Serdica J. Computing **1** (2007), 365–386

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

ALGORITHMIC BACKGROUND OF THE HOST RECOMMENDATION IN THE ADAPTIVE DISTRIBUTED MULTIMEDIA SERVER

Tibor Szkaliczki*, Balázs Goldschmidt, Laszlo Böszörmenyi

Abstract. In a distributed server architecture an obvious question is where to deploy the components. Host recommendation, which gives the answer, faces problems such as server selection, host deployment and, in case of multimedia servers, video replication. It is especially relevant for the Adaptive Distributed Multimedia Server (ADMS) which is dynamically able to add and remove its components to different nodes of the network. The present survey paper introduces the different variants of host recommendation and gives an overview of its possible mathematical approaches. Emphasis is put on the facility location problem and the related approximation algorithms. Finally some algorithms selected for implementation are presented.

ACM Computing Classification System (1998): H.3.4, G.2.2, G.2.3.
Key words: distributed video server, host recommendation, optimisation, facility location problem.

*Partial support of the Hungarian State Eötvös Scholarship, the Hungarian National Science Fund (Grant No. OTKA 42559 and 42706) and the Mobile Innovation Center, Hungary is gratefully acknowledged.

1. Introduction. Host recommendation is a usual task for efficient operation of servers. It answers the question which node to select in a network for hosting a server application or which host to select for a special purpose. Such questions arise in different situations during the operation of a distributed multimedia server: From which server node should a client request be served? Where to put server components in the network? Where to store video instances? It is hard to find the correct answers because of several reasons. First, the operation of the server is influenced by several network and node parameters which are not easy to measure or to calculate accurately. Most of our decisions have effect on the future operation of the server but the real situation is not known exactly in advance. The multimedia server is a quite complex system and the problem model should also be capable to include features such as video replacement or caching. It is unclear in many cases which solution is the best for the host recommendation since the goodness of the system can be evaluated according to several conflicting aspects such as costs, user satisfaction, network load etc. Each video may be delivered with different parameters (resolution, frame rate, audio quality etc.) and it is not enough to select the locations of the hosts, but the video variation also have to be determined. The time-complexity of the algorithm is critical since the distributed video server may extend to large areas through the Internet and the number of the clients may be huge. In many cases, the algorithm should process large problems within strict time constraints.

This problem is especially relevant for the Adaptive Distributed Multimedia Server (ADMS) developed by the Klagenfurt University. ADMS is dynamically able to load and remove its components to different nodes of the network and this novel adaptive architecture enables migrating and replicating the server components in the network. Appropriate host recommendation algorithm with short running time is relevant for the distributed server to adapt quickly to the changing network parameters and client demands. We were motivated to deal with the host recommendation in order to improve the performance of the ADMS. Beside the dynamic placement of the server nodes, we studied other problems related to the host recommendation that are relevant for operating any distributed multimedia.

The present paper gives a survey on the algorithms related to the host recommendation. First, we give a brief overview of the ADMS server. Then we introduce the host recommendation problem in our focus and its variants including server selection, host deployment and video replication. Then we continue our paper by presenting the possible mathematical approaches. First of all, we discuss the facility location problem, its different variants and solution

methods. Then some further mathematical problems are introduced as well that can be connected with host recommendation such as the network flow and the knapsack problem. At last, we give a brief summary of the host recommendation algorithms that we applied.

2. Host recommendation problem.

The Adaptive Distributed Multimedia Server (ADMS). The modular structure of the ADMS [1] makes it possible to automate offensive adaptation strategies [2]. The ADMS has four components: the Data Distributor (DD) distributes the videos, received from a Production Client, onto Data Managers (DM, also called as storage nodes) which store stripe units of the videos. Data Managers storing the stripe units of the same video form a Data Manager group and they are located practically on the same subnet. The Data Collector (DC, also called as proxies or streamer nodes) collects these units, assembles them and streams the requested video to a Retrieval Client. Finally the ADMS Controller (AC) organizes how the server works where the host recommendation runs as well. The nodes that are able to host server components are called as harbours. The server is able to set up new components or remove old ones and also to migrate and replicate videos between DM groups according to the dynamically changing client requests, QoS parameters of the network, and loads of the nodes. To achieve this, the ADMS relies on an appropriate middleware called Vagabond2 [1].

Variants of the problem. The host recommendation is needed in different situations such as server selection, host deployment and video replication.

- Server selection: It assigns server node(s) to the client to serve its request. One client request may be processed at the same time or more. The problem is online in the sense that we do not know the whole series of requests in advance but only the next one.

- Host deployment: It determines nodes in the network where to place individual server components.

- Video replication: It selects storage nodes where the video instances should be placed. The origin of the video instance is also relevant. Usually, it generates huge network load.

We distinguish two main variations of the host recommendation problem: Light and Heavy Weight Recommendation. In case of Light Weight Recommendation (LWR), the task is to find optimal places for data collectors or simply to select one. This kind of recommendation has to be executed immediately after a client request has arrived. The aim of the Heavy Weight Recommendation (HWR) is to place optimally the instances of a video on the DM groups in the system. It is called heavy weight, because large amount of data is transmitted over the network and it is much more time consuming task than replicating or migrating data collectors. In this case the running time of the recommendation algorithm is much less critical. We can distinguish further subcases within these two main categories. Table 1 summarises them. The *static* version of *LWR* is relevant for servers where the location of data collectors is fixed. This is the usual case at multimedia servers. The *dynamic* case refers to the ADMS, where data collectors can be replicated. We speak about *Upload* when a client loads a new video into the system. HWR *Replication* refers to the case when the video is replicated to new storage nodes because the popularity of the video increases.

Type	Client requests	Frequency of calling	Time constraints	Replication	Selection
LWR static	Current	Continuous	Strict	—	DC, DM group, video variation
LWR dynamic	Current	Continuous	Strict	DC	DC, DM group, video variation
HWR Upload	Predicted	Occasional	Moderate	Video (DMs)	DD, DM group, video variation
HWR Replication	Predicted	Periodical	Loose	Video (DMs)	DM group, video variation

Table 1. The main aspects of the host recommendations depending on the time value.

2.3. The optimisation problem. Host recommendation deals with selecting special nodes in the network for a specified purpose. This problem occurs in several different situations during the operation of the server: host recommendation can be used for finding optimal location for the server components in the network or for selecting already occupied host nodes for a given purpose, for example to serve a client request or to store a video.

Unfortunately, the real problem is extremely complicated. For a detailed list of the input parameters see [5]. The parameters may vary depending on the type of the host recommendation problem (LWR or HWR). We mention some of them in order to demonstrate the complexity of the problem:

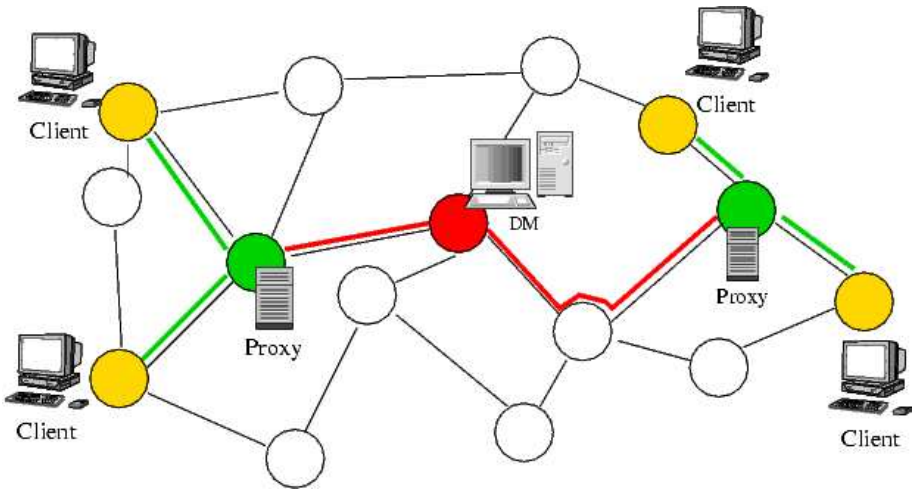

Figure 1. The goal is to find optimal location for multimedia proxies and data managers in a network [4]

- Terminal capability of each client.
- Current or possible locations of the server nodes.
- Locations of the existing or expected possible client requests.
- QoS parameters between the network nodes.
- Possible variations of the videos.
- Resource needs of variations.
- Profits from serving variations.
- Costs of the server nodes.
- Free local resources.
- etc.

The output of the model depends on the type of host recommendation. In case of LWR, the output should tell:

- for each client request, the streamer node serving it.
- for each client request, the storage nodes serving it.

- for each client request, which variation of the video to stream to the client

In case of HWR, it is not enough to determine the storage nodes where to put the video instance. The origin of the video instance is also needed for replicating or migrating the video instance.

The recommendation is done based on several contradictory optimisation criteria whose harmonisation is not a trivial task:

- Maximising the total value of the servable clients.
- Minimising the number of storage nodes.
- Maximising the quality of service between the clients and storage nodes.
- Maximising the quality of delivered video variations.
- Minimising the network load arising from replicating videos.

Multimedia services typically need huge resources. We always have to check whether the available network and the local resources are enough for streaming.

The distributed video server may cover large areas on the Internet and stream data can be delivered to large number of clients. The running time of the host recommendation algorithm becomes crucial for LWR in case of ADMS since the delivery of the data-streams can start only after the placement of the Data Collector nodes. In this case, the time-consuming algorithms aiming at the solution with exact optimum are not applicable.

3. Related Optimisation Problems.

3.1. Facility Location Problem (FLP).

3.1.1. Overview. Suppose that fire stations are planed to place in a city. The city has a number of sites, which can be used for fire stations. The cost of placing and maintaining them is known at each site. The question is where the fire stations should be located in order to minimise the total establishing and maintaining cost and the average time to get to the potential sites of fire in the city. The facility location problem is looking for answer to such questions. This problem is well known in the operation research for decades. This problem is well known in the operation research for decades. Some early works in this topic include [6]–[8]. The facility location problem has many different application areas such as placement of warehouses, telecommunication network design or configuration of distributed servers. To find detailed survey of the problem, see [9]–[10].

The facility location problem belongs to the optimisation problems. We shall focus on the uncapacitated facility location problem (UFLP), the best-studied variant of the problem. Let us consider its specification.

The input of the problem instance consists of the elements as follows:

- The set of facility candidates (F).
- The set of demand (or client) points (D).
- The fixed cost at each facility candidate i (f_i).
- The transportation cost from the facility candidate i to the demand point j (c_{ij}).

A feasible solution contains a set of facilities selected from the candidates and an assignment of each demand point to one selected facility. The objective of the optimisation is to minimise the total cost that can be calculated as the sum of the fixed costs of the selected facility candidates plus the sum of the transportation costs to each demand point from the facility to which it is assigned.

We remark that the facility location problem has many variants. We give a list presenting some different categories of the problem.

- *Deterministic vs. stochastic*. If some parameters are given by probability distribution, the problem is stochastic, otherwise deterministic.
- *Discrete vs. continuos*, according to the type of the set of the facility candidates and the demand points.
- *Static vs. dynamic*. The problem instance is changing in time in the dynamic case.
- *Capacitated vs. uncapacitated*. In the capacitated case there is an upper bound on the number of demand points that can be assigned to a facility.
- *Multi-capacitated case*. It is an extension of the capacitated case when the different types of the facilities have different capacities.
- *Multilevel vs. one level*. In case of the multilevel facility location problem, the facilities form a hierarchy of k level. In this case the demands are satisfied through the sequence of k different facilities, each of which belongs to different level. This model is useful, for example, when central depots are opened beside warehouses.

- *Metric case.* It is an important special case when the transportation costs are induced by a metric. In this case the costs are nonnegative and satisfy the triangle inequality. Since the demand between two demand points is not defined, the triangle inequality contains the sum of three edges instead of two.

- *Splittable vs. unsplittable demands.* In assigning the demand of customers to facilities, there are two natural variations to consider: the demand of a costumer must be met by a single store (unsplittable demands), and the demand of a customer may be divided across any number of facilities (splittable demands). These variants are distinguished only in the capacitated version, because in the uncapacitated variant each demand is satisfied always by the closest facility.

- *Fault tolerant version* In this case, each demand point has to be connected to a given number of facilities. This variation ensures that if a facility breaks down than its clients still can be served with low cost by the other facilities assigned to the client.

- *Multicriteria FLP* Several objectives may exist for evaluation of different facilities. For a multicriteria optimization problem, usually there are no "optimal" solutions as in the case of single criteria problems, but only preferred solutions are available. The preferred solution must be Pareto optimal (also non-dominated solution, that is, improving any of the objectives is possible only with degrading others).

The present paper focuses on the simplest variation of the problem. However, the above variations may be relevant when we apply the facility location problem to the host recommendation.

We still have to mention here the *k-median problem* as well. This problem differs from the facility location problem that there are no costs for opening facilities, instead a number k is specified, which is an upper bound on the number of facilities that can be opened. Many cases, similar methods can be applied to it as to the FLP.

3.1.2. Complexity results. Unfortunately, this problem is NP-complete, as it is proven by Cornuejols et al. [11]. It means that there is no hope to find an algorithm, which finds the exact optimal solution within acceptable time for large problem instances.

However, some special cases are solvable in polynomial time. Kolen [12] proved that the problem is solvable on trees in polynomial time. The facility

location problem is defined on tree if the facility candidates and demand points are located in the nodes of a tree. Nonnegative weights are assigned to the edges of the tree as lengths and the transportation cost between a facility candidate and demand point is the length of the path between the corresponding nodes. Bárány el al. [13] improved the running time of the UFLP algorithm on trees.

3.1.3. Approximation results. Since there is no hope to find the exact solution in polynomial time, the significance of the approximation algorithms has increased. Many constant approximation algorithms have been published for the problem that runs in polynomial time. Unfortunately, there are negative results on the approximation algorithms for the FLP as well. Guha and Khuller [14] proved that it is unlikely to find polynomial time approximation scheme for the FLP, that is, a polynomial time algorithm that finds a solution with cost at most $1 + \varepsilon$ times the optimum for any given ε. Sviridenko [15] proved that the guaranteed approximation ratio cannot be less than 1.467 unless P = NP. However, better result can be achieved in special cases. Arora et al. [16] found randomised polynomial approximation scheme if the FLP is defined in the plane. Kolliopoulos and Rao [17] proved the same result to Euclidean spaces of constant dimension.

Hochbaum [18], Lin and Vitter [19] gave algorithms with $O(\log n)$ approximation ratio for the uncapacitated facility location problem in general case. They proposed a greedy algorithm and a new technique called the filtering technique, respectively. Many constant factor approximation algorithms were found in the special case when the transportation costs are induced by a metric. The table from Bumb's PhD thesis [10] gives a quick overview of the approximation algorithms on metric UFLP (Table 2). n shows the number of facility candidates and demand nodes. At the next section, we introduce the techniques in Table 2.

3.2. Applied methods.

3.2.1. Linear Programming. To derive an integer programming formulation of the facility location problem, two variables have to be introduced as follows:

$X_{i,j}$: (0, 1) variable to indicate whether client i is served by facility j.

Y_j: (0, 1) variable to indicate whether facility j is selected.

The other parameters come from the input of the problem, see the Overview of the FLP.

The Facility Location Problem can be now formulated in the form as follows:

Year	**Performance guarantee**	**Reference**	**Technique**	**Running time**
1997	3.16	Shmoys, Tardos and Aardal	Filtering+LP rounding	
1999	2.408	Guha and Khuller	LP rounding+greedy augmentation	
1998	1.736	Chudak	LP rounding	
1998	$5+\varepsilon$	Korupolu et al.	Local search	$O(n^6 \log(n/\varepsilon))$
2001	3	Jain and Vazirani	Primal-dual method	$O(n^2 \log n)$
2001	3	Arya et al.	Local search	
2000	3	Mettu and Plaxton	Combinatorial	$O(n^2)$
1999	1.853	Charikar and Guha	Primal-dual method+greedy augmentation	$O(n^3)$
1999	1.728	Charikar and Guha	LP rounding + primal-dual method + cost scaling + greedy augmentation	
2001	1.86	Mahdian et al	dual fitting	$O(n^2 \log n)$
2001	1.61	Jain, Mahdian and Saberi	dual fitting	$O(n^3)$
2002	1.582	Sviridenko	LP rounding	
2002	1.52	Mahdian, Ye and Zhang	dual fitting+greedy augmentation	$O(n^3)$

Table 2 Approximation results for the uncapacitated facility location problem [10]

minimize

$$\sum_{i \in D, j \in F} c_{i,j} \cdot X_{i,j} + \sum_{j \in F} f_j \cdot Y_j \tag{1}$$

subject to

$$\sum_{j \in F} X_{i,j} \geq 1, \ \forall i \in D \tag{2}$$

$$X_{i,j} \leq Y_j, \ \forall i \in D, \forall j \in F \tag{3}$$

$$X_{i,j} \in \{0,1\},\ \forall i \in D, \forall j \in F \tag{4}$$

$$Y_j \in \{0,1\},\ \forall j \in F \tag{5}$$

Part (1) is the cost function. Constraint (2) ensures that each client is assigned to a facility. Constraint (3) describes that a facility is selected if there is at least one client assigned to it.

However, the integer program is usually hard to solve. For this reason, the LP-relaxation of the problem is solved. In this case, the (0,1) constraints have to omitted:

$$X_{i,j} \geq 0,\ \forall i \in D, \forall j \in F \tag{4'}$$

$$Y_j \geq 0,\ \forall j \in F \tag{5'}$$

3.2.1.1. Linear Programming Rounding. The linear programming relaxation can be solved using efficient algorithms. If each variable is integer then we found the optimum solution for the integer program. However, this is usually not the case. Different rounding techniques can be applied to get integer solution from the fractional one. Shmoys Tardos and Aardal [20] gave the first constant factor approximation algorithm. They achieved their result applying the LP rounding technique and the filtering technique.

3.2.1.2. Filtering technique. This method consists in constructing a "filtered problem" by setting some variables to zero in the integer programming formulation. Then the feasible integer solution to the filtered program can be found either by LP rounding technique or by using some combinatorial algorithm [19].

3.2.1.3. Primal-dual method, Dual fitting. Jain and Vazarini [21] applied this method first to the facility location problem. Its running time is quite low $O(m \log m)$, where m is the number of edges. This method is also based on the linear programming. But it uses the dual pair of the primal linear program as well presented above. The dual pair of the above linear program is as follows:
maximise

$$\sum_{i \in D} v_i \tag{6}$$

subject to

$$\sum_{i \in D} t_{i,j} \leq f_j, \ \forall j \in F \tag{7}$$

$$v_i - t_{i,j} \leq c_{i,j}, \ \forall i \in D, \forall j \in F \tag{8}$$

$$t_{i,j} \geq 0, \ \forall i \in D, \forall j \in F \tag{9}$$

$$v_i \geq 0, \ \forall i \in D \tag{10}$$

In case of dual program, we introduce new variables, $t_{i,j}$ and v_i and we have to maximise the profit function. The special feature of the dual program that for any feasible solution the profit in the dual program is always smaller or equal than the cost in the primal program. Primal-dual methods can achieve good approximation by considering both the primal and the dual problems at the same time. Dual fitting methods also use the primal-dual technique. Mahdian et al. [22] achieved the best approximation ratio (1.52) so far for FLP using dual fitting.

Combinatorial algorithms.

3.2.2.1. Local search or greedy. The local search heuristic for facility location problems is extremely straightforward. The idea is to start with any feasible solution and then to iteratively improve the solution by repeatedly moving to the best "neighbouring" feasible solution, where one solution is a neighbour of another if it can be obtained by either adding a facility, deleting a facility, or changing the location of a facility. This heuristic was proposed by Kuehn and Hamburger [7] and was subsequently shown to exhibit good practical performance in empirical studies (see e.g., [23]).

3.2.2.2. Greedy strategies. This kind of methods has many variants and it is usually combined with other methods. Mettu and Plaxton [24] developed the first linear time algorithm. The construction applied in the algorithm is similar to the one in the primal-dual method used by Jain and Vazirani [21]. They use a "hierarchically greedy" approach whose basic idea is as follows: Rather than selecting a point based on a single greedy criterion, a region is chosen greedily and then a point is selected recursively within that region. Thus, the choice of point is influenced by a sequence of greedy criteria addressing successively finer levels of granularity.

3.2.2.3. Greedy Augmentation (greedy improvement). If either the service cost or the facility cost is very high, greedy local improvement decreases the total cost by balancing the two. It always selects the facility in each step where the ratio of the decrease in the total cost and the facility cost is the highest. The greedy local improvement by itself yields a very good approximation for facility location in $O(n^2)$ time. This method is applied in many algorithms: [22], [25]–[26].

3.2.2.4. Cost scaling. Charikar and Guha [26] applied this method combined with greedy local improvement and the primal dual method. The idea is to apply the algorithm to the scaled instance and than scale back to get a solution for the original instance. On several occasions, this technique alone improves the approximation ratios significantly.

3.2.2.5. Simulated annealing. Simulated annealing is a generic global optimization method. Its name comes from metallurgy. In that case, the hot material is slowly cooled in order to improve a better quality.

In case of global optimisation, the algorithm proceeds in the search space. The subsequent points are not always better in each step: the algorithm may select with nonzero probability even worse solutions. This probability decreases monotonically while the algorithm is running. The initially high probability ensures that high areas are discovered. Later the small probability ensures that the solution remains near to the optimum.

3.3. Network flow. Let us imagine a water plumbing where the individual pipes may have different widths and so different maximum flow per unit time. There is an inlet and an outlet as well and the question is how much is the maximum rate at which water can flow from the inlet to the outlet.

This is a network flow problem which can be specified more formally: Given a graph $G(V, E)$ with nodes V and edges E. $c(e) \geq 0$ denotes the capacity (maximum flow possible) of edge $e \in E$. Source $s \in V$ and sink $t \in V$ denotes special nodes of the graph G. G, s, t and $c()$ specifies a network. Furthermore, there can be specified a flow $f(e)$ on the edges of the network which is at most the capacity of the edge $c(e) \geq f(e)$ for each $e \in E$ and the total the inflow to a node is equal to the total outflow from the node except the two special nodes. The value of the flow is the outflow from the source or the inflow to the sink which is the same.

The network flow problem can be applied even to find the computer networks where the capacities are the available bandwidths in the network, the

sink is the client and the source is data supplier node and more than one path may be used to transmit the data to the client at the same time.

This problem can be solved efficiently in polynomial time. It remains polynomial if a cost is assigned to each edge and we are looking for a solution with maximal flow and minimal cost. However, the multi-commodity flow problem is already NP-complete, where multiple source and sink pairs are given, and various "commodities" can flow from a given source to a given sink.

3.4. Knapsack Problem. We found the knapsack problem useful to handle with the local resource capacities. The original knapsack problem is as follows: Given a knapsack with maximum volume and several items with different volumes. The optimisation goal is to put as many items as possible into the knapsack. This problem seems to be simple but it is NP-hard, that is, computationally difficult.

Khan [27] applied its variant, namely the multiple-choice multi-dimension knapsack problem (MMKP) for adaptive multimedia problem. The problem can be described as follows. In this case there are some groups with different numbers of items. Each item has a value and resource need. The resource need can be described as a vector because an item need several resources. Furthermore, we know the amounts of available resources. The multi-dimension knapsack problem is to pick exactly one item from each group while maximising the total value of the selected items, subject to the resource constraints.

This model is useful when the overall quality of the delivered multimedia has to be maximised. In this case, the acceptable variations of the requested video can be viewed as a group of items for each client. Each item can be characterised by the resource needs for delivery to the client and by their qualities.

4. Implementations. This section gives a brief overview of the algorithms we selected for host recommendation. We mentioned that the host recommendation has different variants and for example. The algorithms operate on objects which hide the differences between the LWR and HWR in order to avoid the duplication of the algorithms. Due to the underlying common model, the algorithms are able to find solutions for each type of the host recommendation problem but they are not equally appropriate for different cases. For example, facilities are assigned to the client requests to serve the video streams; they are the pairs of streamer and hosting nodes in case of LWR while the streamer node can be neglected at HWR.

4.1. Linear Programming Rounding. We chose an algorithm based on linear programming rounding for the solution of the host recommendation problem. Our linear program is much more complicated than in case of facility location problem. To derive the integer programming formulation of the problem, we had to introduce variables $X_{i,j,k,l}$ to indicate whether the video variation l is served to the client i by DM group k through DC j. Furthermore, S_i indicates whether any request of client i is served or not. Similarly to the facilities, $Y_{j,k}$ indicates whether the connection between DM group k and DC j is used for the service. Weights are assigned to the different optimization criteria. They express their priorities. Each weight is higher than the maximum of the subsequent optimization criteria. We ensure in this way that the subsequent criteria are concerned only if the criteria with higher weights are equal. The exact LP formulation can be found in technical report version of [28]. We had to add constraints to express the network and node resource constraints.

The possible values of the variables are 0 and 1. Since the time complexity to find the exact solution for an integer linear programming problem is large, we consider the LP-relaxation of the problem, where the possible values of the variables can be any positive real number. To solve the linear program, we applied SOPLEX, an object oriented implementation of the simplex algorithm [29]. This software is integrated into the host recommendation algorithm.

If $X_{i,j,k,l} = 1$ in the solution then let the video variation l of client i be served by server k through proxy j. The possible fractional values of X-type variables do not represent legal solutions. We round them in a greedy manner. We take each Xvariables with fractional value one after the other and we try to select the current client-DC-DM group route denoted by the variable. We have to check the resource constraints. The route is selected for video delivery if and only if these conditions are fulfilled after the selection of the route.

4.2. Incremental Algorithm. The incremental algorithm takes the facilities one after the other and the facility is selected if it improves the solution. It has been originally developed for LWR. However, some goals for HWR, such as minimising the number of storage nodes can be easily incorporated. The incremental algorithm is a very simple but efficient method. For large problem inputs a time limit can be set to abort running the algorithm when the running time would be too long, and then apply the current recommendation for configuration (this is, why the algorithm is called incremental).

Some optimization criteria such the number of storage nodes can be easily incorporated but in its original form, the incremental algorithm is unsuitable to minimize the network load arising from migration/replication. This is the

reason while we introduce its extension, namely Complex Incremental Algorithm, which works on three levels: it tries to add facilities one after the other to the recommendation on the highest level, it examines the possible video instances on the current facility on the next level and at last it tries to assign the client requests to the video instances. For more details see [30].

The scheme of the incremental algorithm:

for each facility **do**
    Decide on selecting the current facility
    if the facility is selected **then**
        for each client connected to the current facility **do**
            Decide on assigning the current client to the facility

4.3. Greedy. In this algorithm we start with an empty set of proxies. In each turn that proxy is added that decreases the cost the most. When calculating the cost, in each turn a new configuration has to be generated based on the previous one and the recently extended proxy set. This configuration setup is made by an inner greedy algorithm that for every client selects the best proxy from the set, and tries to modify the index of the chosen demand by adding or subtracting one. Each proxy contacts the server that gives the least cost [31].

4.4. Particle Swarm. The particle swarm algorithm is based on the algorithm of Kennedy and Eberhardt [32]. Their original variant uses a set of particles, which particles describe a concrete configuration each. The particles are connected to each other thus forming a given topology.

The particles (i.e. the configurations they describe) are initialized with random values. The particles remember their own best configuration (b_p), and they know which neighbour of theirs (including themselves) gives the least costy configuration (b_n). The problem solving is divided into rounds. In each round the particles calculate their new configuration by combining b_p and b_n, during which they utilize stochastic values also. The algorithm continues until some predefined condition is reached.

The definition of the original combination looks thus:

$$\begin{cases} \vec{v}_i(t) = \vec{v}_i(t-1) + \varphi_1(\vec{p}_i - \vec{x}_i(t-1)) + \varphi_2(\vec{p}_g - \vec{x}_i(t-1)) \\ \vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \end{cases}$$

where φ_1, φ_2 are random numbers from interval $[0, 1)$, $\vec{v}_i(t)$ is the 'velocity' (amount of change) at time t, $\vec{x}_i(t)$ the value of the vector at time t, $\vec{p}_i$ the least costy vector of particle i, $\vec{p}_g$ is the vector of the actually least costy neighbour.

When the algorithm reaches the predefined condition, it ends. The result is the vector of the least costy particle.

In our current case (ADMS) the elements of vector $\vec{x}$ are the following:

$$\vec{x} = \begin{bmatrix} \vec{\kappa} \\ \vec{\lambda} \\ \vec{\mu} \end{bmatrix} \quad \begin{array}{ll} \vec{\kappa} \in C, & \text{client } i \text{ connects to DC } \kappa_i \\ \vec{\lambda} \in P, & \text{DC } i \text{ connects to DM } \lambda_i \\ \vec{\mu} \in Q_i, & \text{client } i \text{ gets demand } \mu_i \end{array}$$

The algorithm, in its original form cannot be applied to problems similar to the one presented in this paper, when the elements of the representation get their values from non-orderable sets, because in these cases the addition in the above formula makes no sense. (E.g. two DCs can not be sensibly added.)

As a solution, we proposed and applied the following vector-modification:

$$\begin{cases} m_l(t) = \dfrac{h(\vec{x}(t-1), \vec{p})}{L} m_g \\ m_a(t) = (1-\alpha) m_a(t-1) + \alpha \cdot m_l(t) \\ \vec{x}_i(t) = \begin{cases} \left\{ \begin{array}{lll} \vec{p}_i & \text{if} & \varphi_1 \le m_c \\ \vec{x}_i(t-1) & \text{if} & \varphi_1 > m_c \end{array} \right\} & \text{if } \varphi_2 \le m_a(t) \text{ (crossover)} \\ e \in S_i & \text{if } \varphi_2 > m_a(t) \text{ (mutation)} \end{cases} \end{cases}$$

Where p is the vector of the least costy neighbour, $h(\vec{a}, \vec{b})$ is the Hamming distance of the two vectors, L is the length of the vectors, m_g is the global muation rate, m_l is the local mutation rate, m_a is the average mutation rate, α is a scaling factor, m_c is the crossover-rate, S is the possible value set of the given vector element.

The meaning of the formula is that the bigger the difference between the vector of a particle and that of its least costy neighbour, the bigger the mutation probability of the vector elements (taking into account the history of the probability). For each element, that did not mutate, a new probability is counted for getting the respective value from the neighbour. The Hamming distance is used because without that at the end of the calculation a fluctuation can be observed, which prolongs the fulfilment of the stop condition.

The algorithm runs until the cost deviation of the particles is within a given bound.

4.5. The simple algorithm. It assigns each current request one after the other to the facility that can serve the best video instance to the client and if there are more such facilities it selects streaming route with the best QoS

parameters. This method is especially appropriate in LWR for deciding from where to serve a client when the number of clients is small. It is not adequate to handle the facility cost. For an example, see [3].

The scheme of the algorithm:

for each *client* **do**
  best_facility ← 0
  for each *facility* **do**
    if better *facility* for the client than the *best_facility* **then**
      best_facility ← *facility*
  Assign *best_facility* to the client

4.6. Best gain per resource usage first. It is based on the multiple-choice multi-dimension knapsack problem [27]. In each step, the algorithm always selects the possible video instance with highest value per unit resource consumption where the value of a video instance is the total value of the client requests servable from it. This algorithm is developed for HWR. However, it can be adapted to LWR as well.

The scheme of the algorithm:

best_gain ← *1*
while *best_gain* > *0* **do**
  for each $v \in$*possible video instances* **do**

$$ff_v \leftarrow \max_j \left(\frac{needed\_resource_j(v)}{free\_resource_j(v)} \right)$$

$$\textbf{if} \quad ff_v > 1 \textbf{ then } gain_v = \frac{value(v)}{ff_v}$$

    else $gain_v \leftarrow 0$
  $best\_gain \leftarrow \max_v(gain_v)$
  if *best_gain* > *0* **then**
    Select v for usage as video instance

The video instances can be characterised by the variation of the video and the storage node where it is located. ff_i is called feasibility factor which is at most 1 if there are enough resources. The value of an instance can be for example the expected number of clients that can be served from this instance. The *gain* is the ratio of the value and the feasibility factor. For an example, see [3].

5. Conclusions. The area of optimisation algorithms is very rich in different methods and it offers plenty of approaches to solve the host recommendation problem. This problem is much more complex than the well-known optimisation problems. For this reason, the known problems should be extended and different approaches should be combined. In many cases, the running time is more critical than the good approximation and fast heuristic methods have great significance. In an earlier work [28], we conducted tests to compare four different algorithms (greedy, particle swarm, linear programming rounding and incremental algorithm). The particle swarm algorithm produced the best result while the incremental algorithm found the solution in the shortest time.

REFERENCES

[1] Tusch R. Design and Implementation of an Adaptive Distributed Multimedia Streaming Server. PhD Thesis, University Klagenfurt, 2004.

[2] Tusch R., L. Böszörmenyi, B. Goldschmidt, H. Hellwagner, P. Schojer. Offensive and Defensive Adaptation in Distributed Multimedia Systems. *Computer Science and Information Systems (ComSIS),* **1**, No. 1 (Feb. 2004), 49–77.

[3] Kárpáti P. , T. Szkaliczki, L. Böszörmenyi. Abstracting and characterizing distributed VoD servers.In: Proceedings of 3^{rd} Hubuska Open Workshop, Klagenfurt, 27–28 April, 2006.

[4] Hutter O, T. Szkaliczki, B. Goldschmidt. Host Recommendation in the Adaptive Distributed Multimedia Server. *ERCIM News*, **62** (July 2005), 34-35.

[5] Kárpáti P., T. Szkaliczki, L. Böszörményi. Mathematical model for distributed video-on-demand servers. Technical Reports of the Institute of Information Technology, University Klagenfurt, TR/ITEC/05/2.13.

[6] Manne A. Plant location under economies-of-scale-decentralization and computation. *Management Sci.* **11** (1964), 213–235.

[7] Kuehn A., M. Hamburger. A heuristic program for locating warehouses. *Management science* **9** (1963), 643–666.

[8] Stollsteimer J. The effect of technical change and output expansion on the optimum number, size and location of pear marketing facilities in a California pear producing region. PhD Thesis, University of California at Berkeley, California, 1961.

[9] Mirchandani P., R. Francis eds. Discrete Location Theory. John Wiley and Sons, Inc., New York, 1990.

[10] Bumb A. Approximation Algorithms for Facility Location Problems. PhD Thesis, Twente University Press, 2002.

[11] Cornuejols G., G. Nemhauser, L. Wolsey. The uncapacitated facility location problem. In: Discrete Location Theory.(Eds P. Mirchandani, R. Francis), John Wiley and Sons, New York, 1990, 119–171.

[12] Kolen A. Solving covering problems and the uncapacitated plant location problem on trees. *European Journal of Operations Research* **12** (1983), 266–278

[13] Bárány I., J. Edmonds, L. Wolsey. Packing and covering a tree by subtrees. *Combinatorica* **6** (1986), 245–257.

[14] Guha S., S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms* **31** No 1 (1999), 228–248.

[15] Sviridenko M. Personal communication. Cited in S. Guha, Approximation algorithms for facility location problems, PhD Thesis, Stanford, 2000.

[16] Arora S., P. Raghavan, S. Rao. Approximation schemes for Euclidean k-medians and related problems. In: Proceedings of the 30th Annual ACM Symposium on Theory of Computing, 1998, 106-113.

[17] Kolliopoulos S., S. Rao. A nearly linear-time approximation scheme for the Euclidean k-median problem, In: Proceedings of the 7th Annual European Symposium on Algorithms, Lecture Notes in Computer Science, Vol. **1643**, Springer-Verlag, 1999, 378–389.

[18] Hochbaum D. Heuristics for the fixed cost median problem. *Mathematical Programming* **22** (1982), 148–162.

[19] Lin J., J. Vitter. ε-approximation with minimum packing constraint violation. In: Proceedings of the 24th Annual ACM Symposium on Theory of Computing, 1992, 771–782.

[20] Shmoys D., E. Tardos, K. Aardal. Approximation algorithms for facility location problems In: Proceedings of the 29th Annual ACM Symposium on Theory of Computing, Springer, 1997, 265–274.

[21] Jain K., V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM* **48** (2001), 274–296.

[22] Mahdian M., Y. Ye, J. Zhang. Improved approximation algorithms for metric facility location problems. Lecture Notes in Computer Science Vol. **2462**, Springer, 2002 , 229–242.

[23] Hidaka K., H. Okano. An Approximation Algorithm for a Large-Scale Facility Location Problem. *Algorithmica* **35** (2003), 216–224.

[24] Mettu R., C. Plaxton. The online median problem. In: Proceedings of the 41st IEEE Symposium on Foundation of Computer Science, 2000, 339-348.

[25] Guha S., S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms* **31**, No 1, (1999), 228–248.

[26] Charikar M., S. Guha. Improved Combinatorial Algorithms for the Facility Location and k-median Problems. IEEE Symposium on Foundations of Computer Science, 1999, 378-388.

[27] Khan S., K. Li, E. Manning, M. Akbar. Solving the knapsack problem for adaptive multimedia. *Studia Informatica* (special issue on Cutting, Packing and Knapsacking Problems), **2**, No 1 (2002).

[28] Goldschmidt B., T. Szkaliczki, L. Böszörményi. Placement of Nodes in an Adaptive Distributed Multimedia Server. In: Proceedings of the 10th International Euro-Par Conference, 2004, 776–783, Extended version: Technical Reports of the Institute of Information Technology, University Klagenfurt, TR/ITEC/04/2.06.

[29] Wunderling R. Paralleler und Objekt-orientierter Simplex- Algorithmus. ZIB technical report TR 96-09, Berlin, 1996.

[30] Szkaliczki T., L. Böszörményi. Incremental Placement of Nodes in a Large-Scale Adaptive Distributed Multimedia Server. In: Proceedings of International Conference on Distributed and Parallel Systems (DAPSYS

04), September 19–22, 2004, Budapest, Hungary, Series: The Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers, September 2004, Vol. **777**.

[31] Goldschmidt B., Z. László. A Proxy Placement Algorithm for the Adaptive Multimedia Server. In: Proceedings of 9th International Euro-Par Conference, 2003, 1199–1206.

[32] Kennedy J., R. Eberhardt. Swarm Intelligence. Morgan Kaufmann, 2001.

Tibor Szkaliczki
Computer and Automation Research Institute
Hungarian Academy of Sciences
Lágymányosi u. 11.
Budapest H-1111 Hungary
e-mail: `sztibor@sztaki.hu`

Balázs Goldschmidt
Budapest University of Technology and Economics
Magyar tudósok krt. 2
Budapest H-1117 Hungary
e-mail: `balage@iit.bme.hu`

Laszlo Böszörmenyi
Klagenfurt University
Universitätsstraße 65–67
9020 Klagenfurt, Austria, E.U.
e-mail: `laszlo@itec.uni-klu.ac.at`

Received May 25, 2007
Final Accepted September 19, 2007