael Eberhard

Institute of Information Technology
Klagenfurt University
Universitätsstrasse 65-67, 9020 Klagenfurt,
Austria
michael.eberhard@itec.uni-klu.ac.at

Hermann Hellwagner

Institute of Information Technology
Klagenfurt University
Universitätsstrasse 65-67, 9020 Klagenfurt,
Austria
hermann.hellwagner@itec.uni-klu.ac.at

László Szobonya

Computer and Automation Research Institute
of the Hungarian Academy of Sciences
Kende u. 13-17, 1111 Budapest, Hungary
szobonya@sztaki.hu

Abstract: The multimedia systems offer a wide application area for combinatorial optimisation. The presentation introduces variants of knapsack problem and shows their possible applications to the layered piece-picking problem arising when layered video content is streamed in peer-to-peer (P2P) networks. An efficient solution for layered piece-picking can significantly enhance the performance of video streaming and improve the quality of the provided services. A crucial requirement for the solution methods is that they should be performed very fast in order to provide short start-up delay and continuous streaming. The talk gives an overview of the implemented algorithms and provides their brief evaluation.

Keywords: knapsack problem, video streaming, layered/scalable content, multimedia, piece-picking

*Research is supported by grant Nr. OTKA 67651 of the Hungarian National Science Fund, Hungary

1 Introduction

The computer networks and multimedia systems offer a wide application area for combinatorial optimisation. The talk discusses the piece-picking problem arising in peer-to-peer (P2P) networks (e.g. Bittorent). In P2P systems, the P2P clients do not only consume the content but also participate in the distribution which provides a very popular and powerful alternative to the traditional client-server architecture. The content is split into pieces for subsequent time slots and the pieces can be independently downloaded from separate neighbour network nodes called peers.

Nowadays, content adaptation is a key issue in multimedia delivery as the users want to consume the content on various devices like HD TV sets, laptops, or mobile phones. The traditional approach offers the same content in different qualities encoded in different files. A possibility to provide different qualities in a more efficient way within one bitstream is provided by layered codecs (e.g. Scalable Video Coding, SVC) [1]. In case of layered video coding, the bitstream consist of several quality layers. All peers interested in the content can exchange the base layer, and the optional enhancement layers can be shared with all peers interested in the same or higher quality.

The goal of the piece-picking problem is to decide which content pieces should be downloaded at a given time point and from which peer in order to ensure continuous and high-quality streaming. The problem can be decomposed into two main parts: first the piece-selection decides which piece to select and then the peer-selection decides from where to download the pieces. The talk focuses on the first step but the peer-selection will be discussed in brief at the end. The piece selection problem is quite simple in P2P systems with single-layer content: streaming assigns priority to pieces based on their deadlines.

The main goal of our work is to solve the piece-selection problem if layered-content is streamed. In this case, the piece-picking algorithm needs to be modified in order to consider, in addition to the deadline, the layer of the piece. The pieces can be uniquely specified by their time slots and quality layers, therefore the decision should be made in the two-dimensional space. The pieces that can be considered to download at the time point of the decision can be well represented by the so-called sliding window consisting of rows representing quality layers and columns representing time slots.

The problem of finding the best trade-off between smooth playback and displaying the best possible quality without disturbing quality switches represents a challenging optimization problem. The piece-picking problem is very close to the Knapsack Problem (KP). Each piece owns a utility value characterizing its importance and a bandwidth need which play an analogous role as the item and size values in the Knapsack Problem. The pieces may be lost during the delivery and the network conditions may dynamically change, therefore the decision should be repeated for each time slot. A crucial requirement is that the pieces should be selected within strict time constraints.

To find a feasible piece-picking algorithm for layered content, several approaches have been investigated in our earlier paper [2]. In addition to the recommended algorithms introduced in that paper, this talk provides a deeper insight into the relationship between the knapsack problem and piece selection and also considers the peer-selection problem.

In the talk, our model of piece selection is presented first. Then some variants of Knapsack Problem are introduced and their relations to the piece selection problem are shown. Implemented algorithms for the piece selection of layered content that address this optimization

problem are enlisted with a short evaluation. In addition, the peer selection problem is also introduced. Finally, a short conclusion is given.

2 The Problem Model of Piece Selection

Let m and n denote the number of columns (time slots) and rows (layers) in the sliding window, respectively. Let l_i denote the ith layer of the stream ($i = 0..n-1$). Let $P_{i,j}$ denote the piece of the stream of layer l_j for the ith time slot in the sliding window. Let the utility of piece $P_{i,j}$ be denoted by $u_{i,j}$. The utility of a piece is based on its layer, deadline, and download probability (see [3] for the calculation method). Let the size of the piece be denoted by $c_{i,j}$ giving the bandwidth need of its streaming. $x_{i,j}$ shows whether $P_{i,j}$ is selected for download or not. The maximum available download bandwidth S forms an upper bound of the total bandwidth need of the selected pieces.

The aim of the piece selection is to maximise the total utility of the selected pieces considering the dependency between the pieces, while the total size of the pieces is not more than the upper limit S. The piece selection can be defined formally as follows:

Maximise

$$\sum u_{i,j} \cdot x_{i,j} \tag{1}$$

Subject to

$$\sum_{i,j} c_{i,j} \cdot x_{i,j} \leq S \tag{2}$$

$$x_i \in \{0, 1\} \tag{3}$$

$$x_{i,j} \leq x_{i,j-1} \tag{4}$$

$$x_{i,j} \leq x_{i-1,j} \tag{5}$$

The first constraint expresses the limit on the total size of the pieces (Constraint 2). $x_{i,j}$ indicates whether the piece is selected or not selected for download (Constraint 3). The precedence between the pieces is described in constraints 4 and 5. Constraint 4 describes that if a piece is selected then all pieces in lower layers should be also selected. Constraint 5 ensures that a piece is selected only if the piece in the preceding time slot in the same layer is also selected for download. This constraint is added in order to avoid frequent quality switches, because the switches are usually more disturbing for the user than watching the video at slightly lower, but constant quality.

Furthermore, the utility and size values have usually some special properties. Typically, the size values of the pieces depend only on the layer but not on the time slot: $c_{i,j} = c_{i',j}$. The utilities are monotonically decreasing in our piece utility model [3]: $i' < i \Rightarrow u_{i',j} > u_{i,j}$ and $j' < j \Rightarrow u_{i,j'} > u_{i,j}$. In this case, it can easily be proven that a piece is selected in the optimal solution then all pieces in the preceding time slots in the same layer are also selected and, therefore, constraint 5 can be omitted in order to simplify the problem model.

3 Related Variants of Knapsack Problem

The piece selection process tries to maximise the utility of the selected pieces without surpassing the available resource limit. This problem is similar to the knapsack problem (KP), namely to its 0-1 version when at most one sample can occur from each item. The only differences are the last two precedence constraints. In this section, some variants of the KP are shown that are related to the piece selection problem.

3.1 Precedence-Constrained Knapsack Problem (PCKP)

The piece selection problem can be formulated as a special Precedence-Constrained Knapsack Problem (PCKP) that is a generalisation of the knapsack problem. It considers precedence relations between items: an item can be selected only when all of its predecessors are also selected for the knapsack. The problem can be solved optimally by dynamic programming [4]. Open pit mining is a typical application area where the blocks of ore can be represented by the different items to select. Our problem model as well can be considered as PCKP with special precedence constraints.

3.2 Multiple-Choice Knapsack Problem (MCKP)

The precedence constraint w.r.t layers 4 can be omitted introducing the Multiple-Choice Knapsack Problem (MCKP). In this special variant of the knapsack problem, there are several groups of items, and only one item should be selected from each group. Applying it to our case, there are as many groups as time slots. Instead of single pieces, the items correspond to sequences of pieces from the base layer to the particular quality layers at one time slot.

Cheok et al. [5] applied the MCKP to the Multi-Source Multi-Layer Selection (SLS) problem in order to select sources for layered content in a multi-source streaming environment (e.g., surveillance systems). In this case, the different quality streams coming from the same video source form the individual item groups. They adapt standard solutions of the knapsack problem to SLS such as greedy approximation and dynamic programming approaches.

Although P2P systems and multi-source video recording represent a different context for layered streaming (e.g. quality layers are considered as a whole in SLS while quality layers are split into pieces in P2P), the MCKP provides a common underlying algorithmic model for both problems.

3.3 Multiple-Choice Multidimensional Knapsack Problem (MMKP)

A further extension of the MCKP is the Multiple-Choice Multi-Dimensional Knapsack Problem (MMKP). In this case there are several knapsacks (neighbour peers), each of them with limited (download) capacity. The resource needs of the pieces can be described as a vector because the piece can be downloaded from a number of neighbour peers. The goal of applying the MMKP to our problem is to optimise the value of the selected pieces while none of the resources is exceeded. The main advantage of this approach is that it can consider the individual resources (bandwidth) provided by the neighbour peers instead of only the overall bandwidth.

For the MMKP a number of algorithms have been proposed in the literature [6] including heuristic methods and an exact branch-and-bound method that is useful for checking the validity of the solutions. Khan et al. [7] applied the MMKP for adaptive multimedia systems. This

approach proved to be an efficient method to solve problems like quality adaptation, admission control and integrated resource management.

4 Implemented Algorithms and Evaluation

The main motivation of our work is to integrate the support of layered content into an already existing P2P system [8]. This section shows the algorithms that we implemented and their evaluation in a simulation framework. Most of the introduced methods adapt algorithms solving Knapsack Problems. The running time is critical because the piece selection is executed repeatedly at the decision point of each time slot and long running time may cause additional delay for arriving pieces. For more details on the algorithms, see [2].

4.1 Baseline Algorithm

Baseline Algorithm is an efficient simple heuristics. The algorithm considers all pieces within the sliding window that are not currently being downloaded and firstly selects the pieces from the lowest layer, starting with the earliest deadline, and after selecting all pieces in this layer, continues to select pieces from the next higher layer, and so on while the overall capacity is enough to download the selected pieces in parallel. This method was implemented for comparison with the other more complex recommended methods.

4.2 Greedy Algorithm

The greedy solution of the knapsack problem [9] offers a fast solution: select the pieces whose size is not larger than the available resource one after the other in the decreasing order of the ratio of their utilities and size values. This approximation algorithm runs fast but the solution of the algorithm may be far from optimal in the worst case. The overall time complexity of the greedy piece selection method is $O(m \cdot n \cdot log(max(m, n)))$ because of the initial sorting of pieces.

4.3 Dynamic Programming Methods

Exact solutions for the KP using dynamic programming have already been studied extensively in the literature [10]. First, we implemented the standard dynamic programming solution and applied it directly to piece selection. The algorithm proceeds on each possible total size values and on the available pieces and for each size value and piece, it calculates the maximum utility that can be achieved using at most the current size and the pieces up to the current piece. The main disadvantage of this method is that it neglects the precedence constraints. The running time is $O(S \cdot m \cdot n)$ (the cost values go from 0 to S and the number of different pieces is $m \cdot n$). We developed and implemented another dynamic programming algorithm that considers the precedence constraints as well. Its running time is $O(S \cdot n^2 \cdot m)$.

4.4 MMKP-based heuristics

The MMKP can be easily mapped to the piece and peer selection problems. Due to its performance and applicability the HEU algorithm presented in [7] was selected for implementation. Its complexity is $O(m^2 \cdot (n-1)^2 \cdot z)$ (z is the the number of neighbour peers). The special property of this approach is that it performs the peer selection process as well.

4.5 Evaluation

The performance of the implemented algorithms was tested using the Oversim P2P simulation framework [11], which is based on the OMNeT++ simulation framework [12]. The following parameters were adjusted in different test settings: number and bandwidth of the neighbour peers, number and cost of layers, sliding window size, network conditions, and the utility parameters. For more details on the evaluation, see [3].

The performance of layered piece-picking algorithms were evaluated and compared to traditional single layer solutions. The multi layer solution performs clearly better in scenarios with limited bandwidth and/or if peers dynamically leave and join the system during the streaming which is typical in Bittorent-like systems.

We compared the implemented methods with each other. The greedy, both dynamic programming and the HEU methods can be called as KP-based methods in order to distinguish them from the baseline method. The stream gained by using baseline algorithm for piece-selection contained disturbing quality switches. We found that the performance of KP-based methods is very similar in terms of received peak signal-to-noise ratio (PSNR). Nevertheless, the algorithms differ in terms of time complexity (and hence runtime). We found that the dynamic programming method without considering the precedence constraint downloaded useless pieces while the greedy method selected pieces satisfying the precedence constraints due to the monotonic features of pieces (see the end of Section 2 "The Problem Model of Piece Selection"). In our simulations, the greedy algorithm performed as well as the other KP-based algorithms at clearly lower complexity.

5 Peer selection

After selecting pieces to download, the peers also have to be selected from where the pieces are downloaded. The peer selection can be regarded as a special case of host recommendation that answers the question which node to select in a network for hosting a server application or which host node to select for a special purpose. Some possible solution methods are enlisted in our talk held at an earlier Hungarian-Japanese Symposium [13]. The main simplification in our peer selection model compared to the host recommendation is that the underlying network structure is not considered but just the available bandwidth to each peer is taken into account. The peer selection process inputs the list of peers to download from and the list of selected pieces (the output from the piece selection). For each of the peers and selected pieces, the estimated download bandwidth and the utility is given, respectively.

This problem can be regarded as a generalised assignment problem, where tasks have to be assigned to agents. Each task-agent assignment has cost and profit for the agent. The agents have a limited budget. The aim is to maximise the total profit of the assignment while none of the agents exceeds his budget. In our case, the tasks and agents correspond to the selected pieces and peers, respectively. The profit is the utility of a piece while the cost is the bandwidth need and the budget is the available bandwidth to the peer. It is an NP-complete problem. The problem is also close to the Knapsack Problem. There is an approximation method, taking the agents one after the other while trying to assign optimally tasks to the current agent and refresh the profits of the assigned task. Essentially, this approach reduces the problem into a series of knapsack problems for each agent.

As we mentioned, the implemented heuristic algorithm for MMKP is appropriate to solve

this problem as well. Furthermore, we implemented a fast greedy algorithm giving priority on pieces with high utility and peers with large download capability,

6 Conclusion

We examined the piece-picking problem and found it close to the Knapsack and related optimization problems. Therefore, the algorithms developed for the knapsack problem can be adapted to solve the piece picking problem. We found that layered piece-picking can improve the quality of multimedia streaming. Due to the special properties of the utilities and sizes of pieces, the greedy algorithm performed surprisingly well providing similar quality with shorter time than other methods. In the future, we will implement the layered piece-picking algorithm in our real P2P systems and perform detailed measurements to validate the simulation results.

References

[1] H. Schwarz, D. Marpe, T. Wiegand, Overview of the Scalable Video Coding Extension of the H.264/AVC Standard, *IEEE Trans. on CSVT*, **17**, 9, (2007), pp. 1103-1120.

[2] T. Szkaliczki, M. Eberhard, H. Hellwagner, and L. Szobonya, Piece Selection Algorithm for Layered Video Streaming in P2P Networks, *Proceedings of the International Symposium on Combinatorial Optimization (ISCO)*, March 24-26, 2010, Hammamet, Tunisia, Electronic Notes in Discrete Mathematics, Elsevier, **36**, (2010), pp. 1265-1272.

[3] M. Eberhard, T. Szkaliczki, H. Hellwagner, L. Szobonya, C. Timmerer, Knapsack Problem-based Piece-Picking Algorithms for Layered Content in Peer-to-Peer Networks, *Workshop on Advanced Video Streaming Techniques for Peer-to-Peer Networks and Social Networking, ACM Multimedia*, October 25-29, Firenze, Italy (2010).

[4] N. Samphaiboon, T. Yamada, Heuristic and exact algorithms for the precedence-constrained knapsack problem, *Journal of Optimization Theory and Application*, **105**, (2002), pp. 659-676.

[5] L-T. Cheok, A. Eleftheriadis, Operations Research Approach Towards Layered Multi-Source Video Delivery, *Picture Coding Symposium 2004*, San Francisco, CA, USA (2004).

[6] B. Han, J. Leblet, G. Simon, Hard multidimensional multiple choice knapsack problems, an empirical study, *Computers and Operations Research*, **37**, 1, (2010), pp. 172-181.

[7] S. Khan, K.F. Li, E.G. Manning, M. Akbar, Solving the knapsack problem for adaptive multimedia, *Studia Informatica (special issue on Cutting, Packing and Knapsacking Problems)* **2**, 1, (2003), pp. 157-178.

[8] Capovilla, N., Eberhard, M., Mignanti, S., Petrocco, R., and Vehkaper, J. 2010. An Architecture for Distributing Scalable Content over Peer-to-Peer Networks. *Proceedings of the Second MMEDIA Conference*, 1-6.

[9] G. B. Dantzig, Discrete-Variable Extremum Problems, *Operations Research*, **5**, (1957), pp. 266-277.

[10] R. Andonov, V. Poirriez, S. Rajopadhye, Unbounded knapsack problem: Dynamic programming revisited, *European Journal of Operational Research* **123**, 2, (2000), pp. 394-407.

[11] I. Baumgart, B. Heep, S. Krause, Oversim: A Flexible Overlay Network Simulation Framework, *IEEE Internet Symposium* (2007), pp. 79-84.

[12] Varga A., and Hornig R. 2008. An Overview of the Omnet++ Simulation Environment. *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications*, Networks and Systems, 1-10.

[13] Tibor Szkaliczki, Péter Kárpti and Balázs Goldschmidt: Two Combinatorial Optimisation Problems in Multimedia Systems, *In Proceedings of the 5th Japanese-Hungarian Symposium on Discrete Mathematics and its Applications*, April 3-5, 2007, Sendai, Japan, pp. 211-218