Paper

An exact algorithm for design of content delivery networks in MPLS environment

Krzysztof Walkowiak

Abstract—Content delivery network (CDN) is an efficient and inexpensive method to improve Internet service quality. In this paper we formulate an optimisation problem of replica location in a CDN using MPLS techniques. A novelty, comparing to previous work on this subject, is modelling the network flow as connection-oriented and introduction of capacity constraint on network links to the problem. Since the considered optimisation problem is NP-complete, we propose and discuss exact algorithm based on the branch-and-cut and branch-and-bound methods. We present results of numerical experiments showing comparison of branch-and-cut and branch-and-bound methods.

Keywords—content delivery network, optimization, branch-and-cut algorithm.

1. Introduction

In recent years we observe a tremendous increase in data traffic, caused mainly by the growth of the Internet as well as introduction of many new services. Concurrently, corporate and individual users demand more bandwidth and more functions with quality of service (QoS) guarantees. The existing Internet sometimes cannot cope with all challenges that are to be addressed by computer networks in near future. Therefore, new solutions are being developed to overcome most of problems now being encountered by major players of the telecommunication world. Operators focus on new ideas and concepts to enable radical transformation of networks and service infrastructures. In order to achieve a success, the service provider should: develop an efficient transport network; offer and constantly change a huge number of value-added, improved services; construct business plan to make profits delivering those services.

Content delivery network (CDN) is an interesting and robust method to improve the Internet quality. CDN uses many servers offering the same content replicated in various locations. User-perceived latency and other quality of service parameters can be easily and inexpensively improved by various techniques of Web content caching. Every replicated system must deal with two fundamental issues—distributing requests to object replicas and deciding on placement of replicas. In this work we focus on the second problem. The issue of distributing requests to object replicas is strongly discussed in the literature.

In this work we address problems of CDN design in multiprotocol label switching (MPLS) environment. The MPLS approach proposed by the Internet engineering task force (IETF) is a networking technology that enables traffic engineering and QoS performance for carrier networks. MPLS is a connection-oriented technique, which is becoming a popular solution for backbone networks and must be taken into account in the design of Web replication system. Since the considered optimization problem is NP-complete, we propose an exact algorithm using the branch-and-cut approach. It must be noted that results of this work can be applied also to networks using other connection-oriented technologies (e.g., asynchronous transfer mode—ATM) or connectionless protocols (e.g., IP).

The paper is organized as follows. Section 2 presents a brief description of CDNs and Web server caching issues. In Section 3 we report on the previous work in the field of replica placement problems. In Section 4 we formulate an optimization problem of replica location in a CDN using MPLS. Section 5 contains an exact algorithm solving the replica location problem. In Section 6 we present and discuss results of numerical experiments. Last section concludes this work.

2. Content delivery networks and Web caching

Content delivery networks are defined as mechanisms to deliver a range of content to end users on behalf of origin Web servers. The original information is offloaded from source sites to other content servers located in different locations in the network. For each request, the CDN tries to find the closest server offering the requested Web page [16]. CDNs deliver the content from the origin server to replicas located much closer to end-users. The set of content stored in CDNs servers is selected carefully. Therefore, the CDNs' servers can approach the hit ratio of 100%. It means that almost all requests to replicated servers are satisfied. CDNs techniques are based on caching and replication of Web content. The general architecture of CDN system can be found in [23].

Caching is a technique typically applied to bring parts of an overall data set closer to its processing site [3]. A Web cache is an application residing between Web servers providing various content and clients that want to fetch the information [27]. Caching employs the knowledge acquired by several analyses on servers' access logs and by looking

into Web users behavior. Caching can reduce latency experienced by end users when trying to fetch some documents through their Web browser.

Replication can be considered as a kind of caching. Nevertheless, there is some dissimilarity. Replication presumes storing of an object at a place that cannot see the object, while caching is storing of an object at a place that sees the source object. It means that a cache notices both hit and miss requests. Since requests to replicated server arrives only if that server is believed to have a replica of the requested object, the replica notices only hits. In the presented sense, replica is sometimes called push cache [25]. Replication is perceived also as a caching system with only one source Web server generating content, while standard caching must serve a great number of Web servers [17].

An important issue to resolve is the choice between static and dynamic replica placement. In the static replica placement the system administrator, according to observed access and traffic statistics, decides where replicas should be located. Dynamic replica placement assumes that the system monitors access to various servers and adapts set of replicas to changing requirements [25].

One of the most important issues of Web caching is the mechanism used for requests redirection. Transparent replication assumes redirecting a client's request for a document to one of the physical replicas. The most popular practical and theoretical approaches of requests redirection: client multiplexing, IP multiplexing, DNS indirection, HTTP redirection and anycast, peer-to-peer routing have been discussed in [23, 25, 28].

Web caching and replication in CDNs are becoming popular for many reasons. The most important are [6, 28]: reducing the cost of using the Internet, reducing the latency of WWW, bandwidth will always have some cost, non-uniform bandwidth and latencies, network distances grow, bandwidth requirements continue to increase, hot spots in the Web will continue, costs of communication exceed costs of computations, traffic engineering requirements, the need for survivability.

For more information on WWW please refer to [34, 35].

3. Related work

An important issue in the design of robust and survivable CDN is the replica placement. In this section we examine the previous work on replica placement problems. For the context of this paper we are interested in static replica placement. The main problem of static replica placement is to develop effective algorithms for replica location. Some previous authors have developed such algorithms. According to [24], the first work in this area is [19]. Li *et al.* formulate in [19] a problem of proxies' location in a tree topology with the objective function of selection of proxies cost. A dynamic programming algorithm is proposed. The objective function can be calculated as the overall network latency if the link distance is associated with the cost function.

Authors of [17] take into account the cache location problem for transparent caches. The objective function is the cost of serving demands using a cache in a given location. Since the general problem is NP-complete, Krishnan *et al.* analyze only regular topologies: homogenous line, general line and ring.

Qiu *et al.* formulate in [24] problem of the placement of web server replicas as an uncapacitated k-median problem related to the facility location problem. They restrict the maximum number of replicas, but they don't restrict the number of requests served by each replica. The goal of the optimization process is to minimize the total cost of all requests defined as a sum of a distance between origin node and destination node over all requests. A greedy algorithm and a super-optimal algorithm based on the Lagrangian relaxation are proposed.

Guha *et al.* consider in [8] a generalization of the standard facility problem and introduce the requirement for fault-tolerant mechanisms. Every demand point is served by a number of facilities instead of just one. The closest facility is the working one, while other facilities serve as backup facilities. The objective function is a weighted combination of facilities locations' costs. An algorithm using the filtering technique and fractional demands is provided.

Authors of [10] present a simple and natural greedy algorithm for the metric uncapacitated facility location problem and k-median problem.

Arya *et al.* analyze in [2] a local search heuristics for facility location and k-median problems. The main operation of the proposed algorithm is swap, which includes closing one facility and opening another; clients of the closed facility are assigned to other facilities. In [4] an improved combinatorial approximation algorithms for the uncapacitated facility location and k-median problems are proposed and discussed.

The replica placement problem can be modeled as a center placement problem. The k-HST (k-hierarchically well separated tree) approach can solve this problem [11, 23].

Jamin *et al.* propose a topology-informed placement strategy, called "transit node". This heuristic applies the outdegree—information on the number of other nodes connected to a given node. It is assumed that a node with the highest outdegrees can reach more nodes with lower latency. Therefore, the servers are placed in nodes sorted in descending order of outdegrees [12, 23].

Wierzbicki formulates in [36] the Internet cache location problem in a CDN as a mixed integer programming (MILP). New models of cache location are proposed in order to overcome the limitations of the basic model. The complexity of the MILP formulation is evaluated.

The primary concern in most of works discussed above is analyzing the replica location problem as one of well-known optimization problems: k-facility location problem, k-median problem and center placement problem. The first problem consists of assignment of clients to k facilities that can be located in network nodes. The objective is to minimize the total cost including the connection cost of each

client and the facility cost. The k-median problem generally differs from the facility problem in one thing: there is no cost for opening facilities. The main element of both discussed problems is location of k facilities, i.e., selection of k nodes of the network for hosting a facility. Since one can select the closest replica in terms of connection cost, assignment of individual clients to a particular replica is much simpler. Capacity constraints on network links are not considered. However, in a capacitated version of facility location problem there is a capacity constraint on load served by each facility. The center placement problem consists of the placement of a given number of centers in order to minimize the maximum distance between a node and the nearest center.

4. Optimization problem of CDN design in MPLS environment

We propose a different approach then in previous works. Our model is much closer to problems encountered in real computer networks. The main difference is that we take into account capacity constraints on each link of the network. In many cases networks are congested. Therefore, the capacity resources must be used in effective manner. Furthermore, we consider an MPLS network that is a connection-oriented network, i.e., the flow is modeled as a non-bifurcated multicommodity flow. Most of the work in the field of replica placement considers pure IP networks using multicommodity bifurcated flow.

In this section we formulate the optimization problem of the content delivery network design using the MPLS technique. The problem is very close to the replica location (RL) problem discussed in [30–31]. We model the MPLS network flow as non-bifurcated multicommodity flow. However, results of this work can be also applied to connection-less networks. For more information on modeling of flow in MPLS network and non-bifurcated multicommodity flows, see [7, 14, 15, 18, 26, 29].

We begin presentation of the problem by introducing the notation. We will keep the same notation for the rest of the paper.

Indices:

- i used as subscript, denotes the number of considered client of CDN,
- j used as subscript, denotes the number of considered arc or node,
- r used as subscript, denotes the number of considered selection of clients or routes,
- k used as superscript, denotes the number of a route.

Sets:

- V set of $|V|$ vertices representing the network vertices (nodes),
- A set of $|A|$ arcs representing directed links,
- R set of $|R|$ CDN's content servers (replicas); each server must be located in a network vertex,
- P set of $|P|$ CDN's clients; each client is defined by the source vertex s_i, destination vertex t_i and bandwidth requirement Q_i; for each client a set of route proposals is given,
- Π_i set of routes proposals for a client i; $\Pi_i = \{\pi_i^k : k = = 1, \ldots, |\Pi_i|\}$; each route ends in the source node of client i,
- Z_r set of location variables z_i equal to one; the set Z_r is called a selection; each selection Z_r determines the unique assignment of replicas to network nodes,
- X_r set of route selection variables x_i^k equal to one; the set X_r is called a selection; each selection X_r determines the unique set of routes between clients and replicas.

Decision variables:

- z_i binary variable, which is equal to one if a replica is located in the node i and is equal to zero otherwise,
- x_i^k binary variable, which is equal to one if the client i uses the route π_i^k and is otherwise equal to zero.

Other variables:

- f_{jr} flow in link j calculated according to routes defined in selection X_r.

Constants:

- c_j capacity of arc j,
- $C(j)$ capacity of all arcs leaving the node j,
- Q_i bandwidth requirement for a client i,
- $Q(j)$ bandwidth requirement of all clients located at node j,
- a_{ij}^k binary variable, which is equal to one if the jth arc belongs the route π_i^k and is otherwise equal to zero,
- u_{ij} binary variable that equals one if the source node of the arc i is node j,
- u_{ij}^k binary variable that equals one if the source node of the route π_i^k is node j.

We assume that traffic between a replica and a set of clients connected to one node can be aggregated to one or more LSPs. Since clients receive more data than is sent to replicas, we assume that traffic between clients and replicas is generally asymmetric and we ignore the flow from a client to a replica.

The optimization problem of replica location in a CDN is formulated as follows:

$$\min_{X_r, Z_r} D(X_r, Z_r) = \sum_{j \in A} f_{jr} \tag{1}$$

subject to

$$f_{jr} = \sum_{i \in P} \sum_{\pi_i^k \in \Pi_i} a_{ij}^k x_i^k Q_i \quad \forall j \in A, \tag{2}$$

$$\sum_{j \in V} z_j = |R|, \tag{3}$$

$$\sum_{\pi_i^k \in \Pi_i} x_i^k = 1 \quad \forall i \in P, \tag{4}$$

$$f_{jr} \leq c_j \quad \forall j \in A, \tag{5}$$

$$\sum_{j\in V}\sum_{\pi_i^k\in\Pi_i} x_i^k y_j u_{ij}^k = 1 \quad \forall i \in P, \tag{6}$$

$$z_j \in \{0,1\} \quad \forall j \in V, \tag{7}$$

$$x_i^k \in \{0,1\} \quad \forall i \in P;\ \pi_i^k \in \Pi_i. \tag{8}$$

The objective function (1) is the overall flow in the CDN generated by clients. Note that if we introduce a link metric the objective function could represent cost, network latency or other function. Equation (2) is a definition of a link flow. Constraint (3) guarantees that the number of established replicas (content servers) equals the defined number of replicas. We assume that during the CDN design we know how many replicas may be located. The number of replicas can be calculated according to the budget of CDN. The overall budget is divided by the cost of one content server. Thus, we obtain the number of replicas that can be afforded for the particular budget. Constraint (4) ensures that each client uses only one route. Constraint (5) is a capacity constraint. Constraint (6) guarantees that each selected route starts in a node that has a replica. Constraints (7) and (8) ensure that decision variables are binary ones. The condition (8) ensures that the considered flow is non-bifurcated as in MPLS networks. If we relax the constraint (8) to the formula given below, the flow becomes a bifurcated multicommodity flow:

$$0 \le x_i^k \le 1 \quad \forall i \in P;\ \pi_i^k \in \Pi_i.$$

Thus, we obtain the replica location problem for protocols using the bifurcated flow, for instance IP protocol.

Note, that in the problem Eqs. (1)–(8) we don't limit the amount of service that can be provided at any replica. According to [24], it is a reasonable assumption, since increasing the number of replica sites is much more difficult than increasing the capacity of a replica. The number of replicas is frequently given a priori due to cost and administrative reasons, while the capacity constraint can be overcome by adding more machines. Since in many cases the replication traffic and cost of replicas managing can be ignored, we ignore the cost of replica location.

The problem Eqs. (1)–(8) is NP-complete because it has more constraints then the non-bifurcated flow problem which is NP-complete according to [13].

Joint optimization of replica location, clients' assignment and routes' selection must be carried out to find a globally optimal solution of the objective function for a projected traffic demand. Since the optimization is conducted jointly over location and route selection variables, the complexity of the problem grows tremendously. An interesting approach is to partition the problem into two simpler problems: first optimize replica location and next find clients' assignment for already established replicas.

The first subproblem, called only replica location (ORL) consists of selection of $|R|$ nodes to host a replica. This problem is very close to problem RL (1)–(8). However, we don't take into account assignment of clients to replicas. Therefore, we can ignore constraints (2), (4)–(6) and (8). As an objective function we use the function $D(Z_r)$ defined as a solution of clients' assignment to replicas given by the selection Z_r.

The second subproblem is to assign each client i to one replica according to selected criterion. In the optimization problem of clients' assignment to replicas (CATR) we assume that replicas are already located in network nodes and the main goal is to assign clients to replicas minimizing the overall flow. The CATR optimization problem is formulated as follows:

$$\min_{X_r} D(X_r) = \sum_{j\in A} f_{jr} \tag{9}$$

subject to Eqs. (2), (4), (5) and (8).

The CATR problem is similar to the classical non-bifurcated multicommodity flow problem (NBMC) extensively discussed in the literature [7, 14, 33]. The main difference is that in the CATR problem besides route selection for each client we must decide on which replica the client should be assigned to. It is an additional constraint. Since the NBMC problem is NP-complete [13], the CATR problem is also NP-complete.

To solve the ORL problem we must consider many CATR subproblems. For each location of replicas, in order to find the objective function, we must estimate the network flow by assigning clients to already located replicas. For this purpose exact or heuristic algorithms can be used. If a heuristic algorithm treats at least one of ORL or CATR subproblems, the obtained solution of the RL problem cannot be called an optimal one. However, this approach can reduce size of the problem and consequently shorten execution time of the algorithm.

5. Exact algorithm

Optimization of the Web replica placement is a difficult task. In many real life cases, replicas or proxies are placed in fairly obvious nodes, e.g., the Internet service provider gateway [19]. However, in order to improve network parameters some algorithms must be applied to provide optimal or sub-optimal solutions.

As mentioned above, the RL problem is NP-complete. Therefore, heuristic algorithms not always ensure that the solution is optimal. To obtain an optimal solution an exact algorithm must be applied. To construct such an algorithm we propose to use the branch-and-cut (B&C) approach, which is a modification of the branch-and-bound method (B&B). The branch-and-bound approach has become a general solution method for various integer and mixed integer problems. The B&B algorithm is an intelligently structured search over the space of all feasible solutions. The solution space is repeatedly partitioned into smaller subsets, and a lower bound of the objective function is calculated within each subset. Subsets with bound that exceeds the best solution are excluded from further partitioning. For more information on branch-and-bound algorithms refer to [20].

Branch-and-cut is a relatively new but well accepted method proposed by Padberg and Rinaldi [22] for the traveling salesman problem. B&C algorithm is a combination of cutting plane algorithm and branch-and-bound algorithm. Cutting plane procedures are introduced into the bounding phase of B&B, enabling the branching phase to utilize the information on the known cuts, what improves the relaxation of the problem and enables calculation of more effective bounds. The B&C algorithm solves strengthened continuous relaxations of the problem, resulting in fewer analyzed nodes than for the B&B algorithm. The reader interested by branch-and-cut approach is referred to [1, 9, 21, 22].

It must be underlined that in order to find the exact solution of RL we must solve both subproblems concurrently. In this section we focus on the ORL problem and propose a branch-and-cut algorithm to solve this problem. The algorithm guarantees that we analyze the whole solution space of all possible combinations of replica location. However, for each analyzed selection Z_r we must solve the CATR subproblem. If we solve CATR by an exact algorithm, the obtained solution is globally optimal. Otherwise, if we tackle CATR with an heuristic algorithm, the solution can be claimed to be optimal.

5.1. Calculation scheme

In our branch-and-cut algorithm we start with selection Z_1 and generate a sequence of selections Z_r. In order to obtain the initial selection Z_1 we can solve the RL problem using one of heuristic algorithms proposed in [30, 31]. Each new selection Z_s is obtained from a certain selection Z_r of the sequence by complementing a normal variable z_i by a reverse variable z_k in the following way $Z_s := (Z_r - \{z_i\}) \cup \{z_k\}$. It means that we shift the replica from node i to a node k. The generating process can be represented as a branch and bound decision tree. Each node of the decision tree represents a selection. We say that the selection Z_s is a successor of the selection Z_r if there is a path from Z_r to Z_s.

For each set Z_r we constantly fix a set of nodes U_r. The state of nodes included in U_r cannot be changed. It means that nodes included in the set U_r cannot be used in the selection process. If the selection Z_s is obtained from the selection Z_r as $Z_s := (Z_r - \{z_i\}) \cup \{z_k\}$ we update the U_s as follows: $U_s := U_r \cup \{i\}$. There are two key elements of the branch-and-cut algorithm: lower bound of criterion function and branching rules. The lower bound is calculated to check if a "better" solution may be found. If the test result is negative we abandon the considered selection Z_r and backtrack to the selection Z_p from which the selection Z_r was generated. If Z_r was obtained from the selection Z_p in the following way $Z_r := (Z_p - \{z_i\}) \cup \{z_k\}$ we update the U_p as follows: $U_p := U_p \cup \{i\}$. It is a consequence of the fact that variables z_i are binary ones; and if we analyze all selections for which $z_i = 0$ we may constantly fix node i with $z_i = 1$. It must be noted that in branch-and-cut algorithm the lower bound calculation is enriched with the valid inequalities, which can "cut" the solution space.

The basic task of the branching rules is to find the variables for complementing to generate a new selection with the lowest value of criterion function possible. Since in the algorithm we change only location of replica, we use the function D given by (1) as the objective function. However, in order to calculate value of this function we must solve the CATR problem. During the branching operation of the tree we add a node i without a replica (the current variable $z_i = 0$) to the set U_r. When we backtrack, a node i hosting a replica is included in the set of fixed nodes (the current variable $z_i = 1$).

5.2. Branching rules

We define two sets as follows:

$$E_r = \left(\bigcup_{j \in (N - U_r)} \{ j : z_j = 0 \} \right),$$

$$M_r = \left(\bigcup_{j \in (N - U_r)} \{ j : z_j = 1 \} \right).$$

The set E_r comprises all nodes that are not constantly fixed for Z_r and can be selected for complementing. The set M_r includes all nodes that are not constantly fixed for Z_r and can be selected for removing a replica. Since, due to condition (3) the number of replicas must be equal to $|R|$, in the branching rule for a successor of Z_r we must remove a replica from a node hosting a replica, i.e., a node included in the set M_r and locate this replica in a node incorporated in the set E_r.

In order to explain the branching rule we introduce a new function $d_i(Z_r)$ defined as a distance from the node i to the closest replica included in the selection Z_r. To find the $d_i(Z_r)$ we consider only routes from the set Π_i. Using the $d_i(Z_r)$ we define the following function:

$$G(Z_r) = \sum_{i \in P} Q_i \, d_i(Z_r). \tag{10}$$

The function $G(Z_r)$ is only an estimation of the $D(Y_r)$. However, the main benefit of the function $G(Y_r)$ compared to $D(Y_r)$ is that it can be easily calculated. Next, we introduce the function $swap(r,i,k)$ used for selection of the variables for complementing. Without loss of generality, we assume that the selection Z_s is obtained from the selection Z_r in the following way $Z_s := (Z_r - \{z_i\}) \cup \{z_k\}$. According to the above discussion $i \in M_r$ and $k \in E_r$. The function $swap(r,j,k)$ is defined as follows:

$$swap(r,i,k) = G(Z_s) - G(Z_r). \tag{11}$$

According to definition (11), $swap(r,i,k)$ is a "gain" we obtain by moving the replica from the node i to the node k. As mentioned above, the function $G(Z_r)$ used in the definition (11) is an estimate of the objective function $D(Z_r)$. Since in the branching rule we want to generate a new selection and minimize the objective function $D(Z_r)$, we propose to use the function (11) as the decision function.

5.3. Lower bound

The simplest way to calculate a lower bound of an optimization problem is to relax some constraints in order to obtain a much simpler optimization problem in terms of computational complexity. In this case we relax the capacity constraint (5). Therefore, clients can be assigned to the closest node excluding nodes abandoned while generating the decision tree (we don't consider fixed nodes $i \in U_r$ for which $z_i = 0$).
Let N_r denote a set of fixed nodes $i \in U_r$ for which $z_i = 1$:

$$N_r = \left(\bigcup_{j \in U_r} \{ j : z_j = 1 \} \right).$$

For the current selection $|N_r|$ replicas are constantly located. It means that the number of replicas to be located is $(|R| - |N_r|)$. These replicas can be placed only in nodes included in the set $(V - U_r)$. Let $\underline{Z_r}$ denote a set of feasible selections that can be generated from the selection Z_r. We assume that the set $\underline{Z_r}$ compromises also the selection Z_r. According to discussion presented above, the following formula defines the number of elements of the set $\underline{Z_r}$:

$$|\underline{Z_r}| = \left(\begin{array}{c} |V - U_r| \\ |R| - |N_r| \end{array} \right).$$

Now we introduce the cutting inequality. Let $C(j)$ denote the capacity of all arcs leaving the node j:

$$C(j) = \sum_{i \in A} Q_i u_{ij} \tag{12}$$

Recall that u_{ij} is a binary variable that equals one if the source node of the arc i is node j. Due to the capacity constraint (5), a replica located in node j can serve at most $C(j)$ flow. Consequently, $C(Z_r)$ denotes the upper bound of flow that can be served by replicas located in nodes given by the selection Z_r:

$$C(Z_r) = \sum_{j \in V} y_j C(j). \tag{13}$$

The following formula is applied as a cutting plane in the lower bound:

$$C(Z_r) \geq \sum_{j \in V} (1 - y_j) Q(j). \tag{14}$$

The inequality (14) indicates whether or not the location of replicas given in selection Z_r can serve all demands in the network. In the right-hand side we sum bandwidth requirements of demands located in network nodes except for nodes, where replicas are placed.
Let Ψ_r denote a set of selections $Z_r \in \underline{Z_r}$ for which the inequality (14) is satisfied. Note that formula (10) defines a lower bound of the objective function for Z_r. In order to find a lower bound for the selection Z_r and all its successors we apply the following formula:

$$LB_r = \min_{Z_r \in \Psi_r} G(Z_r). \tag{15}$$

In formula (15) we analyze all feasible (in terms of the cutting inequality and fixed variables) selections that can be generated from current selection Z_r. If inequality (14) is satisfied, we calculate the function $G(Z_r)$ for considered replica location. Otherwise, we skip the given selection. Therefore, we perform fewer calculations of $G(Z_r)$. Since to obtain the $LB(Z_r)$ we relax the capacity constraint of the problem Eqs. (1)–(8), the LB_r is a lower bound of the objective function for the selection Z_r and all feasible selections that can be generated from Z_r. The elementary operation of lower bound consists of checking the inequality (14) and if it is satisfied, we must calculate $G(Z_r)$. Otherwise, when the cut (14) fails, we don't examine the given selection any further. To find $G(Z_r)$ we must find the shortest route to the replica for every client. Checking the cut (14) is much simpler, since values of $C(j)$ and $Q(j)$ are constant.
Note that in classic B&B algorithm the following formula can be used as lower bound:

$$LB_r = \min_{Z_r \in \underline{Z_r}} G(Z_r). \tag{16}$$

In formula (16) we don't use the cutting inequality. Therefore, for every selection $Z_r \in \underline{Z_r}$ the value of function $G(Z_r)$ must be found.

5.4. Algorithm

The problem RL (5–12) can be solved using the following algorithm. Let Z_1 denote a feasible initial solution. Set $U_1 := \varnothing$, $D^* := \infty$. The current selection is denoted by Z_r. Let LB_r be a lower bound of Z_r given by (15). We start with $r := 1$.

Step 1: Compute LB_r (15). If $LB_r \geq D^*$ go to Step 4. Otherwise if $LB_r < D^*$ go to Step 2.

Step 2: Compute $D(Z_r)$. If there is a feasible solution of $D(Z_r)$ and $D(Z_r) < D^*$ then set $D^* := D(Z_r)$. Go to Step 3. Otherwise, if there is no feasible solution of $D(Z_r)$ go to Step 4.

Step 3: If $E_r = \varnothing$ or $M_r = \varnothing$ go to Step 4. Otherwise find $i \in M_r$ and $k \in E_r$ for which the value of $swap(r,i,k)$ is lowest. Generate the selection Z_s (successor of Z_r) as follows $Z_s := (Z_r - \{z_i\}) \cup \{z_k\}$, $U_s := U_r \cup \{i\}$. Go to Step 1.

Step 4: Backtrack to the predecessor Z_p of the selection Z_r. If the Z_r has no predecessor, stop the algorithm. The selection Z^* associated with the current D^* is the optimal solution. Otherwise, if Z_r has predecessor, drop the data for Z_r and update data for Z_p as follows. If Z_r has been generated as $Z_r := (Z_p - \{z_i\}) \cup \{z_k\}$ then set $U_p := U_p \cup \{i\}$. Go to Step 1.

To obtain the value of function $D(Z_r)$ calculated in Step 2 we must solve the CATR problem for the particular location of servers given by the selection Z_r.

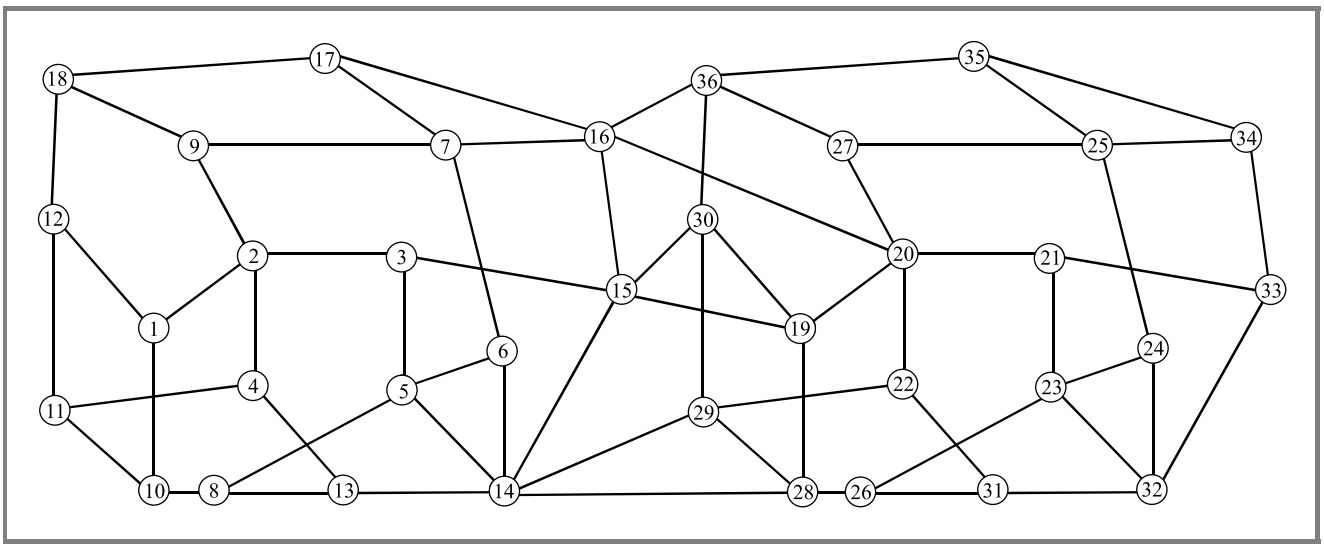

Fig. 1. Tested network.

6. Results

In this section we present results of numerical experiments. The B&C algorithm proposed in previous section was coded in C++. As mentioned above, there is a joint dependency between the replica location and the assignment of routes. Therefore, if a heuristic algorithm solves the CATR subproblem, the solution of the RL problem obtained cannot be called an optimal one. The CATR problem is very complex, even for small networks. Therefore, we decided to use a heuristic algorithm based on the flow deviation method [7] to find feasible solutions of CATR in Step 2 of B&C algorithm. Obviously, the solution of RL problem obtained cannot be called optimal. However, FD algorithm is a very effective method for solving multicommodity flow problems [4, 14, 33]. Consequently, the B&C is used as an intelligent method of searching the solution space of replica location problem.

Results presented in this section are obtained from simulations on a sample networks having 36 nodes and 128 arcs (Fig. 1). Arcs of tested network have various capacities in the range from 2 000 BU (bandwidth units) to 6 000 BU. In the experiment, it is assumed that in every network node there are 5 demands to a CDN server (replica). It means that there are overall 180 clients in the network. For a particular experiment bandwidth requirements are the same for all clients.

We have considered 6 scenarios. In Cases A, B, C and D there are 3 replicas to be located. The starting solutions indicating nodes hosting replicas are $\{2,16,32\}$; $\{1,16,32\}$; $\{1,14,32\}$; $\{2,30,32\}$ respectively for Cases A, B, C and D. In experiment E there are 2 replicas to be located and the initial solution is $\{14,25\}$. Finally, for Case F, 4 replicas are to be placed and the starting solution is $\{2,14,16,30\}$. Initial solutions are found heuristically. We studied the performance of the algorithm for increasing traffic load, examining the evolution of the network status towards a saturation condition. In particular, for every scenario we examine 26 demand patterns having the value of one client's demand between 200 BU and 450 BU.

The first objective of experiments was to investigate how increasing replicas' number changes the network overall flow. In Fig. 2 we report performance of B&C algorithm for Scenarios A, E and F for which the number of replicas to be placed is 2, 3 and 4, respectively. The *x*-axis is the total demand in the network (sum of all clients' demands), and the *y*-axis is the network flow (objective function of the RL problem). It is obvious that increasing the number of replicas decreases the network flow. More replicas means that a replica is closer to clients and the route to the replica is shorter. In Scenario E (2 replicas) the algorithm finds a feasible solution only for first 6 demand patterns. For 3 replicas, 23 of 26 considered demand patterns yield feasible result. Finally, for the last scenario having 4 CDN servers all demand patterns are satisfied.

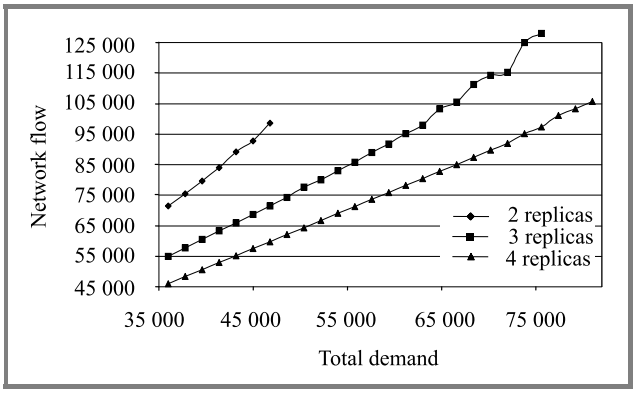

Fig. 2. Network flow as a function of total demand in the network for various number of replicas.

Since the branch-and-cut approach is a relatively new method compared to branch-and-bound algorithm, we made several tests to evaluate performance of B&C against B&B.

To obtain a B&B algorithm we slightly modified the algorithm developed in previous section and applied the lower bound given by (16). All other operations of the B&B algorithm are the same as for B&C.

First, we show how the B&C algorithm reduces the number of nodes in the solution tree of the algorithm. Figure 3 plots the number of nodes in the solution tree for Scenarios A and E as a function of total demand in the network. We show results for both algorithms: B&B and B&C. The *x*-axis is the total demand in the network. The *y*-axis uses logarithmical scale and denotes the number of nodes in the decision tree of the algorithm. We observe similar performance between the bars, reflecting A and E scenarios. For low load (according to the number of replicas), both algorithms need the same number of nodes in the decision tree. For more saturated network, B&C produces significantly less nodes than the B&B. Similar trend can be observed for other scenarios. Summarizing over all experiments, B&B algorithm produces 26 186 nodes, while for B&C the corresponding value is 19 958 nodes. The biggest difference is observed for highly saturated demand pattern in Scenario C, for which B&C needs only 104 nodes compared to 4 184 nodes of B&B. This proves that the branch-and-cut algorithm is more effective then the B&B one. For the problem considered, B&C outperforms B&B, especially for large traffic load that leads to network saturation.

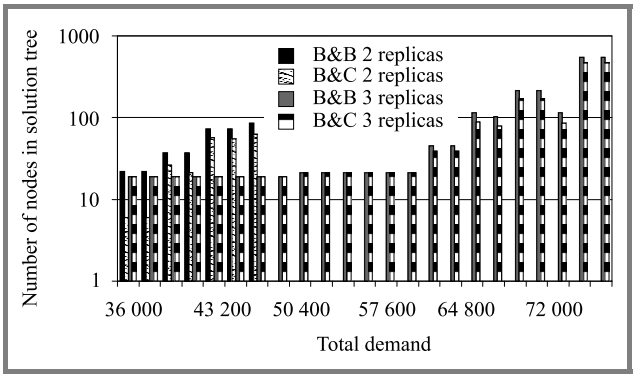

Fig. 3. Number of nodes in the solution tree for B&B and B&C algorithms.

To confirm the advantage of B&C over B&C we present further results. Recall that the main benefit of B&C algorithm is the use of cutting inequality (14) that enables reduction of calculations of function $G(Z_r)$ given by (10). Figure 4 shows the number of elementary operations performed in B&C algorithm. There are two types of elementary operations. The first type is applied when the cut (14) is satisfied and calculation of the function $G(Z_r)$ given by (10) is needed. The second operation consists only of checking the cut inequality and it is used when the cut inequality doesn't hold. The *x*-axis is the total demand in the network, and the *y*-axis denotes the number of operations. Figure 4 shows present results for Scenarios A and D. Generally the trend is the same for both cases. For low loaded networks the number of cuts exceeds the number of function *G* calculations. However, for this experiments the number of decision tree nodes is relatively small. For higher demand patterns curves become stable, number of cuts is about 3 times lower than number of the function *G* calculations. In these cases the number of decision tree nodes grows, and the lower bound is calculated for different selections and combinations of fixed nodes. Similar trend was observed for other scenarios.

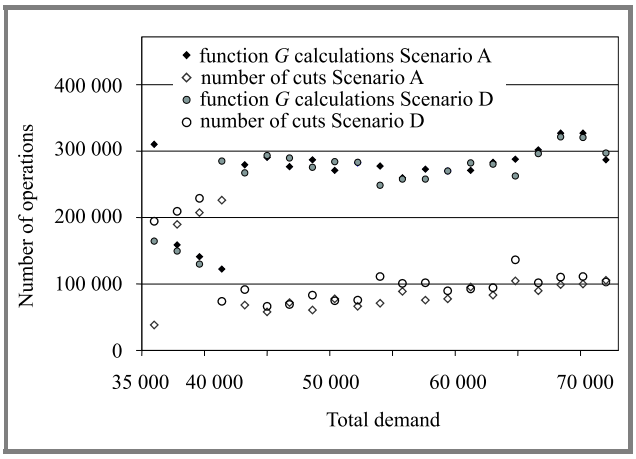

Fig. 4. Number of elementary operations for B&C algorithm.

Recall that for B&B algorithm we don't use the cutting plane. This creates additional overhead. For each analysed selection in the lower bound we must calculate the formula (10). Therefore, for the B&B algorithm the number of function *G* calculations is equal to or bigger than the sum of all elementary operations (of both types) in B&C algorithm.

Next, we present the execution time of B&C and B&C algorithms. The program implementing both algorithms was run on an IBM-compatible PC with 2 GHz Intel processor and 512 MB of RAM. It is worth remarking that decision time does not include I/O time for input of various files. It includes only the time of design output. Figure 5 depicts the decision time of both algorithms for Scenarios A and C.

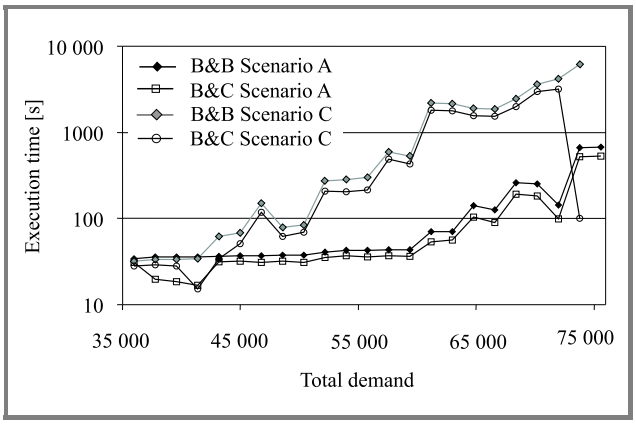

Fig. 5. Execution time of B&B and B&C algorithms for Scenarios A and C.

The *x*-axis is the total demand in the network. The *y*-axis uses logarithmical scale and denotes the decision time in seconds. We observe that B&C outperforms B&B for all demand patterns considered. The gap is similar for different network loads. It is worth remarking that when the network load grows, the execution time also increases.
Comparing Figs. 3 and 4 to Fig. 5 we can reach interesting conclusions. Analysis of B&C and B&B decision times obtained for various demand patterns shows only slight differences. On the other hand, observation of decision nodes' number and effectiveness of the cut inequality shows many differences in performance for various total loads in the network. For low loads the cut inequality works more effectively and gives relatively more positive tests. For more saturated networks the B&C produces much less solution tree nodes then B&B. Thus, these two effects combine to yield similar performance of B&C and B&B in terms of decision time for all considered demand patterns.
It should be noted that the decision time of B&B and B&C algorithms is influenced strongly by the execution time of the heuristic algorithm applied to solve the CATR subproblem. It was observed that the execution time of heuristic algorithm depends on the solved problem; the time is not constant. Therefore, analysis of the exact algorithms' decision time only from the perspective of decision tree nodes' numbers or effectiveness of cut inequality is not always sufficient.

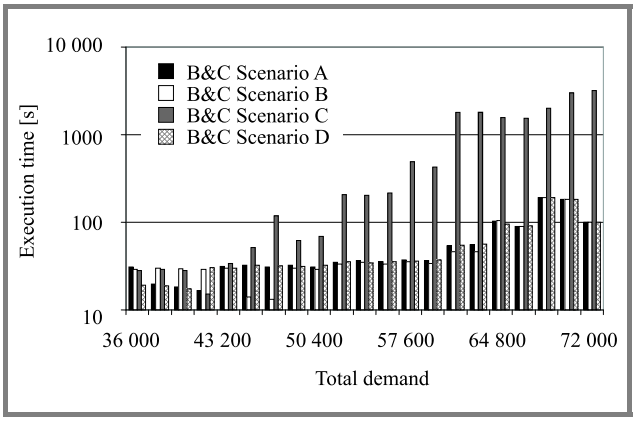

Fig. 6. Execution time of B&C algorithm for Scenarios A, B, C, and D.

Another important issue we have examined is the impact of the starting solution on the performance of the B&C algorithm. Figure 6 shows the decision time of B&C algorithm for Scenarios A, B, C and D. Recall that all these cases have 3 replicas to be located; however the initial solution of each scenario is different. The *x*-axis is the total demand in the network. The *y*-axis uses logarithmical scale and denotes the decision time in seconds. We observe very similar performance for three bar series, reflecting Scenarios A, B and D. For Case C the performance is much worse and the decision time is about 16 times longer then for other cases. This becomes evident when we analyse the quality of starting solutions applied in individual scenarios. The starting solution used in Scenario C gives an average result about 10% worse than the result obtained for B&C. The corresponding difference is 1%, 7% and 5% for Scenarios A, B and D, respectively. We can conclude that the starting solution is an important issue in the B&C algorithm, which has a strong effect on the execution time.
In summary, we must underline that experimental data showing comparison of B&B and B&C methods is reasonably well explained by the theoretical foundations of both algorithms presented in previous section.

7. Conclusion

This paper deals with the problem of replica location in a content delivery network. We have presented and discussed basic information on CDNs and MPLS. We have formulated an optimisation problem of replica location in a CDN. The network flow has been modelled as a connection-oriented flow. Furthermore, the capacity constraint has been incorporated into the model. This problem is NP-complete. The objective function is the overall flow in the network. To our knowledge, this problem has not received much attention in the literature. Using optimisation model, an exact algorithm based on the branch and cut approach has been developed. Two main operations of the algorithm: lower bound and branching rule have been discussed in detail. Results of numerical experiments have been discussed. From both experimental and analytical viewpoints, we have concluded that when applied to replicas location problem, the branch-and-cut algorithm outperforms branch-and-bound method in terms of execution time and number of analysed nodes of the decision tree. In future work we want to make more extensive tests in order to evaluate this algorithm and compare it with other algorithms.

References

[1] K. I. Aardal and C. P. M. van Hoesel, "Polyhedral techniques in combinatorial optimization II: computations and applications", *Stat. Nederl.*, vol. 53, pp. 129–178, 1999.

[2] V. Arya, N. Garg, R. Khandekar, A. Meyerson, and V. Pandit, "Local search heuristics for *k*-median and facility location problems", in *Proc. ACM Symp. Theory Comput.*, Hersonissos, Crete, Greece, 2001, pp. 21–29.

[3] M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm, "World wide Web caching: the application-level view of the Internet", *IEEE Commun. Mag.*, pp. 170–178, June 1997.

[4] J. Burns, T. Ott, A. Krzesiński, and K. Muller, "Path selection and bandwidth allocation in MPLS networks", *Perform. Eval.*, vol. 52, pp. 133–152, 2003.

[5] M. Charikar and S. Guha, "Improved combinatorial algorithms for the facility location and *k*-median problems", in *Proc. IEEE Symp. Found. Comput. Sci.*, New York, USA, 1999, pp. 378–388.

[6] B. Davison, "The design and evaluation of Web prefetching and caching techniques", Ph.D. thesis, 2002, http://www.cse.lehigh.edu/∼brian/pubs/2002/thesis/

[7] L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation method: an approach to store-and-forward communication network design", *Networks*, vol. 3, pp. 97–133, 1973.

[8] S. Guha, A. Meyerson, and K. Munagala, "Improved approximation algorithms for fault-tolerant facility location", in *Proc. ACM-SIAM Symp. Discr. Algor.*, Washington, USA, 2001, pp. 636–641.

[9] O. Gunluk, "Branch-and-cut algorithm for capacitated network design problems", *Math. Programm.*, vol. 86, pp. 17–39, 1999.

[10] K. Jain, M. Mahdin, and A. Saberi, "A new greedy approach for facility location problems", in *Proc. ACM Symp. Theory Comput.*, Montreal, Canada, 2002.

[11] S. Jamin, Ch. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "On the placement of Internet instrumentation", in *Proc. IEEE INFOCOM 2000*, Tel-Aviv, Israel, 2000, pp. 295–304.

[12] S. Jamin, Ch. Jin, Y. Jin, A. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the Internet", in *Proc. IEEE INFOCOM 2001*, Anchorage, Alaska, USA, 2001, pp. 31–40.

[13] R. Karp, "On the computational complexity of combinatorical problems", *Networks*, vol. 5, pp. 45–68, 1975.

[14] A. Kasprzak, *Topological Design of Wide Area Networks*. Wrocław: Wrocław University of Technology Press, 2001.

[15] L. Kennington, "A survey of linear cost multicommodity networks flows", *Oper. Res.*, vol. 26, pp. 209–236, 1978.

[16] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content delivery networks", in *Proc. ACM SIGCOMM Internet Measur. Worksh.*, San Francisco, USA, 2001.

[17] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem", *IEEE/ACM Trans. Netw.*, vol. 8, pp. 568–582, 2000.

[18] T. Li, "MPLS and the evolving Internet architecture", *IEEE Commun. Mag.*, pp. 38–41, Dec. 1999.

[19] B. Li, M. Golin, G. Italiano, X. Deng, and K. Sohraby, "On the optimal placement of Web proxies in the Internet", in *Proc. IEEE INFOCOM'99*, New York, USA, 1999, pp. 1282–1290.

[20] J. Mitchell and E. Lee, "Branch-and-bound methods for integer programming", in *Handbook of Applied Optimization*. Oxford: Oxford University Press, 2002.

[21] J. Mitchell, "Branch-and-cut methods for combinatorial optimization problems", in *Handbook of Applied Optimization*. Oxford: Oxford University Press, 2002.

[22] M. Padberg and G. Rinaldi, "Optimization of a 532-city traveling salesman problem by branch-and-cut", *Oper. Res. Lett.*, vol. 6, 1987.

[23] G. Peng, "CDN: content distribution networks", Tech. Rep., 2003, http://www.sunysb.edu/tr/rpe13.ps.gz

[24] L. Qiu, V. Padmanabhan, and G. Voelker, "On the placement of Web server replicas", in *Proc. IEEE INFOCOM 2001*, Anchorage, Alaska, USA, 2001, pp. 1587–1596.

[25] M. Rabinovich, "Issues in Web content replication", *Data Eng. Bull.*, vol. 21, no. 4, 1998.

[26] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture", RFC 3031, Jan. 2001.

[27] A. Vakali, "An evolutionary scheme for Web replication and caching", in *Proc. 4th Int. Web Cach. Worksh.*, San Diego, USA, 1999.

[28] K. Walkowiak, "Designing of survivable Web caching", in *Proc. 8th Polish Teletraffic Symp.*, Zakopane, Poland, 2001, pp. 171–181.

[29] K. Walkowiak, "Modelling of Web server replication system in MPLS networks", in *Proc. Inform. Syst. Model. ISM 2002*, Roznov pod Radhostem, Czech Republic, 2002, pp. 213–220.

[30] K. Walkowiak, "A new exact algorithm for Web replica location problem in MPLS networks", in *Proc. Polish-Germany Teletraffic Symp. PGTS 2002*, Gdańsk, Poland, 2002, pp. 307–314.

[31] K. Walkowiak, "Some approaches to solve a Web replica location problem in MPLS networks", in *Internet Technologies, Applications and Societal Impact*, W. Cellary and A. Iyengar, Eds. Boston [etc.]: Kluwer, 2002, pp. 61–72.

[32] K. Walkowiak, "On application of genetic algorithms to replica location problem", in *Proc. Comput. Recogn. Syst. KOSYR 2003*, Milków, Poland, 2003, pp. 445–450.

[33] K. Walkowiak, "A new approach to survivability of connection oriented networks", in *Lectures Notes in Computer Science*. Berlin, Heidelberg: Springer-Verlag, 2003, vol. 2657, pp. 501–510.

[34] J. Wang, "A survey of Web caching schemes for the Internet", *ACM Comput. Commun. Rev.*, pp. 36–46, Oct. 1999.

[35] B. Williams, "Transparent Web caching solutions", in *Proc. 3rd Int. WWW Cach. Worksh.—TF-Cache Meet.*, Manchester, England, 1998.

[36] A. Wierzbicki, "Internet cache location and design of content delivery networks", in *Lectures Notes in Computer Science*. Berlin, Heidelberg: Springer-Verlag, 2002, vol. 2376, pp. 69–82.

Krzysztof Walkowiak was born in Poland in 1973. He received the M.Sc. and Ph.D. degrees in computer science from Wrocław University of Technology in 1997 and 2000, respectively. Since 2001 he has been an Assistant Professor at the Chair of Systems and Computer Networks, Faculty of Electronics, Wrocław University of Technology. His research interest is mainly focused on optimization of connection-oriented networks, survivability issues of ATM, MPLS, and application of soft-optimization techniques for design of computer networks, Web caching.

e-mail: Krzysztof.Walkowiak@pwr.wroc.pl
Chair of Systems and Computer Networks
Faculty of Electronics
Wrocław University of Technology
Wybrzeże Wyspiańskiego st 27
50-370 Wrocław, Poland